APRENDER A PROGRAMAR

José Luis Rodríguez Muñoz

La empresa de Inteligencia Artificial OpenAI liberó, en diciembre de 2022, la última versión de su aplicación ChatGPT basada en el modelo del lenguaje GPT-3. Esta herramienta se alimenta de datos de lenguaje natural para generar respuestas a preguntas. Alguien la lanzó el siguiente reto en forma de pregunta.

Pregunta:
Escribe un breve poema en inglés con rima que explique la Teoría de la Relatividad General de Einstein en términos sencillos pero precisos.

Respuesta (original en inglés con rima):

If you want to understand Einstein's thought

It's not that hard if you give it a shot

General Relativity is the name of the game

Where space and time cannot remain the same

Mass affects the curvature of space

Which affects the flow of time's race

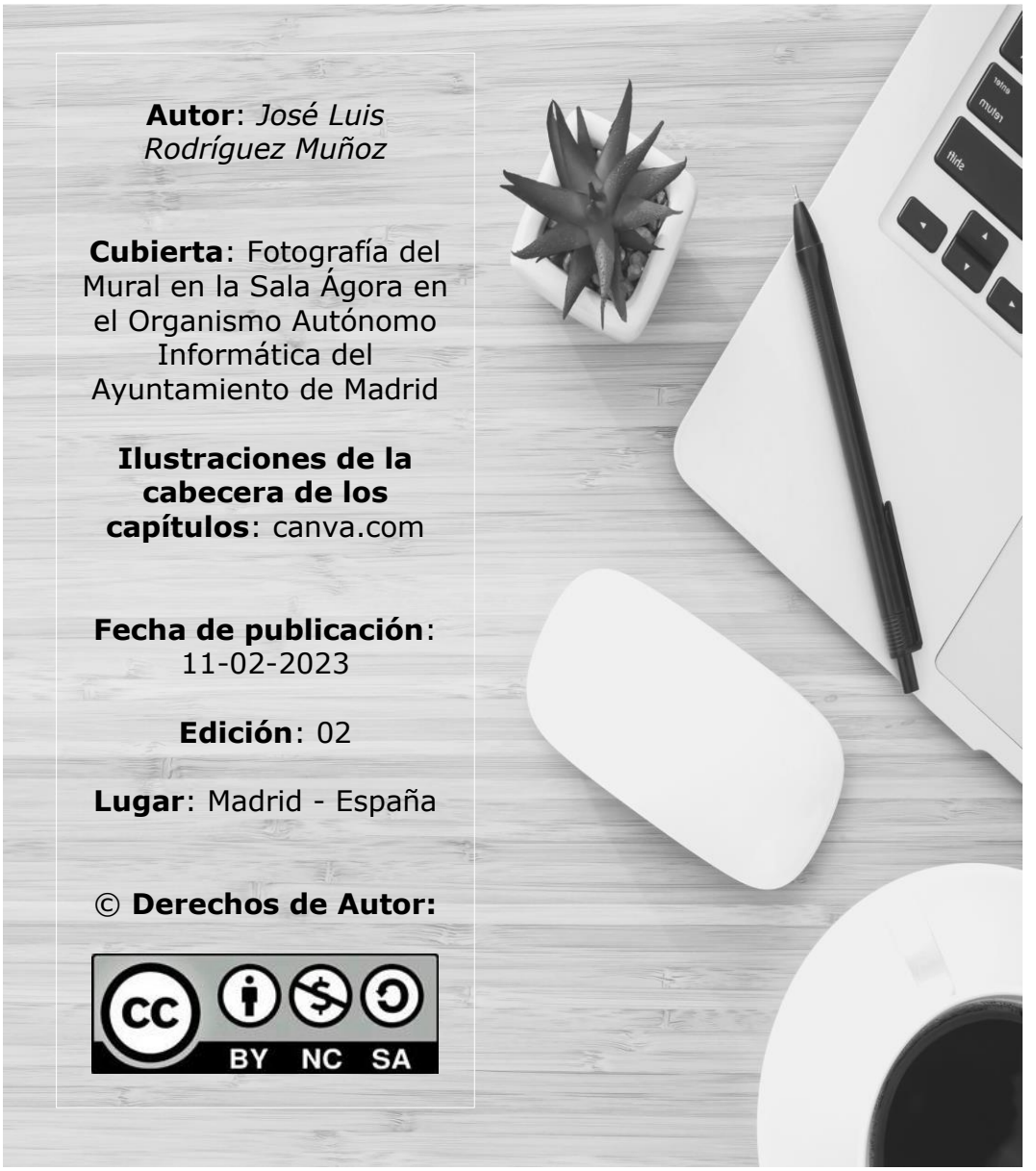
An object's motion will be affected

By the distortion that is detected

The closer you are to a large mass

The slower time will seem to pass

The farther away you may be



Autor: *José Luis Rodríguez Muñoz*

Cubierta: Fotografía del Mural en la Sala Ágora en el Organismo Autónomo Informática del Ayuntamiento de Madrid

Ilustraciones de la cabecera de los capítulos: canva.com

Fecha de publicación: 11-02-2023

Edición: 02

Lugar: Madrid - España

Tipo de Licencia de los Derechos de Autor

Aprender a Programar | Ejemplos en PSeInt

- No comercial (**BY-NC**): Esta licencia permite modificar la obra y construir otra a partir de la original pero siempre y cuando su finalidad no sea comercial.

- Compartir igual (**BY-SA**): Para usar las obras bajo esta licencia es imprescindible citar al autor y licencien sus nuevas obras bajo los mismos términos, estos deben ser idénticos a: CC-BY-NC-SA.

A mi familia

Prefacio

Aprender a Programar | Ejemplos en PSeInt

Esta obra está destinada a quienes quieren utilizar el programa informático de software libre **PSeInt** para aprender a realizar programas de ordenador, a través de ejemplos de código fuente (algoritmos) escritos en lengua castellana.

Por lo tanto, el objetivo del libro es que aprenda a programar mediante la práctica.

En el mismo, podrá encontrar el aprendizaje de la escritura de programas informáticos partiendo del nivel cero, como son los tipos de sentencias básicas, los tipos de datos simples, las tablas, las subrutinas, recursividad, paso de valores por valor y por referencia, el ámbito de las variables.
Y temas profesionales como el desarrollo de software siguiendo los principios de la Programación Estructurada, la validación de los datos de entrada al programa, el formateo de los datos de salida y la creación de datos de prueba del algoritmo realizado, entre otros.

Además, encontrará enunciados de **47 ejercicios** a resolver, así como **61 soluciones** comentadas a los mismos. También encontrará la explicación del desarrollo de **7 aplicaciones** de codificación de programas en **PSeInt**.

El libro está respaldado por una página web que se encuentra en:

[Consulte el apartado Contacto en el capítulo Despedida]

Donde podrá copiar el código fuente de los ejemplos, ejercicios, soluciones y las 7 aplicaciones presentadas.

Espero que sea de su interés y le resulte útil.

José Luis Rodríguez Muñoz

A quién se dirige este libro

Este libro está destinado a quienes quieren aprender a programar y no tienen ningún conocimiento de programación de computadoras.

A programar se aprende programando y no ha sido intención del autor recoger o aglutinar una batería de ejercicios propuestos a modo de biblioteca o recopilación de ejercicios para resolver.
El objetivo de los ejercicios ha sido mostrar situaciones y decisiones, en las soluciones, que le resulten enriquecedoras en su aprendizaje de la creación de programas informáticos.

Es recomendable que lea el libro de forma secuencial: un capítulo detrás de otro; y, si no se encuentra muy seguro de poder afrontar el capítulo de las **Aplicaciones**, se recomienda que use este capítulo como apartado de consulta. Es decir, para aprender a realizar ejercicios de mayor tamaño que los expuestos en capítulos anteriores.

En el capítulo del **Anexo** encontrará apartados que le serán de utilidad en diferentes momentos de su aprendizaje. En el mismo, se encuentra el proceso de instalación del programa PSeInt y su **configuración**: elegir las opciones adecuadas para que pueda llevar a cabo el seguimiento de los ejemplos y ejercicios en las mismas condiciones y parámetros que se han utilizado para escribir el libro.

En la página web del libro encontrará el código fuente de los capítulos del libro, pero recomiendo que escriba los algoritmos usted mismo y que utilice el código, de la página web, solamente a título informativo y para aprender a leer algoritmos que ha pensado y creado otro programador. Le adelanto que en una empresa de informática hay más aplicaciones a modificar, que aplicaciones a realizar desde un principio.

Es decir, su primer trabajo será, muy probablemente, entrar a programar la modificación de una aplicación que está en la **Fase de Mantenimiento** de una aplicación de su empresa, por lo tanto, será una aplicación que ya está desarrollada y en funcionamiento; y, tendrá que aprender a modificarla leyendo el código fuente (programa) que ha escrito otro programador/a.

Sumario

Aprender a Programar | Ejemplos en PSeInt

Ejercicios para practicar → Capítulo 6

Aplicaciones → Capítulo 7

Anexo → Capítulo 8

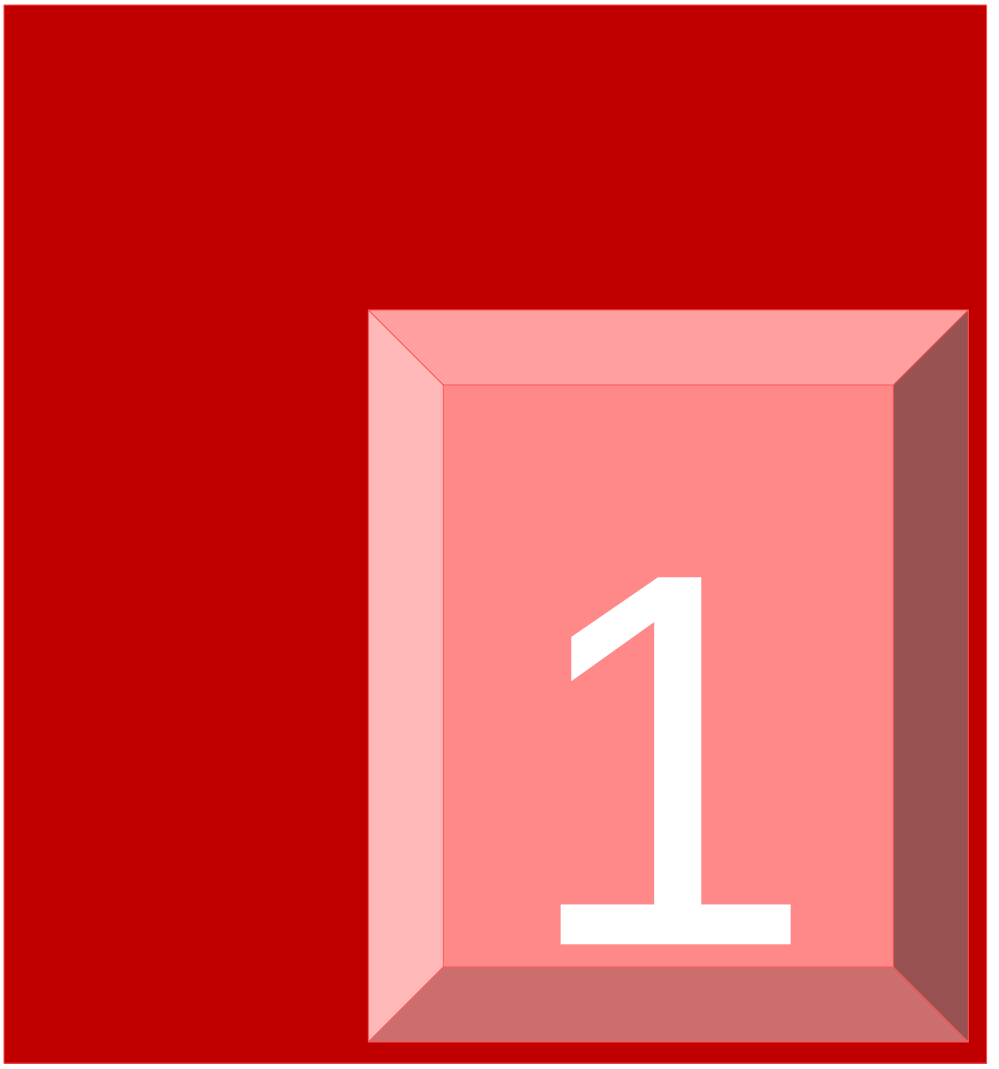
Despedida → Capítulo 9

Glosario de Términos – Páginas **316** - **317**

Índice del Contenidos

Aprender a Programar | Ejemplos en PSeInt

Un programa informático



Capítulo 1

Un programa informático

Aprender a Programar | Ejemplos en PSeInt

Un **problema** es algo que nos surge en nuestra vida cotidiana o laboral. Los problemas son una dificultad, un impedimento o una necesidad a resolver. Según la **Real Academia Española** (https://www.rae.es/) un problema es un planteamiento de una situación cuya respuesta desconocida debe obtenerse a través de métodos científicos.

Resolver un problema

Cuando pensamos en la forma de afrontar un problema, estos son los pasos que hemos interiorizado, como especie humana, para resolverlo:

- → Comprender el problema
- → Buscar una solución o un conjunto de soluciones
- → Elegir la solución adecuada
- → Implementar la solución, es decir, ponerla en práctica
- → Verificar si esta solución tuvo los resultados deseados de forma que hayamos resuelto el problema

Comprender el problema

El primer paso para diseñar y resolver un problema es comprender el problema. Es una etapa particularmente crítica porque de ella depende el desarrollo de la solución ya que, si no identificamos correctamente el problema, obtendremos una solución que no resolverá el problema planteado. Esta etapa incluye los siguientes pasos:

→ Descripción del problema: La descripción del problema la realiza nuestro usuario o cliente, que es el que conoce y plantea el problema; también nosotros nos podemos haber dado cuenta que nuestro cliente tiene una dificultad o problema y, entonces, seremos nosotros mismos los que plantearemos el problema para su solución.

→ Encontrar los datos: Hay que encontrar los elementos (datos) en los que se basa el problema. Por ejemplo, en una ecuación:

$$ax + b = 0$$

Los datos son los factores "a" y "b".

→ Encontrar lo que se pide: Tenemos que encontrar la información que necesitamos para solucionar el problema. Es decir, debemos encontrar qué se pide. Continuando con el ejemplo anterior, la información es la "x" de la ecuación: $ax + b = 0$

En este paso, es importante aclarar todos los puntos "oscuros" del problema y no quedarnos con ninguna duda del alcance (situación en la que se plantea el problema) y cuál es la problemática "real" del problema que vamos a solucionar.

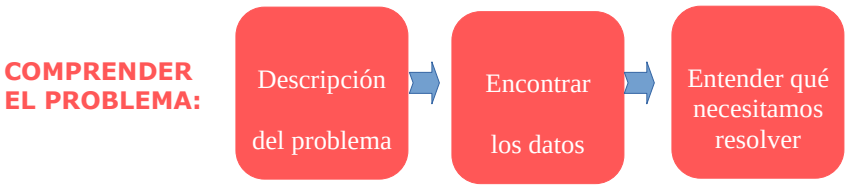


Ilustración 01.01 | Pasos para comprender un problema

Buscar una solución o un conjunto de soluciones

Una vez que hemos reconocido los datos y lo que se solicita, se debe buscar una solución para resolver el problema. A menudo eso no es inmediato o trivial. Por lo tanto, es necesario buscar un método o **un conjunto lógico de pasos que conduzca a la solución**. Estos pasos deben conducir a una **solución determinista**. Es decir, al aplicar ese conjunto de pasos para un mismo problema, debemos obtener **siempre la misma solución** o resultado. Y, además, debe de ser una **solución finita**. Es decir, **los pasos se deben realizar en un período relativamente corto de tiempo y con un final** o término con el que resolvamos el problema.

En el caso de los problemas solucionados por ordenador, debemos utilizar un **algoritmo**.

ojo

> En matemáticas y ciencias de la computación, un algoritmo es una secuencia finita de instrucciones bien definidas que se pueden implementar por computadora, típicamente para resolver una clase de problemas o realizar un cálculo.

Elegir la solución correcta

Normalmente, un problema puede tener más de una solución o algunas soluciones son más eficientes (tienen menos gastos) que otras. Elegir la solución adecuadamente conduce a un resultado más corto en tiempo, más barato en el uso de recursos o a obtener un resultado más confiable: con menos margen de error en la solución obtenida.

Implementar la solución

Una vez seleccionado el método de resolución, se deben tomar una serie de acciones para implementarlo: llevarlo a cabo. En otras palabras, aplicar **el Algoritmo**

Verificar si esta solución obtuvo los resultados deseados

Finalmente, tras llevar a cabo la aplicación de la solución, es necesario probar el algoritmo (los pasos a realizar) con varios datos para examinar si obtenemos resultados correctos cada vez que se aplica el algoritmo en diferentes escenarios de datos de prueba.

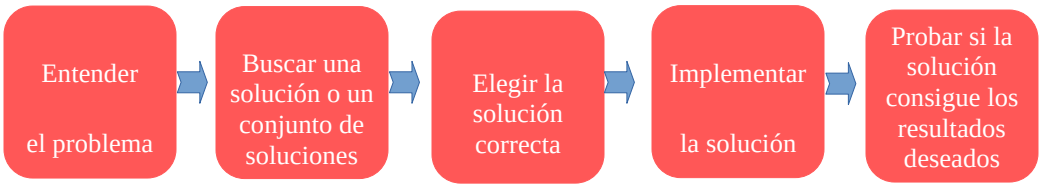


Ilustración 01.02 | Pasos para resolver un problema

Producto Informático

La industria informática se basa en la creación de productos informáticos que resuelven un problema por medio de computadoras y accesorios. Todo producto informático está formado de una parte Software (programas informáticos) y otra parte Hardware (elementos físicos: ordenadores, móviles, teclados, pantallas, impresoras, ratones, escáneres, cámaras, relojes inteligentes, coches, lavadoras, etc.). Por lo tanto, un **programa informático** es un producto software que funciona en un elemento físico (hardware).

El desarrollo de programas informáticos no deja de ser otro producto más que desarrolla una empresa u organización. Como todo desarrollo de producto en una empresa, un **desarrollo de producto informático empresarial** se afronta mediante el siguiente gráfico, dentro de un Proyecto para la construcción de un Producto.

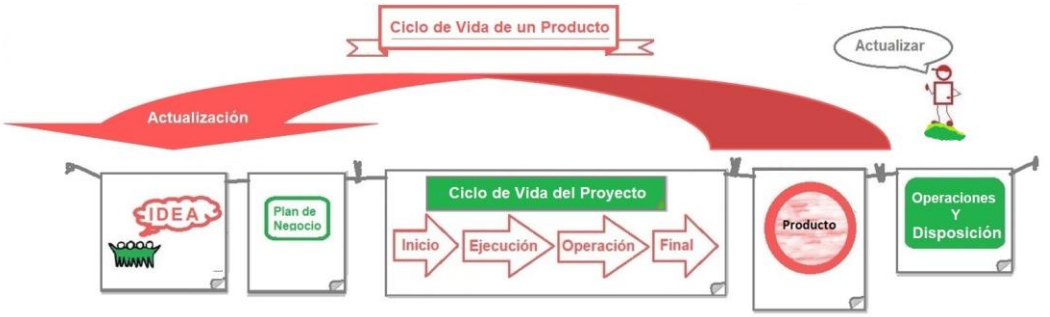


Ilustración 01.03 | Ciclo de Vida de un Producto

Dentro del Ciclo de Vida de un Proyecto, los programas informáticos se producen en un tipo de Proyecto especial que se denomina Proyecto Informático. El Proyecto Informático sirve para el desarrollo (construcción) de Programas Informáticos como un producto más de la empresa.

Un **Proyecto** es un esfuerzo temporal, llevado a cabo para crear un producto, servicio o resultado único; que tiene un principio y fin definidos, para obtener un resultado duradero

Fases de un Proyecto Informático de software

Las Fases de un Proyecto Informático de Software responde al siguiente Ciclo de Vida.

Ilustración 01.04 | Ciclo de Vida o Fases de un Proyecto Informático

Estas fases se resumen en los siguientes apartados.

Fase 1: Planificación y Especificación de Requisitos

Durante esta primera fase se fijan los objetivos que debe cumplir el proyecto, teniendo en cuenta los objetivos estratégicos y tácticos de la empresa; gracias al Alcance (conjunto de actividades que deben realizarse), Coste (precio del proyecto) y Tiempo (duración) que debe cubrir la aplicación (programas informáticos) de acuerdo con las necesidades que muestran los usuarios finales mediante entrevistas y encuestas. Además, se debe entregar una Planificación de los trabajos a realizar que debe respetar el Alcance, Coste y Tiempo que se pretende obtener con este proyecto.

Es importante señalar la importancia del **documento de Especificación de Requisitos**; este contiene todos los puntos que deberá cubrir la aplicación, teniendo en cuenta el Alcance del proyecto, y respetando el Coste y el Tiempo asignado para construir ese Alcance; sin entrar en detalles internos de cómo construir el proyecto.

El responsable de esta fase 1 es el Analista Funcional

Fase 2: Análisis

En la segunda etapa se descompone el sistema por partes que puedan desarrollarse por separado. Esta parte es fundamental para el desarrollo, porque se describen paso a paso los procesos y tareas a realizar, así como las tecnologías a emplear para resolver el "**qué** hacer" para cumplir con el documento de Especificación de Requisitos.

El responsable de esta fase 2 es el Analista Orgánico

Fase 3: Diseño

En esta etapa se ajusta y depura la fase de Análisis para resolver el "**cómo**" llevar a cabo la funcionalidad requerida en el documento de Especificación de Requisitos.
Gran parte de la documentación de todo el desarrollo se realiza en esta fase, describiendo textual y gráficamente las partes y funciones del desarrollo.

El responsable de esta fase 3 es el Analista Programador

Fase 4: Implementación

En esta fase, también llamada Codificación; es en la que se invierte más tiempo, en torno al 75% de los recursos. Durante esta fase de Implementación (**construcción**) se realizan diversas modificaciones del producto diseñado; debido a errores, adecuaciones del código a las necesidades del usuario (cambio de requisitos), etcétera.

El responsable de esta fase 4 es el Programador

Fase 5: Pruebas

Esta etapa es fundamental, y debe contemplarse en las fases de análisis, diseño e implementación; ya que de las pruebas **se conseguirá que el usuario final apruebe la aplicación**. Si bien los desarrolladores ya han comprobado que el código esté libre de fallos en la fase de implementación; y, que cumple con las especificaciones fijadas en los requisitos del usuario en las fases de análisis y diseño. En esta fase se produce la primera toma de contacto del usuario final con la aplicación; y puede que, para el usuario, la aplicación no se adapte exactamente a aquello que había imaginado en su mente.

El responsable de esta fase 5 es el Testeador o Especialista de Pruebas

Fase 6: Mantenimiento

Esta etapa se produce una vez que la aplicación se ha entregado (es decir, el producto informático se ha finalizado, se ha aprobado por los usuarios y se ha puesto en funcionamiento). Consiste en introducir los ajustes necesarios para mejorar el rendimiento (velocidad de ejecución) y corregir los problemas que puedan surgir, al utilizar datos reales. Así como actualizar el producto entregado (aplicación) a los cambios tecnológicos del entorno hardware y software; y, realizar los cambios evolutivos para adaptarse a las nuevas funcionalidades deseadas por los usuarios, que pidan a la aplicación informática (programas informáticos).
Las acciones que se realizan en la fase de mantenimiento se denominan Mantenimiento **Evolutivo o Perfectivo** (nuevos requerimientos o funcionalidades), **Correctivo** (corrección de errores), y **Adaptativo** (actualización tecnológica al cambio de hardware y/o software) y **Preventivo** (se aplican correcciones para mejorar el software y prevenir posibles errores que se puedan producir).

El responsable de esta fase 6 es el Equipo de Desarrollo: todos los perfiles profesionales expuestos anteriormente. El equipo de desarrollo puede estar formado por los mismos participantes que realizaron la creación del producto o por un equipo completamente nuevo y separado del equipo que desarrolló la aplicación

Fase 0: Integración

Esta fase se aplica a todas las otras y, es responsabilidad del Jefe de Proyecto que debe coordinar la integración de los diferentes equipos participantes en las diferentes fases, así como del traspaso de entregables de documentación. También debe supervisar la calidad del producto y vigilar por la calidad en la comunicación entre los diferentes equipos que intervienen en el desarrollo.

El **Jefe de Proyecto** es, por tanto, el integrador y el responsable del éxito del proyecto informático del que es imprescindible disponer de una documentación detallada y actualizada.

El responsable de esta fase 0 es el Jefe de Proyecto

A continuación, se visualiza (haciendo un zoom o ampliación de la ilustración del Ciclo de Vida de un Producto) la relación entre las etapas del Ciclo de Vida de un Proyecto, dentro de la construcción de un Producto; y, del Ciclo de Vida de un Proyecto Informático. Es decir, **cómo encajan las Fases de un Proyecto Informático con las etapas de un Proyecto**, para la construcción de un producto empresarial.

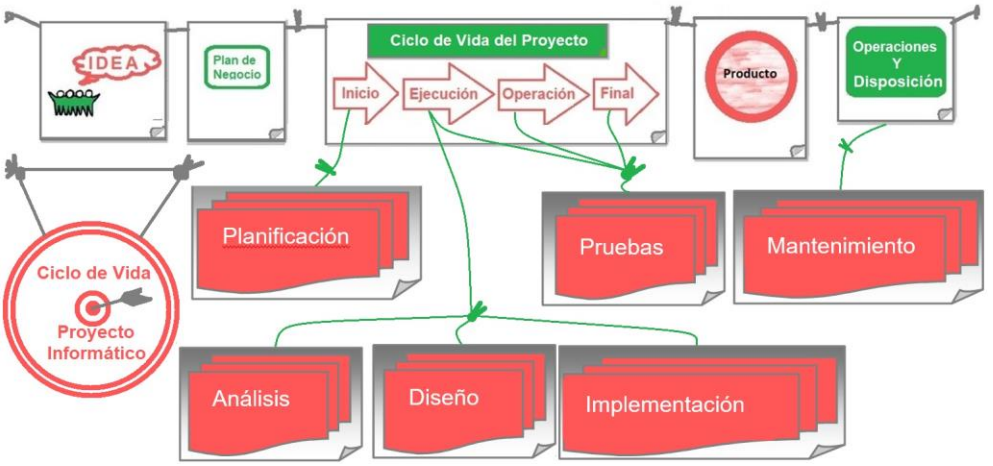


Ilustración 01.05 | Ciclo de Vida de un Proyecto para construir cualquier Producto empresarial vs. Ciclo de Vida de un Proyecto Informático

La secuencia de ejecución, u orden de realización, de las fases de un proyecto informático se pueden llevar a cabo como se muestra seguidamente.

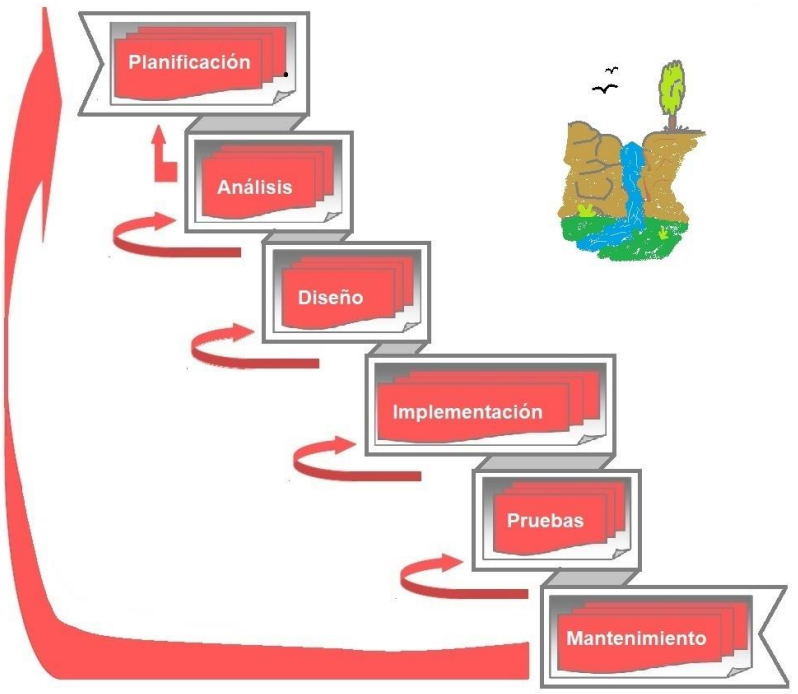


Ilustración 01.06 | Secuencia de ejecución en cascada de las Fases de un Proyecto Informático

A la anterior secuencia de ejecución de fases, se la denomina **Desarrollo de Proyectos Informáticos en Cascada,** también se la llama Ciclo de Vida en Cascada o Waterfall (en inglés)**.**

Existen otros tipos de ejecución de las fases de un proyecto informático como son el Iterativo, el Incremental, en Espiral, en V; y otros.

Conclusión

1 La resolución de un problema tiene unos pasos bien definidos. Un algoritmo no deja de ser la implementación de la solución a un problema informático de software.

2 Los productos informáticos son un compendio de programas informáticos (software) y elementos físicos materiales (hardware).

3 Los programas informáticos utilizan su propio Ciclo de Vida, denominado Proyecto Informático, para desarrollarse como un producto empresarial.

4 Los programas informáticos se desarrollan en unas fases que se pueden realizar en diferentes tipos de secuencias u orden de ejecución; por ejemplo, en cascada o, de manera iterativa e incremental; entre otras.

Fundamentos de Programación

Capítulo 2

Fundamentos de Programación

Aprender a Programar | Ejemplos en PSeInt

Este capítulo resume los fundamentos básicos de los conceptos de la programación informática (desarrollo de programas informáticos).

Ordenador

Un **ordenador** es una máquina electrónica capaz de realizar un tratamiento automático de la información y de resolver con gran rapidez problemas matemáticos y lógicos mediante programas informáticos que ejecutan unos algoritmos.

En España se usa, preferentemente, el término ordenador frente a otras partes del mundo de habla hispana, en las que se utilizan los términos computadora o bien computador. El término ordenador está tomado del francés "ordinateur"; y, los franceses lo denominan así porque un ordenador invierte el 50% de su tiempo de trabajo en ordenar información (datos) entre los diferentes elementos de los que está compuesto. Se entiende que el resto del tiempo de trabajo del ordenador es de cómputo o ejecución de operaciones y cálculos matemáticos. De ahí que en otros países se denomine computadora o computador.

Un ordenador es una máquina compuesta por una Unidad Central de Proceso (CPU) y unos periféricos que nos permiten interactuar con ella. Estos periféricos pueden ser de entrada de datos (teclado, ratón) o de salida (impresora, monitor o pantalla de ordenador); y, de entrada-salida (una pantalla táctil).

La CPU (Unidad Central de Proceso – Central Processing Unit) es la parte del ordenador que realiza todo el procesamiento y control de todos sus componentes.

NOTA: A partir de ahora omitiremos la traducción de las siglas al inglés. Es decir, haremos referencia únicamente a las siglas en su traducción al castellano.

Funcionamiento de una CPU en un ordenador

La siguiente ilustración representa la denominada **Arquitectura de Von Neumann** que es la arquitectura clásica de una única CPU (**Unidad Central de Proceso**) en el ordenador.

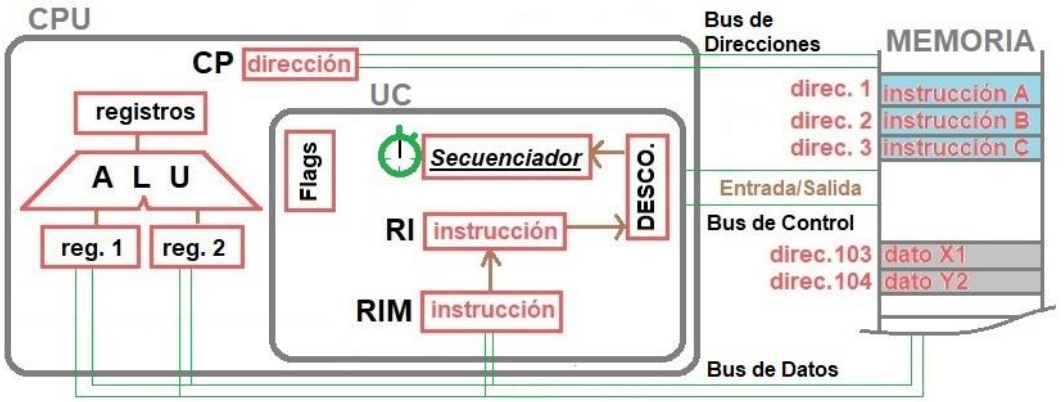


Ilustración 02.01 | Arquitectura Von Neumann de ordenadores

En la parte de la izquierda se encuentra la CPU con sus diferentes componentes. En la parte central se encuentra la UC (**Unidad de Control**). A la derecha esta la **Unidad de Memoria** del ordenador que almacena las instrucciones del programa y los datos (todo almacenado en ceros y unos). Las instrucciones y sus datos se almacenan en sitios diferenciados: el segmento de instrucciones y el segmento de datos.

Vamos a presentar el ciclo de ejecución de una instrucción de nuestro programa.

Supongamos que queremos ejecutar la siguiente instrucción de un programa: **X1 + Y2**. Las instrucciones están en Memoria Principal, que es la memoria RAM (Memoria de Acceso Aleatorio). Para acceder a la siguiente instrucción a ejecutar, que se encuentra en memoria, se accede a la dirección de la memoria que se encuentra alojada en el componente de la CPU: CP (Contador de Programa).

Esta dirección del CP se transmite por el Bus de Direcciones, que es un conjunto de "hilos" que encaminan la conexión, a modo de caminito, entre la CPU y la Memoria. Este camino del bus de direcciones está formado por hilos que transmiten los ceros y unos que contiene la dirección, que es un número de posición en la memoria, y la UC tiene hilos de control (Bus de Control); cuyo cometido es indicar si la operación a realizar en la memoria es una entrada de datos (escritura de la memoria) o salida de contenido de la memoria (lectura de la memoria). Luego, el primer paso es volcar el contenido del Contador de Programa (que es una dirección) en el Bus de Direcciones del ordenador.

La Memoria localiza la dirección dentro de ella y nos devuelve su contenido (el dato de la instrucción) por el otro camino que es el Bus de Datos que conecta la Memoria con la UC. El Bus de Datos deja su contenido en el componente de la Unidad de Control: Registro RIM (Registro Intermedio) que maneja instrucciones. Cuando los datos de la instrucción están completos, es decir, ha llegado todo el dato al registro RIM, el dato se pasa al Registro RI (Registro de Instrucciones).

Pero ese dato de instrucción hay que traducirlo. Ese dato se analiza en el Registro DECO (Registro Decodificador). Una vez analizado en el registro decodificador el contenido de la instrucción, es decir, su operación (acción; que en nuestro caso es una suma) y sus datos (X1 y Y2). Entonces se cargan, desde la memoria, los datos de los valores de la instrucción en los registros de la CPU (Registro 1 y Registro 2). El contenido de estos registros 1 y 2 (que son los datos que utiliza la instrucción decodificada); se pasan al componente más importan de la CPU: ALU (Unidad Aritmético y Lógica). La ALU está compuesta de unos registros de entrada (registro 1 y registro 2) y otros registros de salida con el resultado de la operación que ha realizado la ALU.

Pero cómo se orquesta todo este paso de mensajes de información entre los diferentes componentes de la CPU y la UC. El Secuenciador es el encargado de dirigir y automatizar ese trasiego de información entre todos los componentes de la CPU, la memoria y la UC. El Secuenciador es el que tiene alojado el Reloj del ordenado: el que ordena en qué momento actúa cada componente de la máquina. El Secuenciador es el que tiene la "inteligencia" para orquestar todo el tráfico de ceros y unos (de instrucciones y datos) entre los componentes de la CPU, la Memoria y la UC.

La ALU es la unidad encargada de hacer las operaciones aritméticas y lógicas con los valores almacenados en los registros de entrada (es la "calculadora" del ordenador).

Muy resumidamente, la CPU está formada por la ALU, los registros; y, por la UC (Unidad de Control). La Unidad de Control está formada por el Descodificador, el Secuenciador y sus registros y los "Flags" o dispositivos de alertas (banderas o bits de alerta) de anomalías e incidencias en la descodificación de la instrucción.

El Secuenciador no hace "una cosa tras otra". Es decir, mientras se está componiendo, analizando y ejecutando una instrucción en la ALU, en ese momento, el Secuenciador está preparando con el manejo del Bus de Direcciones, la Memoria, y, con el Bus de Datos y el Registro RIM la extracción de la siguiente instrucción que tocaría ejecutar.

Por lo tanto, todo tratamiento de una instrucción de un programa se divide en dos fases: la fase de búsqueda de la instrucción (fetch, en inglés) y la fase de ejecución de la instrucción.

Algoritmo

Un **algoritmo** es un método para resolver un problema, en este caso computacional (mediante un ordenador), utilizando una secuencia de pasos bien definidos, ordenados y finitos.
Un **programa de ordenador** es una secuencia de órdenes (instrucciones) que describen un algoritmo (conjunto de acciones), escritas de forma que puedan ser entendidas por el ordenador.

Datos

Un programa informático maneja **datos** para obtener un resultado con el que se resuelve un problema computacional.

Todo programa se base en una entrada de datos, sobre los que se realiza un procesamiento de esos datos a través de un programa, y con los que se obtiene una salida o resultado del procesamiento.

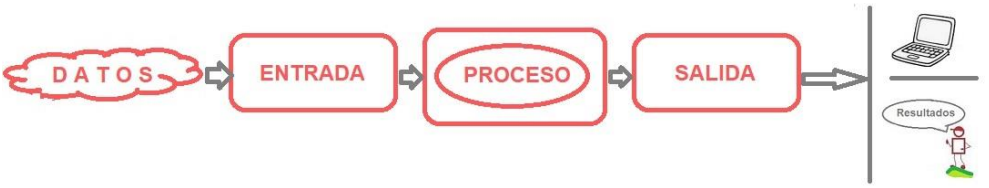


Ilustración 02.02 | Secuencia de obtención de información

Los resultados pueden tener como destinatario a un ser humano o, a otro ordenador.

Este proceder es la forma habitual en la que realizamos nuestras tareas los humanos. Por ejemplo, para tomar un café con leche. Primero obtenemos la cantidad de café molido y de leche que queremos (datos de entrada), los juntamos en una cazuela y los calentamos al gusto (proceso) para, posteriormente, echarlos en una taza para servirlos (salida).

Ahora veamos un caso práctico de procesamiento de datos, de forma computerizada (mediante un ordenador).

Problema

Vamos a calcular la altura en pulgadas de una persona cuya altura nos la comunica en centímetros.

→DATOS:

Necesitamos que nos introduzcan, en el ordenador, la altura en centímetros de la persona.

Necesitamos saber la equivalencia que hay entre la altura en centímetros y pulgadas. Para ello, buscamos en Internet y obtenemos que 1 centímetro de altura equivale a 0,394 pulgadas.

→ENTRADA:

La entrada de datos, se realiza con la instrucción:

Leer centimetrosAltura;

Esta instrucción lee de la pantalla del ordenador el valor introducido por el usuario con el teclado y guarda el valor introducido en el contenido de la **variable** ①

centimetrosAltura

Y solamente necesitamos ese dato de entrada de la persona.

① Una "Variable" es una cajita, que tiene un nombre único en el programa; que se guarda en la memoria del ordenador y tiene como característica que se puede actualizar (modificar), el contenido (valor) que se almacena en la cajita de la memoria.

→PROCESO:

Necesitamos ejecutar la operación aritmética siguiente:

alturaEnPulgadas = centimetrosAltura x 0.394

Pero esta operación no la entiende el ordenador. Porque para el ordenador el símbolo matemático del producto (multiplicación) de dos operandos, no lo escribe con un carácter **x** porque para el ordenador, una **x** es una letra del alfabeto, en vez de un símbolo matemático. Por lo tanto, el ordenador, necesita cambiar el operador aritmético del producto (multiplicación) por otro símbolo diferente de la **x**

Ese símbolo de multiplicación entre dos operandos es, para el ordenador, el símbolo del asterisco *****

Por lo tanto, la operación de multiplicación, para que sea entendida por un ordenador, se debe escribir:

alturaEnPulgadas = centimetrosAltura * 0.394

Esta instrucción guarda en el contenido de la variable alturaEnPulgadas el resultado de hacer la operación aritmética de multiplicación del valor contenido en la variable centrimetrosAltura multiplicado por el valor 0,394

→SALIDA:

La salida de datos se realiza con la instrucción:

Escribir alturaEnPulgadas;

Esta instrucción escribe en la pantalla del ordenador el valor contenido en la variable alturaEnPulgadas

Y solamente necesitamos ese dato de salida de la persona.

Si escribimos el programa informático (algoritmo) completo, que recoge la entrada de datos, los procesa y realiza la salida del proceso; sería el siguiente:

```
Algoritmo deCentimetrosAPulgadas

    //DATOS:
      //se define una variable para contener números enteros
      //y sirve para almacenar un valor de la altura en centímetros.
      Definir centimetrosAltura Como Entero; //Valor de Entrada

      //se define una variable que pueda contener decimales
      //y sirve para almacenar un valor de la altura en pulgadas.
      Definir alturaEnPulgadas Como Real; //Valor de Salida

    //ENTRADA:
      //se escribe en pantalla la pregunta.
      Escribir "Teclee los centímetros de altura: ";
      Leer centimetrosAltura; //se almacena la altura en centímetros.

    //PROCESO:
      //se calcula los centímetros de la altura tecleados, en pulgadas.
      alturaEnPulgadas = centimetrosAltura * 0.394;
      //En los decimales se usa un carácter punto y no una coma.

    //SALIDA:
      //se escribe en pantalla el mensaje del significado del resultado.
      Escribir "La altura en pulgadas es: ";
      //se escribe en pantalla el resultado.
      Escribir alturaEnPulgadas; //se escribe el valor de la variable.
FinAlgoritmo
```

Las nuevas cosas que le pueden sorprender son la aparición de los dos símbolos de la barra inclinada **//**

Las dos barras inclinadas se llaman ***comentarios*** y son mensajes que utiliza el programador (el autor del programa) para documentar el programa de forma que, pasado el tiempo, o si el programa lo lee otro programador; le resulte más fácil saber lo que hace el algoritmo (programa), tanto al autor del programa como a otro compañero.

Otra cosa que le debe llamar la atención es que cada instrucción se finaliza con un carácter de punto y coma **;**
Este carácter de punto y coma sirve para que el ordenador entienda que la instrucción se ha terminado de escribir. Por lo tanto, es un carácter de finalización de sentencia (instrucción) que es como se denomina formalmente.

Operaciones con datos

Con los datos se pueden realizar una serie de operaciones, durante la fase de procesado, es decir, durante la ejecución del algoritmo (programa).

Operaciones aritméticas

Como puede comprobarse son los mismos operadores utilizados normalmente en el álgebra de las matemáticas.

Operador	Significado	Ejemplo	Resultado
+	Suma	5 + 3	8
-	Resta	5 – 3 3 - 5	2 -2
*	Multiplicación	5 * 3	15
/	División	5 / 3	1.6666666667
^	Potenciación	5 ^ 3	125
% MOD	Módulo	5 MOD 3	2
-variable	Invierte el signo de la variable	En la asignación: A = 2; A = -A;	El valor de la variable A pasa a ser -2

Le recuerdo que el **operador Potenciación**: 5 ^ 3 obtiene como resultado 125. Que es el resultado de hacer la multiplicación: 5 * 5 * 5. Es decir, el primer operando, que es el 5, se multiplica 3 veces, por sí mismo. Por lo tanto, es un 5 elevado a 3 que, en matemáticas, se escribe: 5^3

Otro operador que le puede confundir es el **operador de Módulo**: **%** ó **MOD**. Nosotros utilizaremos en esta obra el operador Módulo con el símbolo: MOD; por ser más fácil de recordar su significado. Este operador consiste en obtener el resto de la división entera.

En el ejemplo expuesto:

```
5  | 3
     1
2
```

Es decir, el dividendo 5, dividido por el divisor 3, da como cociente 1 y como resto el número 2. Pues bien, el módulo de 5 MOD 3 es 2; que es el resto de la división entera.

La **operación de Resta** admite números negativos. Por eso, si realizamos la operación aritmética: **3 – 5** se obtiene un resultado: **-2**
Si realizamos la operación **-3 – 5** se obtiene un resultado: **-8**

Operaciones relacionales

Operador	Significado	Ejemplo	Resultado Lógico
>	Mayor que	5 > 3	VERDADERO
<	Menor que	5 < 3	FALSO
=	Igual que	5 = 3	FALSO
<=	Menor o igual que	5 <= 3	FALSO
>=	Mayor o igual que	5 >= 3	VERDADERO
<>	Distinto que	5 <> 3	VERDADERO

La evaluación de una expresión con operadores relacionales obtiene como resultado un valor lógico o booleano de valor VERDADERO (true, en inglés) o FALSO (false, en inglés).

Operaciones lógicas

Operador	Significado	Ejemplo	Resultado Lógico
& ó Y	Conjunción (y)	(5 > 3) & (2=1)	FALSO
| ó O	Disyunción (o)	(5 <> 3) | (5 <= 3)	VERDADERO
NO	Negación (no)	NO (5 > 3)	FALSO

Veamos cada caso por separado.

→ Operador & ó Y: También llamado **AND**, en inglés. Solamente es VERDADERO cuando ambos operandos de la expresión son verdaderos. Es decir, este operador responde a la Tabla de Verdad siguiente.

Operador1	Operador2	Operación Lógica a Evaluar	Resultado Lógico
VERDADERO	VERDADERO	VERDADERO y VERDADERO	VERDADERO
FALSO	VERDADERO	FALSO y VERDADERO	FALSO
VERDADERO	FALSO	VERDADERO y FALSO	FALSO
FALSO	FALSO	FALSO y FALSO	FALSO

Veamos el ejemplo de partida: (5 > 3) Y (2 = 1)
|
Resultado evaluación: VERDADERO Y FALSO
|
Resultado de la Tabla de Verdad: FALSO

→ Operador | ó O: También llamado **OR**, en inglés. Solamente es FALSO cuando ambos operandos de la expresión son falsos. Es decir, este operador responde a la Tabla de Verdad siguiente.

Operador1	Operador2	Operación Lógica a Evaluar	Resultado Lógico
VERDADERO	VERDADERO	VERDADERO o VERDADERO	VERDADERO
FALSO	VERDADERO	FALSO o VERDADERO	VERDADERO
VERDADERO	FALSO	VERDADERO o FALSO	VERDADERO
FALSO	FALSO	FALSO o FALSO	FALSO

Veamos el ejemplo de partida: (5 <> 3) O (5 <= 3)

Resultado evaluación: VERDADERO O FALSO

Resultado de la Tabla de Verdad: VERDADERO

→ Operador NO: También llamado **NOT**, en inglés. Solamente es VERDADERO cuando el operador al que se le aplica la expresión es falso. Es decir, este operador responde a la Tabla de Verdad siguiente.

Operador	Operación Lógica a Evaluar	Resultado Lógico
VERDADERO	NO(VERDADERO)	FALSO
FALSO	NO(FALSO)	VERDADERO

Veamos el ejemplo de partida: NO (5 > 3)

Resultado evaluación de la expresión: NO VERDADERO

Resultado de la Tabla de Verdad: FALSO

Cuidado con el operador lógico O, y el uso de la conjunción "o" en español.

Hay que aclarar algo sobre los operadores lógicos y el idioma castellano. En el caso del operador Y, este operador "funciona" como lo utilizamos habitualmente en el idioma español. Es decir, solamente es verdadero (cierto) lo que nos dicen, si el operador1 es cierto Y el operador2 es cierto. Por ejemplo, la frase: "hoy es lunes y empieza la semana", es cierta solamente cuando el operador1: "hoy es lunes", sea cierta. Y, cuando lógicamente "empieza la semana". Esta frase es falsa en los países anglosajones, en los que la semana "empieza" el domingo. Luego, la frase "hoy es lunes y empieza la semana", es cierta solamente cuando hablamos de países de habla hispana; porque se cumple (son ciertos) los operandos 1 y 2.

Vamos ahora con el operador O. Este operador **no funciona** como la lógica habitual que lo utilizamos en el idioma castellano. En español, la conjunción "o" la usamos como: una cosa u otra (pero **no** ambas cosas a la vez). Pero el operador O funciona en la Tabla de Verdad como "una cosa u otra". Y también funciona como "una cosa y otra". Es decir, no es "una cosa u otra". Es a la vez "una cosa u otra" y "una cosa y otra"; cuando se vuelve cierta la expresión. Fíjese en la Tabla de Verdad, es cierta cuando es "una cosa u otra" y cuando es cierta "una cosa y otra". Es decir, solamente es FALSO, si ambos operandos son falsos a la vez.

Dicho lo anterior, la frase "hoy es lunes O empieza la semana" (siendo O un operador lógico) sería cierta cuando sea lunes en los países hispanos y, sería cierta también en los países anglosajones, cuando estuviéramos en domingo. Porque sería cierto el operando2: "empieza la semana" independientemente de si es "lunes" o "domingo". Luego, la frase de la expresión lógica "hoy es lunes O empieza la semana" se volvería VERDADERA cuando fuera "lunes" o "domingo", independientemente de si estamos en un país hispano o bien anglosajón.

Orden de evaluación de las operaciones

Dentro de una expresión no todas las operaciones son resueltas (evaluadas) a la vez. Por ejemplo, todos sabemos desde pequeños que una multiplicación tiene más prioridad que una suma.

El orden de prioridad de los operadores, para la evaluación de cualquier tipo de expresión, es el siguiente.

Nivel de Prioridad (el 1 es el más prioritario: se evalúa antes)	Operador	Significado
1	( )	Paréntesis, comenzando la evaluación por los más internos hacia los más externos
2	^	Potenciación
3	* / MOD	Multiplicación División Módulo: Resto de la división entera
4	+ -	Suma Resta
5	>, <, =, <=, >=, <>	Operador Relacional
6	NO	Negación(no)
7	Y	Conjunción(y)
8	O	Disyunción(o)

Ante una igualdad en la prioridad, la expresión se evaluará de izquierda a derecha. Es decir, en el sentido de la lectura de un texto escrito en castellano.

Ejemplo: Se desea realizar la evaluación de la siguiente expresión aritmética. Aplicar el orden de evaluación de las operaciones (su prioridad).

(3 * 3) + (5 MOD 2) * 2 - 1 - 1

Paso 1: (3 * 3) → Resultado1: 9
Paso 2: (5 MOD 2) → Resultado2: 1
Paso 3: (5 MOD 2) * 2 → Resultado2 anterior: 1 * 2 → Resultado3: 2
Paso 4: (3 * 3) + (5 MOD 2) * 2 → Resultado1 + Resultado3: 9 + 2 →
→ Resultado4: 11
Paso 5: (3 * 3) + (5 MOD 2) * 2 - 1 → Resultado4 anterior: 11 – 1 →
→ Resultado5: 10
Paso 6: (3 * 3) + (5 MOD 2) * 2 - 1 - 1 → Resultado5 anterior: 10 – 1→
→ ResultadoFINAL: 9

Variables y Constantes

Los datos de un programa de ordenador se pueden guardar (almacenar) en variables o bien en constantes, para poderlos procesar.

Variables

Una variable, como su propio nombre indica, puede cambiar (variar) su valor durante la ejecución de un programa (algoritmo).
Es decir, se la puede asignar un valor con el símbolo de asignación **=**
Y se la puede dar un valor tantas veces como se quiera, a lo largo de la ejecución del programa.

Por ejemplo, podemos asignar un valor Cadena (conjunto de caracteres) a una variable de nombre:

mesActual

Pero su valor dependerá del tipo del dato que almacene. Es decir, si declaramos (definimos) una variable como de tipo Cadena. Podemos hacer la siguiente asignación de un valor:

mesActual = "Diciembre"

Pero, si definimos (declaramos) una variable como número Entero, solamente puede tener un valor numérico. Y deberíamos asignarle el valor numérico 12 como se muestra seguidamente.

mesActual = 12

Para definir una variable como Cadena, debemos usar una forma especial que entienda el ordenador. Para ello, usamos la palabra reservada Definir; seguida por el nombre de la variable y, a continuación, el tipo de dato que va a contener la variable.

Definir mesActual Como Cadena

Esta **Declaración de Tipo**, que es como se denomina formalmente, reserva un espacio de Memoria en el ordenador: una "cajita" de tipo Cadena para un valor.

El resultado de la instrucción (sentencia) Definir en Memoria es el siguiente:

mesActual → 

El resultado de la instrucción de asignación con el símbolo **=** es el siguiente:
mesActual = "Diciembre"

Obtenemos en memoria:

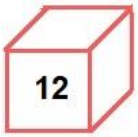
mesActual → 

Si lo ponemos en su orden correcto y en su sintaxis correcta (su forma de escribirlo para que lo entienda un ordenador) es:

```
Definir mesActual Como Cadena;
mesActual = "Diciembre";
```

Por el contrario, si realizamos una definición (declaración de tipo) de tipo Entero, la secuencia de instrucciones y el resultado en memoria será:

```
Definir contadorMesActual Como Entero;
contadorMesActual = 12;
```

MEMORIA: contadorMesActual → 

Constantes

Una constante es también una "cajita" en memoria; pero, una vez que le asignamos un valor no se puede cambiar su contenido (el valor que tiene); durante toda la ejecución del programa (algoritmo). Es decir, es un valor fijo e inmutable (no variable).

Ejemplo: por convencionalismo entre los programadores (un acuerdo entre los programadores), las constantes se nombran en letras mayúsculas. Ejemplo:

```
Definir DIASSEMANA Como Entero;
DIASSEMANA = 7;
```

Sentencia de asignación

Hemos visto la sentencia (instrucción) de asignación con el símbolo **=** en el extracto de código siguiente.

```
Definir mesActual Como Cadena;
mesActual = "Diciembre";
```

Y hemos visto que este código tiene como consecuencia o resultado, la creación de una "cajita" en Memoria y la asignación de un valor, como se muestra seguidamente.

mesActual → 

Pero también hemos anunciado que una variable, cambia de valor a lo largo del código del programa (conjunto secuencial de sentencias).

Luego, ¿cómo se cambia el valor (contenido) de una variable?; y, ¿cómo quedaría este resultado en la memoria del ordenador?.

```
mesActual = "Enero";
```

MEMORIA: mesActual → 

Pues bien, una sentencia de asignación consigue asignar a la variable de nombre de la izquierda del símbolo **=** el resultado de la **evaluación de la expresión** del lado de la derecha del signo **=**

Esto es lo que hemos hecho al asignar un valor "Diciembre" y luego un valor "Enero". La expresión de la derecha del signo **=** era un valor concreto que no necesitaba evaluación ya que era, en sí mismo, un valor original (sin necesidad de evaluar una expresión).

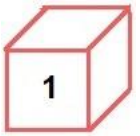
Pero, ¿cómo es una expresión a evaluar?. Una expresión puede ser de tipo aritmético: da como valor evaluado un valor de Tipo Numérico; o bien una expresión de Tipo Lógica: da como valor evaluado el valor VERDADERO (true) o FALSO (false).

Veamos un ejemplo de una expresión a evaluar de Tipo Aritmética.

```
Definir contadorMesActual Como Entero;
contadorMesActual = 1;
```

El resultado en memoria del ordenador de la sentencia de asignación es la evaluación del valor 1.

contadorMesActual →

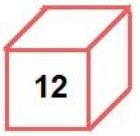


Si ahora, realizamos en el programa una sentencia de asignación que necesita una evaluación de una expresión aritmética, obtendremos un valor de tipo numérico.

```
contadorMesActual = 1 + 11;
```

Al evaluar la expresión, obtenemos el valor 12. El resultado en memoria es:

contadorMesActual →



Pero, esto ya lo hace una calculadora, es decir, no necesitamos escribir un programa de ordenador para sumar 1 + 11

Entonces, cuál es realmente la potencia de un ordenador. La ventaja de un ordenador es que podemos ordenarle, la misma sentencia de asignación anterior, pero de forma computerizada (para que la resuelva informáticamente).

```
contadorMesActual = contadorMesActual + 11;
```

Esta instrucción, ¿cómo la va a ejecutar el ordenador?.

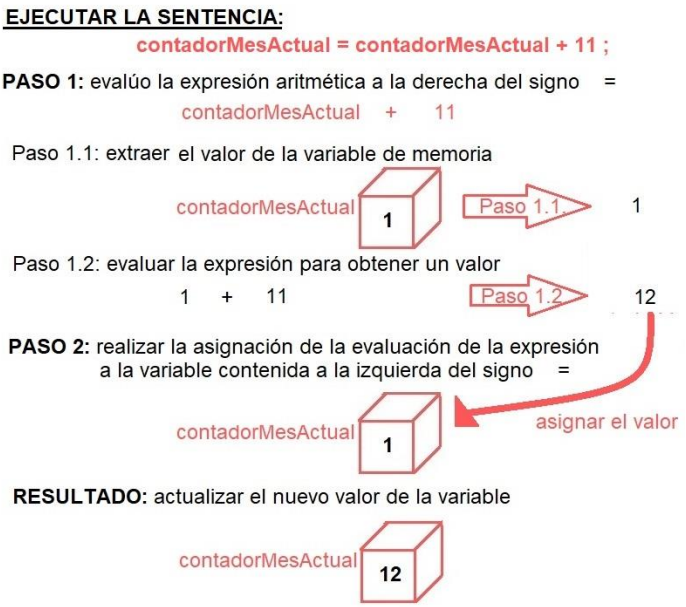


Ilustración 02.03 | Ejecución en memoria de una sentencia de asignación

Dejo al lector la evaluación del siguiente código:

```
Definir contadorMesActual Como Entero;
Definir incremento Como Entero;
contadorMesActual = 1;
incremento = 11;
contadorMesActual = contadorMesActual + incremento;
```

¿Qué valor obtendríamos en la variable contadorMesActual ?.

PISTA: contadorMesActual = 1 + 11

Conclusión

1 Los Programadores resuelven los problemas informáticos mediante la confección de Algoritmos utilizando un lenguaje de programación.

2 Todo algoritmo (programa) se basa en unos Datos de entrada que mediante su Proceso se obtiene un Resultado a modo de salida del mismo. Esta salida puede ser la entrada para otro algoritmo en un ordenador y/o bien para un ser humano.

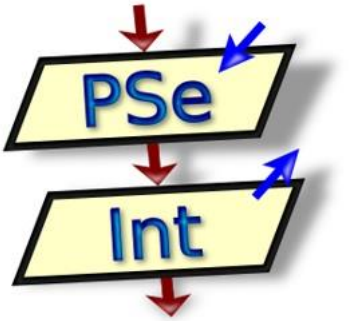
3 Los operadores en expresiones a evaluar, las variables y las sentencias de asignación son los pilares básicos de todo programa de una computadora.


3

Capítulo 3

PSeInt

Aprender a Programar | Ejemplos en PSeInt



Este capítulo ofrece la oportunidad de conocer los conceptos básicos de programación (desarrollo de programas informáticos) por medio de la utilización del programa **PSeInt**.

PSeInt es la abreviatura de "Pseudocode Interpreter". Por lo tanto, como su propio nombre indica, PseInt se trata de un Entorno de Desarrollo Integrado (IDE) para escribir programas de ordenador, que se basa en un intérprete de instrucciones, en este caso de instrucciones (órdenes al ordenador) que están escritas en "pseudocódigo".

El **pseudocódigo** es un lenguaje de descripción algorítmico (programa de ordenador) que utiliza una descripción de alto nivel, similar al lenguaje natural de las personas, para escribir las órdenes (instrucciones) que es capaz de entender un ordenador.

Es decir, un programa de ordenador escrito en pseudocódigo utiliza las convenciones estructurales, en forma de instrucciones, de un lenguaje de programación real; pero está diseñado para la lectura humana en lugar de la lectura mediante una máquina (ordenador); y, con independencia de cualquier otro lenguaje de programación.

El principal objetivo del pseudocódigo es el de representar la solución a un algoritmo (conjunto de instrucciones que ejecuta un ordenador) de la forma más detallada posible, y a su vez lo más parecido posible al lenguaje que posteriormente se utilizará para la codificación (escritura del programa con un lenguaje de programación real) del mismo.

Por lo tanto, PSeInt utiliza instrucciones de programación, que forman un algoritmo para el ordenador, en forma de pseudocódigo. Estas instrucciones escritas en un lenguaje accesible para los humanos se pueden usar alternativamente con **Diagramas de Flujo** que son el equivalente al programa escrito en pseudocódigo, pero con una representación gráfica.

En esta obra, nos centraremos en el pseudocódigo escrito, ya que, el pseudocódigo es más legible y cómodo de escribir frente a los ordinogramas o diagramas de flujo.

Pero, si PSeInt es un intérprete… ¿qué es un Intérprete de Instrucciones?. Un intérprete es diferente a un "compilador de Instrucciones".

Compilador

El compilador es el programa informático que traduce el código fuente (algoritmo) a un lenguaje intermedio llamado código objeto que por medio del programa enlazador (linker) recoge e integra las funciones (subrutinas) de las librerías necesarias para ejecutar el código máquina (lenguaje en ceros y unos que entiende el ordenador).

Una vez que un programa ha sido compilado, puede ser ejecutado repetidamente en el ordenador sin necesidad de más traducción.

Ilustración 03.01 | Compilador

Por utilizar un símil de la vida real, podríamos decir que cuando se le encarga a un Traductor Profesional el encargo de traducir un libro del inglés al castellano. El Código Fuente es el libro en lengua inglesa, el Compilador es la persona que hace la traducción del inglés al español. Que genera como resultado el Código Objeto, que es el texto resultante de su traducción a la lengua española. Y por medio de un programa de edición de libros, que en este caso hace las funciones del proceso de Enlazador; se termina visualizando en el ordenador por medio de un editor de documentos en formato PDF, por ejemplo.

Intérprete

Un Intérprete es el programa informático encargado de traducir los programas fuentes (algoritmos) a lenguaje máquina (lenguaje en ceros y unos que entiende el ordenador) y comprobar que las llamadas a funciones de las librerías se realizan de forma correcta.

Ilustración 03.02 | Intérprete

Siguiendo con el símil del encargo a un Traductor Profesional de la traducción de un libro de lengua inglesa al castellano. El Código Fuente es el libro en lengua inglesa que se debe traducir. Este libro lo cargamos en el "***Intérprete***" de idiomas que tiene en la web de Internet la empresa Google: Traductor de Google; y, la página web de Google nos muestra inmediatamente la traducción del inglés al español en la pantalla del ordenador en Internet.

Una de las diferencias entre el compilador y el intérprete es que el compilador obtiene un producto intermedio, llamado Código Objeto, que hay que transformar en sentencias ejecutables por el ordenador (sentencias en el lenguaje de ceros y unos). Mientras que el intérprete ejecuta (realiza) directamente, desde el Código Fuente, las sentencias que necesita el ordenador, sin necesidad de obtener un objeto intermedio o resultado parcial como realiza el compilador.

Las principales **características de PSeInt** son:

→ Permite escribir programas (algoritmos) en **castellano**. Es decir, en PSeInt se programa en pseudocódigo.

→ Es **multiplataforma.** Es decir, permite crear y ejecutar programas en los sistemas operativos más populares: Microsoft Windows, GNU/Linux, macOS.

→ Es un software **Libre** (cualquiera lo puede utilizar y modificar ya que se dispone del código fuente de la aplicación); y, es **gratis**.

Interfaz de usuario

Pantalla de trabajo de PSeInt en modo edición de pseudocódigo (código fuente): Comenzamos mostrando el Interfaz general de la pantalla de inicio de PSeInt, una vez que se ha escrito un algoritmo y se ha ejecutado dentro del programa, al pulsar sobre el botón del Triángulo Verde (Ejecutar) en la "Barra de Acceso Directo".

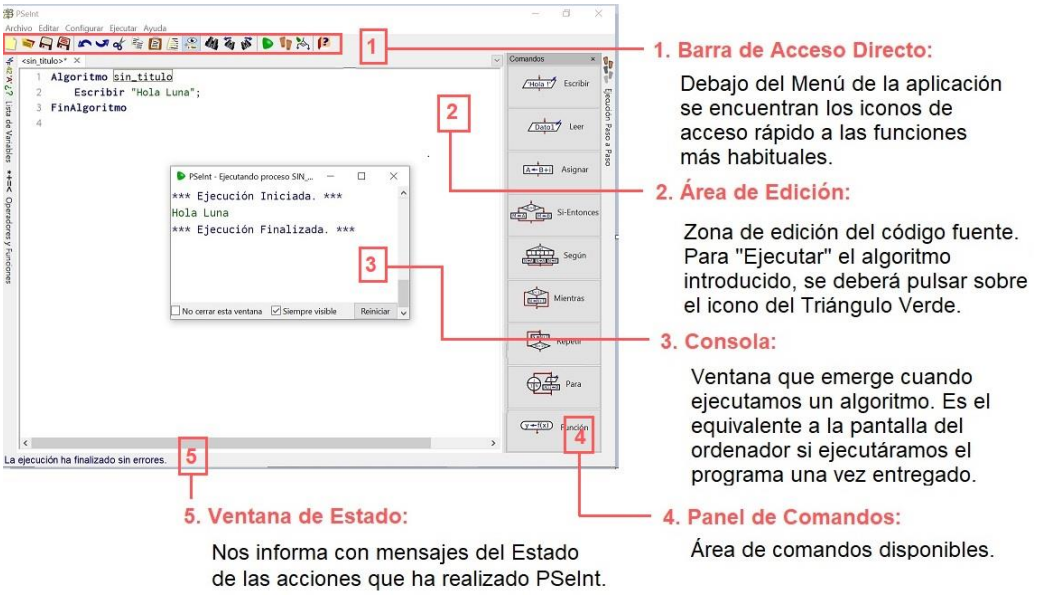


Ilustración 03.03 | Pantalla de Trabajo de PSeInt

Como abreviaturas de teclado más habituales, el usuario dispone de los comandos de acción para la edición del código fuente siguientes:

TECLAS	SIGNIFICADO
Ctrl + D	Para comentar las líneas de código: añadir los símbolos **//** al principio de la línea
Ctrl + Shift + D	Para eliminar las líneas de **comentarios** que se han escrito (descomentar) : eliminar los símbolos **//** al principio de la línea
Escape	Para abandonar una sugerencia de sintaxis, por parte de PSeInt, a la hora de editar el código

Los dos primeros comandos se encuentran también en el Menú **Editar**.

Barra de Acceso Directo

Visualizamos cada uno de los iconos del interfaz de usuario de PSeInt y exponemos su significado en una tabla.

Archivo

Nuevo	El **Post-it** sirve para abrir un nuevo fichero en el que crear un algoritmo.	**Abrir**	La **Carpeta Abierta** sirve para buscar un fichero existente en el ordenador para editarlo al abrirlo.
Guardar	El **Disquete con el Texto en Gris** sirve para guardar los cambios, en un fichero del ordenador, con un algoritmo ya creado en el disco del ordenador.	**Guardar como**	El **Disquete con el Texto en Rojo** sirve para guardar el algoritmo en un nuevo fichero en el disco del ordenador, con el nombre de fichero que le asignemos.

Edición

Deshacer	La **Flecha Azul hacia Atrás** sirve para deshacer la última acción de edición que hayamos realizado sobre el algoritmo.	**Rehacer**	La **Flecha Azul hacia Delante** sirve para deshacer la última acción de deshacer en la edición. Es decir, vuelve a rehacer lo deshecho.
Cortar	La **Tijera** sirve para almacenar, en memoria del ordenador, el texto previamente seleccionado del editor; y se elimina del mismo. Comando de teclado **Ctrl+X**	**Copiar**	Los **Folios Duplicados** sirven para almacenar, en memoria del ordenador, el texto previamente seleccionado del editor; y se saca una copia del texto seleccionado. Comando de teclado **Ctrl+C**

Edición

Pegar

El **Folio sobre el Portapapeles** sirve para pegar, el contenido previamente almacenado en la memoria del ordenador, en la posición actual del cursor del ratón en el editor.
Comando de teclado **Ctrl+V**

Mostrar Errores

El **Ojo** sirve para mostrar los mensajes de error del código fuente (algoritmo) de los que nos advierte PSeInt

Corregir Indentación

El **Folio con Correcciones en Rojo** sirve para mover las líneas a la derecha, al igual que se si introdujeran espacios en blanco por la izquierda de la línea, con la tecla de la Barra Espaciadora; o se pulsase sobre la tecla de Tabulador.

Búsqueda

Buscar

Los **Prismáticos** sirven para buscar el texto, que se introduzca posteriormente a la posición del cursor, en el algoritmo en edición.

Buscar anterior

Los **Prismáticos con Flecha hacia Atrás** sirven para buscar el texto introducido en la posición anterior del texto editado del algoritmo.

Buscar siguiente

Los **Prismáticos con Flecha hacia Delante** sirven para buscar el texto introducido en la posición siguiente del texto editado del algoritmo.

Ejecución

Ejecutar

El **Triángulo Verde (Play)** sirve para ejecutar el algoritmo que se está editando en pantalla.

Ejecutar Paso a Paso

Los **Dos Pies** sirven para ejecutar instrucción a instrucción, una detrás de otra, del algoritmo en edición. Es lo que se llama el Depurador de PSeInt.

Dibujar Diagrama de Flujo

El **Rombo con Flechas** sirve para dibujar el algoritmo en edición, en forma de gráfico de Diagrama de Flujo.

Ayuda

Ayuda

El **Signo de Interrogación** sirve para visualizar la Ayuda del programa PSeInt.

Tipos de Datos

Los programas informáticos utilizan datos. En PSeInt los tipos de datos que se manejan son:

Entero: cualquier valor numérico sin decimales (Ej.: 18. Ej.: -15)

Real: permite contener un número, tanto un valor entero (Ej.: 2023) como un número decimal (Ej: 3.1416). La parte decimal se separa con un carácter punto: .

Cadena: secuencia de caracteres alfanuméricos (letras, símbolos y números). Las cadenas se escriben delimitadas por el carácter Comilla Doble (el símbolo "). (Ej.: "España". Ej.: ":)". Ej.: "Madrid 360". Ej.: "Calle Alcalá, 45". Ej.: "18". Ej.: "3.1416". Ej.: "-54"). Cuando es un único carácter se puede usar también el tipo de dato: **Caracter**. (Ej.: "A". Ej.: "1". Ej.: "p". Ej.: "/". Ej.: "-")

Logico: es un tipo de dato que puede tomar el valor VERDADERO o FALSO. También se les llaman: Valores Booleanos (en inglés: true o false).

Variables y Constantes

Dependiendo de su uso, los datos se pueden almacenar en variables o bien en constantes.

Aunque PSeInt no es "case sensitive". Es decir, no distingue entre un texto escrito en mayúsculas o minúsculas: Conviene acordar ciertas reglas de escritura del código fuente (programa). Por ejemplo, una constante se recomienda que se escriba siempre en mayúsculas.

Ejemplo:

PI = 3.1416;

Un **nombre de variable** puede escribirse con caracteres alfanuméricos de tipo letras (de la "**a**" a la "**z**" o bien de la "**A**" a la "**Z**") y con caracteres de tipo numérico (del "**0**" al "**9**") y con el símbolo del guion de subrayado (Ej. si_no).

Como nombre de una variable se pueden utilizar cualquiera de los caracteres de los expuestos, pero el nombre debe empezar con un carácter de tipo letra (de la "a" a la "z" o bien de la "A" a la "Z"), necesariamente o dará un error en el editor de PSeInt.

Ejemplo:

mes12 = "Diciembre";

Como regla de escritura o nombrado (dar nombre) de una variable, se recomienda que empiece por una letra en minúscula, siguiendo por el resto de las palabras empezando en una letra con mayúscula.

Ejemplo:

diaSemana = "Lunes";

No utilizar artículos. No usar: diaDeLaSemana

En los nombres de las variables y de las constantes no se puede utilizar una vocal con tilde, ni la letra "ñ". Como caracteres de símbolos solamente se pueden utilizar el guion de subrayado (Ejemplo: equipo_01).

Ejemplo: para nombrar una variable **año**, se recomienda usar el identificador: **anio**

Una **constante** mantiene su valor asignado durante todo el programa. Una variable va cambiando de valor (de ahí su nombre "variable").

Sentencias

Según su función, las sentencias pueden ser del siguiente tipo de instrucciones.

Instrucciones de Entrada

Recogen un dato del Teclado del ordenador y lo asignan a la variable de la instrucción.

Ejemplo:

Leer nombre;

No olvidar terminar la instrucción con un ;

Se pueden incluir tantas variables de entrada de datos como se quiera, entonces los valores se irán asignando a los valores según el orden en el que fueron introducidos por Teclado. Los valores, se irán introduciendo en las variables, según la aparición de un espacio en blanco de separación entre los datos.

Ejemplo:

Leer numero1, numero2, numero3;

Instrucciones de Asignación

Recogen el dato de la derecha y lo asignan (introducen) en la variable de la izquierda.

Ejemplo: con un dato simple de tipo Entero.

edad = 18;

Ejemplo: con un dato simple de tipo Cadena.

miCiudad = "Madrid";

También pueden recoger una expresión de la derecha, la evalúan para obtener un dato y, asignan el dato obtenido en la evaluación a la variable de la izquierda.

Ejemplo: con una expresión que es una sola variable.

```
altura = centimetrosMedidos;
```

Ejemplo: con una expresión que es una operación entre variables.

```
areaRectangulo = base * altura;
```

Ejemplo: con una expresión. Suponemos un valor del Impuesto del Valor Añadido (IVA) del 21%.

```
precioDeVentaTotal = precioDeVenta + (precioDeVenta * 0.21);
```

Instrucciones de Salida

Envían un dato de tipo cadena a la Pantalla del ordenador. También sirven para mostrar el resultado de la evaluación de una expresión.

Ejemplo: con un dato simple.

```
Escribir "Hola PSeInt";
```

Ejemplo: con un dato simple y una variable.

```
Escribir "Hola ", nombre;
```

Lógicamente, la variable ***nombre*** debe de tener asignado previamente un valor en la variable.

Ejemplo: con un dato simple, una expresión y un dato simple.

```
Escribir "Tu altura es ", centimetrosDeAltura * 0.394, " en pulgadas";
```

Hay una instrucción especial que sirve para documentar el código fuente y que no se tiene en cuenta por el programa PSeInt al ejecutar el algoritmo. Esta instrucción especial es la de los **Comentarios**.

Los **Comentarios** son unos símbolos que se introducen al principio de la línea del código fuente y que consiguen que el intérprete de ejecución del código de PSeInt, ignore esas instrucciones. Los símbolos utilizados en los comentarios son dos caracteres seguidos de la Barra Inclinada **//**

Ejemplo:

```
Algoritmo Bienvenida
    //Autor: José Luis Rodríguez Muñoz
    Escribir "Hola PSeInt";
    Escribir "Buenos días";   // Si no, Escribir "Buenas tardes";
FinAlgoritmo
```

El intérprete del programa PSeInt al detectar los caracteres **//** ignora el resto del contenido de la línea del editor.

Como puede apreciarse, los símbolos de los **comentarios** pueden aparecer al principio de la línea o bien en medio de una línea. Lo que quieren indicar, al intérprete de PSeInt, es que, a partir de la aparición en una línea de estos símbolos, el intérprete ignorará el resto de lo que aparezca en la línea (sin importar lo que contenga: instrucciones, símbolos y/o texto).

Estructuras de Control

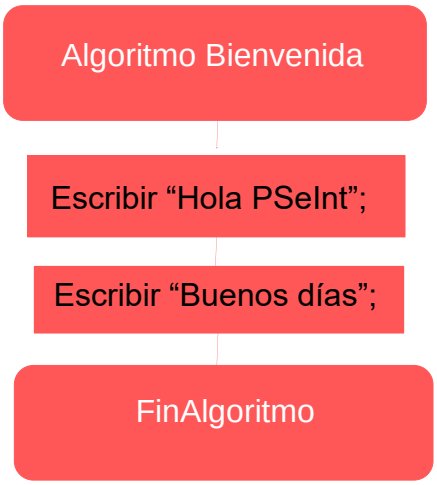
La **Programación Estructura** postula que todo algoritmo (programa) solamente necesita hacer uso de tres tipos de *estructuras de control* para resolver un problema computacional mediante el uso de un algoritmo: Secuencial, Selectiva (condicional), y/o Repetitiva (Iterativa).

Pasamos a presentar cada una de estas estructuras de control.

Estructura Secuencial

La estructura secuencial ejecuta las órdenes (instrucciones) de forma consecutiva (una detrás de otra) en el orden en las que están escritas. En un momento dado de la ejecución de un programa (algoritmo) siempre existirá una instrucción anterior (ya ejecutada), una actual (que se está ejecutando), y otra posterior que se ejecutará seguidamente.

Veamos gráficamente en un ordinograma o **Diagrama de Flujo**, cuál es su aspecto visual.



Estructura Selectiva o Condicional

La estructura selectiva o condicional se utiliza para tomar una decisión sobre qué instrucción es la siguiente a ejecutar.

Condicional Simple (sentencia SI-ENTONCES)

En esta estructura condicional simple, primero se evalúa la condición, y si se cumple: la evaluación de la expresión de la condición de la decisión da como resultado el valor lógico verdadero (true); es decir, la expresión se evalúa como correcta; entonces se ejecuta la instrucción o instrucciones que contiene.

Ejemplo:

```
Algoritmo Bienvenida
    Definir hoy Como Cadena; //se define una variable
                                //para el día de la semana.
    Escribir “Hoy es: ”;  //se escribe en pantalla la pregunta.
    Leer hoy; //se solicita que se introduzca un día por pantalla.
    Si hoy = “lunes”
      Entonces //si al evaluar la expresión es VERDADERA:
                  //la variable “hoy” ‘es igual’ a “lunes”
                  //Es decir, tiene el valor correcto: “lunes”
                  // entonces, la expresión es cierta y
                  //se ejecuta la sentencia de salida que contiene.
               Escribir “Buenos días”; //se saluda.
    Finsi //en caso de no ser el valor “lunes”, NO se hace nada
    //porque termina el algoritmo (proceso).
FinAlgoritmo
```

ojo

Hay que resaltar que la condición solamente evalúa por un valor correcto, que es cuando la condición se vuelve cierta (verdadera). Es decir, el algoritmo solamente "*saluda*" si se introduce, únicamente, el valor de la cadena **lunes**
Es decir, exactamente como está: sin mayúsculas ni espacios en blanco.

Condicional Compuesta (sentencia SI-ENTONCES-SINO)

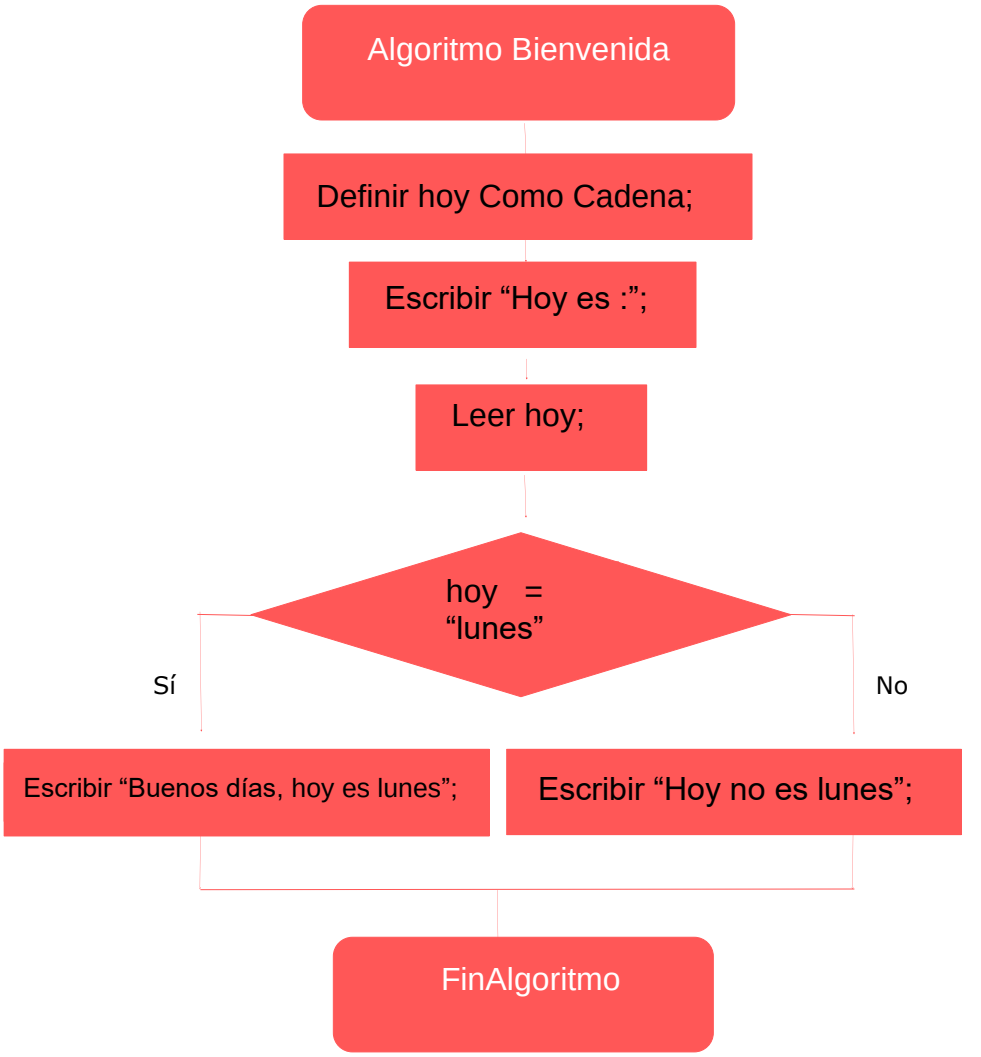
La estructura condicional compuesta es como la condicional simple, pero se añade una condición de "SI NO" a la evaluación de la expresión de la decisión.

Ejemplo:

```
Algoritmo Bienvenida
    Definir hoy Como Cadena; //se define una variable
                                //para el día de la semana.
    Escribir "Hoy es: ";  //se escribe en pantalla la pregunta.
    Leer hoy; //se solicita que se introduzca un día por pantalla.
    Si hoy = "lunes"
      Entonces //si al evaluar la expresión es VERDADERA:
               //la variable "hoy" 'es igual' a "lunes"
               //Es decir, tiene el valor correcto: "lunes"
               //con lo cual, la expresión es cierta
            Escribir "Buenos días, hoy es lunes"; //se saluda.
       SiNo
            Escribir "Hoy no es lunes"; //es otro día u otra cosa
                                          //diferente de:  lunes.
    Finsi //fin de la estructura Si-Entonces-Sino

FinAlgoritmo
```

Hay que destacar que la condición solamente es cierta para el único valor **lunes** o ,si no , se escribe el valor que hace que la condición sea FALSO (false). Es decir, la condición se vuelve falsa, siempre que no sea un valor **lunes**. Pero OJO, puede ser cualquier otro valor (no tiene porqué ser un día de la semana); ya que el usuario de la aplicación puede haber introducido por pantalla cualquier otro valor, por ejemplo, un número entero (porque no hay "nada" en el algoritmo que le obligue a introducir un día de la semana con caracteres de letras del alfabeto español).

Vamos a mostrar el gráfico o **Diagrama de Flujo** de una estructura condicional compuesta.



Condicional Múltiple (sentencia SEGUN)

La estructura condicional múltiple comienza evaluando una expresión de tipo numérica (devuelve como dato evaluado un valor de tipo entero) u otro tipo de dato. Una vez obtenido el dato evaluado en la expresión condicional, se comprueba si coincide ese valor con cada uno de los valores de las opciones disponibles. En el caso de coincidencia del valor de la expresión con un valor de las opciones, se ejecutan las instrucciones que contiene esa opción. Cuando no se encuentra ningún valor coincidente con las opciones disponibles, entonces se ejecutan las instrucciones de la opción del valor que no se ha encontrado entre ninguna de las opciones disponibles, y entonces se ejecutan las sentencias de dentro de la cláusula: De Otro Modo.

Pero, veamos esta estructura condicional múltiple con un ejemplo.

Ejemplo:

```
Algoritmo Bienvenida
    //se define una variable de tipo entero
    //para el número del día de la semana.
    Definir numeroDiaSemana Como Entero;

    //se escribe en pantalla la pregunta del valor numérico.
    Escribir "En qué número de día de la semana estamos: ";
    //se recoge un número de día de la semana por pantalla.
    Leer numeroDiaSemana;

    Segun numeroDiaSemana Hacer
      1: Escribir "Hoy es lunes"; ////se escribe por ser el día 1.
      2: Escribir "Hoy es martes"; //se escribe por ser el día 2
      3: Escribir "Hoy es miércoles"; //se escribe por ser el día 3
      4: Escribir "Hoy es jueves"; ////se escribe por ser el día 4
      5: Escribir "Hoy es viernes"; ////se escribe por ser el día 5
            De Otro Modo: //es otro día u otra cosa diferente
                  Escribir "Hoy es fin de semana";
    FinSegun //fin de la estructura Segun

FinAlgoritmo
```

Los valores opcionales, en una estructura de condición múltiple, se pueden agrupar.

Veamos un ejemplo de agrupación de los valores de las opciones disponibles en una sentencia de estructura condicional múltiple.

Ejemplo:

```
Algoritmo Bienvenida
    //se define una variable de tipo entero
    //para el número del día de la semana.
    Definir numeroDiaSemana Como Entero;

    //se escribe en pantalla la pregunta del valor numérico.
    Escribir "En qué número de día de la semana estamos: ";
    //se recoge un número de día de la semana por pantalla.
    Leer numeroDiaSemana;

    Segun numeroDiaSemana Hacer
      1,2,3,4,5: Escribir "Hoy es día laborable";
            De Otro Modo: //es otro día u otra cosa diferente
                  Escribir "Hoy es fin de semana";
    FinSegun //fin de la estructura Segun

FinAlgoritmo
```

Como se puede deducir, la cláusula "De Otro Modo" es opcional (puede aparecer o no, a elección del autor del programa: programador).

Estructura Repetitiva o Iterativa

La estructura repetitiva o iterativa se basa en ejecutar un conjunto de instrucciones en forma de bucle (ciclo). Es decir, el conjunto de instrucciones se ejecutará varias veces. Para ello, se debe evaluar una condición y, se vuelve al inicio del conjunto de instrucciones (cuerpo del bucle), dependiendo de si la condición es verdadera o falsa.

Bucle REPETIR

El intérprete de PSeInt, al encontrar la palabra reservada **Repetir**, entra directamente a ejecutar el cuerpo de instrucciones del bucle. Después de ejecutar todas las instrucciones de dentro del bucle, pasa a evaluar la condición de finalización de ejecución del cuerpo de instrucciones del bucle y si la evaluación de la condición es falsa (false), entonces se vuelve a comenzar a ejecutar el bucle Repetir por la instrucción de inicio del cuerpo del bucle de instrucciones.

Ejemplo:

```
Algoritmo Bienvenida
        //se define una variable de tipo entero
        //para el número del día de la semana.
        Definir numeroDiaSemana Como Entero;

        Repetir
            //se escribe en pantalla la pregunta del valor numérico.
            Escribir "En qué número de día de la semana estamos: ";
            //se recoge un número de día de la semana por pantalla.
            Leer numeroDiaSemana;

              Si numeroDiaSemana = 1
              Entonces //si al evaluar la expresión es VERDADERA:
                        //la variable "numeroDiaSemana"
                        // 'es igual' a 1
                        //Es decir, tiene el valor correcto: 1
                        // entonces, la expresión es cierta y
                        //se ejecuta la sentencia de salida que contiene.
                    Escribir "Hoy es lunes";
              Finsi //en caso de no ser el valor 1, NO se hace nada
                    //en la sentencia SI-ENTONCES, pero se sigue
                    //con la ejecución del cuerpo de instrucciones
                    //del bucle. Y al no haber más instrucciones,
                    //se pasa a evaluar la expresión de la condición de
                    //finalización.
        Hasta Que numeroDiaSemana = 1
FinAlgoritmo
```

El pseudocódigo resumido de esta sentencia del bucle Repetir, en palabras, sería:

REPETIR las siguientes instrucciones HASTA que el número introducido por la pantalla sea un número 1. En cuyo caso, saldrá del bucle.

O en otras palabras (concretando):

El programa ESCRIBE una pregunta, LEE un número; y, SI este número es igual al valor 1, ENTONCES ESCRIBE en pantalla el mensaje "Hoy es lunes". Las anteriores instrucciones, que son el cuerpo de instrucciones de un bucle REPETIR, se ejecutan HASTA que el número introducido por pantalla es el valor igual a 1.

En un bucle Repetir siempre se ejecutan las instrucciones del cuerpo del bucle, al menos una vez; porque la condición de finalización del bucle se encuentra al final del cuerpo de instrucciones del bucle; que es cuando se evalúa la expresión de la condición de finalización.

Un bucle **Repetir** se ejecuta **hasta** que la condición de finalización es VERDADERA (cierta o true, en inglés).

Bucle MIENTRAS

En un bucle **Mientras**, la expresión de la condición de finalización del bucle se evalúa **antes** de entrar a ejecutar el cuerpo de instrucciones del bucle. Si la evaluación de la condición es verdadera (cierta o true), entonces se entra a ejecutar el bucle Mientras por la instrucción de inicio del cuerpo del bucle de instrucciones.

Por lo tanto, las instrucciones del cuerpo del bucle Mientras se ejecutan MIENTRAS la condición de finalización se evalúa como VERDADERA (cierta o true). Es decir, se sale del bucle MIENTRAS cuando la condición del bucle se evalúe en su totalidad como FALSO (falsa o false, en inglés).

Ejemplo:

```
Algoritmo Bienvenida
        //se define una variable de tipo entero
        //para el número del día de la semana.
        Definir numeroDiaSemana Como Entero;

        numeroDiaSemana = 0; //se asigna este valor
                                        //para que la condición de finalización
                                        //del bucle Mientras se evalúe como
                                        //cierta; para que entre a ejecutarse
                                        //el bucle la primera vez.
```

```
Mientras numeroDiaSemana <> 1 Hacer
        //se escribe en pantalla la pregunta del valor numérico.
        Escribir "En qué número de día de la semana estamos: ";
        //se recoge un número de día de la semana por pantalla.
        Leer numeroDiaSemana;

            Si numeroDiaSemana = 1
            Entonces //si al evaluar la expresión es VERDADERA:
                        //la variable "numeroDiaSemana"
                        // 'es igual' a 1
                        //Es decir, tiene el valor correcto: 1
                        // entonces, la expresión es cierta y
                        //se ejecuta la sentencia de salida que contiene.
                    Escribir "Hoy es lunes";
            Finsi //en caso de no ser el valor 1, NO se hace nada
                  //en la sentencia SI-ENTONCES, pero se sigue
                  //con la ejecución del cuerpo de instrucciones
                  //del bucle. Y al no haber más instrucciones,
                  //se pasa a evaluar la expresión de la condición de
                  //finalización, del principio del bucle.
    FinMientras
FinAlgoritmo
```

Un bucle Mientras se ejecuta MIENTRAS la condición es verdadera (true). Pero puede que no se ejecute nunca (que el cuerpo de instrucciones del bucle no se ejecute), que es cuando la condición de finalización del bucle se evalúa, la primera vez, como falsa (false, en inglés).

Para diferenciar en qué ocasión utilizar un bucle Repetir o bien un bucle Mientras. Hay que preguntarse si el cuerpo del bucle se va a ejecutar SIEMPRE al menos una vez, entonces utilizaremos un bucle Repetir. Sin embargo, si no es necesario que el cuerpo del bucle se ejecute al menos una vez, entonces utilizaremos un bucle Mientras.

Bucle PARA

Cuando se desea ejecutar un conjunto de instrucciones, un número determinado de veces, que es conocido antes de empezar a ejecutarse el cuerpo de instrucciones del bucle; entonces se utiliza el bucle **Para**.

Ejemplo: Se desea escribir por pantalla el nombre de los días laborables de la semana.

```
Algoritmo Bienvenida
        //se define una variable de tipo entero
        //para contener el número de veces que se ejecutará el bucle.
        Definir contador Como Entero;

        Para contador = 1 Hasta 5 Con Paso 1 Hacer

                Segun contador Hacer
                        1: Escribir "Lunes";
                        2: Escribir "Martes";
                        3: Escribir "Miércoles";
                        4: Escribir "Jueves";
                        5: Escribir "Viernes";
                FinSegun //fin de la estructura Según

        FinPara //fin del bucle Para
FinAlgoritmo
```

Sí, sí ya sé que no tiene mucha utilidad. Es más, es un error grave utilizar un bucle cuando se podría haber escrito un algoritmo más simple: únicamente escribiendo los cinco días de la semana con una sentencia de salida ESCRIBIR. ¡Es cierto!, no se necesita un bucle para escribir por pantalla los días laborables de la semana. Pero, avancemos un poco más.

Uniendo todas las estructuras

Los bucles se pueden anidar entre sí (utilizar una estructura dentro de otra). En realidad, cualquier tipo de instrucción Secuencial, Condicional, e Iterativa; se puede anidar e intercalar como queramos, para resolver el problema algorítmico (programa).

Ejemplo: Se desea escribir por pantalla los días de la semana que quedan hasta el domingo, a partir del número de un día de la semana que se lee por pantalla. El programa se finalizará cuando el usuario introduzca el número cero.

```
Algoritmo Bienvenida
        //se define una variable de tipo entero
        //para el número del día de la semana.
        Definir numeroDiaSemana Como Entero;
        //para el contador que se va incrementando
        //con el día de la semana.
        Definir contador Como Entero;

        Repetir
            //mensaje al usuario para avisarle que se sale del bucle
            //cuando se introduce un número igual a 0
            Escribir "Teclee el número 0 para terminar la ejecución";
            //se escribe en pantalla la pregunta del valor numérico.
            Escribir "En qué número de día de la semana empezamos:";
```

```
        //se recoge un número de día de la semana por pantalla.
        Leer numeroDiaSemana;

        Para contador = numeroDiaSemana Hasta 7 Hacer

                Segun contador Hacer
                        1: Escribir "Lunes";
                        2: Escribir "Martes";
                        3: Escribir "Miércoles";
                        4: Escribir "Jueves";
                        5: Escribir "Viernes";
                        6: Escribir "Sábado";
                        7: Escribir "Domingo";
                FinSegun //fin de la instrucción Según

        FinPara //fin del bucle Para

    Hasta Que numeroDiaSemana = 0  //condición de finalización
                                   //del bucle Repetir
FinAlgoritmo
```

Al no haber utilizado la cláusula opcional Con Paso, el programa PSeInt interpreta que la variable "contador" se incrementa automáticamente de uno en uno, en cada iteración del bucle Para.

Se ha elegido un bucle Repetir en el exterior (el bucle que se ejecuta primero en el anidamiento) porque al menos se va a solicitar e introducir SIEMPRE UNA VEZ el mensaje de cómo finalizar el programa y se va a leer el número de la semana necesario para que se ejecute el programa.

Bucle infinito

La ejecución de un bucle puede dar como resultado un bucle infinito. Un bucle infinito es un bucle en el que la condición de finalización no permite que la ejecución salga del ciclo, y, por lo tanto, el cuerpo del bucle se ejecuta una y otra vez, de forma indefinida y sin final.

Cuando tenemos un bucle infinito, decimos coloquialmente que la ejecución se "ha colgado" (porque se ejecutan siempre las mismas instrucciones sin que haya un final de ejecución).

Subrutinas

La **Programación Estructura** permite la programación modular: dividir el programa en subrutinas (módulos) reutilizables en otros programas o en diferentes partes del propio programa original.

Estos módulos de subrutinas se componen de dos tipos de estructuras: Subprocesos o Procedimientos y Funciones.

La razón por la que surge el concepto de subrutina se debe a que en la escritura de programas hay muchas partes del código fuente que deben repetirse: bucles que solamente cambian su condición de finalización por los valores de sus variables, secuencia de instrucciones consecutivas en las que solamente cambian los valores iniciales o de partida de las variables; por ejemplo.

Para estos casos y con la finalidad de no estar copiando y pegando partes del código que tienen la misma funcionalidad (utilidad) y que solamente cambian en unos valores concreto de las variables y/o condiciones, surge el concepto de subrutina.

Por lo tanto, una subrutina es un conjunto de instrucciones que se repiten siempre todas, pero cambiando unos valores de entrada, que los denominaremos parámetros de entrada; desde los cuales se realizarán siempre ese conjunto de instrucciones y se obtendrán un valor de retorno o devolución en el caso de las subrutinas llamadas funciones; y, ninguno o uno o varios valores de retorno en el caso de las subrutinas que llamamos subproceso (procedimiento).

Subproceso (Procedimiento)

Ejemplo: Dado un número de día de la semana, recibido como parámetro de entrada en un procedimiento; se escribe en pantalla el nombre del día de la semana desde dentro del procedimiento.

```
SubProceso DiaSemana (dia)
        Segun dia Hacer
                1: Escribir "Lunes";
                2: Escribir "Martes";
                3: Escribir "Miércoles";
                4: Escribir "Jueves";
                5: Escribir "Viernes";
                6: Escribir "Sábado";
                7: Escribir "Domingo";
        FinSegun //fin de la instrucción Según
FinSubProceso
```

```
Algoritmo Bienvenida
        //se define una variable de tipo entero
        //para el número del día de la semana.
        Definir numeroDiaSemana Como Entero;
        //para el contador que se va incrementando
        //con el día de la semana.
        Definir contador Como Entero;

        Repetir
            //mensaje al usuario para avisarle que se sale del bucle
            //cuando se introduce un número igual a 0
            Escribir "Teclee el número 0 para terminar la ejecución";
            //se escribe en pantalla la pregunta del valor numérico.
            Escribir "En qué número de día de la semana empezamos:";
            //se recoge un número de día de la semana por pantalla.
            Leer numeroDiaSemana;

            Para contador = numeroDiaSemana Hasta 7 Hacer

                DiaSemana(contador);

            FinPara //fin del bucle Para

        Hasta Que numeroDiaSemana = 0  //condición de finalización
                                       //del bucle Repetir
FinAlgoritmo
```

El subproceso se denomina **DiaSemana** y, al utilizar PSeInt un intérprete para ejecutar el código, cuando se ejecuta el algoritmo e interpretarse la línea de código **DiaSemana(contador)** este subproceso DiaSemana debe "conocerlo" antes de ejecutarlo.

La instrucción **DiaSemana(contador)** evalúa el valor de la **variable contador**, en cada iteración del bucle Para y en cada iteración (ciclo de la vuelta del bucle Para) se recoge el valor de la variable **contador**, se hace una copia de ese valor de la variable y se pasa (se asigna), el valor de la variable **contador,** en la "cajita" en memoria de la variable **día**, que es el parámetro de entrada del subproceso **DiaSemana**.

Explicaremos, más adelante, el paso de valores a los parámetros por Valor y por Referencia.

Pero hay un error o, mejor dicho, un inconveniente a esta solución ya que hemos acoplado el Proceso de los datos con la Salida de los datos. Ya vimos que todo programa informático se basa en una entrada de datos, un procesamiento y una salida de datos. Una subrutina, y por lo tanto un subproceso, estaría dentro de la etapa del Proceso de datos. Por lo tanto, no es muy estructurado en módulos, realizar la salida de datos desde una subrutina que pertenece a la fase de Proceso de un algoritmo (programa).

Para desacoplar la integración entre módulos de Entrada de datos, Proceso y Salida de datos, quizás sea más visible el utilizar el concepto de subrutina que llamaremos función.

Ejemplo: Dado un número de día de la semana, recibido como parámetro de entrada en una función, se escribe en pantalla el nombre del día de la semana desde el cuerpo principal del programa que llama a la función que devuelve el nombre de ese día de la semana.

```
Funcion resultado = DiaSemana (dia)
        Segun dia Hacer
                1: resultado = "Lunes";
                2: resultado = "Martes";
                3: resultado = "Miércoles";
                4: resultado = "Jueves";
                5: resultado = "Viernes";
                6: resultado = "Sábado";
                7: resultado = "Domingo";
        FinSegun //fin de la instrucción Según
FinFuncion


Algoritmo Bienvenida
        //se define una variable de tipo entero
        //para el número del día de la semana.
        Definir numeroDiaSemana Como Entero;
        //para el contador que se va incrementando
        //con el día de la semana.
        Definir contador Como Entero;

        Repetir
            //mensaje al usuario para avisarle que se sale del bucle
            //cuando se introduce un número igual a 0
            Escribir "Teclee el número 0 para terminar la ejecución";
            //se escribe en pantalla la pregunta del valor numérico.
            Escribir "En qué número de día de la semana empezamos:";
            //se recoge un número de día de la semana por pantalla.
            Leer numeroDiaSemana;

            Para contador = numeroDiaSemana Hasta 7 Hacer

                Escribir "Día: ", DiaSemana(contador);

            FinPara //fin del bucle Para

        Hasta Que numeroDiaSemana = 0  //condición de finalización
                                       //del bucle Repetir
FinAlgoritmo
```

Es decir, es idéntico que el ejemplo anterior, pero para cada opción de la instrucción Según, se utiliza una instrucción de retorno sobre la variable de retorno de la función **resultado**.

Por lo tanto, la diferencia entre una sentencia de llamada a una subrutina Subproceso y una sentencia de llamada a una subrutina Función, es que ésta última se puede utilizar en una sentencia como parte de una expresión a evaluar. Ya que una Función devuelve un valor a la instrucción de llamada en el propio lugar de la llamada a la Función.
Es decir, la llamada a una Función lleva aparejada la devolución de un valor al lugar desde el que se llamó a la Función. Por eso, se puede llamar a una Función desde una sentencia de salida como es Escribir.

A la variable **contador** se la denomina Parámetro de Llamada a la subrutina. Y a la variable **día** se la denomina Parámetro de Entrada a la subrutina.

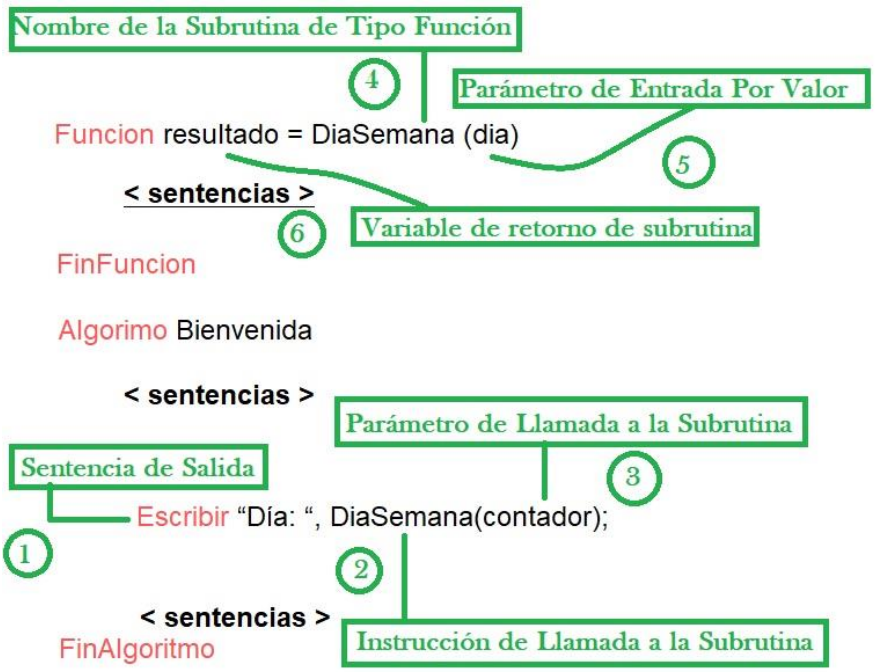


Ilustración 03.04 | Secuencia de instrucciones de llamadas a una Subrutina

La variable **día** es un parámetro de entrada que se ha recibido en la subrutina **por Valor**, que es la forma de paso de variables y parámetros a las subrutinas por defecto (prefijado) en el programa **PSeInt**.

El parámetro de llamada y el parámetro de entrada pueden tener el mismo nombre. Y no existe confusión para el ordenador porque son posiciones físicas de memoria diferentes, aunque se las llame con el mismo nombre. Es decir, son el mismo nombre, pero están en "cajitas" diferentes en memoria.

Al utilizar una subrutina de tipo Función, hemos conseguido desacoplar la parte de Proceso de un programa y la parte de Salida del programa. Esto se debe hacer siempre "que se pueda", porque cada parte de un programa tiene su funcionalidad bien definida: la **fase de Entrada** obtiene los datos de entrada al programa; la **fase de Procesado** los trata y hace las gestiones necesarias con ellos para obtener los resultados que se presentarán al usuario en la **fase de Salida**. A esto se le denomina las **Capas de un Programa**.

Parámetros de entrada por Valor y por Referencia

Los parámetros de llamada a una subrutina, ya sea un Subproceso o una Función (devuelve un valor en la llamada a la función), se pueden "pasar" (recoger en la subrutina) bien por Valor o bien por Referencia en los parámetros de entrada a la subrutina.

Si no se especifica nada (no usamos ninguna cláusula adicional) en los parámetros de entrada, el programa PSeInt interpreta que los parámetros de entrada se reciben (pasan) por defecto en forma de **por Valor**.

Como es "por defecto", la subrutina del apartado anterior se puede declarar, indistintamente, de las formas:

```
Funcion resultado = DiaSemana (dia)
        < sentencias >
FinFuncion
```

O bien:

```
Funcion resultado = DiaSemana (día Por Valor)
        < sentencias >
FinFuncion
```

En ambas definiciones de subrutina, se obtiene el mismo resultado en la ejecución.

Por Valor

Para pasar **Por Valor** un parámetro de llamada a un parámetro de entrada, se debe sacar en memoria una copia del valor del parámetro de llamada en el parámetro de entrada.
Si en la subrutina se modifica el contenido (el valor) del parámetro de entrada, al ser el parámetro de entrada una copia en memoria del parámetro de llamada, entonces el parámetro de llamada, del programa principal, no se ve afectado por el cambio del valor del parámetro de entrada.

Ejemplo:

```
Subproceso DiaSemana (numeroDiaSemana Por Valor)
        Escribir "Día del parámetro de entrada: ", numeroDiaSemana;

        //Incrementamos el parámetro de entrada
        numeroDiaSemana = numeroDiaSemana + 1;

        //Aviso:
        //Escriba en PSeInt la siguiente instrucción en una sola línea del Editor
        Escribir "Día del parámetro de entrada: ", numeroDiaSemana,
                " después de incrementar en UNO el valor de entrada";

FinSubproceso


Algoritmo Bienvenida
        //se define una variable de tipo entero, en el programa principal,
        //para el número del día de la semana.
        Definir numeroDiaSemana Como Entero;

        //se escribe en pantalla la pregunta del valor numérico.
        Escribir "En qué número de día de la semana estamos:";
        //se recoge un número de día de la semana por pantalla.
        Leer numeroDiaSemana;

        Escribir "Día del parámetro de llamada: ", numeroDiaSemana;

        Escribir "Llamamos a la subrutina";
        DiaSemana(numeroDiaSemana);
        Escribir "FIN de la llamada a la subrutina";

        //Aviso:
        //Escriba en PSeInt la siguiente instrucción en una sola línea del Editor
        Escribir "Día del parámetro de llamada: ", numeroDiaSemana,
                " después de la llamada a la Subrutina";
FinAlgoritmo
```

Como ha podido comprobar, aunque se llamen igual, el parámetro de llamada y el parámetro de entrada, son variables diferentes para el programa porque son "cajitas" de memoria diferentes.

Por Referencia

Para pasar **Por Referencia** un parámetro de llamada a un parámetro de entrada, se debe pasar como referencia la dirección en memoria (la "cajita" del parámetro de llamada) que será el valor de la cajita de llamada, con el que se trabaje desde la subrutina en el parámetro de entrada.
Si en la subrutina se modifica el contenido (el valor) del parámetro de entrada, verá que el parámetro de entrada se ve igualmente modificado en memoria en el parámetro de llamada. Esto ocurre así porque el parámetro de llamada y el parámetro de entrada, comparten la misma "cajita" de memoria (trabajan con la misma referencia a memoria: la misma dirección de memoria).

Ejemplo:

```
Subproceso DiaSemana (numeroDiaSemana Por Referencia)
        Escribir "Día del parámetro de entrada: ", numeroDiaSemana;

        //Incrementamos el parámetro de entrada
        numeroDiaSemana = numeroDiaSemana + 1;

        //Aviso:
        //Escriba en PSeInt la siguiente instrucción en una sola línea del Editor
        Escribir "Día del parámetro de entrada: ", numeroDiaSemana,
                " después de incrementar en UNO el valor de entrada";
FinSubproceso

Algoritmo Bienvenida
        //se define una variable de tipo entero, en el programa principal,
        //para el número del día de la semana.
        Definir numeroDiaSemana Como Entero;

        //se escribe en pantalla la pregunta del valor numérico.
        Escribir "En qué número de día de la semana estamos:";
        //se recoge un número de día de la semana por pantalla.
        Leer numeroDiaSemana;

        Escribir "Día del parámetro de llamada: ", numeroDiaSemana;

        Escribir "Llamamos a la subrutina";
        DiaSemana(numeroDiaSemana);
        Escribir "FIN de la llamada a la subrutina";

        //Aviso:
        //Escriba en PSeInt la siguiente instrucción en una sola línea del Editor
        Escribir "Día del parámetro de llamada: ", numeroDiaSemana,
                " después de la llamada a la Subrutina";

FinAlgoritmo
```

Como ha podido comprobar, aunque se llamen igual o con nombres diferentes, el parámetro de llamada y el parámetro de entrada, son la misma variable porque son la misma "cajita" de memoria.

La cláusula **Por Referencia** significa que el parámetro de llamada pasa a ser el mismo que el parámetro de entrada: la referencia en memoria de la cajita del parámetro de llamada es igual al del parámetro de entrada. Es decir, el parámetro de llamada le comunica al parámetro de entrada: "quiero que trabajes con el valor que hay en esta cajita de memoria de la que te paso la dirección en memoria de la cajita que te referencio". Por lo tanto, a esa dirección de memoria se la llama Paso Por Referencia (la variable de entrada a la subrutina se "refiere" a la variable de llamada del programa principal: a la misma dirección de la memoria).

Veamos que "el truco" del paso **Por Referencia**, no está en que las variables de llamada y de entrada se llamen con el mismo nombre. De hecho, lo que las hace "iguales" es la cláusula Por Referencia y NO el nombre de las variables.

Compruebe el siguiente código.

Ejemplo:

```
Subproceso DiaSemana (dia Por Referencia)

        Escribir "Día del parámetro de entrada: ", dia;

        //Incrementamos el parámetro de entrada
        dia = dia + 1;

        //Aviso:
        //Escriba en PSeInt la siguiente instrucción en una sola línea del Editor
        Escribir "Día del parámetro de entrada: ", dia,
                " después de incrementar en UNO el valor de entrada";

FinSubproceso

Algoritmo Bienvenida
        //se define una variable de tipo entero, en el programa principal,
        //para el número del día de la semana.
        Definir numeroDiaSemana Como Entero;

        //se escribe en pantalla la pregunta del valor numérico.
        Escribir "En qué número de día de la semana estamos:";
        //se recoge un número de día de la semana por pantalla.
        Leer numeroDiaSemana;

        Escribir "Día del parámetro de llamada: ", numeroDiaSemana;

        Escribir "Llamamos a la subrutina";
        DiaSemana(numeroDiaSemana);
        Escribir "FIN de la llamada a la subrutina";
```

```
//Aviso:
//Escriba en PSeInt la siguiente instrucción en una sola línea
Escribir "Día del parámetro de llamada: ", numeroDiaSemana,
        " después de la llamada a la Subrutina";

FinAlgoritmo
```

Tampoco influye si se trata de una subrutina de tipo Subproceso o Función (devuelve un valor a la sentencia que la llama), insisto, todo el procedimiento del paso de valores y variables los realiza la cláusula Por Referencia.

Animo al lector a que consulte la Ayuda del programa PSeInt en el apartado Ejecución Paso a Paso para una demostración gráfica del valor que van obteniendo las variables a lo largo de la ejecución de los programas anteriores.

A la pantalla de Ejecución Paso a Paso de la Ayuda de PSeInt, se accede desde el menú Ayuda y tiene la siguiente apariencia:

Ilustración 03.05 | Cómo realizar la Ejecución Paso a Paso con la Ayuda de PSeInt

Vamos a hacer la **tabla de seguimiento** de la llamada **Por Valor**.

Código Fuente	Número de la Dirección de la Memoria	Contenido en Memoria (Valor)	Nombre de esa Dirección de Memoria
Algoritmo Bienvenida Definir numeroDiaSemana Como Entero; Escribir "En qué número de día de la semana estamos:"; Leer numeroDiaSemana;	1001	0 1001	numeroDiaSemana numeroDiaSemana

Escribir "Día del parámetro de llamada: ", numeroDiaSemana;
 	
Escribir "Llamamos a la subrutina";
 	
DiaSemana(numeroDiaSemana);
 	
Escribir "FIN de la llamada a la subrutina";
 	
Escribir "Día del parámetro de llamada: ", numeroDiaSemana, " después de la llamada a la Subrutina";
FinAlgoritmo
//PARA EJECUTAR LA
//INSTRUCCIÓN
//DiaSemana
//PRIMERO SE EJECUTA LA
// LLAMADA A LA SUBRUTINA:
//Subproceso DiaSemana
//PASANDO UNA COPIA DEL VALOR
//DEL PARÁMETRO DE LLAMADA
// numeroDiaSemana
	1001
	
	1001
	
	1001
	
	1001
		7
		
		Parámetro de Llamada
		
		7
		
		7
			numeroDiaSemana
			
			numeroDiaSemana
			
			numeroDiaSemana
			
			numeroDiaSemana

Subproceso DiaSemana (numeroDiaSemana Por Valor) Escribir "Día del parámetro de entrada: ", numeroDiaSemana; numeroDiaSemana = numeroDiaSemana + 1; Escribir "Día del parámetro de entrada: ", numeroDiaSemana, " después de incrementar en UNO el valor de entrada"; FinSubproceso	Parámetro de Entrada 1100 1100 1100	7 8 8	numeroDiaSemana numeroDiaSemana numeroDiaSemana

Código Fuente	Número de la Dirección de la Memoria	Contenido en Memoria (Valor)	Nombre de esa Dirección de Memoria
Algoritmo Bienvenida Definir numeroDiaSemana Como Entero; Escribir "En qué número de día de la semana estamos:"; Leer numeroDiaSemana; Escribir "Día del parámetro de llamada: ", numeroDiaSemana; Escribir "Llamamos a la subrutina"; DiaSemana(numeroDiaSemana); Escribir "FIN de la llamada a la subrutina"; Escribir "Día del parámetro de llamada: ", numeroDiaSemana, " después de la llamada a la Subrutina"; FinAlgoritmo			
		NOS QUEDAMOS EN LA EJECUCIÓN DE ESTA INSTRUCCIÓN. Y SE CONTINÚA CON EL PROGRAMA PRINCIPAL	
	1001	7	numeroDiaSemana

Hay que fijarse en que la columna del número de la dirección de la memoria va cambiando.

Ahora, vamos a hacer la **tabla de seguimiento** de la llamada **Por Referencia**.

Código Fuente	Número de la Dirección de la Memoria	Contenido en Memoria (Valor)	Nombre de esa Dirección de Memoria
Algoritmo Bienvenida Definir numeroDiaSemana Como Entero; Escribir "En qué número de día de la semana estamos:"; Leer numeroDiaSemana;	1001	0	numeroDiaSemana
Escribir "Día del parámetro de llamada: ", numeroDiaSemana;	1001	7	numeroDiaSemana
Escribir "Llamamos a la subrutina";	1001	7	numeroDiaSemana
		Parámetro de Llamada	

DiaSemana(numeroDiaSemana);
 	
Escribir "FIN de la llamada a la subrutina";
 	
Escribir "Día del parámetro de llamada: ", numeroDiaSemana,
" después de la llamada a la Subrutina";
FinAlgoritmo
//PARA EJECUTAR LA
//INSTRUCCIÓN
//DiaSemana
//PRIMERO SE EJECUTA LA
// LLAMADA A LA SUBRUTINA:
//Subproceso DiaSemana
//PASANDO UNA DIRECCIÓN
//DEL PARÁMETRO DE LLAMADA
// numeroDiaSemana
	1001	7	numeroDiaSemana

		Parámetro de Entrada	
Subproceso DiaSemana (dia Por Referencia)	1001	7	Día
Escribir "Día del parámetro de entrada: ",dia;	1001	7	Día
dia = dia + 1;	1001	8	Día
Escribir "Día del parámetro de entrada: ",dia, " después de incrementar en UNO el valor de entrada"; FinSubproceso	1001	8	Día

Código Fuente	Número de la Dirección de la Memoria	Contenido en Memoria (Valor)	Nombre de esa Dirección de Memoria
Algoritmo Bienvenida Definir numeroDiaSemana Como Entero; Escribir "En qué número de día de la semana estamos:"; Leer numeroDiaSemana; Escribir "Día del parámetro de llamada: ", numeroDiaSemana; Escribir "Llamamos a la subrutina"; DiaSemana(numeroDiaSemana); Escribir "FIN de la llamada a la subrutina"; Escribir "Día del parámetro de llamada: ", numeroDiaSemana, " después de la llamada a la Subrutina"; FinAlgoritmo	NOS QUEDAMOS EN LA EJECUCIÓN DE ESTA INSTRUCCIÓN. Y SE CONTINÚA CON EL PROGRAMA PRINCIPAL		
	1001	8	numeroDiaSemana

Hay que fijarse en que la columna del número de la dirección de memoria **NO** va cambiando.

Recursividad

Toda subrutina se puede llamar desde cualquier parte del programa. Pero, además, una subrutina se puede llamar a sí misma. Esto es en lo que consiste la recursividad: una subrutina que se llama a sí misma. Entonces diremos que la subrutina es recursiva.

Veamos un ejemplo de un algoritmo no recursivo. No se asuste, es el tipo de algoritmo que hemos utilizado hasta ahora. Y lo vamos a convertir en un algoritmo recursivo.

Ejemplo: Cálculo del Factorial de un número.

Le recuerdo que el Factorial de 5, es el resultado de realizar la operación aritmética:

5 * 4 * 3 * 2 * 1

Que tiene como resultado el valor 120.

```
// Factorial: función que calcula
// el factorial de un número introducido como parámetro.

Funcion resultado = Factorial (numeroFactorial)

//CAPA DE ENTRADA DE DATOS:
        // La variable resultadoIntermedio no puede inicializarse a cero
        // porque vamos a hacer una multiplicación con la variable.
        Definir resultadoIntermedio Como Entero;
        Definir contadorIteraciones Como Entero;
        Definir i Como Entero; //para la variable contador del bucle Para
        resultadoIntermedio = 1;
        contadorIteraciones = 0;

        //Solamente se puede calcular el Factorial de
        //un número positivo y distinto del cero

//CAPA DE PROCESADO DE DATOS:

        Si numeroFactorial <= 0 Entonces
                resultado = 0
        SiNo
                // Calcular el Factorial de forma no recursiva: iterativa
                contadorIteraciones = numeroFactorial;

                Para i = 1 Hasta numeroFactorial Con Paso 1 Hacer

                        resultadoIntermedio = resultadoIntermedio *
                                                contadorIteraciones;

                        contadorIteraciones = contadorIteraciones - 1;

                FinPara
        FinSi

//CAPA DE SALIDA DE DATOS: en este caso la salida del algoritmo
//                                es para otro algoritmo, que en este
//                                caso está en este mismo programa,
//                                y no es para una persona (usuario)
        //Devolvemos el valor de la variable resultado
        //en el parámetro de salida del algoritmo de la función
        resultado = resultadoIntermedio;
FinFuncion
```

```
// ALGORITMO: Calcular el factorial de un número introducido por el usuario.

Algoritmo  ProcesarFactorialNumero

        Definir numeroFact Como Entero;
        Definir resultadoFactorial Como Entero;

        // Solamente se admite el cálculo del factorial de números
        // positivos y distintos de cero

        Repetir
                Escribir "Introduzca el número a calcular el Factorial: ";
                Leer numeroFact;
        Hasta Que numeroFact > 0

        // La condición del bucle Hasta elimina los números negativos y el numero cero
        // Calculamos el Factorial del número
        resultadoFactorial = Factorial(numeroFact);

        // Visualizamos por pantalla el resultado obtenido
        Escribir "El Factorial del número ", numeroFact, " es: ",
                resultadoFactorial;
FinAlgoritmo
```

Para transformar el anterior algoritmo no recursivo en recursivo, debemos analizar, primeramente, la causa de cómo es el algoritmo recursivo. Ya que todo algoritmo recursivo tiene su equivalente no recursivo. Pero esta premisa no se cumple a la inversa: todo algoritmo no recursivo (iterativo), no tiene por qué tener un algoritmo recursivo equivalente.

Veamos si se puede hacer un algoritmo recursivo del factorial de un número. Primero, vamos a pensar cómo se calcula el factorial de un número.
Hemos dicho que el factorial del número 5 se calcula con la operación de multiplicación:

$$5 * 4 * 3 * 2 * 1$$

De este ejemplo, ¿podemos extraer un comportamiento general que valga para todos los números posibles?.

Podemos, deducir la siguiente fórmula:

Factorial de un número N = N * (N – 1) * (N – 2) * (N – 3) *…

En general: N * (N – (N - 1)))

Es decir, el cálculo del factorial de un número N es igual al número N multiplicado por el número N decrementado en 1 unidad, y así sucesivamente.

Luego, ese será el algoritmo que seguirá la **función de recursividad**:

$$N* (N-1)$$

Ahora bien, cuándo se termina el ciclo de decremento de la parte de
N-1

Pues se termina cuando llegamos a 0. Ya que, por definición, el factorial de un 0 es el número 1. A este final de ejecución o parada del ciclo de llamadas (que es el de llegar al factorial de 0 es igual a 1); se le denomina: condición de finalización; o también, **condición de parada** de la recursividad: la sucesivas llamadas recursivas acotando el campo de la solución en cada llamada.

Advierta el lector que, si no hay una condición de parada, el ciclo de llamadas recursivas, sobre sí misma, sería infinita y el programa no acabaría nunca (sería un **bucle infinito de llamadas recursivas**).

Por lo tanto, ya que tenemos la condición de la función del algoritmo de la recursividad, y, la condición de parada. Vamos a codificar el algoritmo recursivo del Factorial de un número.

Ejemplo: Cálculo del Factorial de un número (con recursividad).

```
// Factorial: función que calcula recursivamente
// el factorial de un número introducido como parámetro.

Funcion resultado = Factorial (numeroFactorial)
        // La variable resultado no puede inicializarse a cero
        // porque vamos a hacer una multiplicación con la variable.

        //Solamente se puede calcular el Factorial de */
        //un número positivo y distinto del cero */
        Si numeroFactorial = 0 Entonces
                // Condición de Finalización de las llamadas recursivas:
                // el factorial de cero es UNO
                // Recordar que es una multiplicación:
                // no se puede multiplicar por cero
                // porque la multiplicación de "todo", también daría cero
                resultado = 1
        SiNo
                // Calcular el Factorial de forma recursiva
                // Recordatorio:
                // el factorial de 5 es el resultado de la operación:
                // 5 * 4 * 3 * 2 * 1
                // cuyo resultado es 120
                resultado = numeroFactorial *
                                Factorial(numeroFactorial - 1); //esta es la Función de
                                                                //Recursividad: N * (N – 1)
        FinSi
FinFuncion


// ALGORITMO: Calcular el factorial de un número, de forma recursiva,
//                  de un número introducido por el usuario.
```

```
Algoritmo FactorialRecursivo

        Definir numeroFact Como Entero;
        Definir resultadoFactorial Como Entero;

        // Solamente se admite el cálculo del factorial de números
        // positivos y distintos de cero

        Repetir
                Escribir "Introduzca el número a calcular el Factorial: ";
                Leer numeroFact;
        Hasta Que numeroFact > 0

        // La condición del bucle Hasta elimina el numero cero
        // Calculamos el Factorial del número
        resultadoFactorial = Factorial(numeroFact);

        // Visualizamos por pantalla el resultado obtenido
        Escribir "El Factorial del número ", numeroFact, " es: ",
                resultadoFactorial;
FinAlgoritmo
```

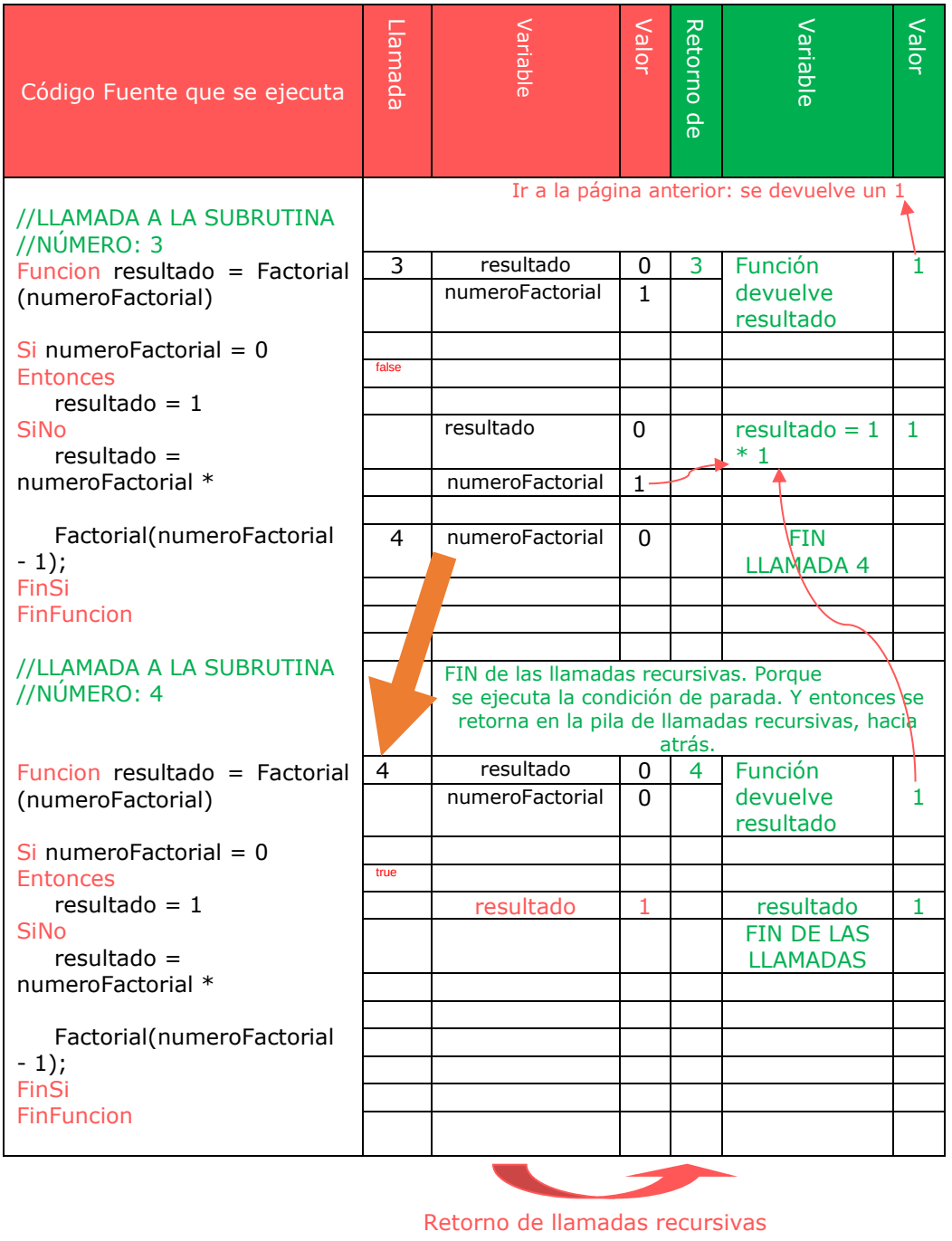
La recursividad no solamente se realiza utilizando funciones. Si necesitamos pasar varios valores entre las llamadas recursivas, podemos hacer la recursividad utilizando un procedimiento (SubProceso), siempre que los valores devueltos, para conseguir el control de la recursividad, se pasen como parámetros Por Referencia en el subproceso.

Seguidamente **mostramos la tabla del seguimiento** de la función recursiva del cálculo recursivo del Factorial de un número. En concreto, **el Factorial del número 3**.

Las columnas rojas son las llamadas secuenciales y se leen de arriba a abajo. Las columnas verdes son el retorno (vuelta atrás: retorno o devolución del resultado obtenido) de las llamadas recursivas, y se leen de abajo hacia arriba: una vez que hemos bajado hasta abajo de las columnas rojas.

Código Fuente que se ejecuta	Llamada	Variable	Valor	Retorno de	Variable	Valor
Algoritmo FactorialRecursivo						
Definir numeroFact Como Entero;		numeroFact	0			
Definir resultadoFactorial Como Entero;		resultadoFactorial	0			
Repetir						
Escribir "Introduzca el número a calcular el Factorial: ";		numeroFact	3			
Leer numeroFact;						
Hasta Que numeroFact > 0	true	numeroFact	3			
resultadoFactorial = Factorial(numeroFact);		resultadoFactorial	0	1	resultadoFactorial	6
	1	numeroFact	3		FIN LLAMADA 1	
					SEGUIMOS CON LA EJECUCIÓN	
Escribir "El Factorial del número ", numeroFact, " es: ", resultadoFactorial; //Escribe un 6					numeroFact	3
FinAlgoritmo					resultadoFactorial	6
//LLAMADA A LA SUBRUTINA						
//NÚMERO: 1						
Funcion resultado = Factorial (numeroFactorial)	1	resultado	0		Función devuelve resultado	6
		numeroFactorial	3			
Si numeroFactorial = 0 Entonces						
resultado = 1	false					
SiNo						
resultado = numeroFactorial *		resultado	0		resultado = 3 * 2	6
		numeroFactorial	3			
Factorial(numeroFactorial - 1);	2	numeroFactorial	2		FIN LLAMADA 2	
FinSi						
FinFuncion						
//LLAMADA A LA SUBRUTINA						
//NÚMERO: 2						
Funcion resultado = Factorial (numeroFactorial)	2	Resultado	0		Función devuelve resultado	2
		numeroFactorial	2			
Si numeroFactorial = 0 Entonces						
resultado = 1	false					
SiNo						
resultado = numeroFactorial *		Resultado	0		resultado = 2 * 1	2
		numeroFactorial	2			
Factorial(numeroFactorial - 1);	3	numeroFactorial	1		FIN LLAMADA 3	
FinSi						
FinFuncion						

Viene de la página siguiente: se devuelve un 1



Hay soluciones algorítmicas que son más fáciles de entender (implementar o codificar el programa) en forma recursiva que iterativa (no recursiva). El **inconveniente de un algoritmo recursivo** es que ocupa más memoria y, al realizar llamadas sobre la propia subrutina realiza más consumo de tiempo y de recursos del ordenador (como el uso de la CPU, de la memoria, realización de cambios de contexto, etcétera.).

Funciones predefinidas

Las funciones prefinidas son utilidades que nos proporciona de forma nativa (propias o prefijadas) el entorno de desarrollo del programa PSeInt.
Son funciones habituales y de uso generalizado que se ha probado que son de utilidad universal para todo programador en el uso del lenguaje, en nuestro caso, en pseudocódigo de PSeInt.

Pasamos a enumerarlas según su finalidad y/o tipo de dato al que se aplica.

Funciones utilizadas con cadenas

Función	Significado	Ejemplo
LONGITUD (<cadena>)	Cantidad de caracteres de la cadena	Escribir LONGITUD (“Hola”);
MAYUSCULAS (< cadena >)	Devuelve una copia de la cadena con todos sus caracteres en mayúsculas	Escribir MAYUSCULAS (“Hola”);
MINUSCULAS (< cadena >)	Devuelve una copia de la cadena con todos sus caracteres en minúsculas	Escribir MINUSCULAS (“Hola”);

Función	Significado	Ejemplo
SUBCADENA (< cadena >, <valor1>, <valor2>)	Devuelve una nueva cadena que consiste en la parte de la <cadena> que va desde la posición <valor1> hasta la posición <valor2> (incluyendo ambos extremos). Las posiciones utilizan la misma base que las tablas (arreglos), por lo que la primera letra de la <cadena> devuelta, será la 0 o la 1 dependiendo del perfil del lenguaje que tengamos asignado en PSeInt: “estricto” o “flexible”	Escribir SUBCADENA (“Hola”,1,3);
CONCATENAR (< cadena1 >, < cadena2 >)	Devuelve una nueva cadena resultante de unir las cadenas < cadena1 >, y, < cadena2 >	Escribir CONCATENAR (“Hola,”, “buenos días”);

Función	Significado	Ejemplo
CONVERTIRANUMERO (<cadena>)	Recibe una <cadena> de caracteres que contiene un número y devuelve una variable numérica con el mismo	Escribir CONVERTIRANUMERO ("3.1416");
CONVERTIRATEXTO (<valor>)	Recibe un <valor> que es un número y devuelve un resultado de tipo cadena con la representación como cadena de caracteres de dicho número	Escribir CONVERTIRATEXTO (3.1416);

Funciones aritméticas

Función	Significado	Ejemplo
RAIZ(<valor>)	Raíz Cuadrada del <valor> numérico	Escribir RAIZ(5);
ABS(<valor>)	Valor Absoluto del <valor> numérico	Escribir ABS(-5);
LN(<valor>)	Logaritmo Natural del <valor> numérico	Escribir LN(5);
EXP(<valor>)	Función Exponencial del <valor> numérico	Escribir EXP(5);
TRUNC(<valor>)	Función Truncar: Parte entera del <valor> numérico	Escribir TRUNC(3.1416);
REDON(<valor>)	Función Redondear: Entero más cercano al <valor> numérico	Escribir REDON(3.1416);
AZAR(<valor>)	Función Azar: Entero aleatorio en el rango [0;<valor> - 1]	Escribir AZAR(5);
ALEATORIO(<valor1>, <valor2>)	Función Aleatorio: Entero aleatorio en el rango especificado [<valor1>, <valor2>]	Escribir ALEATORIO(0,1);

Funciones trigonométricas

Función	Significado	Ejemplo
SEN(<valor>)	Seno del <valor> numérico	Escribir SEN(5);
COS(<valor>)	Coseno del <valor> numérico	Escribir COS(5);
TAN(<valor>)	Tangente del <valor> numérico	Escribir TAN(5);
ASEN(<valor>)	Arcoseno del <valor> numérico	Escribir ASEN(5);
ACOS(<valor>)	Arcocoseno del <valor> numérico	Escribir ACOS(5);
ATAN(<valor>)	Arcotangente del <valor> numérico	Escribir ATAN(5);

Funciones propias del entorno PSeInt

→ La instrucción **Borrar Pantalla** (Limpiar Pantalla) permite borrar la pantalla y colocar el cursor en la esquina superior izquierda de la pantalla del ordenador.

```
Borrar Pantalla;
```

→ La instrucción **Esperar Tecla** detiene el algoritmo hasta que el usuario presione una tecla cualquiera del teclado.

```
Esperar Tecla;
```

Esta instrucción es útil en programas de videojuegos o en programas que informan de algo al usuario.
Tenga en cuenta que PSeInt no envía, a su algoritmo, cuál es la tecla que pulsó el usuario. Es decir, desde el programa el programador no puede saber si el usuario pulsó la **W** o la **A**, por ejemplo.

→ La instrucción **Esperar** también puede utilizarse para pausar el algoritmo durante un intervalo de tiempo predefinido, indicando a continuación de la palabra clave Esperar la longitud en tiempo de la unidad de dicho intervalo. Las unidades válidas son Segundos y Milisegundos.

```
Esperar 3 Segundos;
```

Tenga en cuenta que 1000 milisegundos son 1 segundo.

Por regla general, cualquier espera superior a 3 segundos se considera una falta o carencia de Usabilidad del programa para el usuario final: quien maneja el programa.

Tablas

Las tablas son una estructura de datos que permite el almacenamiento estático (tamaño fijo y conocido previamente antes de la ejecución del programa con datos reales). En PSeInt las tablas se caracterizan por tener (almacenar), todos sus elementos, el mismo tipo de dato.

Las tablas se almacenan en una dimensión (unidimensionales), dos dimensiones (bidimensionales); y, multidimensionales (más de dos dimensiones).
Gráficamente se representan:

Ilustración 03.06 | Representación gráfica de una tabla de una, dos y tres dimensiones

En una tabla, los datos están indexados (tienen un índice). Es decir, se les accede mediante un índice de su posición relativa.

Ilustración 03.07 | Representación gráfica del "índice de tabla" y acceso a un elemento de la tabla

En PSeInt, los índices pueden empezar en 0 ó en 1. Si estamos utilizando un Perfil de Usuario de PSeInt de tipo "estricto", entonces empiezan en 0. Si estamos en un Perfil de Usuario de tipo "flexible", entonces los índices de una tabla empiezan en 1.

En esta obra, trabajaremos con el Perfil de Usuario de tipo "flexible": los índices de las tablas empiezan en 1. Para saber cuál es perfil utilizado en esta obra, consulte el capítulo **Anexo**.

Para declarar una tabla de una dimensión, se utiliza:

Dimension <identificador>[tamaño]

Ejemplo de la *ilustración 03.07*:

Dimension tabla[5];

Para declarar una tabla de dos dimensiones, se utiliza:

Dimension <identificador>[tamañoFilas, tamañoColumnas]

Ejemplo de la *ilustración 03.07*:

Dimension tabla[6,5];

Cuando trabajamos con el editor de PSeInt, nos dará un error de sintaxis (de escritura de código) si dejamos un espacio en blanco entre el identificador de la tabla y el corchete que incluye el tamaño de la tabla.

Tablas unidimensionales (array/arreglo)

Ejemplo: Dada una tabla de una dimensión con los elementos previamente definidos, mostrar el número de vocales que tiene almacenada.

```
Algoritmo ContarVocales

//ENTRADA DE DATOS
        Dimension tablaVocales[5];
        Definir contadorVocales Como Entero;
        Definir i Como Entero;
        // Inicialización de variables
        contadorVocales = 0;

        // Realizamos una declaración de datos
        // de los elementos de la tabla
        // en el propio código fuente
        tablaVocales[1]= "a";
        tablaVocales[2]= "b";
        tablaVocales[3]= "c";
        tablaVocales[4]= "d";
        tablaVocales[5]= "e";
```

```
//PROCESADO DE LOS DATOS
        Para i = 1 Hasta 5 Con Paso 1 Hacer
                Si tablaVocales[i] = "a" O tablaVocales[i] = "e" O
                   tablaVocales[i] = "i" O tablaVocales[i] = "o" O
                   tablaVocales[i] = "u" Entonces
                        contadorVocales = contadorVocales + 1;
                FinSi
        FinPara

//SALIDA DE DATOS
        // Visualizamos por pantalla el resultado obtenido
        Escribir "El número de vocales en la tabla es: ", contadorVocales;

FinAlgoritmo
```

Seguidamente visualizamos el resultado de la ejecución del algoritmo en PSeInt.

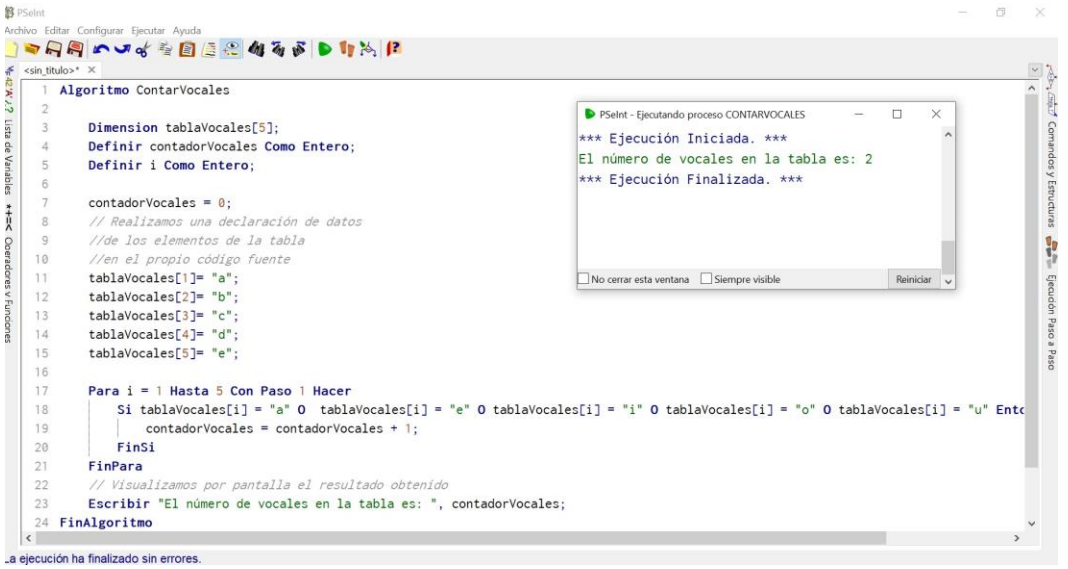


Ilustración 03.08 | Pantalla de la ejecución del ejemplo ContarVocales

El bucle Para es el más recomendado para trabajar con tablas, porque al ser una estructura de datos de dimensión estática, se conoce a priori el número de veces que se realizará el bucle de manejo de la tabla. Es decir, la condición de fin del bucle se conoce antes de entrar en el bucle porque se conoce el número de elementos de la tabla (dimensión), antes de entrar en el bucle Para.

ojo

Hay que fijarse en que la declaración del tipo de dato de los elementos de la tabla, no se define en la declaración de la tabla. Porque PSeInt asigna un tipo de dato a los elementos de la tabla, cuando se realiza la primera asignación de un valor a uno de los elementos de ella.

Tablas bidimensionales (matriz)

Ejemplo: Leer por teclado los elementos de la tabla de dos dimensiones de la ***ilustración 03.07***. Y visualizar después los elementos de la tabla por pantalla.

```
Algoritmo LeerDatosTabla

//ENTRADA DE DATOS
        Dimension tabla[6,5];
        Definir i, j Como Entero;
        // Realizamos la lectura de datos por teclado
        // de los elementos de la tabla
        Para i = 1 Hasta 6 Con Paso 1 Hacer
                Escribir "Introduzca los elementos de la fila ", i;
                Para j = 1 Hasta 5 Con Paso 1 Hacer
                        Escribir "Teclee el elemento de la fila: ", i, " y columna: " j;
                        Leer tabla[i,j];
                FinPara
        FinPara
//PROCESADO DE LOS DATOS
        //No hay ningún tratamiento.
        //Luego, el procesamiento de datos está vacío.
//SALIDA DE DATOS
        // Visualizamos por pantalla el resultado obtenido
        // en la lectura de datos por teclado
        // de los elementos de la tabla
        Escribir "La tabla que ha tecleado está almacenada así";
        Para i = 1 Hasta 6 Con Paso 1 Hacer
                //Escribimos un salto de línea entre fila y fila
                Escribir " ";
                Para j = 1 Hasta 5 Con Paso 1 Hacer
                        //Escribimos un espacio en blanco antes
                        //y después del elemento de la tabla
                        Escribir Sin Saltar " ", tabla[i,j], " ";
                FinPara
        FinPara
FinAlgoritmo
```

El resultado, en la pantalla de la consola de PSeInt, es:

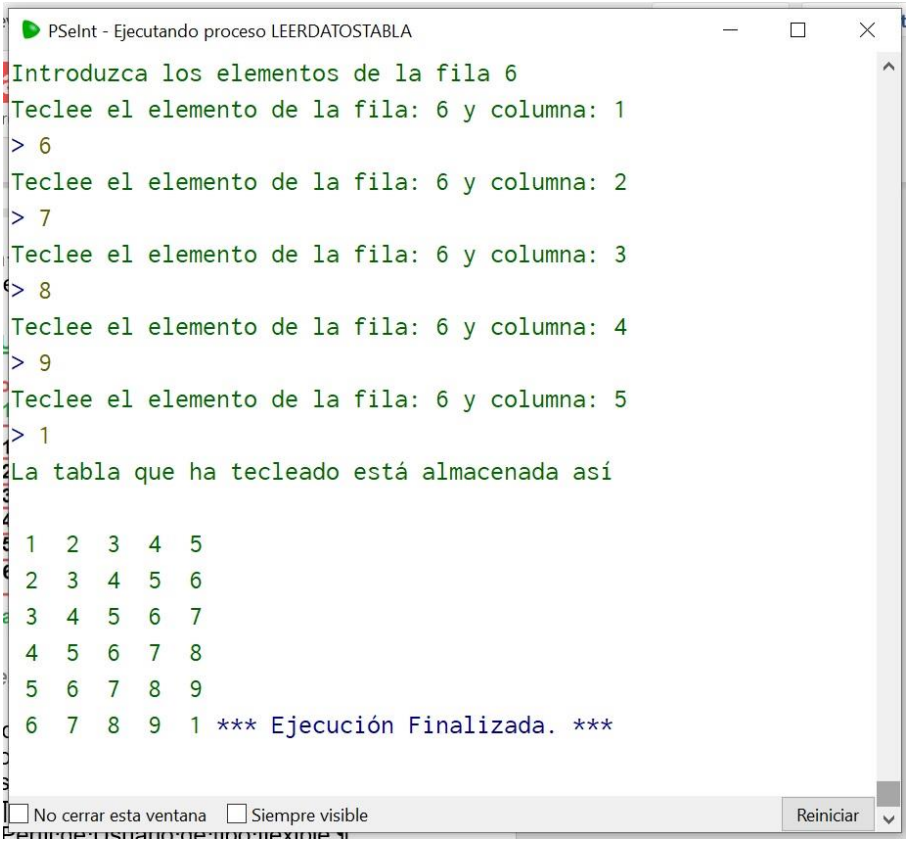


Ilustración 03.09 | Pantalla del resultado de la ejecución del ejemplo LeerDatosTabla

Paso de una tabla como parámetro a una subrutina

El paso de una tabla como parámetro de entrada a una subrutina, siempre se realiza **Por Referencia**. Independientemente de si se pone la cláusula de Por Referencia o no se pone en el parámetro de entrada a la subrutina.

Establecimos que, en el paso de una variable a una subrutina, si no aparecía una cláusula de especificación del paso Por Valor, la ausencia de esta cláusula en el parámetro de Entrada, la interpretaba el programa PSeInt, que era un paso de parámetro Por Valor. Pues bien, esto es cierto excepto en el paso de un parámetro de una variable de tipo tabla; en este caso, se pasa siempre Por Referencia (esté o no la cláusula).

Comprobemos este punto, de paso de un parámetro a una subrutina siempre **Por Referencia** de una tabla; con un ejemplo de código.

Ejemplo: Se desea escribir el día de la semana en letras, a partir de un número de día de la semana introducido por teclado. Los nombres de los días de la semana se guardarán en una tabla que se pasará a una subrutina que devolverá el día de la semana en letras (en una cadena).

```
Funcion nombreDia = DiaSemana (tablaDias, numeroDia)
        //Esta instrucción solamente tiene por objetivo
        //confirmar que la tabla se modifica al haberse
        //pasado como Parámetro de Entrada POR REFERENCIA
        tablaDias[numeroDia] = MAYUSCULAS(tablaDias[numeroDia]);
        //Retornamos el valor de la función
        nombreDia = tablaDias[numeroDia];
FinFuncion

Algoritmo PasoDeUnaTabla_A_Subrutina
//DATOS
        Dimension tabla[7];
        //se define una variable de tipo entero
        //para el número del día de la semana.
        Definir numeroDiaSemana Como Entero;

        // Inicialización
        tabla[1] = "Lunes";
        tabla[2] = "Martes";
        tabla[3] = "Miércoles";
        tabla[4] = "Jueves";
        tabla[5] = "Viernes";
        tabla[6] = "Sábado";
        tabla[7] = "Domingo";
```

```
//ENTRADA
        //se escribe en pantalla la pregunta del valor numérico.
        Escribir "En qué número de día de la semana estamos: ";
        //se recoge un número de día de la semana por pantalla.
        Leer numeroDiaSemana;
//PROCESO y SALIDA
        Escribir "Hoy es ", DiaSemana(tabla, numeroDiaSemana);
        Escribir "El valor modificado de la tabla es ", tabla[numeroDiaSemana];
FinAlgoritmo
```

El resultado, de la ejecución del programa, sería el siguiente:

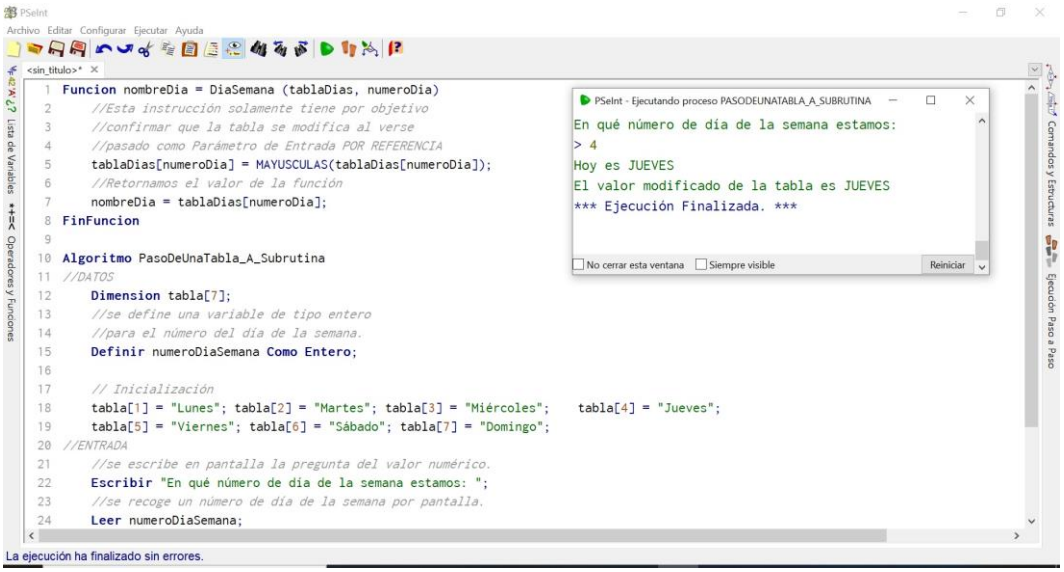


Ilustración 03.10 | Pantalla del resultado de la ejecución del ejemplo PasoDeUnaTabla_A_Subrutina

Ámbito o Vida de una variable

En los lenguajes reales, el ámbito de una variable se refiere al espacio del código fuente en el que la definición y uso de una variable tiene su alcance: puedo acceder a su valor que contiene.
En estos lenguajes se suele estructurar, el contenido de una variable o alcance, en tres ámbitos de validez. En el más interior o restrictivo, está el ámbito de la sentencia de tipo Estructura de Control a la que pertenece la variable, luego está recubriéndolo el ámbito de la subrutina y, finalmente, el ámbito del programa o algoritmo. A este último se le denomina ámbito o alcance global. Y quiere decir que una variable definida a nivel de programa (algoritmo) se puede usar desde el programa y desde las estructuras que contiene. Dícese de las subrutinas y dentro de estas, en las Estructuras de Control (sentencias).

Pero cuál es el ámbito de una variable en PSeInt. Ya hemos determinado que una estructura de datos de tipo tabla, tiene un ámbito de actuación de tipo global (afecta o tiene su uso desde cualquier parte del programa), ya vimos que una tabla se pasa a una subrutina como parámetro de entrada Por Referencia, independiente de si utilizamos la cláusula Por Referencia o no la usamos.

Veamos con un ejemplo, el ámbito del alcance de una variable.

Ejemplo: detectar el ámbito de uso o alcance de una variable definida a nivel global, a nivel de subrutina y a nivel de estructura de control.

```
Algoritmo AmbitoDeVariable1
        // Ámbito Global
        Definir i Como Entero;
        i = 1;
        // Entramos en el Ámbito de una sentencia de tipo Estructura de Control
        Si i = 1 Entonces
                // Variable definida a nivel Global con el mismo nombre
                Definir i Como Entero;
                // Estamos dentro del Ámbito Estructura de Control
                i = 2;

                Escribir "Valor de la variable i dentro de la Estructura de Control: ", i;
                Definir j Como Entero;
                j = 2;
                Escribir "Valor de la variable j dentro de la Estructura de Control: ", j;
        FinSi
        // Comprobar los valores al salir del Ámbito Estructura de Control y
        // volver al Ámbito Global
        Escribir "Valor de la variable i dentro del Ámbito Global: ", i;
        Definir j Como Entero;
        j = 1;
        Escribir "Valor de la variable j dentro del Ámbito Global: ", j;
FinAlgoritmo
```

Como puede apreciarse, nos da un error de sintaxis. Pero, aun así, si ejecutamos el programa, vemos que PSeInt no admite el ámbito Estructura de Control (sentencia que define un bloque de instrucciones dentro de ella).

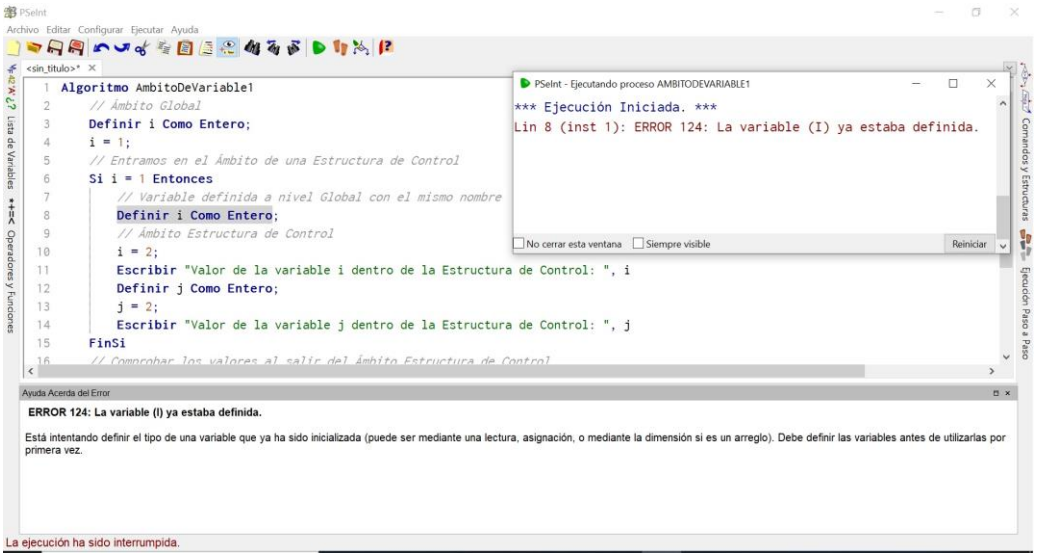


Ilustración 03.11 | Pantalla del resultado de la ejecución del ejemplo AmbitoDeVariable1

Vamos a comprobar que PSeInt solamente admite el ámbito Global y Ámbito Subrutina. Pero, antes de nada, arreglaremos el error del programa anterior.

```
Algoritmo AmbitoDeVariable2
        // Ámbito Global
        Definir i Como Entero;
        i = 1;
        // Entramos en el Ámbito de una Estructura de Control
        Si i = 1 Entonces
                // Variable definida a nivel Global con el mismo nombre
                //Definir i Como Entero;
                // Ámbito Estructura de Control
                i = 2;
                Escribir "Valor de la variable i dentro de la Estructura de",
                        " Control: ", i;
                Definir j Como Entero;
                j = 2;
                Escribir "Valor de la variable j dentro de la Estructura de",
                        " Control: ", j;
        FinSi
        // Comprobar los valores al salir del Ámbito Estructura de Control y
        // volver al Ámbito Global
        Escribir "Valor de la variable i dentro del Ámbito Global: ", i;
        Definir j Como Entero;
        j = 1;
        Escribir "Valor de la variable j dentro del Ámbito Global: ", j;
FinAlgoritmo
```

El resultado que obtenemos es el de un error por definir una variable j en el ámbito de Estructura de Control e intentarla redefinirla en el ámbito Global. El resultado de la ejecución se muestra seguidamente.

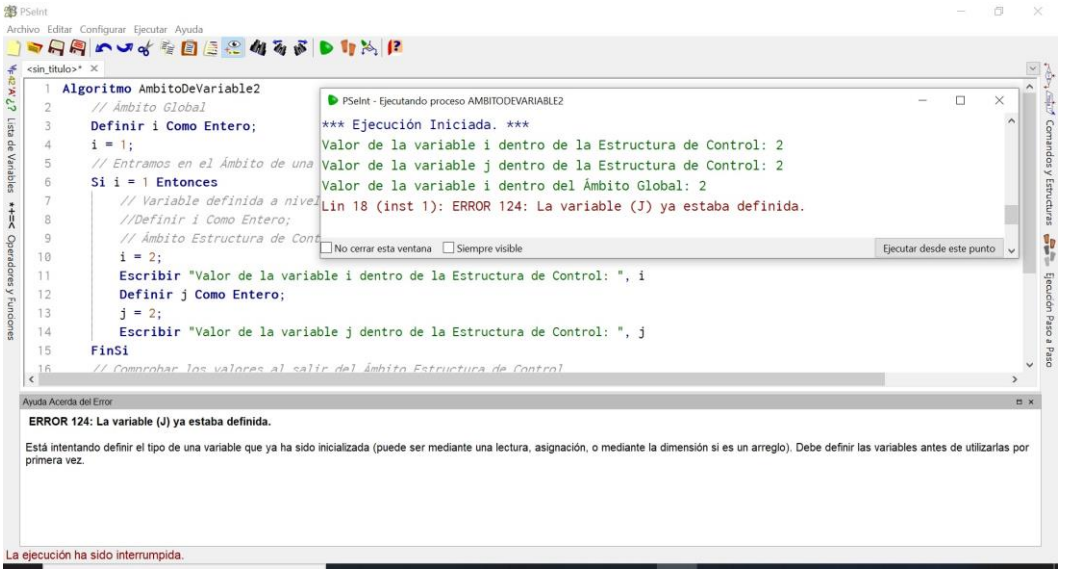


Ilustración 03.12 | Pantalla del resultado de la ejecución del ejemplo AmbitoDeVariable2

Dejamos al lector que demuestre, con la práctica de un algoritmo de su autoría, que el programa PSeInt solamente admite dos ámbitos del alcance o "vida" de una variable: el Global y el de Subrutina.

Conclusión

1 La escritura de programas informáticos en pseudocódigo es la forma más sencilla de aprender programación.

2 Los ordinogramas o diagramas de flujo son tediosos de utilizar y desproporcionados de manejar en algoritmos complejos.

3 PSeInt utiliza un intérprete para ejecutar los programas.

4 La Programación Estructurada se fundamenta en la utilización de tres tipos de instrucciones: secuencial, condicional, e iterativa. De forma que cualquier problema computacional (informático) se puede resolver algorítmicamente (mediante un programa de ordenador), únicamente, con la combinación de estos tres tipos de instrucciones (sentencias).

5 Toda subrutina puede llamarse desde cualquier parte del programa y tantas veces como sea necesario. Una subrutina puede llamar a otra u otras subrutinas.

6 Una subrutina puede llamarse a sí misma. A esa subrutina se la denomina Subrutina Recursiva.

7 Toda subrutina recursiva debe tener definida una "condición de parada" de las llamadas recursivas; que finalice la "condición de la recursividad" (función recursiva) de la solución recursiva del algoritmo.

8 Toda función recursiva tiene una solución algorítmica en forma no recursiva. Este principio no es cierto al revés. Es decir, de cualquier algoritmo de instrucciones no recursivo (iterativo), no existe su equivalente recursivo.

9 Como curiosidad, todos nuestros programas que creemos, son ejecutados por el sistema operativo como una subrutina del sistema operativo del ordenador.

Enunciados de ejercicios propuestos

Capítulo 4

Enunciados de ejercicios propuestos

Aprender a Programar | Ejemplos en PSeInt

En este capítulo se encuentra una relación de enunciados de ejercicios que conviene que resuelva usted mismo, con su propio criterio, antes de consultar el siguiente capítulo: Soluciones de ejercicios propuestos.

Es muy importante que intente resolver cada enunciado ya que, a programar se aprende programando (y como dice el dicho: "la práctica hace al artesano").

Cuando sea necesario, se le ofrecerá una "pista" para su resolución.

Mucho ánimo y, adelante.

Ejercicio 1: Escribir tu nombre en PSeInt

Hacer un programa que solicite tu nombre por pantalla y lo imprima de la forma:

Hola XXXX bienvenido

PISTA: Como mi nombre es José Luis, en mi caso el resultado sería:

Hola José Luis bienvenido

Ejercicio 2: Escribir tu nombre y edad en PSeInt

Hacer un programa que solicite tu nombre por pantalla y tu edad; y, lo imprima de la forma:

Hola XXXX tienes YY años

PISTA: Como mi nombre es José Luis; y, tengo 53 años, en mi caso el resultado sería:

Hola José Luis tienes 53 años

Ejercicio 3: Error de PSeInt

En un algoritmo vacío, teclee el siguiente código:

```
Algoritmo ErrorPSeInt
        Definir unaVariable Como Entero;
        unaVariable = "2022";
FinAlgoritmo
```

Ejecute el programa, pulsando sobre el botón del Triángulo Verde (play o ejecutar).

Explique por qué se ha producido el error que nos reporta el intérprete de PSeInt.

Ejercicio 4: Otro error de PSeInt

En un algoritmo vacío, teclee el siguiente código:

```
Algoritmo ErrorPSeInt
        Definir unaVariable Como Real;
        unaVariable = 2022;
        Escribir "Resultado ", unaVariable;
FinAlgoritmo
```

Ejecute el programa, pulsando sobre el botón del Triángulo Verde (play o ejecutar).

Explique por qué se ha producido el resultado que nos reporta el intérprete de PSeInt.

PISTA: Pregúntese: ¿la asignación es un tipo de dato diferente al que se ha declarado de la variable?.

Ejercicio 5: Manejo de variables

Teclear dos números enteros, realizar con ellos una suma, resta, multiplicación, división, resto de la división entera y potenciación. Realizar cada operación sobre una misma variable y mostrar su resultado, después de cada operación, sobre esa variable.

Ejercicio 6: Manejo de operadores relacionales

Teclear dos números enteros: A y B; realizar con ellos las siguientes **operaciones relacionales**:

A = B	B = A
A > B	B > A
A < B	B < A
A >= B	B >= A
A <= B	B <= A
A <> B	B <> A

Mostrando el resultado en pantalla de si cada expresión relacional es verdadera (se cumple) o es falsa (no se cumple).

Ejercicio 7: Evaluar una expresión de operadores relacionales y lógicos

Teclear un número entero y mostrar por pantalla si el número introducido por teclado está verdaderamente o no, dentro del rango de valores 0 y 10.

Ejercicio 8: Evaluar otra expresión de operadores relacionales y lógicos

Teclear un número entero y mostrar por pantalla si el número introducido por teclado es cierto o no que es mayor que 10, pero no igual; o bien es un número impar.

Ejercicio 9: Operaciones aritméticas

Cuál es el resultado de las siguientes operaciones aritméticas.

2.1) 3 - 5 – 2

2.2) 3 – 5 * 2

2.3) 3 + 5 / 2

2.4) (3 – 5) * 2

2.5) 3 ^ 5 * 2

2.6) 3 ^ (5 / 2)

2.7) (3 + 5) * (2 – 1)

2.8) ((3 + 5) * 2) – 2

2.9) 3 MOD 5 + 2

2.10) 3 MOD (5 – 2)

Razone la respuesta que nos muestra la salida de cada operación aritmética.

Ejercicio 10: Incrementar un valor a una variable de tipo Entero

Dada la siguiente definición de variable:

```
Definir contador Como Entero;
contador = 1;
```

Hay que conseguir que la variable "contador" tenga el valor 2, pero sin asignarle el valor 2. Es decir, no es válido realizar la asignación:

```
contador = 2;
```

PISTA: Advierta que en el título del ejercicio aparece la palabra: incrementar. Y el nombre de la variable se denomina "contador". Es decir, debido al nombre que tiene la variable sus valores, es previsible que contenga los valores sucesivos: 1, 2, 3, 4, 5…

Ejercicio 11: Intercambio de valores entre dos variables

Dada la definición de variables:

```
Definir variableUno Como Entero;
Definir variableDos Como Entero;

variableUno = 1;
variableDos = 2;
```

El ejercicio consiste en que la variableUno termine con el valor 2. Y que la variableDos termine con el valor 1.

PISTA: Si su primera idea ha sido realizar las asignaciones:

```
variableUno = variableDos;
variableDos = variableUno;
```

Ya le puedo adelantar que NO es correcto. Es más, es un error muy grave realizar esas asignaciones.

Ejercicio 12: El mayor de dos valores

Teclear dos números: A y B; y mostrar qué número de los dos es el mayor.

Ejercicio 13: Sumar dos valores, siempre que ambos sean menores de 10

Teclear dos números: A y B; y mostrar la suma siempre que ambos números sean menores de 10. En caso contrario, mostrar el número que lo incumple.

Ejercicio 14: Multiplicar dos valores, y comprobar si es mayor, menor o igual que 100

Teclear dos números: A y B; y multiplicarlos. Si el resultado es mayor que 100, mostrar el mensaje: El resultado es mayor que 100. Si el resultado es menor que 100, mostrar el mensaje: El resultado es menor que 100. Y si el resultado es igual a 100, mostrar el mensaje: El resultado es igual a 100.

Ejercicio 15: Valor par o impar

Teclear un número. Imprimir en pantalla si el número introducido es par o impar.
Restricción: es obligatorio utilizar la sentencia Segun.

Ejercicio 16: Menú de pantalla

En el programa, primeramente el usuario tecleará dos números. Después se mostrará un menú con las opciones:

1. Suma
2. Resta
3. Multiplicación
4. División

0 Finalizar programa

Al seleccionar cada opción, se mostrará el resultado de la operación aritmética elegida. El menú se visualizará repetidamente, hasta que el usuario introduzca la opción 0, en cuyo momento se concluirá la ejecución del programa.

PISTA: Hacer uso de la **función predefinida**: Borrar Pantalla

Ejercicio 17: Contar desde un número hasta 10

Teclear un número. Imprimir en pantalla el número introducido y los sucesivos valores que toma hasta el valor 10, incrementándolo cada vez en una unidad.

Ejemplo: Si introducimos el 7
Saldrá:

7
8
9
10

Restricción: es obligatorio utilizar la sentencia Mientras. El programador debe **validar que el número introducido** es menor o igual que 10.

Ejercicio 18: Contar desde un número menor que 10 hasta 0

Teclear un número. Imprimir en pantalla el número introducido y los sucesivos valores que toma hasta el valor 0, decrementándolo cada vez en una unidad.

Ejemplo: Si introducimos el 7
Saldrá:

7
6
5
4
3
2
1
0

Restricción: es obligatorio utilizar la sentencia Mientras. El programador debe validar que el número introducido es menor o igual que 10.

Ejercicio 19: Contar desde un número hasta 10 con un bucle Repetir

Realizar el mismo ejercicio que el enunciado del Ejercicio 17 pero con la restricción: es obligatorio utilizar la sentencia Repetir. El programador debe **validar** que el número introducido es menor o igual que 10.

Ejercicio 20: Lectura de líneas de pantalla y concatenarlas en un texto

El programa deberá leer líneas de pantalla sucesivamente, hasta que el usuario introduzca en una línea la palabra **exit;** únicamente.
El algoritmo concatenará, en un texto, todas las líneas introducidas por el usuario, excepto la última línea que se introdujo con la palabra **exit**.
El resultado de la concatenación del texto, de las líneas introducidas, se mostrará por pantalla.

Ejercicio 21: Seguimiento de código con bucle Para

Realice un gráfico en papel que recoja el seguimiento, instrucción a instrucción, del siguiente algoritmo.

```
Algoritmo Ejercicio21
        //se define una variable de tipo entero
        //para contener el número de veces que se ejecutará el bucle.
        Definir i Como Entero;
        //se define una variable de tipo entero
        //para contener el resultado de la ejecución.
        Definir j Como Entero;

        j = 1;

        Para i = 1 Hasta 10 Hacer

                j = j + 1;
        FinPara //fin del bucle Para

FinAlgoritmo
```

¿Cuál es el resultado de la variable "**j**"?.

Ejercicio 22: Segundo seguimiento de código con bucle Para

Realice un gráfico en papel que recoja el seguimiento, instrucción a instrucción, del siguiente algoritmo.

```
Algoritmo Ejercicio22
        //se define una variable de tipo entero
        //para contener el número de veces que se ejecutará el bucle.
        Definir i Como Entero;

        //se define una variable de tipo entero
        //para contener el resultado de la ejecución.
        Definir j Como Entero;

        j = 1;

            Para i = 1 Hasta 10 Hacer
                i = i + 1;
                j = j + 1;
            FinPara //fin del bucle Para
FinAlgoritmo
```

¿Cuál es el resultado de la variable "**i**"?.
¿Cuál es el resultado de la variable "**j**"?.

Ejercicio 23: Contar desde un número hasta 0 con un bucle Para

Teclear un número. Imprimir en pantalla el número introducido y los sucesivos valores que toma hasta el valor 0, decrementándolo cada vez en una unidad.

Ejemplo: Si introducimos el 4
Saldrá:

```
4
3
2
1
0
```

Restricción: es obligatorio utilizar la sentencia Para.

Ejercicio 24: Calcular la media de varios números

Teclear un número que es la cantidad de números que introducirá el usuario para calcular la media de esos números e imprimir en pantalla el resultado.

Ejercicio 25: Seguimiento de código con bucles anidados

Ejecutar el siguiente código y explicar el resultado que se obtiene.

```
Algoritmo Ejercicio25
	//Para contener el número de veces que se ejecutará el bucle.
	Definir i, j Como Entero;
	i = 10;
	j = 1;

	Mientras i <= 10 Hacer
		i = i - 1;
		Repetir
			j = j + 1;
		Hasta Que j > 2
		Escribir "Valor de i: ", i;
	FinMientras
FinAlgoritmo
```

Ejercicio 26: Bucles anidados

Modifique el siguiente código para que el programa no escriba los días de la semana, cuando el usuario teclee el número cero.

```
Algoritmo Ejercicio26
	//se define una variable de tipo entero
	//para el número del día de la semana.
	Definir numeroDiaSemana Como Entero;
	//para el contador que se va incrementando
	//con el día de la semana.
	Definir contador Como Entero;
```

```
Repetir
    //se escribe en pantalla la pregunta del valor numérico.
    //mensaje al usuario para avisarle que se sale del bucle
    //cuando se introduce un número igual a 0
    Escribir "Teclee el número 0 para terminar la ejecución";
    Escribir "En qué número de día de la semana empezamos:";
    //se recoge un número de día de la semana por pantalla.
    Leer numeroDiaSemana;

    Para contador = numeroDiaSemana Hasta 7 Hacer

                Segun contador Hacer
                        1: Escribir "Lunes";
                        2: Escribir "Martes";
                        3: Escribir "Miércoles";
                        4: Escribir "Jueves";
                        5: Escribir "Viernes";
                        6: Escribir "Sábado";
                        7: Escribir "Domingo";
                FinSegun //fin de la instrucción Según
        FinPara //fin del bucle Para
Hasta Que numeroDiaSemana = 0  //condición de finalización
                                        //del bucle Repetir
FinAlgoritmo
```

Ejercicio 27: Números pares con un bucle Para

Hay que escribir por pantalla los números pares hasta el número 10,

Restricción: es obligatorio utilizar la sentencia Para.

PISTA: No hace falta hacer ninguna división por 2 para ver si el número es par. Tampoco hace falta el uso de ninguna sentencia condicional SI-ENTONCES.

Ejercicio 28: Números pares con un bucle Mientras

Hay que repetir el anterior ejercicio, con un bucle Mientras. Es decir, hay que escribir por pantalla los números pares hasta el número 10, pero utilizando un bucle Mientras.

Ejercicio 29: Eliminar la parte decimal de un número entero

Dado el código fuente siguiente.

```
Algoritmo EjercicioParteEntera
        Escribir "Al dividir un 5 y un 3, se obtiene: ", 5 / 3;
FinAlgoritmo
```

Al ejecutarlo, se obtiene el siguiente mensaje en la pantalla:

Al dividir un 5 y un 3, se obtiene: 1.6666666667

Hay que modificar el algoritmo para que se obtenga el siguiente mensaje:
Al dividir un 5 y un 3, se obtiene: 1

PISTA: Consulte las funciones predefinidas de PSeInt; ya que se trata de quedarse con la parte entera del número decimal 1.6666666667

Ejercicio 30: Calcular el resto de una división entre dos números enteros

Dado el siguiente código fuente.

```
Algoritmo EjercicioRestoDivision

        Escribir "Al dividir un 5 y un 3, se obtiene: ", 5 / 3;
FinAlgoritmo
```

Al ejecutarlo, se obtiene el siguiente mensaje en la pantalla:

Al dividir un 5 y un 3, se obtiene: 1.6666666667

Se desea modificar el algoritmo, añadiéndole una instrucción Escribir; para que nos devuelva el resto de la división entre 5 y 3, devolviendo el siguiente mensaje de salida:

El resto de dividir un 5 y un 3, es: 2

Ejercicio 31: Contar vocales de un texto

Construir un programa que, dado un texto introducido por el usuario, devuelva el número de vocales que contiene.
Ejemplo: En un lugar de la Mancha, de cuyo nombre no quiero acordarme.

Restricción: Usar una función que dado un texto devuelva el número de vocales que tiene.

Ejercicio 32: Contar los caracteres diferentes a una vocal de un texto

Construir un programa que, dado un texto introducido por el usuario, devuelva el número de vocales que contiene.
Ejemplo: En un lugar de la Mancha, de cuyo nombre no quiero acordarme.

Restricción: Usar una función que dado un texto devuelva el número de caracteres que no son una vocal.

Ejercicio 33: Contar recursivamente las vocales de un texto

Repetir el ejercicio 31 pero con una función recursiva.
Ejemplo: En un lugar de la Mancha, de cuyo nombre no quiero acordarme.

Ejercicio 34: Eliminar recursivamente los espacios en blanco de un texto

Dado el texto de ejemplo: En un lugar de la Mancha, de cuyo nombre no quiero acordarme.

Debemos obtener el texto:
EnunlugardelaMancha,decuyonombrenoquieroacordarme.

Ejercicio 35: Encontrar recursivamente si existe un carácter de espacio en blanco en un texto

Usar una función que pasando un texto devuelva si hay un espacio en blanco o no.

Ejemplo: En un lugar de la Mancha, de cuyo nombre no quiero acordarme.

Ejercicio 36: Calcular la media de varios números, almacenando los números

Volver a realizar el ejercicio 24, pero esta vez se desea, además, que se impriman en pantalla los números introducidos por el usuario, cuando se imprima el resultado de la media de los números.

PISTA: Como no sabemos el número de datos que nos va a introducir el usuario, le recomiendo utilizar una **Tabla "Dinámica"**: se define su dimensión, una vez que conocemos el número de datos que nos va a introducir el usuario.

Ejercicio 37: demostrar que PSeInt solamente tiene dos ámbitos de variables: el Global y el de Subrutina

Se debe demostrar, con la práctica de un algoritmo, que el programa PSeInt solamente admite dos **ámbitos de vida de las variables**: el Global y el de Subrutina.

PISTA: en el capítulo 3 ya se comprobó que el programa PSeInt no dispone de ámbito de las variables de las sentencias de tipo Estructura de Control. Hay que definir una variable en el ámbito Global y la misma variable en el ámbito de subrutina y comprobar que tienen valores diferentes en tiempo de ejecución.

Soluciones de ejercicios propuestos

Capítulo 5

Soluciones de ejercicios propuestos

Aprender a Programar | Ejemplos en PSeInt

El presente capítulo recoge las soluciones a los enunciados de los ejercicios propuestos en el capítulo anterior.

Las soluciones son algoritmos propuestos, por lo tanto, son soluciones aproximadas a los ejercicios. Es decir, seguramente usted encuentre soluciones más acertadas y simples, que estaría muy agradecido que me las hiciera llegar a mi correo de contacto, que se detalla al final de la obra.

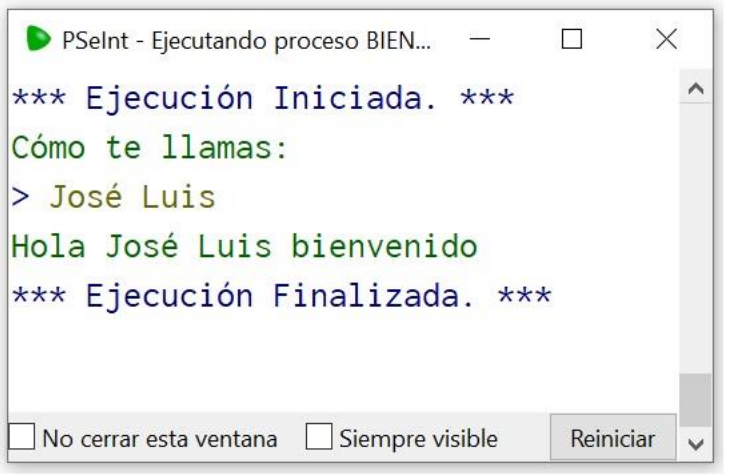
Además, se trata pormenorizadamente la prueba de las soluciones, a los algoritmos propuestos, con el apartado: **Datos de Prueba**

Solución Ejercicio 1: Escribir tu nombre en PSeInt

```
Algoritmo Bienvenida
        //Declaración de datos
        Definir nombre Como Cadena;
        //Solicitud de entrada de datos
        Escribir "Cómo te llamas: ";
        Leer nombre;
        //Procesado de datos y salida
        Escribir "Hola ", nombre, " bienvenido";

FinAlgoritmo
```

Fíjese que hemos dejado un espacio en blanco después de la cadena "Hola" y antes de la cadena "bienvenido". Lo hemos hecho así para que no se junte su nombre con las palabras "Hola" y "bienvenido".

Resultado de la ejecución.

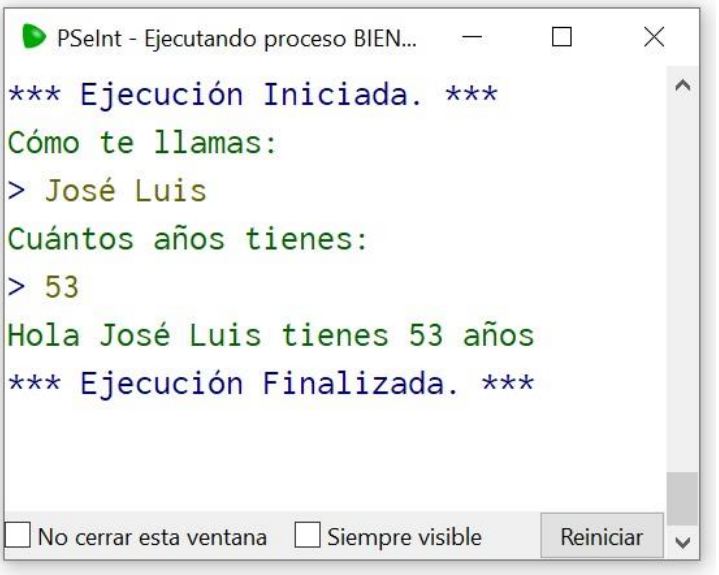


Solución Ejercicio 2: Escribir tu nombre y edad en PSeInt

```
Algoritmo Bienvenida
        //Declaración de datos
        Definir nombre Como Cadena;
        Definir edad Como Entero;
```

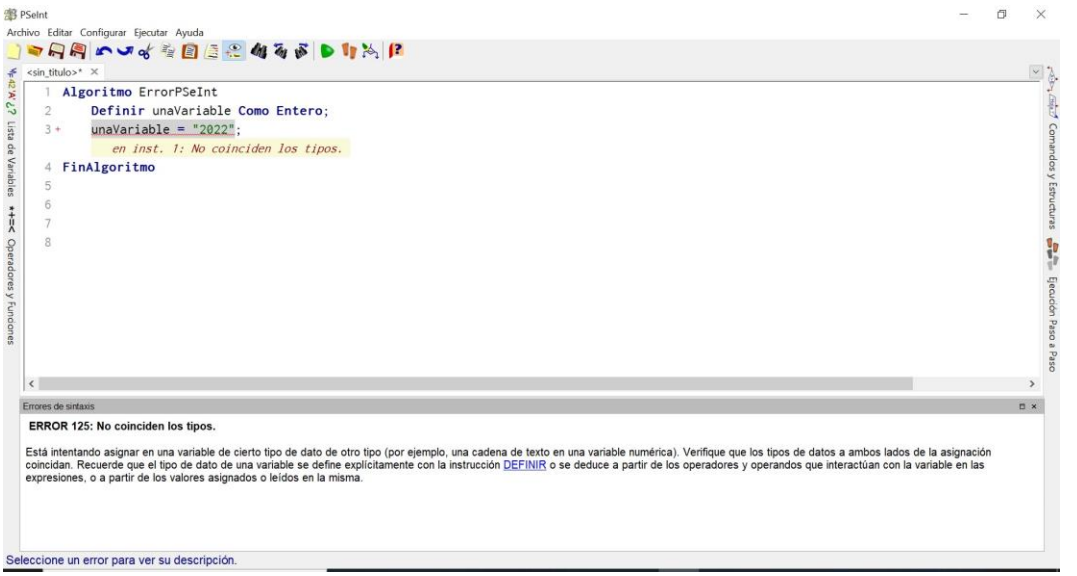
```
        //Solicitud de entrada de datos
        Escribir "Cómo te llamas: ";
        Leer nombre
        Escribir "Cuántos años tienes: ";
        Leer edad;

        //Procesado de datos y salida
        Escribir "Hola ", nombre, " tienes ", edad, " años";
FinAlgoritmo
```

Fíjese que cada variable tiene un tipo de dato diferente y que está en relación con el tipo de tratamiento del dato que vamos a hacer. Es decir, el tipo de dato Cadena se utiliza para un texto y el tipo de dato Entero para un número sin decimales.



Solución Ejercicio 3: Error en PSeInt

Aunque nos ha avisado de un error de sintaxis, aun así, hemos pulsado el icono de Ejecutar: el Triángulo Verde (play); y hemos obtenido lo siguiente.

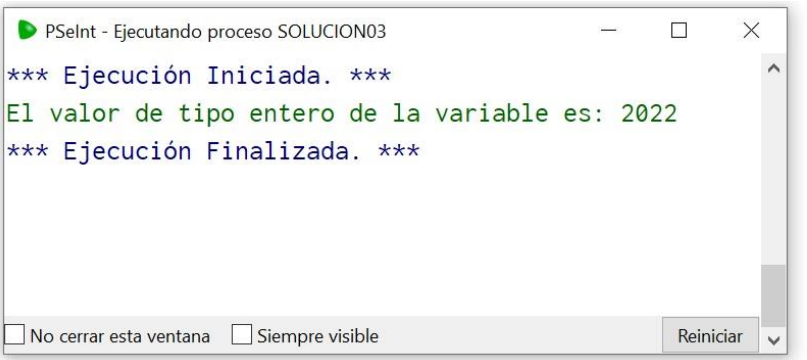


Efectivamente, a las variables de tipo Entero deben asignárseles valores de tipo Entero, y no como hemos intentado hacer: asignarle un valor de tipo Cadena. Pero tenga en cuenta que este error nos hubiera ocurrido siempre que le intentemos asignar un valor de tipo diferente a Entero.

Pero, cómo se soluciona este error de asignación entre tipos diferentes. Para resolverlo tenemos que hacer uso de una función predefinida del lenguaje PSeInt: CONVERTIRANUMERO.

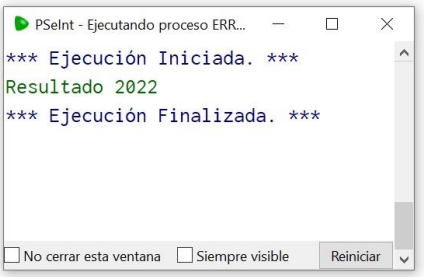
```
Algoritmo Solucion03
        Definir unaVariable Como Entero;
        unaVariable = CONVERTIRANUMERO("2022");
        Escribir "El valor de tipo entero de la variable es: ", unaVariable;
FinAlgoritmo
```

Comprobación del correcto funcionamiento del código anterior:



Solución Ejercicio 4: Otro error de PSeInt

Como ha podido comprobar, el intérprete de PSeInt no nos reporta ningún mensaje de error de sintaxis. Y al ejecutarlo, tampoco nos da ningún error. Como puede verse seguidamente.

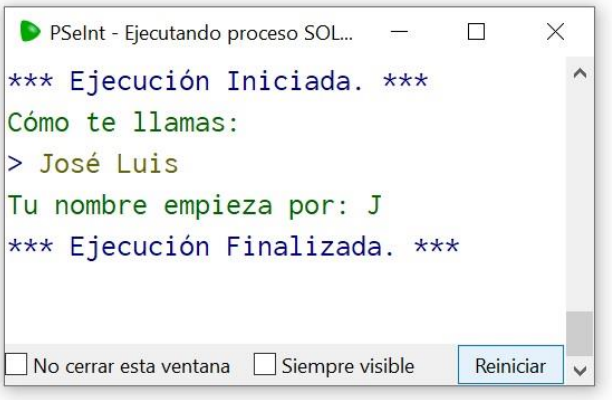


Quizás se esté preguntando, ¿por qué no ha aparecido un error de asignación de tipo de dato?, cuando hemos declarado una variable como Real y la hemos asignado un número sin decimales. Esto es correcto porque los números reales (con decimales) incluyen al conjunto de los números enteros (sin decimales). Por lo tanto, son tipos de datos compatibles.

Existen otros tipos de datos compatibles como son el tipo de dato Cadena y el tipo de dato Carácter. Veamos un ejemplo de código.

```
Algoritmo Solucion04
        //Declaración de datos
        Definir nombre Como Cadena;
        Definir primeraLetra Como Caracter;

        //Solicitud de entrada de datos
        Escribir "Cómo te llamas: ";
        Leer nombre

        //Procesado de los datos
        primeraLetra = SUBCADENA(nombre,1,1);

        //Salida de datos
        Escribir "Tu nombre empieza por: ", primeraLetra;

FinAlgoritmo
```

El resultado es:



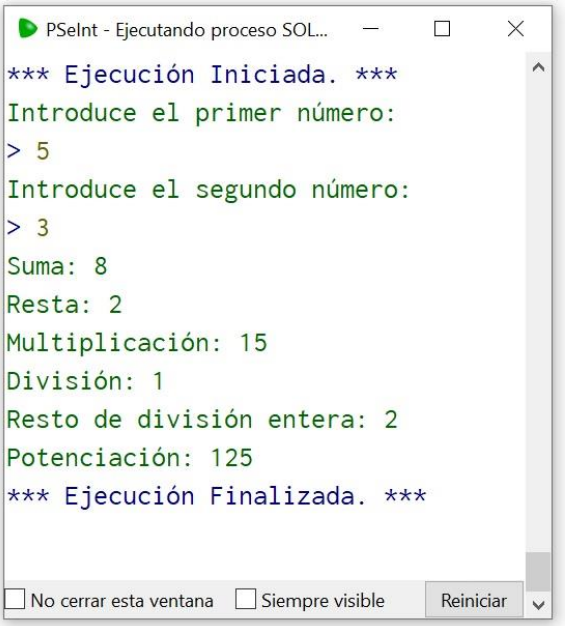
Un tipo de dato Caracter es una Cadena de longitud 1. Es por eso por lo que los caracteres están incluidos dentro del tipo de dato Cadena.

En los lenguajes "reales", al tipo de dato Caracter de PSeInt se le suele llamar CHAR. Y, al tipo de dato Cadena de PSeInt se le suele llamar STRING.

Solución Ejercicio 5: Manejo de variables

```
Algoritmo Solucion05
        Definir numero1, numero2, resultado Como Entero;
        Escribir "Introduce el primer número: ";
        Leer numero1;
        Escribir "Introduce el segundo número: ";
        Leer numero2;
        //Realizamos las operaciones y mostramos el resultado
        resultado = numero1 + numero2;
        Escribir "Suma: ", resultado;
        resultado = numero1 - numero2;
        Escribir "Resta: ", resultado;
         resultado = numero1 * numero2;
        Escribir "Multiplicación: ", resultado;
        //Debemos truncar el resultado porque la división de
        //dos números enteros devuelve un número Real: con decimales
        //Y hemos declarado la variable "resultado" de tipo Entero, y no de tipo Real
        resultado = TRUNC(numero1 / numero2);
        Escribir "División: ", resultado;
        resultado = numero1 MOD numero2;
        Escribir "Resto de división entera: ", resultado;
        resultado = numero1 ^ numero2;
        Escribir "Potenciación: ", resultado;
FinAlgoritmo
```

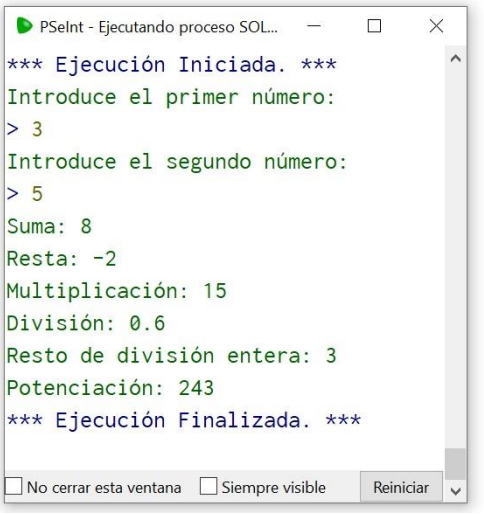
Resultado:



Le sugiero que cambie la definición de los tres números enteros por el tipo de dato Real, elimine TRUNC; y, vuelva a ejecutar el programa.

```
Algoritmo Solucion05_02
        Definir numero1, numero2, resultado Como Real;
        Escribir "Introduce el primer número: ";
        Leer numero1;
        Escribir "Introduce el segundo número: ";
        Leer numero2;
        //Realizamos las operaciones y mostramos el resultado
        resultado = numero1 + numero2;
        Escribir "Suma: ", resultado;
        resultado = numero1 - numero2;
        Escribir "Resta: ", resultado;
         resultado = numero1 * numero2;
        Escribir "Multiplicación: ", resultado;

        //NO debemos truncar el resultado porque la división de
        //dos números reales devuelve un número Real: con decimales
        resultado = numero1 / numero2;
        Escribir "División: ", resultado;
        resultado = numero1 MOD numero2;
        Escribir "Resto de división entera: ", resultado;
        resultado = numero1 ^ numero2;
        Escribir "Potenciación: ", resultado;

FinAlgoritmo
```

Resultado:

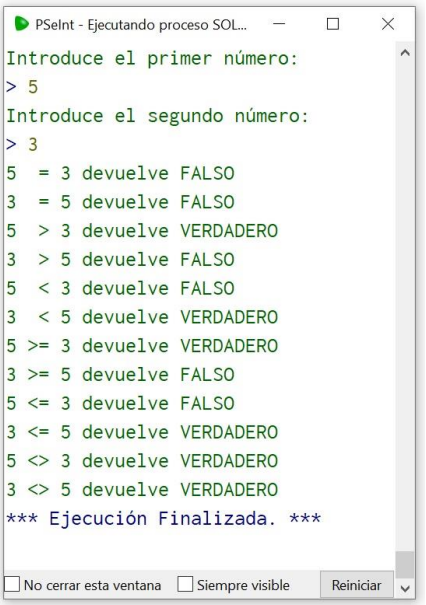


Solución Ejercicio 6: Manejo de operadores relacionales

```
Algoritmo Solucion06
        Definir numero1, numero2 Como Entero;
        Escribir "Introduce el primer número: ";
        Leer numero1;
        Escribir "Introduce el segundo número: ";
        Leer numero2;

        //Realizamos las operaciones relacionales
        //y mostramos el resultado del valor Lógico que obtenemos
        Escribir numero1, "  = ", numero2, " devuelve ",
                  numero1    = numero2;
        Escribir numero2, "  = ", numero1, " devuelve ",
                  numero2    = numero1;
        Escribir numero1, "  > ", numero2, " devuelve ",
                  numero1    > numero2;
        Escribir numero2, "  > ", numero1, " devuelve ",
                  numero2    > numero1;
        Escribir numero1, "  < ", numero2, " devuelve ",
                  numero1    < numero2;
        Escribir numero2, "  < ", numero1, " devuelve ",
                  numero2    < numero1;
        Escribir numero1, " >= ", numero2, " devuelve ",
                  numero1    >= numero2;
        Escribir numero2, " >= ", numero1, " devuelve ",
                  numero2    >= numero1;
        Escribir numero1, " <= ", numero2, " devuelve ",
                  numero1    <= numero2;
```

```
        Escribir numero2, " <= ", numero1, " devuelve ",
                numero2    <= numero1;
        Escribir numero1, " <> ", numero2, " devuelve ",
                numero1    <> numero2;
        Escribir numero2, " <> ", numero1, " devuelve ",
                numero2    <> numero1;
FinAlgoritmo
```

Resultado:



Solución Ejercicio 7: Evaluar una expresión de operadores relacionales y lógicas

```
Algoritmo Solucion07
        Definir numero Como Entero;
        Definir resultado Como Logico;
        Escribir "Introduce el número a evaluar: ";
        Leer numero;
        resultado = numero >= 0 Y numero <= 10;
        Escribir "El número: ", numero, " es ", resultado,
                " que está entre el rango de números 0 y 10";
FinAlgoritmo
```

Le recuerdo que los operadores **relacionales** tienen más prioridad que los operadores **lógicos**, en este caso el operador Y. Es decir, el tener más prioridad quiere decir que se evalúan antes los operadores relacionales que los operadores lógicos, en este caso el operador Y se evalúa después que el operador relacional >= y después que el operador relacional <=. Además, tenga en cuenta que, ante igual prioridad, se evalúa de izquierda a derecha. Es decir, se evalúa antes el operador relacional >= que el operador relacional <= ; e insisto, después de los operadores relacionales se evalúa el Y: operador lógico.

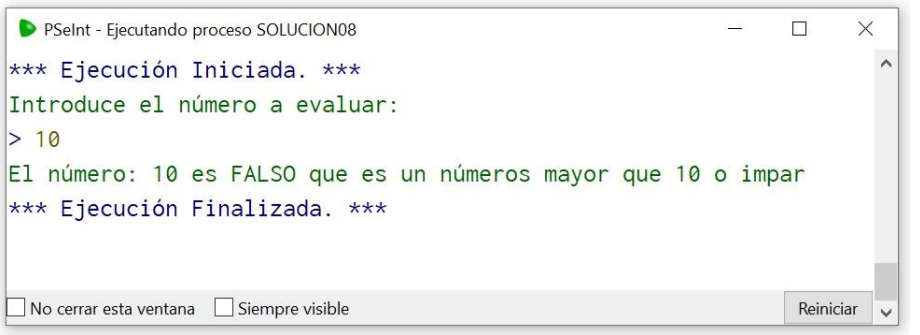
Resultado:

Solución Ejercicio 8: Evaluar otra expresión de operadores relacionales y lógicas

```
Algoritmo Solucion08
        Definir numero, impar Como Entero;
        Definir resultado Como Logico;
        Escribir "Introduce el número a evaluar: ";
        Leer numero;
        resultado = numero > 10;
        impar = numero MOD 2;
        resultado = resultado O impar = 1;
        Escribir "El número: ", numero, " es ", resultado,
                " que es un números mayor que 10 o impar";
FinAlgoritmo
```

Le recuerdo que los operadores **relacionales** tienen más prioridad que los operadores **lógicos**, en este caso el operador O

Resultado:

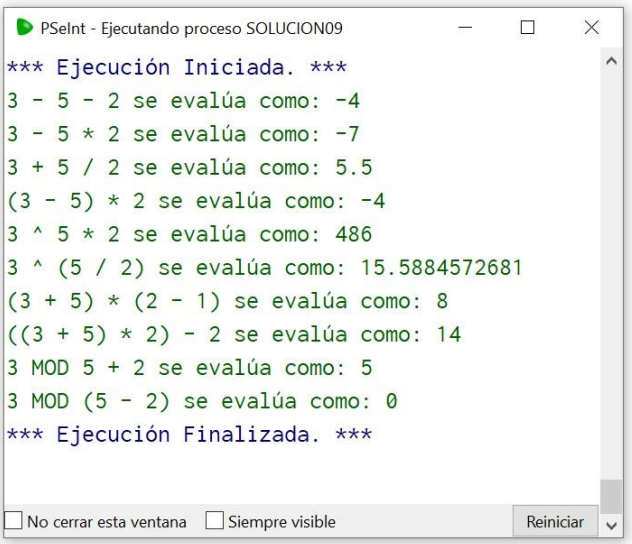


Solución Ejercicio 9: Operaciones aritméticas

```
Algoritmo Solucion09
        Escribir "3 - 5 - 2 se evalúa como: ", 3 - 5 - 2;
        Escribir "3 - 5 * 2 se evalúa como: ", 3 - 5 * 2;
        Escribir "3 + 5 / 2 se evalúa como: ", 3 + 5 / 2;
        Escribir "(3 - 5) * 2 se evalúa como: ", (3 - 5) * 2;
        Escribir "3 ^ 5 * 2 se evalúa como: ", 3 ^ 5 * 2;
        Escribir "3 ^ (5 / 2) se evalúa como: ", 3 ^ (5 / 2);
        Escribir "(3 + 5) * (2 - 1) se evalúa como: ", (3 + 5) * (2 - 1);
        Escribir "((3 + 5) * 2) - 2 se evalúa como: ", ((3 + 5) * 2) - 2;
        Escribir "3 MOD 5 + 2 se evalúa como: ", 3 MOD 5 + 2;
        Escribir "3 MOD (5 - 2) se evalúa como: ", 3 MOD (5 - 2);
FinAlgoritmo
```

Hay que aplicar los principios de prioridad de evaluación de los operadores **aritméticos**: primero los paréntesis más internos; luego, la prioridad se resuelve evaluando la expresión de izquierda a derecha y, ante la igualdad de prioridad de equivalencia, se evalúa de izquierda a derecha.

El resultado es:

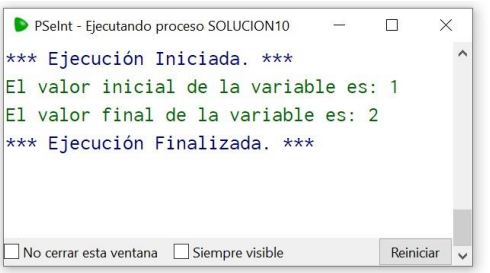


Solución Ejercicio 10: Incrementar un valor a una variable de tipo Entero

```
Algoritmo Solucion10
        Definir contador Como Entero;

        contador = 1;

        Escribir "El valor inicial de la variable es: ", contador;
        //Incrementamos la variable en una unidad
        contador = contador + 1;

        Escribir "El valor final de la variable es: ", contador;
FinAlgoritmo
```

Efectivamente, primero se evalúa la expresión de la instrucción a la derecha del signo de asignación =
Al tener el contenido de la variable **contador** con el valor 1, éste se suma al valor 1. Luego, 1+1 se evalúa como 2
Que se asigna a la variable **contador** del lado izquierdo del signo = de asignación.
Por lo tanto, la sentencia de asignación concluye con el valor de la variable **contador** con el valor 2

Veamos el resultado:



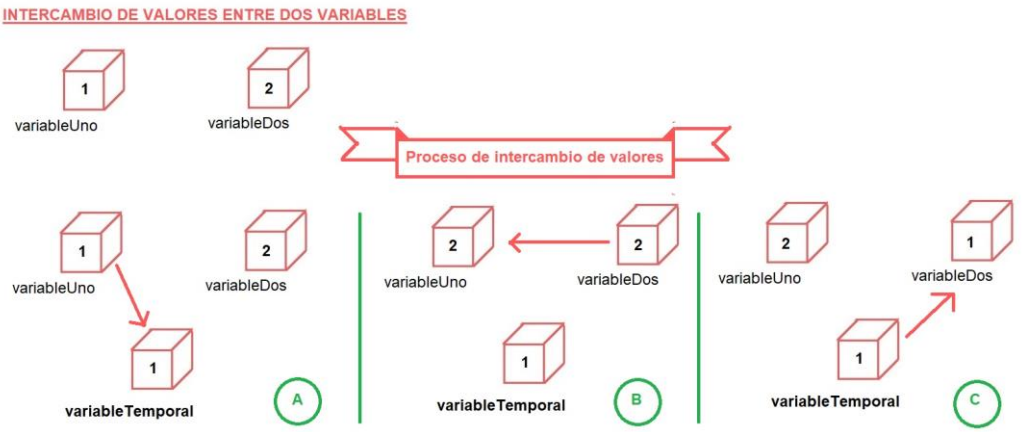
Solución Ejercicio 11: Intercambio de valores entre dos variables

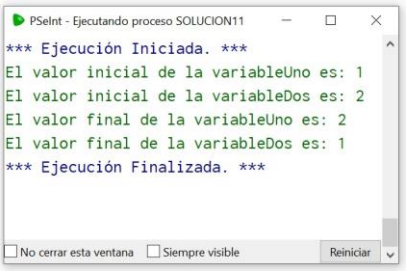
Necesariamente, la solución está en utilizar una **Variable Intermedia Temporal**:

```
Algoritmo Solucion11
        Definir variableUno Como Entero;
        Definir variableDos Como Entero;
        Definir variableTemporal Como Entero; //Variable de Intercambio
```

```
        //Inicialización de las variables
        variableUno = 1;
        variableDos = 2;
        Escribir "El valor inicial de la variableUno es: ", variableUno;
        Escribir "El valor inicial de la variableDos es: ", variableDos;

        //Intercambio de valores, por medio de la variable temporal
        variableTemporal = variableUno;     //Paso A
        variableUno = variableDos;     //Paso B
        variableDos = variableTemporal;     //Paso C

        Escribir "El valor final de la variableUno es: ", variableUno;
        Escribir "El valor final de la variableDos es: ", variableDos;
FinAlgoritmo
```

Comprobemos gráficamente el proceso de forma pormenorizada:
El resultado es:





Solución Ejercicio 12: El mayor de dos valores

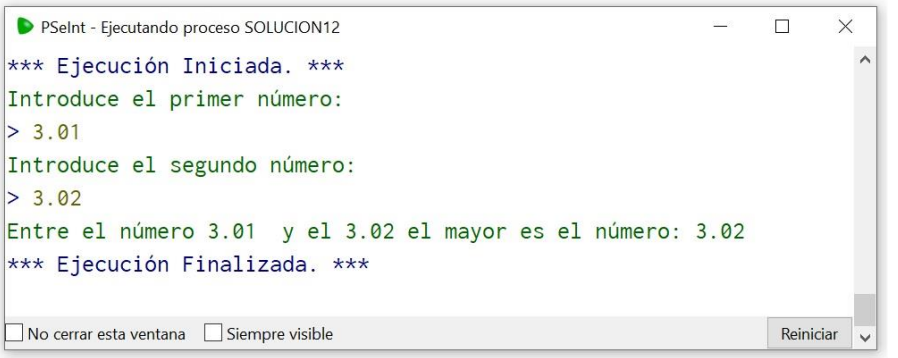
```
Algoritmo Solucion12
        Definir numero1, numero2, numeroMayor Como Real;
```

```
        Escribir "Introduce el primer número: ";
        Leer numero1;
        Escribir "Introduce el segundo número: ";
        Leer numero2;

        Si numero1 > numero2 Entonces
                numeroMayor = numero1
        SiNo
                numeroMayor = numero2;
        FinSi

        //Mostramos el resultado del valor obtenido
        Escribir "Entre el número ", numero1, "  y el ", numero2,
                  " el mayor es el número: ", numeroMayor;
FinAlgoritmo
```

Como puede haber apreciado, ahora no hemos declarado las variables de tipo entero, porque los números reales incluyen a los números enteros.

De esta forma, el usuario puede introducir cualquier tipo de dato numérico, bien sea entero o real.

Vemos el resultado:

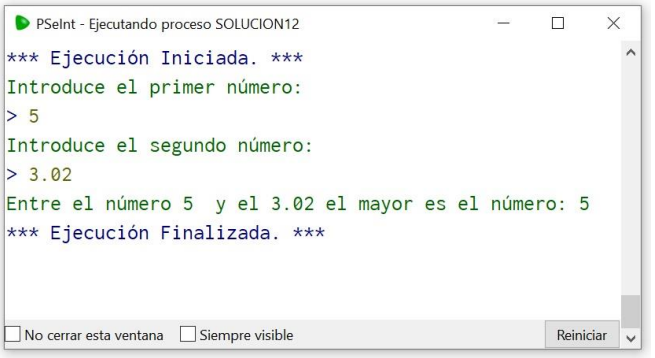


Cuando se trabaja con operadores relacionales, hay que contemplar todas las posibilidades (bifurcaciones) de los datos de entrada.
Es decir, si **numero1** es mayor que **numero2**: la condición del Si-Entonces se resuelve correctamente: se asigna el **número1**.
Si **numero1** es menor que **numero2**: la condición del Si-Entonces se resuelve correctamente: se asigna el **número2**.
Pero hay una tercera condición que hay que evaluar con los datos de entrada: cuando los números son iguales… ¿la condición del Si-Entonces es correcta?. Es decir, está bien definido el operador relacional "**>**" o debería ser el "**>=**". En este caso, está bien la condición del Si-Entonces porque si **numero1** es igual que **numero2**: la condición del Si-Entonces se resuelve correctamente: se asigna el **número2**. Porque la condición del operador relacional "**>**" es falso y se "entra" por el SiNo.

Por lo tanto, hay que "probar" razonadamente si el código fuente responde a todos los datos posibles de prueba.

Pero, luego hay que hacer las pruebas del código fuente con datos reales. Ya hemos probado el código fuente con datos de tipo Real. Hemos probado con datos enteros. Pero, se le ha ocurrido la idea de probar los tipos de datos mezclados.

Veámoslo:



Hay que razonar mentalmente, la simulación de la ejecución del código fuente con datos reales: datos de verdad; antes de ejecutar el algoritmo. Con esto garantizamos que la lógica del código fuente es correcta. Pero, además, por supuesto, hay que probar la ejecución del código fuente con todos los datos reales que podamos. A esto se le denomina: crear una batería de datos de prueba.

Solución Ejercicio 13: Sumar dos valores, siempre que ambos sean menores de 10

```
Algoritmo Solucion13
        Definir numero1, numero2, numeroMayor, suma Como Real;
        Definir sonMenores Como Logico;
        Escribir "Introduce el primer número: ";
        Leer numero1;
        Escribir "Introduce el segundo número: ";
        Leer numero2;

        Si numero1 < 10 Entonces
                Si numero2 < 10 Entonces
```

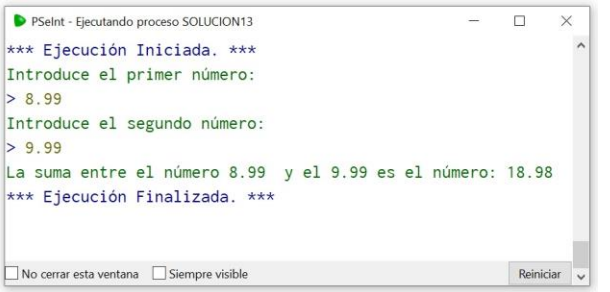
```
                suma = numero1 + numero2;
                sonMenores = VERDADERO;
        SiNo
                numeroMayor = numero2;
                sonMenores = FALSO;
        FinSi
SiNo
                numeroMayor = numero1;
                sonMenores = FALSO;
FinSi

//Mostramos el resultado del valor obtenido
Si sonMenores Entonces
        Escribir "La suma entre el número ", numero1,
                 "  y el ", numero2, " es el número: ", suma;
SiNo
        Escribir "Entre el número ", numero1,
                 "  y el ", numero2,
                 " el mayor de 10 es el número: ", numeroMayor;
FinSi
FinAlgoritmo
```

No hace falta que la condición del Si de la fase de Salida de Datos sea:

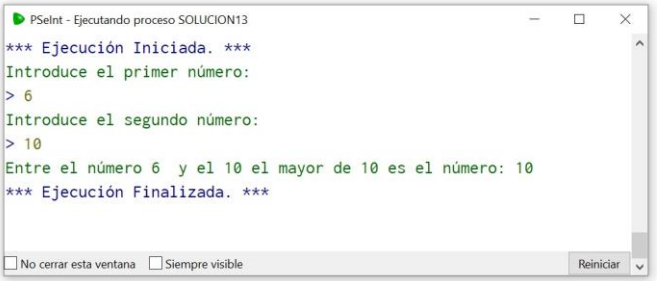
```
Si sonMenores = VERDADERO Entonces
```

Ya que la condición de una sentencia Si-Entonces se evalúa siempre como VERDADERO, entonces se realizan sentencias de detrás de la palabra reservada Entonces; o FALSO, entonces se realizan las sentencias de detrás de la cláusula SiNo.

El resultado es:



Hemos probado la primera salida del programa. Pero tenemos que probar con todos los valores posibles, por ejemplo, vamos a probar con los **Valores Límite**: qué pasa si uno de los números es, precisamente, el valor 10.



Se comprueba, que el algoritmo del código fuente es correcto, ya que el Valor Límite, que es cuando uno de los números es igual a 10 (recordemos que se pregunta por la condición "<10"), vemos que el programa muestra la sentencia Escribir que es la correcta.

Datos de prueba

En la Solución del Ejercicio 12 hemos visto que hay que realizar pruebas con una Batería de Datos de Prueba.
Pero, ¿cómo se crea esa batería de pruebas con datos reales?.

En el trabajo en informática, siempre hay que contar con tres casos de prueba de datos:

→ Ausencia de valor en la variable. Lo llamaremos: **cero valores**.

→ Cuando la variable toma un valor "esperado". Lo llamaremos: **valores esperados**.

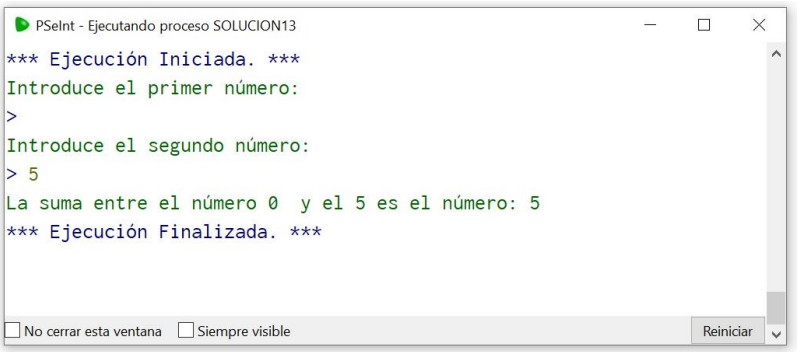
→ Cuando la variable toma un rango de valores, dentro de su dominio de valores. Los llamaremos: **N valores**.

Cero Valores

La primera consideración que debemos pensar es cuando hay "**Cero Valores**". Veamos el caso anterior de la Solución del Ejercicio 13.
Primero hay que probar: qué pasa en el caso de Ausencia de valor en las variables. Pretendemos responder a la pregunta: ¿cómo responde el código si le faltan valores a las variables?. Al tratarse de dos variables de entrada, tenemos que probar 3 casos:

→ Ausencia de valor en la variable numero1.

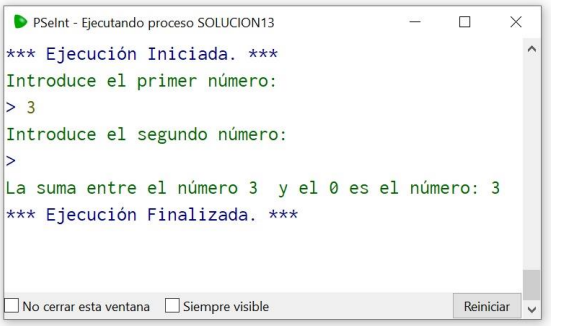
Ejecutemos el código de la Solución del Ejercicio 13 sin dar valor a la variable numero1. Obtenemos el resultado:



Hemos comprobado que ante la ausencia de valor en la variable numero1, el entorno de ejecución del programa PSeInt (su intérprete) asigna a la variable numero1 el valor cero. Por lo tanto, el programa funciona correctamente: el programa responde correctamente a la ausencia de valor.

→ Ausencia de valor en la variable numero2.

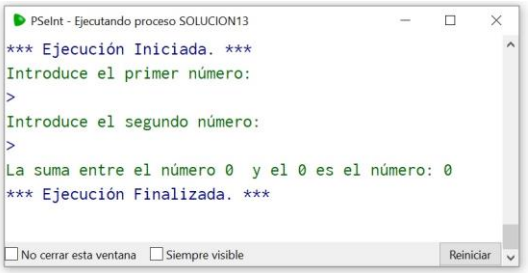
Ejecutemos el código de la Solución del Ejercicio 13 sin dar valor a la variable numero2. Obtenemos el resultado:



Hemos comprobado que ante la ausencia de valor en la variable numero2, el entorno de ejecución del programa PSeInt (su intérprete) asigna a la variable numero2 el valor cero. Por lo tanto, el programa funciona correctamente: el programa responde correctamente a la ausencia de valor.

→ Ausencia de valor en ambas variables a la vez.

Ejecutemos el código de la Solución del Ejercicio 13 sin dar valor a la variable numero1 y numero2. Obtenemos el resultado:



Al lector puede parecerle esta prueba de Ausencia de Valor, un "sin sentido": ejecutar el programa sin ningún dato de entrada. Pero, como programadores nos debemos cerciorar que nuestro código funciona ante cualquier error, descuido o accidente del usuario de la aplicación.
Porque, luego, cuando entreguemos el programa al usuario final, este puede dar dos veces a la tecla y no introducir ningún valor. Y, si no lo hemos probado, el programa se podría colgar (entrar en bucle infinito, por ejemplo) o bien terminar con una interrupción abrupta de la ejecución y el programa se finaliza con un "error fatal": abortar la ejecución. Este programa no tiene "consecuencias" pero un "error fatal" en un software del vuelo de un avión de pasajeros… queda claro que es fatal.

Valores Esperados

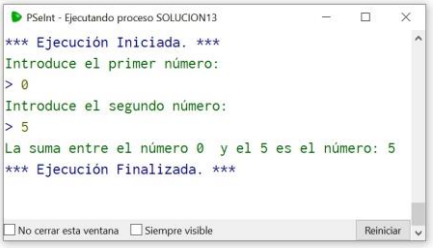
El segundo caso, es preguntarse por el caso de "**Valores Esperados**".
En este caso, probaremos el código fuente con números enteros, con números reales y, con mezcla de números enteros y reales.
A este tipo de valores se les llama la batería de pruebas con valores "esperados". Es decir, este es el caso de los valores del 99 por ciento de las veces: la ejecución del programa con los valores "normales" o "habituales" con los que va a trabajar el programa.

Dejamos al lector que haga sus propias pruebas con estos **valores esperados**. Estos son los valores con los que hemos estado probando todos nuestros ejercicios hasta el momento. Y son los que seguiremos utilizando para probar.

Pero, hay un caso especial en el "Valores Esperados" que es cuando aparece la ejecución con los **Valores Límites**. En el análisis de nuestro código fuente, los valores límites son el cero y el 10 (porque preguntamos, en la condición, por la expresión "<10").

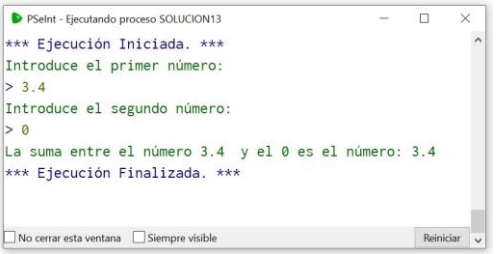
→ **Probaremos con el valor límite 0**.

Qué resultado obtenemos si numero1 es igual a 0.

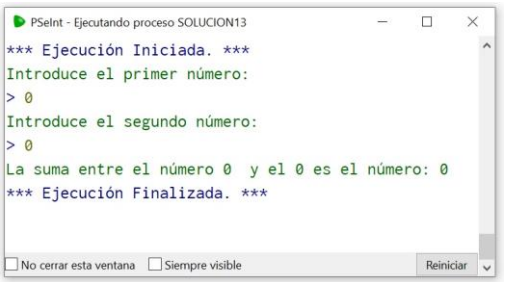


Quizás el lector se pregunte: pero si esos valores ya los hemos utilizado como prueba. Permítame sacarle del error. Una cosa es dejar la variable numero1 sin valor, que entonces PSeInt le asigna el valor 0. Y otra cosa distinta es que el usuario asigne el valor 0 a la variable numero1.
Créame, no es lo mismo. Si el usuario no introduce un valor a la variable numero1, el programa podría tener un "valor indefinido" en la variable numero1, y al ejecutar el programa se podría obtener un "error fatal". Este caso está contemplado por el programa PSeInt, y por eso, nos asigna su intérprete un valor cero a la variable numero1, ante una ausencia de valor de la variable numero1.

Qué resultado obtenemos si numero2 es igual a 0.

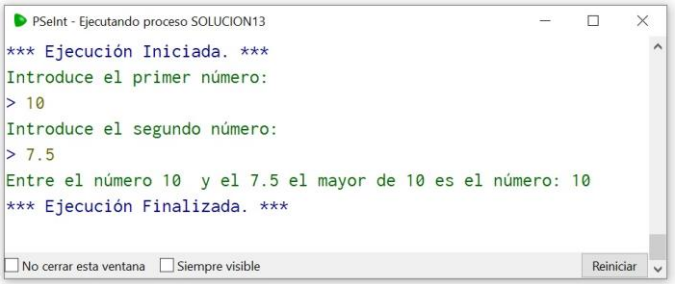


Qué resultado obtenemos si numero1 y numero2 son iguales a 0.

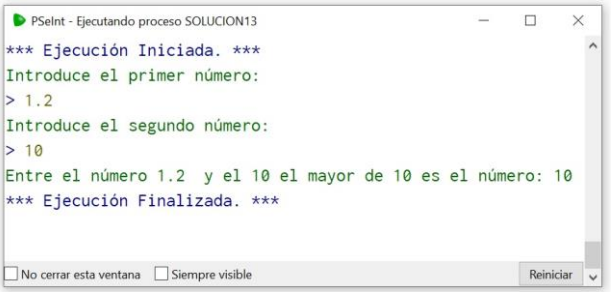


→ **Probaremos con el valor límite 10**.

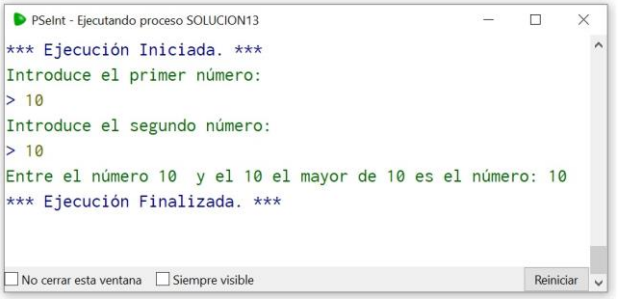
Qué resultado obtenemos si numero1 es igual a 10.



Qué resultado obtenemos si numero2 es igual a 10.



Qué resultado obtenemos si numero1 y numero2 son iguales a 10.

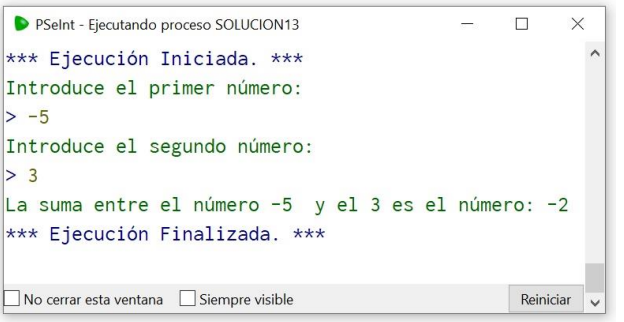


N Valores

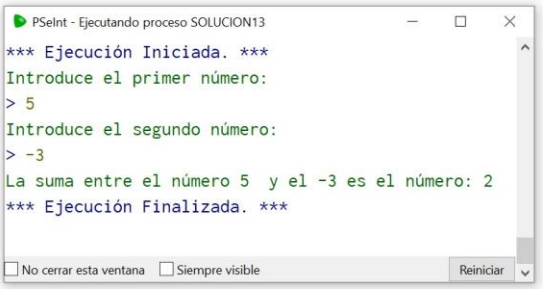
El último caso es el de "**N Valores**". Es decir, qué ocurre a nuestro código si vienen "muchos valores" a nuestro programa.
En nuestro caso, sería probar: ¿qué pasa al ejecutar el programa si nos introducen números con valores negativos?.

→ **Probaremos con el valor de números negativos**.

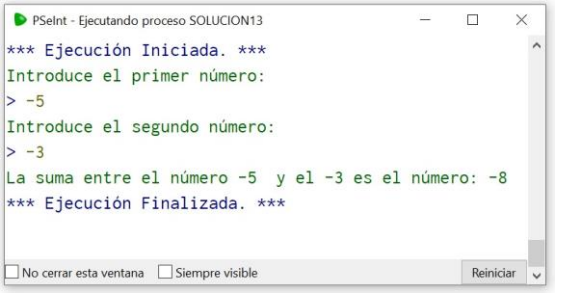
Qué resultado obtenemos si numero1 es igual a un número negativo.



Qué resultado obtenemos si numero2 es igual a un número negativo.



Qué resultado obtenemos si numero1 y numero2 son iguales a un número negativo.

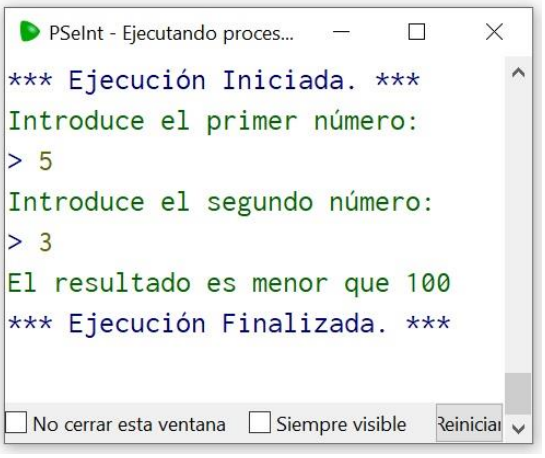


Después de considerar este epígrafe, espero que, seguramente, el lector se habrá dado cuenta, por qué se necesita un tipo de profesional especializado en pruebas de software, que hemos denominado Testeador o Especialista de Pruebas en el Capítulo 1.

Solución Ejercicio 14: Multiplicar dos valores, y comprobar si es mayor, menor o igual que 100

```
Algoritmo Solucion14
        Definir numero1, numero2, multiplicacion Como Real;
        Definir mensaje Como Cadena;
        Escribir "Introduce el primer número: ";
        Leer numero1;

        Escribir "Introduce el segundo número: ";
        Leernumero2;

        //Obtenemos el resultado
        multiplicacion = numero1 * numero2;

        Si multiplicacion = 100 Entonces
                mensaje = "El resultado es igual a 100";
        SiNo
                Si multiplicacion > 100 Entonces
                        mensaje = "El resultado es mayor que 100"
                SiNo
                        mensaje = "El resultado es menor que 100";
                FinSi
        FinSi

        //Mostramos el resultado del mensaje obtenido
        Escribir mensaje;
FinAlgoritmo
```

Resultado de la ejecución:

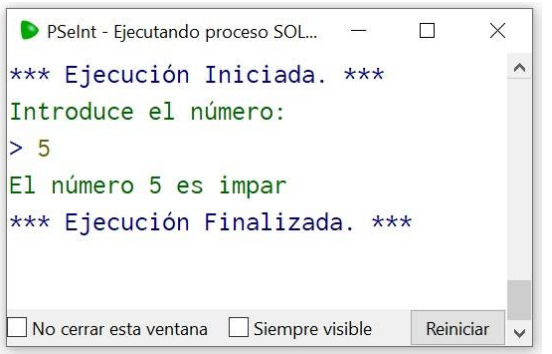


Solución Ejercicio 15: Valor par o impar

```
Algoritmo Solucion15
        Definir numero, resultado Como Real;
        Escribir "Introduce un número: ";
        Leer numero;

        //Obtenemos el resultado
        resultado = numero MOD 2;

        //Imprimimos en pantalla el resultado
        Segun resultado Hacer
                1: Escribir "El número ", numero, " es impar";
                0: Escribir "El número ", numero, " es par";
        FinSegun
FinAlgoritmo
```

Fíjese que las opciones en la sentencia Segun no tienen por qué seguir un orden creciente, o un orden determinado.
Aunque siempre es preferible ser ordenado y escribir el código de forma "previsible" para otro programador: el uso común y corriente del lenguaje en pseudocódigo y del lenguaje escrito y de las matemáticas.

Resultado:



Solución Ejercicio 16: Menú de pantalla

```
Algoritmo Solucion16
        Definir numero1, numero2, resultado Como Real;
        Definir unaOpcion Como Entero;
```

```
        //Introducimos los números a realizar las operaciones
        Escribir "Introduce el primer número: ";
        Leer numero1;
        Escribir "Introduce el segundo número: ";
        Leer numero2;
        Repetir
                Borrar Pantalla;
                //Mostramos las opciones del menú
                Escribir "1. Suma";
                Escribir "2. Resta";
                Escribir "3. Multiplicación";
                Escribir "4. División";
                Escribir "0. Finalizar programa";
                Escribir "Teclee la opción numérica de la operación a realizar: ";
                Leer unaOpcion;
                //Obtenemos el resultado
                Segun unaOpcionHacer
                        1:      resultado = numero1 + numero2;
                                Escribir "El número ", numero1, " sumado",
                                    " al número ", numero2," es ", resultado;
                        2:      resultado = numero1 - numero2;
                                Escribir "El número ", numero1, " restado",
                                    " al número ", numero2," es ", resultado;
                        3:      resultado = numero1 * numero2;
                                Escribir "El número ", numero1,
                                    " multiplicado por",
                                    " el número ", numero2," es ", resultado;
                        4:      resultado = numero1 / numero2;
                                Escribir "El número ", numero1,
                                    " dividido por",
                                    " el número ", numero2," es ", resultado;
                        0:       Escribir "Hasta pronto";

                        De Otro Modo: Escribir "Opción errónea";
                FinSegun
                Esperar 3 segundos;
        Hasta Que unaOpcion= 0
FinAlgoritmo
```

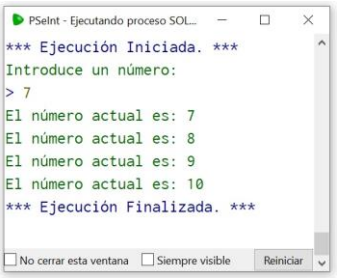
Dejamos al lector que realice las pruebas del anterior código.

Solución Ejercicio 17: Contar desde un número hasta 10

```
Algoritmo Solucion17
        Definir unNumero, resultado Como Real;

        //Introducimos los números para realizar la cuenta
        Escribir "Introduce un número: ";
```

```
        Leer unNumero;

        //Inicializamos la variable de salida del resultado
        resultado = unNumero;

        Mientras resultado <= 10 Hacer
                Escribir "El número actual es: ", resultado;
                resultado = resultado + 1;
        FinMientras
FinAlgoritmo
```

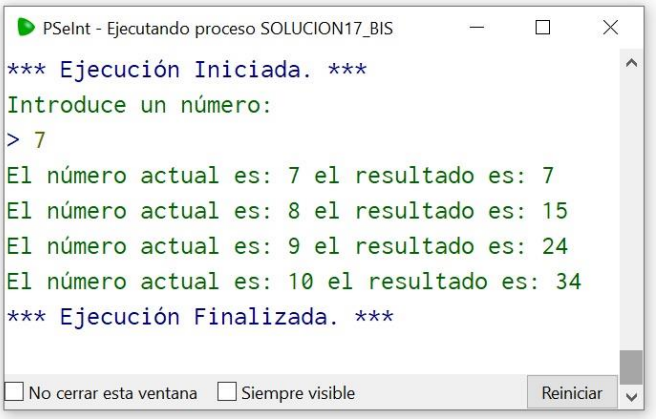
Resultado para el valor introducido: 7



Piense una solución al algoritmo anterior si nos hubieran pedido que sumáramos todos los números que hay entre el número introducido y el número 10. Es decir, hacer la suma de los valores: 7+8+9+10

```
Algoritmo Solucion17_bis
        Definir unNumero, resultado Como Real;

        //Introducimos los números para realizar la cuenta
        Escribir "Introduce un número: ";
        Leer unNumero;

        //Inicializamos la variable de salida del resultado
        resultado = unNumero;

        Mientras unNumero <= 10 Hacer
                Escribir "El número actual es: ",unNumero,
                        " el resultado es: ", resultado;
                unNumero = unNumero + 1;
                resultado = resultado + unNumero;
        FinMientras
FinAlgoritmo
```

Resultado:



Piense ahora en la siguiente plantilla de programa. La intención de la plantilla es realizar una única salida del programa al final del algoritmo con la suma total del ejemplo: 7+8+9+10

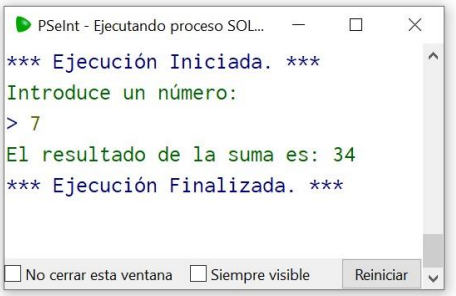
```
Algoritmo Solucion17_bis_bis
        Definir unNumero, resultado Como Real;

        //Introducimos los números para realizar la cuenta
        Escribir "Introduce un número: ";
        Leer unNumero;

        //Inicializamos la variable de salida del resultado
        resultado = ¿?;

        Mientras unNumero ¿? 10 Hacer
                ¿ESTÁ BIEN EL ORDEN DE LAS SENTENCIAS?
                unNumero = unNumero + 1;
                resultado = resultado + unNumero;
        FinMientras

        Escribir "El resultado de la suma es: ", resultado;
FinAlgoritmo
```

Resultado del código:

```
Algoritmo Solucion17_bis_bis
        Definir unNumero, resultado Como Real;

        //Introducimos los números para realizar la cuenta
        Escribir "Introduce un número: ";
        Leer unNumero;

        //Inicializamos la variable de salida del resultado
        resultado = 0;
```

```
        Mientras unNumero <= 10 Hacer
                resultado = resultado + unNumero;
                unNumero = unNumero + 1;
        FinMientras

        Escribir "El resultado de la suma es: ", resultado;
FinAlgoritmo
```

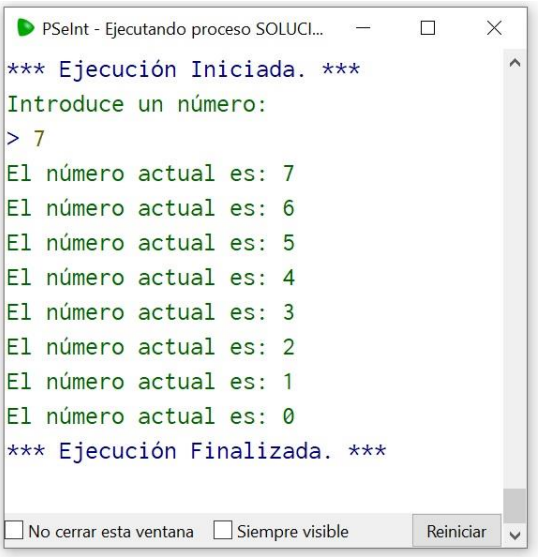
Resultado de la ejecución para el número de entrada: 7



Solución Ejercicio 18: Contar desde un número menor que 10 hasta 0

```
Algoritmo Solucion18
        Definir unNumero Como Real;
        Definir contador Como Entero;

        //Introducimos los números para realizar la cuenta
        Escribir "Introduce un número: ";
        Leer unNumero;

        Si unNumero > 10 Entonces
                Escribir "Error: el número introducido ", unNumero,
                         " es mayor de 10"
        SiNo
                //Inicializamos la variable de salida del resultado
                contador = unNumero;

                Mientras contador >= 0 Hacer
                        Escribir "El número actual es: ", contador;
                        contador = contador - 1;
                FinMientras
        FinSi
FinAlgoritmo
```

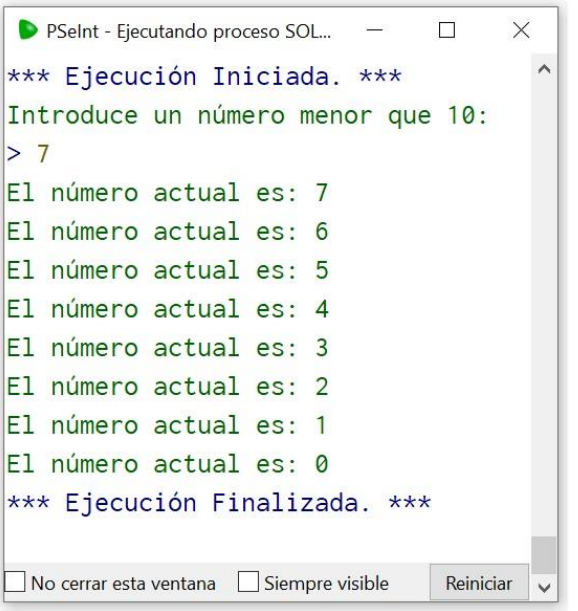
Resultado para el valor introducido: 7



Otra posible solución es la siguiente:

```
Algoritmo Solucion18_bis
        Definir unNumero Como Real;
        Definir contador Como Entero;

        //Introducimos los números para realizar la cuenta
        Escribir "Introduce un número menor que 10: ";
        Leer unNumero;

        //Inicializamos la variable de salida del resultado
        contador = unNumero;

        Mientras contador >= 0 Y unNumero <= 10 Hacer
                Escribir "El número actual es: ", contador;
                contador = contador - 1;
        FinMientras

FinAlgoritmo
```

Resultado para el valor introducido: 7



Como puede comprobar, hemos modificado el mensaje al usuario de entrada de datos. Y, en la condición del bucle, hemos añadido "Y unNumero <= 10". Este "<=" impide que se entre en el bucle si el número introducido es mayor que 10. Por eso hay que modificar el mensaje de la introducción del número, para que el usuario no se equivoque, ya que hemos quitado la sentencia Si que avisaba al usuario, mediante una sentencia Escribir, de su error al introducir un número mayor que 10.

Pero, ¿cuál es la consecuencia de esta nueva condición del bucle Mientras?.

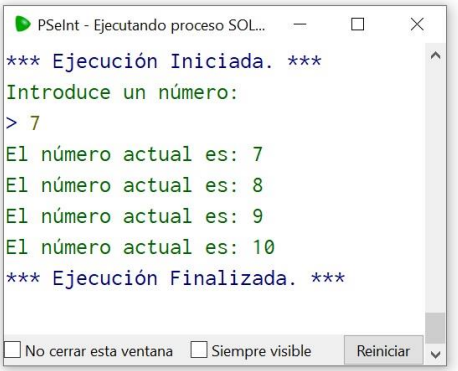
Esa condición del bucle tiene como consecuencia que se evalúe, como mucho diez veces más, la condición de "unNumero <= 10". Porque el bucle va a evaluarla cada vez que intente ejecutarse. Sin embargo, en la primera solución, solamente habremos evaluado la condición de la sentencia Si, una única vez.
Es decir, la segunda solución es más lenta que la primera, porque la condición de menor de 10 se evalúa SIEMPRE que se ejecute el bucle, por el contrario, la solución primera solamente ejecuta la condición del Si una única vez.

Cuando programamos un algoritmo también hay que tener en cuenta el rendimiento del código: cuánto tiempo tarda en ejecutarse. Para saber esto, hay que analizar las sentencias usadas en el código fuente, a esto se le denomina **Analizar la Complejidad del Algoritmo**. Pero también, hay que contar con los datos que trabaja el programa cuando esté entregado al usuario final. Es decir, de todo programa, se necesita someterlo a unas **Pruebas de Rendimiento**, también llamadas Pruebas de Estrés. Es decir, si el programa es capaz de procesar, en un tiempo razonable, el volumen de datos a los que se le va a someter una vez entregada la aplicación al usuario final.

Solución Ejercicio 19: Contar desde un número hasta 10 con un bucle Repetir

```
Algoritmo Solucion19
        Definir unNumero Como Real;
        //Introducimos el valor numérico para realizar la cuenta
        Escribir "Introduce un número: ";
        Leer unNumero;

        Repetir
                Escribir "El número actual es: ", unNumero;
                unNumero = unNumero + 1;
        Hasta Que unNumero > 10

FinAlgoritmo
```

Resultado para el valor introducido: 7



Lo primero que debe de haber notado es que la condición de finalización del bucle ha cambiado. Ya no es "<=" que 10; sino ">" que 10. Esto es así porque, como ya sabe, el bucle Mientras se ejecuta mientras que la condición es cierta (verdadera: mientras se cumple). Es decir, se ejecuta hasta que la condición es falsa. Por el contrario, el bucle Repetir se ejecuta hasta que la condición es verdadera (se cumple). Es decir, el bucle Repetir se ejecuta mientras que la condición es falsa (no se cumple).

Pero, ¿no le ha llamado la atención algo más?.

Debería haber notado, incluso en las pruebas que debe haber realizado que, si el número introducido es mayor que 10, TAMBIÉN se ejecuta el bucle Repetir. Esto es incorrecto, porque las especificaciones del enunciado nos informan explícitamente, en las restricciones del algoritmo, que el número introducido debe verificar el programador que es menor de 10 (se trata de sacar todos los números consecutivos, menores de 10, terminando en el número 10).

Luego la solución correcta al enunciado es:

```
Algoritmo Solucion19_bis
        Definir unNumero Como Real;
        //Introducimos el valor numérico para realizar la cuenta
        Escribir "Introduce un número: ";
        Leer unNumero;

        Si unNumero > 10 Entonces
                Escribir "El número introducido debe ser menor de 10";
        SiNo
                Repetir
                        Escribir "El número actual es: ", unNumero;
                        unNumero = unNumero + 1;
                Hasta Que unNumero > 10
        FinSi
FinAlgoritmo
```

Con esta solución, evitamos que nos muestre un valor cuando el número introducido es mayor que 10 (porque nos obligan, en las restricciones del enunciado, a usar un bucle Repetir) y cumplimos con la restricción que como programadores debemos validar que el número introducido no sea mayor que 10, porque utilizamos una sentencia Escribir donde imprimimos el error del usuario, después de la sentencia Si.

Ejercicio 20: Lectura de líneas de pantalla y concatenarlas en un texto

```
Algoritmo Solucion20
        Definir linea, unTexto Como Cadena;

        //Inicializamos la variable que tiene el resultado del algoritmo
        unTexto = ""; //Inicializamos la variable al valor "Cadena Vacía"

        Repetir
                Escribir "Introduce una línea de texto: ";
                Leer linea;

                Si linea <> "exit" Entonces
                        unTexto = CONCATENAR(unTexto, linea);
                FinSi

                Borrar Pantalla;

        Hasta Que linea = "exit"

        Escribir "Las líneas introducidas corresponden al texto: ", unTexto;
FinAlgoritmo
```

Vamos a introducir por pantalla la secuencia de líneas:

lunes
martes
miércoles
exit

Resultado:

En el anterior algoritmo hay dos errores:

1.- Las palabras de las líneas, que hemos introducido, no están separadas por un espacio en blanco.

2.- Si el usuario teclea cualquier cosa diferente de "exit", por ejemplo "Exit", no podría salir del bucle Repetir. Porque "Exit" no es igual a "exit" como hemos codificado en la condición de finalización del bucle Repetir.

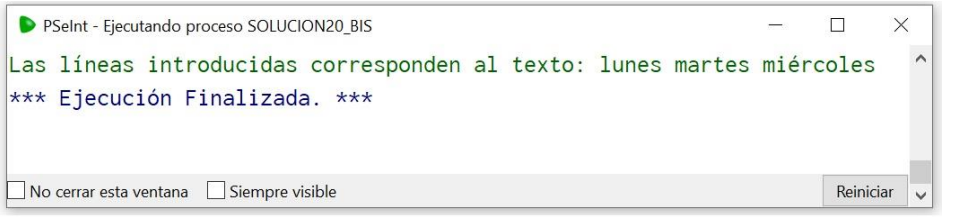
El algoritmo corregido con las anteriores indicaciones sería:

```
Algoritmo Solucion20_bis
        Definir linea, lineaConEspacio, unTexto Como Cadena;

        //Inicializamos la variable que tiene el resultado del algoritmo
        unTexto = ""; //Inicializamos la variable al valor "cadena vacía"

        Repetir
                Escribir "Introduce una línea de texto: ";
                Leer linea;

                Si MAYUSCULAS(linea) <> MAYUSCULAS("exit")
                Entonces
                        //Introducimos un espacio en blanco al final
                        //de la línea introducida para separar el texto

                        lineaConEspacio = CONCATENAR(linea, " ")
                        unTexto = CONCATENAR(unTexto, lineaConEspacio);
                FinSi
                Borrar Pantalla;

        Hasta Que MAYUSCULAS(linea) = MAYUSCULAS("exit")

        Escribir "Las líneas introducidas corresponden al texto: ", unTexto;
FinAlgoritmo
```

Vamos a introducir por pantalla la secuencia de líneas:

lunes
martes
miércoles
Exit

El resultado es:



Fíjese que hemos utilizado una nueva variable "lineaConEspacio", le sugiero que la cambie por la variable "línea" y comprobará que el resultado del "texto", se convierte o bien en un texto con un espacio en blanco al principio del "texto" o bien con un espacio en blanco al final del texto.

Practique con este código.

Pero, además, el utilizar la variable "línea" produce otro error: cuando convirtamos el código en pseudocódigo a un lenguaje de programación "real", podemos obtener un comportamiento erróneo del programa, porque la función predefinida MAYUSCULAS de PSeInt elimina los espacios en blanco de la variable "línea", pero al convertir al lenguaje "real", no sabemos cómo lo va a convertir el programa PSeInt en cada uno de los lenguajes reales.

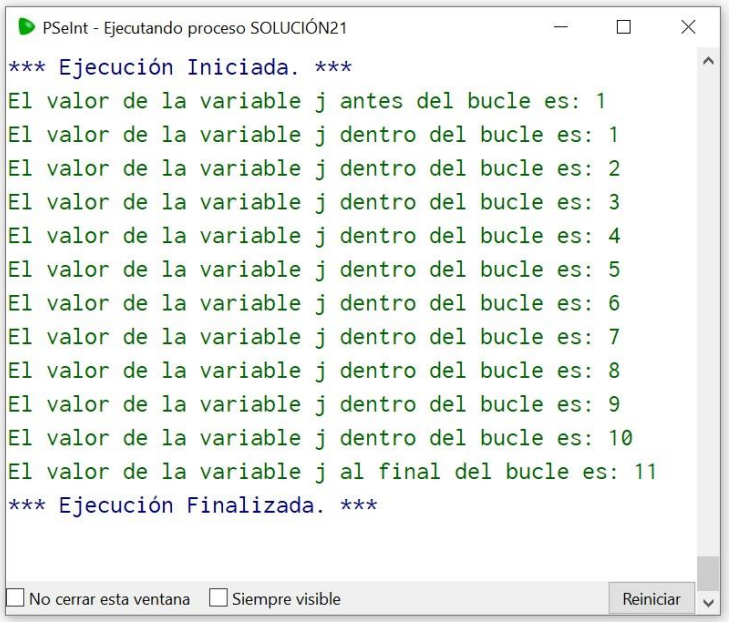
Practique con la conversión del pseudocódigo a un lenguaje de programación real, utilizando el menú **Archivo** en la opción **Exportar**.

Solución Ejercicio 21: Seguimiento de código con bucle Para

```
Algoritmo Solución21
    //se define una variable de tipo entero
    //para contener el número de veces que se ejecutará el bucle.
    Definir i Como Entero;
    //se define una variable de tipo entero
    //para contener el resultado de la ejecución.
    Definir j Como Entero;
    j = 1;
    Escribir "El valor de la variable j antes del bucle es: ", j;
```

```
    Para i = 1 Hasta 10 Hacer
        Escribir "El valor de la variable j dentro del bucle es: ", j;
        j = j + 1;
    FinPara //fin del bucle Para

    Escribir "El valor de la variable j al final del bucle es: ", j;

FinAlgoritmo
```

Resultado:

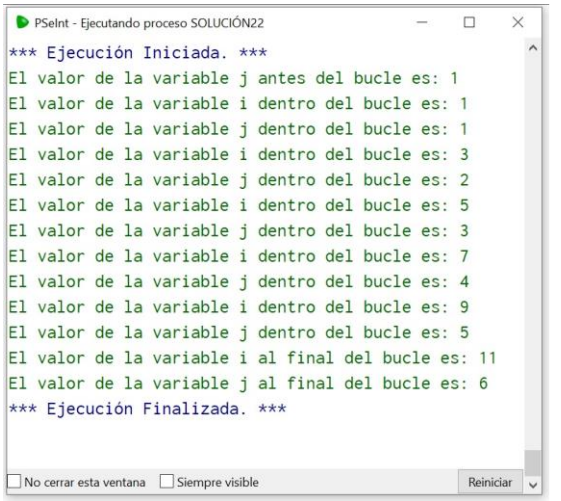


Solución Ejercicio 22: Segundo seguimiento de código con bucle Para

```
Algoritmo Solución22
    //se define una variable de tipo entero
    //para contener el número de veces que se ejecutará el bucle.
    Definir i Como Entero;
    //se define una variable de tipo entero
    //para contener el resultado de la ejecución.
    Definir j Como Entero;
    j = 1;
    Escribir "El valor de la variable j antes del bucle es: ", j;
```

```
        Para i = 1 Hasta 10 Hacer
                Escribir "El valor de la variable i dentro del bucle es: ", i;
                i = i + 1;
                Escribir "El valor de la variable j dentro del bucle es: ", j;
                j = j + 1;
        FinPara //fin del bucle Para

        Escribir "El valor de la variable i al final del bucle es: ", i;
        Escribir "El valor de la variable j al final del bucle es: ", j;
FinAlgoritmo
```

Resultado:

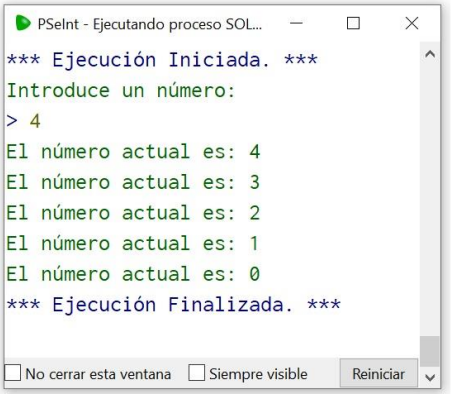


Verifique que el bucle se ha ejecutado solamente 5 veces, porque hemos modificado el valor de la variable contador de iteración del bucle Para, dentro del propio bucle: variable "i".

Solución Ejercicio 23: Contar desde un número hasta 0 con un bucle Para

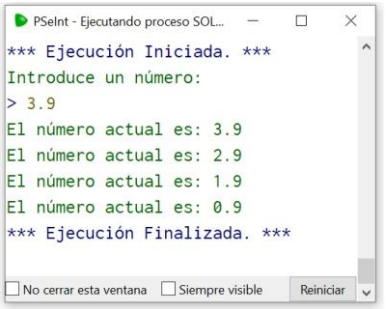
```
Algoritmo Solucion23
        Definir unNumero Como Real;
        Definir contador Como Entero;
        //Introducimos el número para realizar la cuenta atrás
        Escribir "Introduce un número: ";
        Leer unNumero;

        Para contador = unNumero Hasta 0 Con Paso -1 Hacer
                Escribir "El número actual es: ", contador;
        FinPara
FinAlgoritmo
```

Resultado:



Pero, ¿qué ocurre cuando introducimos un número de tipo real?.

Resultado:



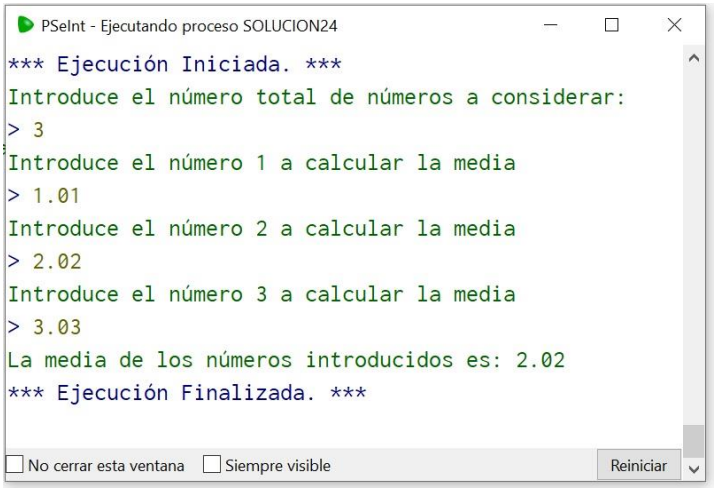
El programa funciona, pero hay que fijarse que el bucle se ejecuta una vez menos que con un número entero.

Solución Ejercicio 24: Calcular la media de varios números

```
Algoritmo Solucion24
        Definir unNumero, resultado Como Real;
        Definir totalNumeros, contador Como Entero;

        resultado = 0;
        //Introducimos la cantidad de números a calcular la media
        Escribir "Introduce el número total de números a considerar: ";
        Leer totalNumeros;
```

```
    Para contador = 1 Hasta totalNumeros Con Paso 1 Hacer
        Escribir "Introduce el número ", contador, " a calcular la media";
        Leer unNumero;

        resultado = resultado + unNumero;
    FinPara

    resultado = resultado / totalNumeros;
    Escribir "La media de los números introducidos es: ", resultado;

FinAlgoritmo
```

Resultado:



Solución Ejercicio 25: Seguimiento de código con bucles anidados

La ejecución produce un bucle infinito.

PISTA: Cuando se trabaja con **bucles anidados**, hay que tener dos precauciones:

→ Comprobar el final de la condición de finalización de los bucles, para que no se conviertan en bucles infinitos.

→ Cuando los bucles modifican variables que intervienen en la condición de finalización de los bucles, hay que **validar** los valores que tomarán las variables porque estos valores pueden provocar que la ejecución de un bucle se convierta en un **bucle infinito**.

Solución Ejercicio 26: Bucles anidados

```
Algoritmo Solucion26
        //se define una variable de tipo entero
        //para el número del día de la semana.
        Definir numeroDiaSemana Como Entero;
        //para el contador que se va incrementando
        //con el día de la semana.
        Definir contador Como Entero;
        //se escribe en pantalla la pregunta del valor numérico.
        //mensaje al usuario para avisarle que se sale del bucle
        //cuando se introduce un número igual a 0
        Escribir "Teclee el número 0 para terminar la ejecución";
        Escribir "En qué número de día de la semana empezamos:";
        //se recoge un número de día de la semana por pantalla.
        Leer numeroDiaSemana;

        Mientras numeroDiaSemana <> 0  Hacer

            Para contador = numeroDiaSemana Hasta 7 Hacer

                        Segun contador Hacer
                                1: Escribir "Lunes";
                                2: Escribir "Martes";
                                3: Escribir "Miércoles";
                                4: Escribir "Jueves";
                                5: Escribir "Viernes";
                                6: Escribir "Sábado";
                                7: Escribir "Domingo";
                        FinSegun //fin de la instrucción Según

                FinPara //fin del bucle Para

            //se escribe en pantalla la pregunta del valor numérico.
            //mensaje al usuario para avisarle que se sale del bucle
            //cuando se introduce un número igual a 0
            Escribir "Teclee el número 0 para terminar la ejecución";
            Escribir "En qué número de día de la semana empezamos:";

            //se recoge un número de día de la semana por pantalla.
            Leer numeroDiaSemana;

        FinMientras //fin bucle Mientras
FinAlgoritmo
```

Efectivamente, como NO siempre se desea que se ejecute el cuerpo del bucle exterior, siempre al menos una vez; se debe cambiar el bucle exterior de un bucle Repetir a un bucle Mientras.

Pero, para que salga el mensaje de aviso de fin de ejecución con el número cero, hay que sacar del bucle Mientras (ponerlo antes del bucle Mientras), las instrucciones del mensaje del "cero para terminar" y la lectura del valor para la condición de finalización del bucle Mientras.

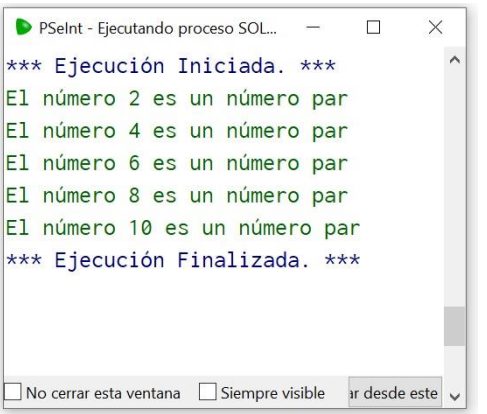
Además, si solamente sacáramos fuera del bucle exterior el mensaje del cero para terminar y la lectura del valor tecleado; esas instrucciones solamente se ejecutarían una vez. Por lo tanto, también hay que meter dentro del bucle las instrucciones de escribir el mensaje de fin de ejecución con el cero y la lectura del valor introducido (sentencia Leer). Advierta el lector que se incluyen estas instrucciones fuera del bucle, justo antes de evaluar la condición de finalización del bucle Mientras y como últimas instrucciones al final del bucle Mientras porque, al ser un bucle, la siguiente instrucción a ejecutar es la evaluación de la condición de final del bucle Mientras, que es

numeroDiaSemana <> 0.

Solución Ejercicio 27: Números pares con un bucle Para

```
Algoritmo Solucion27

        Para i = 2 Hasta 10 Con Paso 2 Hacer
            Escribir "El número ", i, " es un número par";
        FinPara //fin del bucle Para

FinAlgoritmo
```

Resultado:

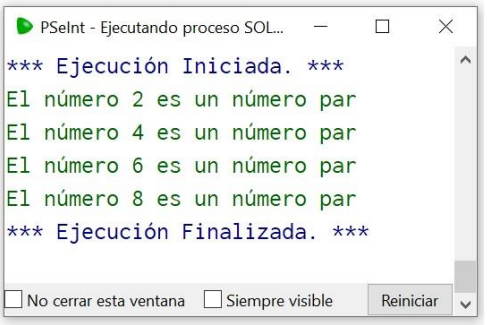


Merece destacarse que no siempre la cláusula "Con Paso" de un bucle Para tiene por qué ser siempre un 1. En éste caso, es un Con Paso" con un incremento de 2 en 2 valores.

Solución Ejercicio 28: Números pares con un bucle Mientras

```
Algoritmo Solucion28
        //se define una variable de tipo entero
        //para el contador de números que se va incrementando.
        Definir contador Como Entero;

        //se inicializa la variable antes de entrar en el bucle.
        //Si inicializásemos la variable dentro del bucle
        //habríamos conseguido hacer un bucle infinito.
        contador = 2;

        Mientras contador <> 10 Hacer

            Escribir "El número ", contador , " es un número par";
            //se incrementa en dos unidades la variable contador.
            contador = contador + 2;
        FinMientras
FinAlgoritmo
```

Resultado:



Todo bucle Para se puede transformar por un bucle Mientras. Esto no ocurre para un bucle Repetir: ya que un bucle Para no se puede sustituir por un bucle Repetir.
Le dejo al lector que lo piense, razonadamente.

Advierta el lector que, si cambiamos por error, la condición de finalización del bucle Mientras a la expresión: contador <> 11

Se habría conseguido hacer un bucle infinito, ya que el valor de la variable "contador", nunca alcanzaría a obtener el valor 11; porque los valores que obtiene la variable "contador" son, sucesivamente, los valores: 2, 4, 6, 8, 10, 12, 14, 16, 18, …

Es decir, la condición de finalización del bucle nunca se volvería cierta (verdadera o true en inglés) al no evaluarse nunca la expresión con un valor 11 como cierta. Y nunca se saldría del bucle (eso es un bucle infinito).

Recuerde el lector, que un bucle Mientras se ejecuta hasta que la condición es falsa (false). Es decir, el bucle se ejecuta mientras que la condición es verdadera. Es decir, MIENTRAS el valor de la variable "contador" sea distinto de 10 (mientras la condición se cumpla), se entra en el bucle y se ejecutan las instrucciones del cuerpo del bucle.

Solución Ejercicio 29: Eliminar la parte decimal de un número entero

El código fuente quedaría como sigue.

```
Algoritmo Solucion29
        Escribir "Al dividir un 5 y un 3, se obtiene: ", TRUNC(5 / 3);
FinAlgoritmo
```

Solución Ejercicio 30: Calcular el resto de una división entre dos números enteros

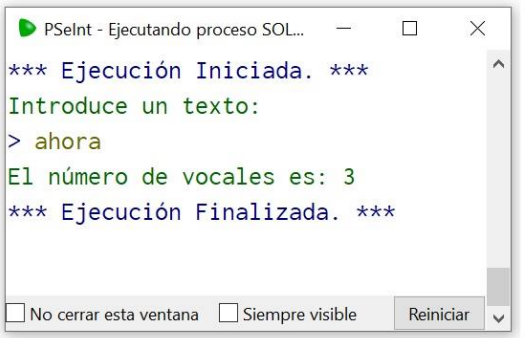
```
Algoritmo Solucion30

        Escribir "Al dividir un 5 y un 3, se obtiene: ", 5 / 3;
        Escribir "El resto de dividir un 5 y un 3, es: ", 5 MOD 3;
FinAlgoritmo
```

Solución Ejercicio 31: Contar vocales de un texto

```
Funcion numeroVocales = ContarVocales( unTexto )

        Definir contadorVocales, longitudTexto, i Como Entero;
        Definir letra Como Caracter;

        // Inicialización de variables
        contadorVocales = 0;
        longitudTexto = LONGITUD(unTexto);
```

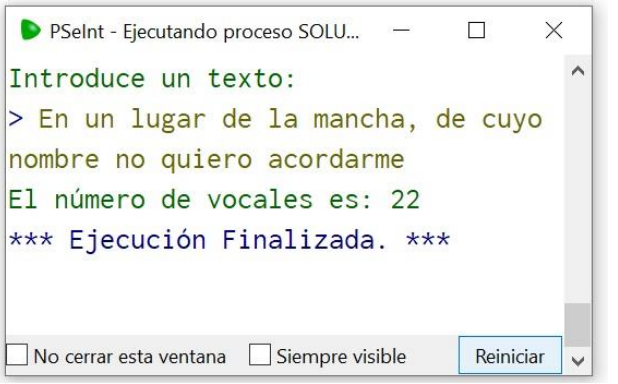
```
        Para i = 1 Hasta longitudTexto Con Paso 1 Hacer
                letra = MAYUSCULAS(SUBCADENA(unTexto,i,i));
                Si letra = MAYUSCULAS("a") O
                   letra = MAYUSCULAS("e") O
                   letra = MAYUSCULAS("i") O
                   letra = MAYUSCULAS("o") O
                   letra = MAYUSCULAS("u") Entonces

                        contadorVocales = contadorVocales + 1;
                FinSi
        FinPara
        //Devolvemos el resultado en la variable de la función
        numeroVocales = contadorVocales;
FinFuncion

Algoritmo Solucion31
        Definir unTexto Como Cadena;

        Escribir "Introduce un texto: ";
        Leer unTexto;

        Escribir "El número de vocales es: ", ContarVocales(unTexto);
FinAlgoritmo
```

Resultado:



Este tipo de prueba nos sirve para saber si un tipo de dato Cadena empieza su primera letra por el índice 0 ó en 1. Como podemos comprobar, las cadenas son como "tablas" que empiezan a numerarse en el índice 1.

El ejemplo del enunciado es: "En un lugar de la Mancha, de cuyo nombre no quiero acordarme".
Si lo usamos para una prueba, el resultado es:



Ahora, vamos a analizar el código ejecutado en la sentencia de asignación:

```
letra = MAYUSCULAS(SUBCADENA(unTexto,i,i));
```

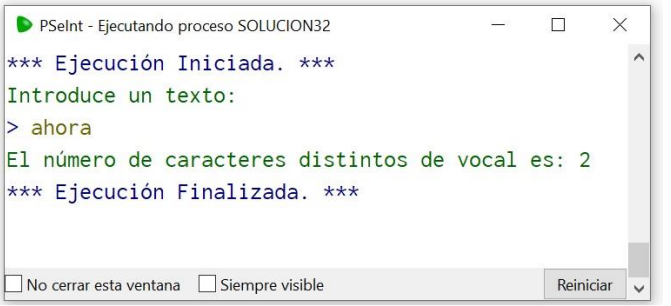
Como puede verificar en las soluciones aportadas, una función predefinida es una subrutina como otra cualquiera: dados unos datos de entrada, devuelve un valor. Por eso, podemos utilizar el resultado de una función predefina, para aplicárselo como entrada a otra función (subrutina); en este caso es también una función predefinida.

Solución Ejercicio 32: Contar los caracteres diferentes a una vocal de un texto

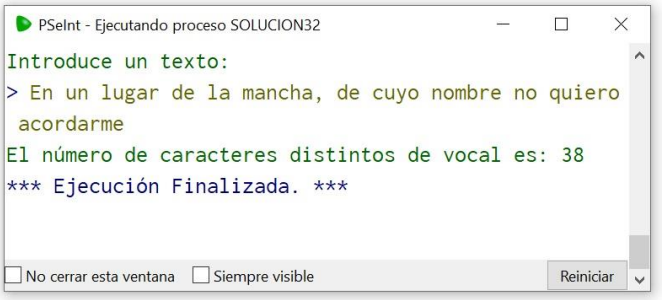
```
Funcion numeroDistintoVocal = ContarDistintoVocales ( unTexto Por Referencia )

        Definir contadorDistintoVocal, longitudTexto, i Como Entero;
        Definir letra Como Caracter;
        // Inicialización de variables
        contadorDistintoVocal = 0;
        longitudTexto = LONGITUD(unTexto);

        Para i = 1 Hasta longitudTexto Con Paso 1 Hacer
                letra = MAYUSCULAS(SUBCADENA(unTexto,i,i));
                Si NO ( letra = MAYUSCULAS("a") O
                   letra = MAYUSCULAS("e") O
                   letra = MAYUSCULAS("i") O
                   letra = MAYUSCULAS("o") O
                   letra = MAYUSCULAS("u")) Entonces
                        contadorDistintoVocal=contadorDistintoVocal + 1;
                FinSi
        FinPara
```

```
        //Devolvemos el resultado en la variable de la función
        numeroDistintoVocal = contadorDistintoVocal;
FinFuncion


Algoritmo Solucion32
        Definir unTexto Como Cadena;

        Escribir "Introduce un texto: ";
        Leer unTexto;

        Escribir "El número de caracteres distintos de vocal es: ",
                ContarDistintoVocales(unTexto);
FinAlgoritmo
```

Hacemos las mismas pruebas para comprobar los resultados.
Resultado con la palabra: ahora

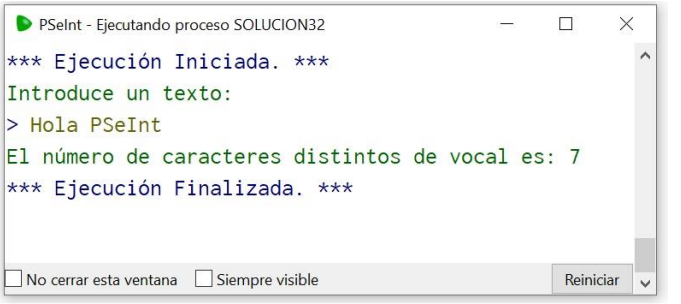


Resultado con el texto del ejemplo propuesto en el enunciado.



Quizás le haya sorprendido que haya dado como resultado un número igual a 38. Ese es el resultado correcto porque se cuentan todos los caracteres que sean distintos de vocal; y por supuesto el carácter de espacio en blanco es un carácter diferente a una vocal; y, por supuesto, el carácter de la coma también es un carácter diferente a una vocal del castellano.

Veamos en el siguiente resultado, cómo se tienen en cuenta los espacios en blanco:



ojo

Le recomiendo que no utilice la variable de retorno del valor de la Función como variable intermedia de contador o de almacenamiento temporal de un valor. En programa estructurada le recuerdo que todo programa o subprograma, tiene una entrada de datos, un procesamiento de los datos y una salida. Luego, la salida es única.

Como ha podido comprobar, básicamente el algoritmo es el mismo código, pero negando la condición de la sentencia Si.

Pero, ¿es necesario reescribir toda la función del ejercicio 31 en el ejercicio 32?. Pruebe, el siguiente código.

```
Funcion numeroVocales = ContarVocales( unTexto )

        Definir contadorVocales, longitudTexto, i Como Entero;
        Definir letra Como Caracter;
        // Inicialización de variables
        contadorVocales = 0;
        longitudTexto = LONGITUD(unTexto);

        Para i = 1 Hasta longitudTexto Con Paso 1 Hacer
                letra = MAYUSCULAS(SUBCADENA(unTexto,i,i));
                Si letra = MAYUSCULAS("a") O
                   letra = MAYUSCULAS("e") O
                   letra = MAYUSCULAS("i") O
                   letra = MAYUSCULAS("o") O
                   letra = MAYUSCULAS("u") Entonces

                        contadorVocales = contadorVocales + 1;
                FinSi
        FinPara
```

```
        //Devolvemos el resultado en la variable de la función
        numeroVocales = contadorVocales;

FinFuncion


Algoritmo Solucion32_bis
        Definir unTexto Como Cadena;

        Escribir "Introduce un texto: ";
        Leer unTexto;

        Escribir "El número de caracteres distintos de vocal es: ",
                (LONGITUD(unTexto) - ContarVocales(unTexto));
FinAlgoritmo
```

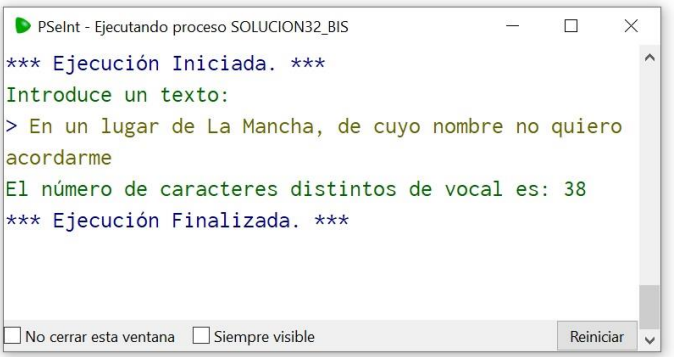
Es decir, dejamos la función sin modificar el contador de las vocales y realizamos la operación del total de caracteres en el texto, restando el número de vocales.

Con esto, nos ahorramos dos cosas:

→ Tener que reescribir toda la función para, simplemente, añadirla un NO en la condición de la sentencia Si.

→ El mantenimiento de otra función que no aporta ninguna funcionalidad nueva al conjunto de funcionalidades que ya tenemos en otras funciones codificadas por nosotros.

Comprobemos el resultado, para los mismos valores de entrada.

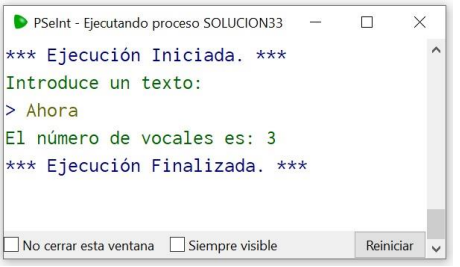


Ejercicio 33: Contar recursivamente las vocales de un texto

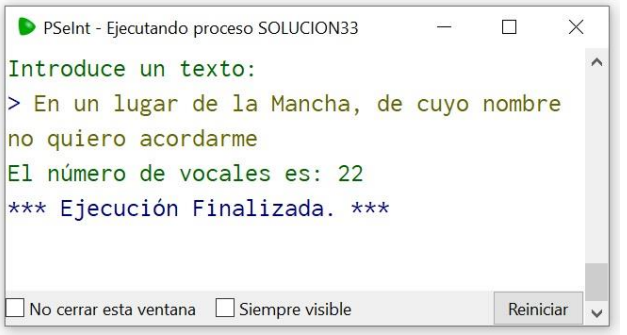
```
Funcion numeroVocales = ContarVocales( unTexto Por Valor)
        //Recibimos los caracteres del texto en letras mayúsculas

        Definir contadorVocales, longitudTexto Como Entero;
        Definir letra Como Caracter;
        // Inicialización de variables
        contadorVocales = 0;
        longitudTexto = LONGITUD(unTexto);

        Si longitudTexto = 1 Entonces
                //Asignamos la condición de finalización de la recursividad
                Si (unTexto = "A") O (unTexto = "E") O
                   (unTexto = "I") O (unTexto = "O") O (unTexto = "U")
                Entonces
                        contadorVocales = 1;
                FinSi
        SiNo
                //Verificamos únicamente la primera letra del texto
                letra = SUBCADENA(unTexto,1,1);
                Si (letra = "A") O (letra = "E") O (letra = "I") O
                   (letra = "O") O (letra = "U")
                Entonces

                        contadorVocales = contadorVocales + 1;
                FinSi

                //Realizamos la llamada recursiva,
                //eliminando del texto la primera letra
                contadorVocales = contadorVocales +
                ContarVocales(SUBCADENA(unTexto, 2, longitudTexto));
        FinSi

        //Devolvemos el resultado en la variable de la función
        numeroVocales = contadorVocales;

FinFuncion

Algoritmo Solucion33
        Definir unTexto Como Cadena;

        Escribir "Introduce un texto: ";
        Leer unTexto;

        //Realizamos la llamada a la función
        //con el parámetro de entrada Por Valor
        //y con sus caracteres en mayúsculas
```

```
        Escribir "El número de vocales es: ",
                 ContarVocales(MAYUSCULAS(unTexto));
FinAlgoritmo
```

Resultado con la palabra: Ahora



Resultado con el texto de ejemplo del enunciado:



Hemos puesto una restricción o condicionante, al parámetro de entrada de la función recursiva: los caracteres del texto tienen que venir en mayúsculas.
Esto nos aumenta el rendimiento (es una solución más eficiente) de la ejecución del algoritmo respecto al que se ejecuta en el ejercicio 31 porque cada carácter se pasa a mayúsculas una sola vez, en vez de hacer cinco llamadas a la función predefinida MAYUSCULAS por cada letra del texto para compararla con una vocal. Por lo tanto, al hacer menos llamadas, el algoritmo se ejecuta más rápido.

La función SUBCADENA no funciona, da error de sintaxis, cuando la intentamos usar para un parámetro de entrada Por Referencia. Por lo tanto, el texto solamente se puede pasar a la función Por Valor.

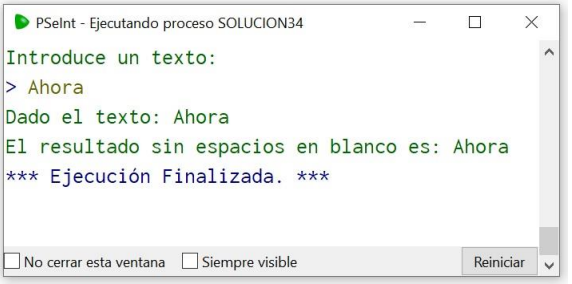
Solución Ejercicio 34: Eliminar recursivamente los espacios en blanco de un texto

```
Funcion blancosEliminados = EliminarBlancos( unTexto Por Valor)

        Definir longitudTexto Como Entero;
        Definir sinBlancos Como Cadena;
        Definir BLANCO, letra Como Caracter;

// Inicialización de variables
BLANCO = " "; //Definimos una constante de un carácter en blanco:un espacio en blanco
longitudTexto = LONGITUD(unTexto);


Si longitudTexto = 1 Entonces
        //Asignamos la condición de finalización de la recursividad
        Si unTexto = BLANCO Entonces
                //Es un espacio un blanco,
                //entonces lo borramos del texto
                sinBlancos = ""; //Asignamos la cadena vacía
        SiNo
                //Es un carácter distinto de un espacio un blanco,
                //entonces lo "devolvemos" al texto
                sinBlancos = unTexto;
        FinSi
SiNo

        //Verificamos únicamente la primera letra del texto
        letra = SUBCADENA(unTexto,1,1);
        Si letra = BLANCO Entonces
                //Realizamos la llamada recursiva,
                //eliminando del texto la primera letra
                //porque es un espacio en blanco
                sinBlancos = EliminarBlancos(
                                        SUBCADENA(unTexto, 2,
                                                        longitudTexto));
        SiNo
                //La letra es un carácter distinto de un espacio en blanco,
                //entonces lo "devolvemos" al texto
                //concatenándolo con el resto del texto pero quitándolo
                //de la llamada recursiva porque ya hemos visto
                //que es distinto de un espacio en blanco
                sinBlancos =
                        CONCATENAR(letra,
                        EliminarBlancos(SUBCADENA(unTexto,
                                                        2,
                                                        longitudTexto)));
        FinSi
FinSi
```

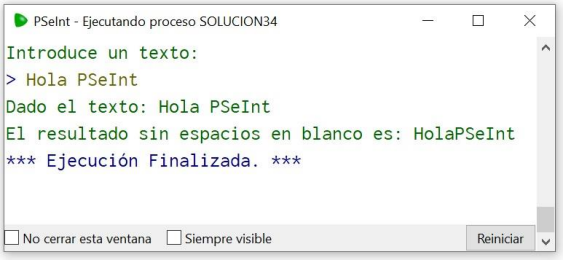
```
        //Devolvemos el resultado en la variable de la función
        blancosEliminados = sinBlancos;
FinFuncion

Algoritmo Solucion34
        Definir unTexto Como Cadena;
        Escribir "Introduce un texto: ";
        Leer unTexto;

        //Realizamos la llamada a la función
        //con el parámetro de entrada Por Valor
        Escribir "Dado el texto: ", unTexto;
        Escribir "El resultado sin espacios en blanco es: ", EliminarBlancos(unTexto);
FinAlgoritmo
```

Aplicamos la Batería de Pruebas:

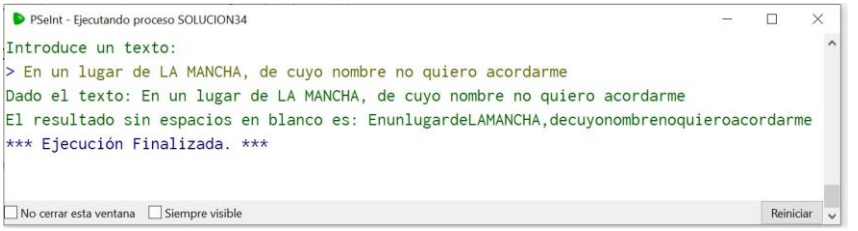
1.- Sin ningún dato: **Cero Valores**. Es decir, sin espacios en blanco.
Resultado:



2.- Con un dato: **Valores Esperados**. Es decir, con un espacio en blanco.
Resultado:



3.- Con muchos datos: **N Valores**. Es decir, con un número grande de espacios en blanco.



Solución Ejercicio 35: Encontrar recursivamente si existe un carácter de espacio en blanco en un texto

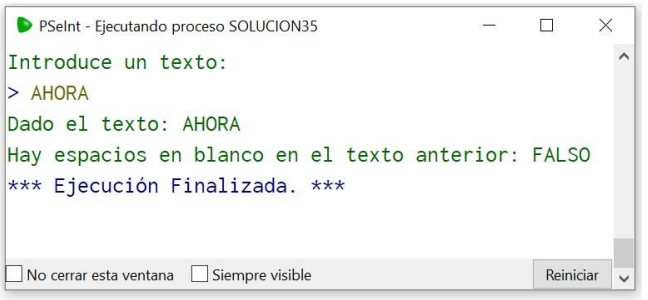
```
Funcion blancosEncontrados = HayBlancos( unTexto Por Valor)
        Definir longitudTexto Como Entero;
        Definir esBlanco Como Logico;
        Definir BLANCO, letra Como Caracter;

// Inicialización de variables
BLANCO = " "; //Definimos una constante
longitudTexto = LONGITUD(unTexto);

Si longitudTexto = 1 Entonces
        //Asignamos la condición de finalización de la recursividad
        Si unTexto = BLANCO Entonces
                //Es un espacio en blanco,
                //entonces lo borramos del texto
                esBlanco = VERDADERO; //Asignamos el valor cierto
        SiNo
                //Es un carácter distinto de un espacio en blanco,
                //entonces devolvemos el valor falso
                esBlanco = FALSO;
        FinSi
SiNo
        //Verificamos únicamente la primera letra del texto
        letra = SUBCADENA(unTexto,1,1);
        Si letra = BLANCO Entonces
                //Hay un espacio en blanco,
                //entonces abortamos las llamadas recursivas
                esBlanco = VERDADERO;
        SiNo
                esBlanco = HayBlancos(SUBCADENA(unTexto, 2, longitudTexto));
        FinSi

FinSi
        //Devolvemos el resultado en la variable de la función
        blancosEncontrados = esBlanco;
FinFuncion
```

```
Algoritmo Solucion35
	Definir unTexto Como Cadena;

	Escribir "Introduce un texto: ";
	Leer unTexto;

	//Realizamos la llamada a la función
	//con el parámetro de entrada Por Valor
	Escribir "Dado el texto: ", unTexto;
	Escribir "Hay espacios en blanco en el texto anterior: ", HayBlancos(unTexto);
FinAlgoritmo
```

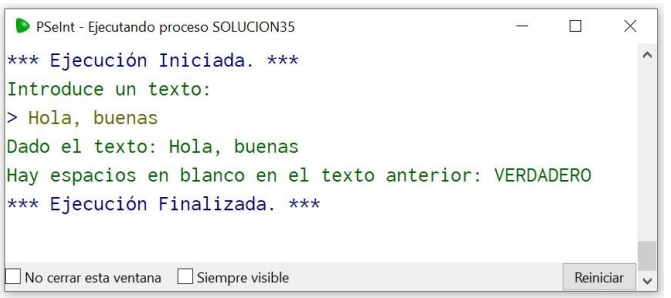
Aplicamos la Batería de Pruebas:

1.- Sin ningún dato: **Cero Valores**. Es decir, sin espacios en blanco.

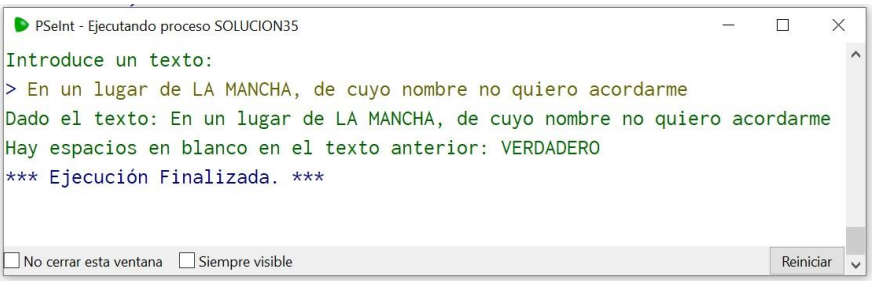
Resultado:



2.- Con un dato: **Valores Esperados**. Es decir, con un espacio en blanco.
Resultado:



3.- Con muchos datos: **N Valores**. Es decir, con un número grande de espacios en blanco.



Se ha demostrado que el algoritmo funciona perfectamente.

Vamos a analizar la solución. Este código fuente recursivo es completamente diferente a los anteriores. En los otros, se recorrían todos los caracteres del texto uno a uno hasta el final. Pero en este código, ante la primera aparición del primer espacio en blanco en el texto, las llamadas recursivas se detienen, se devuelve el valor a las anteriores llamadas recursivas y, por lo tanto, no hace falta llegar al final del recorrido de todos y cada uno de los caracteres del texto, porque es una "función lógica" (booleana: verdadero o falso): o bien el texto tiene un espacio en blanco (con uno basta) o bien no tiene un espacio en blanco en todo el texto.

Pues todo aclarado, ¿no?.
Pues no.

De hecho, es un pésimo algoritmo ya que esta solución recursiva nunca de debió resolver con una función recursiva.
Un algoritmo iterativo era más fácil de implementar (codificar) y, además, consume muchos menos recursos de máquina en cuestiones de uso de memoria (por las llamadas recursivas reiterativas) y es mucho más lento de ejecución (tiene peor rendimiento al ejecutarse, considerando el tiempo de resolución del problema por usar la recursividad como solución algorítmica).

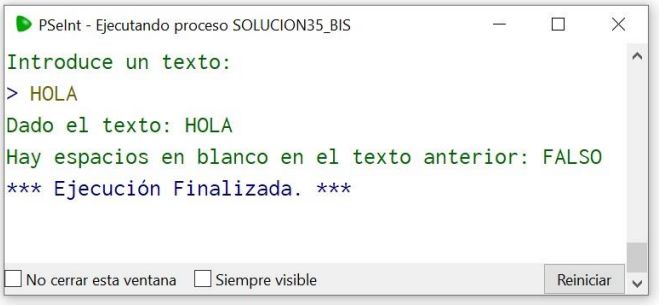
Una posible solución iterativa correcta, sería:

```
Algoritmo Solucion35_bis
        Definir unTexto Como Cadena;
        Definir longitudTexto Como Entero;
        Definir esBlanco Como Logico;
        Definir BLANCO, letra Como Caracter;
        //Solicitamos los datos de entrada al usuario
        Escribir "Introduce un texto: ";
        Leer unTexto;

        // Inicialización de variables
        contador = 1; //Las cadenas de texto empiezan en el valor 1
        esBlanco = FALSO;
        BLANCO = " "; //Definimos una constante
        longitudTexto = LONGITUD(unTexto);
```

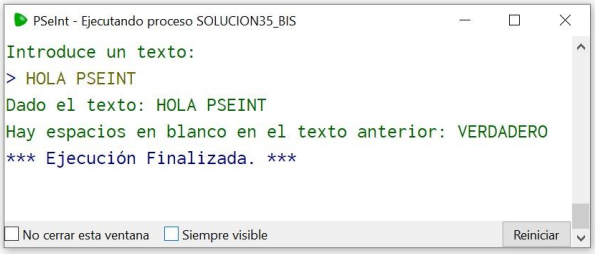
```
        Repetir
                letra = SUBCADENA(unTexto,contador,contador);
                Si letra = BLANCO Entonces
                        esBlanco = VERDADERO;
                SiNo
                        contador = contador + 1;
                FinSi

        Hasta Que contador = longitudTexto O esBlanco = VERDADERO

        Escribir "Dado el texto: ", unTexto;
        Escribir "Hay espacios en blanco en el texto anterior: ", esBlanco;
FinAlgoritmo
```

Aplicamos la Batería de Pruebas:

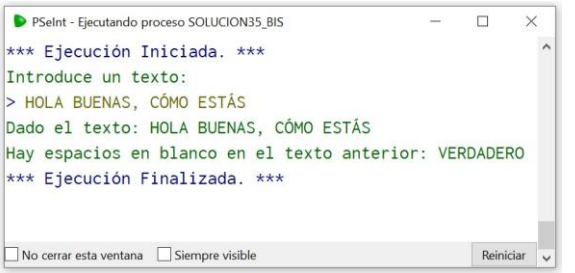
1.- Sin ningún dato: **Cero Valores**. Es decir, sin espacios en blanco.
Resultado:



2.- Con un dato: **Valores Esperados**. Es decir, con un espacio en blanco.
Resultado:



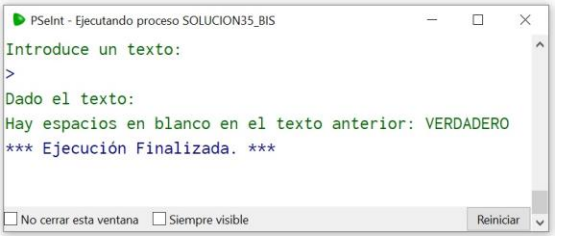
3.- Con muchos datos: **N Valores**. Es decir, con un número grande de espacios en blanco.



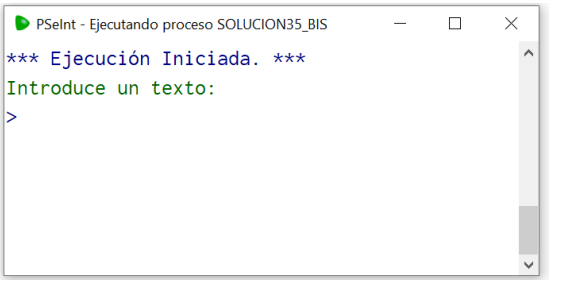
Se ha demostrado que el algoritmo funciona perfectamente. Y es más eficiente, porque consume menos recursos del ordenador, como la memoria; y, es más rápido al ser iterativo.

Pues todo aclarado, ¿no?.
Pues no.

Y, ¿qué pasa con las pruebas de los **Valores Límites**?.
Probemos el algoritmo anterior, introduciendo un único espacio en blanco en el texto.
El resultado es:



Probemos el algoritmo anterior, introduciendo la cadena vacía, es decir sin ningún carácter, por lo tanto, sin ningún espacio en blanco ni de otro tipo en el texto.
El resultado es:



Se ha colgado el programa. Hemos obtenido un **bucle infinito**.

Bienvenido al mundo del programador informático.

¿Cómo se soluciona?.

PISTA: Al tener el algoritmo un bucle, seguramente es una evaluación incorrecta de la condición de finalización del bucle, que lo transforma en un bucle infinito.

¡Efectivamente!.

Pruebe a cambiar la condición del bucle Repetir por la siguiente línea de código:

Hasta Que contador >= longitudTexto O esBlanco = VERDADERO

El error se produce porque ante una cadena vacía, nunca se sale por **esBlanco** = VERDADERO, porque nunca es igual a BLANCO. Y **contador** tiene el valor 1, 2, 3, 4, etc. Y, sin embargo, **longitudTexto** es siempre igual a 0. Porque la cadena vacía tiene una longitud de 0 caracteres. Por lo tanto, NUNCA se cumple la condición de finalización del operador relacional "=", ante una cadena vacía

Por supuesto, dejo al lector que vuelva a probar todas y cada una de las condiciones de la Batería de Pruebas que hemos realizado, incluyendo las pruebas de los Valores Límite.

Dentro de los valores límite, yo incluiría las dos pruebas siguientes:

1.- Un Texto que empieza por un único espacio en blanco. Por ejemplo:
" hola"

2.- Un Texto que termina por un único espacio en blanco. Por ejemplo:
"hola "

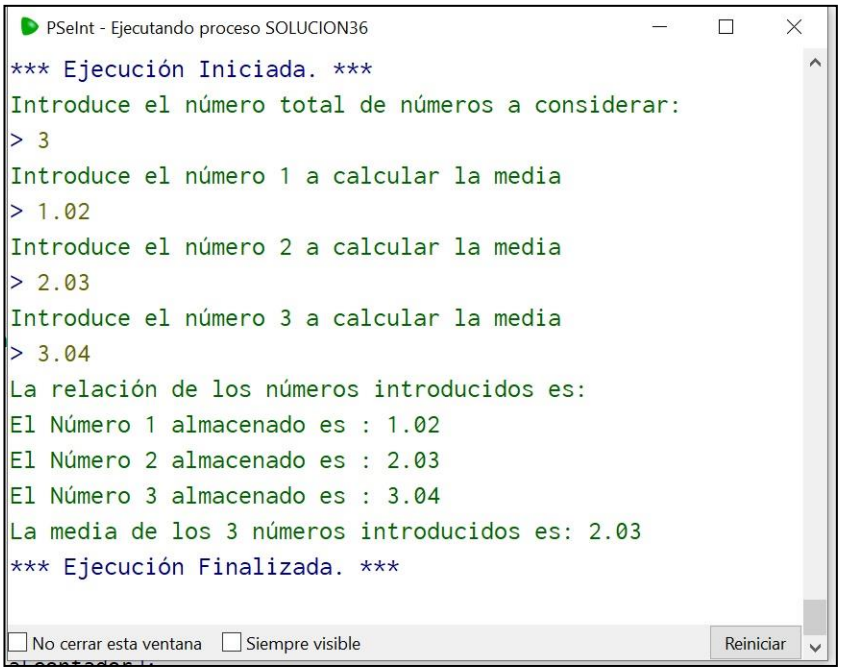
Por supuesto, la prueba es sin utilizar el carácter " que se usa únicamente para delimitar la posición del espacio en blanco, en el texto escrito de esta obra.

Aunque es cierto que la solución recursiva es la única para ciertos problemas; y que, algunas veces es la más fácil de implementar para resolver algún problema. Es la última opción, por la que debemos optar, para resolver un problema algorítmico.

Solución Ejercicio 36: Calcular la media de varios números, almacenando los números

```
Algoritmo Solucion36
        Definir unNumero, resultado Como Real;
        Definir totalNumeros, contador Como Entero;

        resultado = 0;
        //Introducimos la cantidad de números a calcular la media
        Escribir "Introduce el número total de números a considerar: ";
        Leer totalNumeros;

        //Definimos la dimensión variable de la tabla. Es decir, es una Tabla Dinámica
        Dimension tabla[totalNumeros];

        Para contador = 1 Hasta totalNumeros Con Paso 1 Hacer
                Escribir "Introduce el número ", contador, " a calcular la media";
                Leer unNumero;
                tabla[contador] = unNumero;

                resultado = resultado + unNumero;
        FinPara

        resultado = resultado / totalNumeros;

        Escribir "La relación de los números introducidos es: ";
        Para contador = 1 Hasta totalNumeros Hacer
                Escribir "El Número ", contador, " almacenado es : ", tabla[contador];
        FinPara

        Escribir "La media de los ", totalNumeros, " números introducidos es: ",
                 resultado;

FinAlgoritmo
```

Resultado:



En esta solución hemos definido el tamaño de una tabla en tiempo de ejecución: mientras el programa se ejecuta por el usuario. De forma que, en cada ejecución del programa, la tabla tomará un tamaño diferente (no fijo): el tamaño que quiera el usuario en cada ejecución. A este tipo de tablas que toman su dimensión en el momento de ejecutar el programa, se las denomina **Tablas Dinámicas**. Porque en el momento de crear el algoritmo (código fuente) no se conoce el valor estático del tamaño de la tabla.

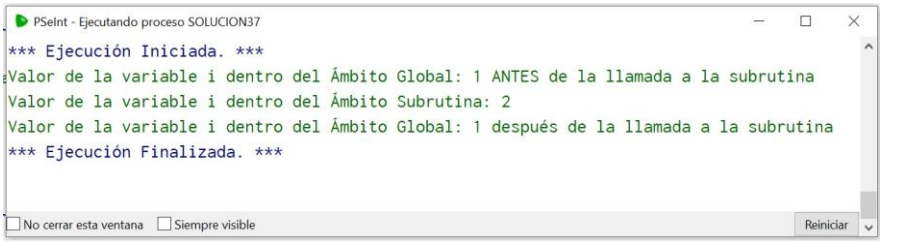
Ejercicio 37: Demostrar que PSeInt solamente tiene dos ámbitos de variables: el Global y el de Subrutina

Por supuesto, la variable no puede pasarse por referencia a la subrutina, porque ese es el ámbito Global. Es decir, al pasar una variable como parámetro de entrada Por Referencia, en realidad estamos pasando el ámbito global de la variable, al ámbito de la subrutina.

```
Subproceso UnaSubrutina()

    // Ámbito Subrutina
    Definir i Como Entero;
    i = 2;
    Escribir "Valor de la variable i dentro del Ámbito Subrutina: ", i;

FinSubproceso
```

```
Algoritmo Solucion37
        // Ámbito Global
        Definir i Como Entero;
        i = 1;
        Escribir "Valor de la variable i dentro del Ámbito Global: ", i,
                        " ANTES de la llamada a la subrutina";

        // Entramos en el Ámbito de una Subrutina
        UnaSubrutina();
        // Comprobar los valores al salir del Ámbito Subrutina
        Escribir "Valor de la variable i dentro del Ámbito Global: ", i,
                        " después de la llamada a la subrutina";
FinAlgoritmo
```

Resultado:

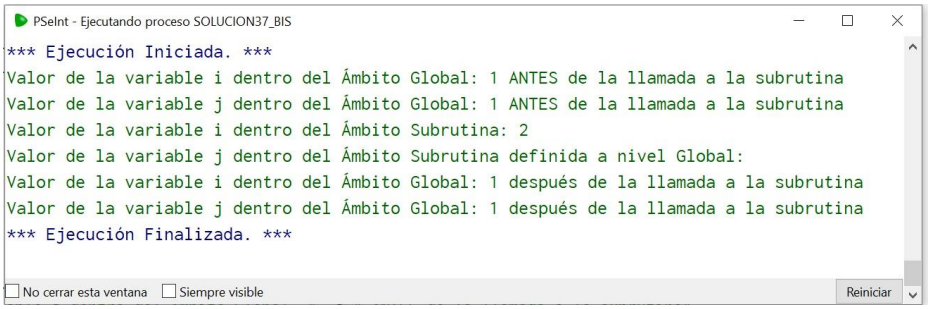


Pero comprobemos "un poco más". Vamos a definir una variable a nivel Global y acceder a la misma desde el ámbito Subrutina.

```
Subproceso UnaSubrutina()

        // Ámbito Subrutina
        Definir i Como Entero;
        i = 2;
        Escribir "Valor de la variable i dentro del Ámbito Subrutina: ", i;
        Escribir "Valor de la variable j dentro del Ámbito Subrutina",
                " definida a nivel Global: ", j;

FinSubproceso

Algoritmo Solucion37_bis
        // Ámbito Global
        Definir i, j Como Entero;
        i = 1;
        j = 1;
        Escribir "Valor de la variable i dentro del Ámbito Global: ", i,
                        " ANTES de la llamada a la subrutina";
```

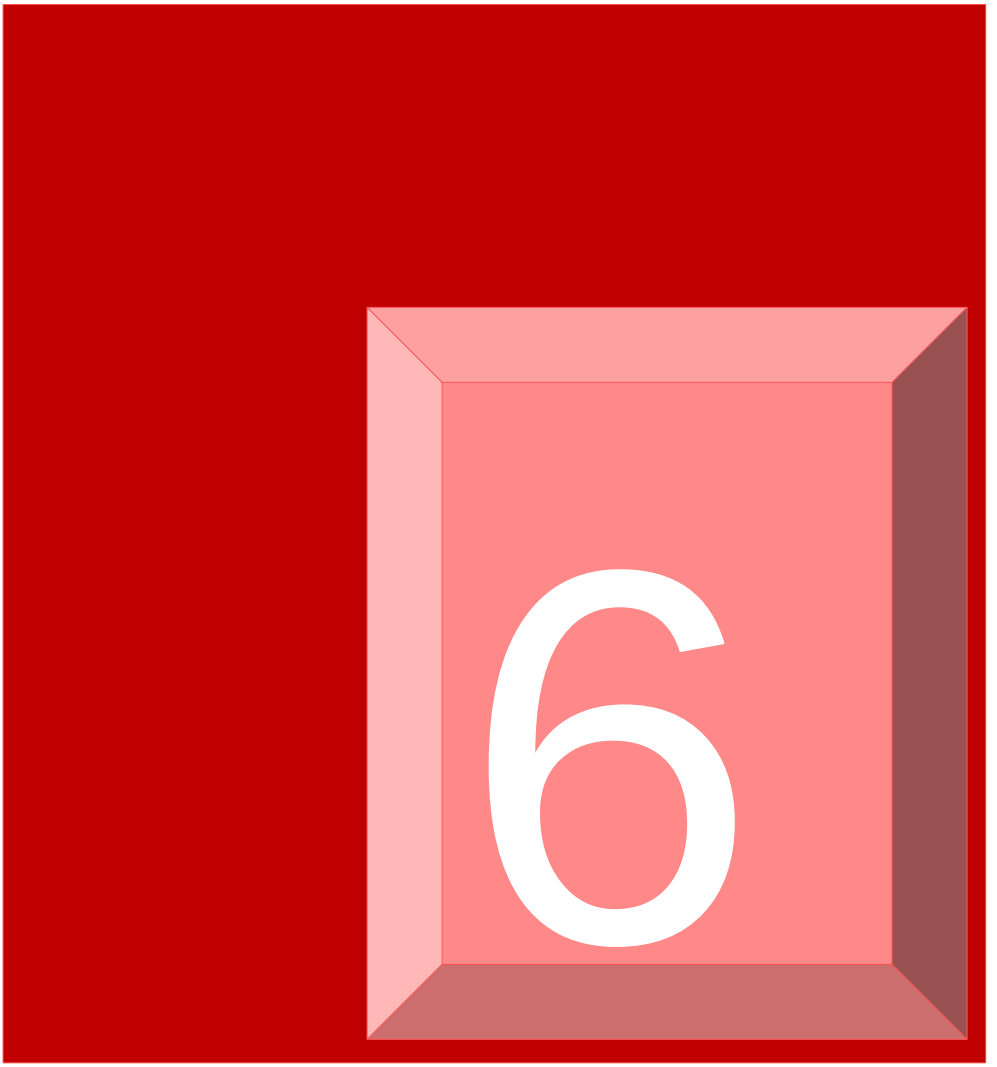
```
    Escribir "Valor de la variable j dentro del Ámbito Global: ", j,
                    " ANTES de la llamada a la subrutina";

    // Entramos en el Ámbito de una Subrutina
    UnaSubrutina();

    // Comprobar los valores al salir del Ámbito Subrutina
    Escribir "Valor de la variable i dentro del Ámbito Global: ", i,
                    " después de la llamada a la subrutina";
    Escribir "Valor de la variable j dentro del Ámbito Global: ", j,
                    " después de la llamada a la subrutina";

FinAlgoritmo
```

Resultado:



Como se puede comprobar, la variable j está indefinida en el ámbito Subrutina (no se ha escrito ningún valor de la variable en la sentencia Escribir de la subrutina), porque desde este ámbito no se "alcanza" el ámbito global; a no ser que pasemos la variable j como parámetro de entrada Por Referencia.

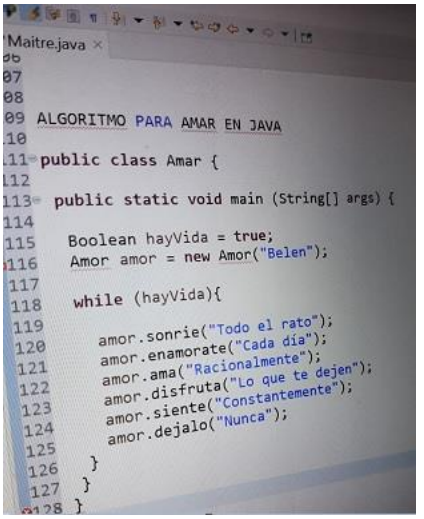
Ejercicios para practicar



Capítulo 6

Ejercicios para practicar

Aprender a Programar | Ejemplos en PSeInt



En este apartado aparecen ejercicios a resolver sin un orden correlativo o incremental en su dificultad de resolución. Además, dentro de las soluciones de los ejercicios, se tratan los temas de "***Validar la Entrada de Datos***" al programa y "***Formatear la Salida de Datos***" del programa.

Mucho ánimo; y, seguro que usted puede mejorar la simplicidad y eficacia de las soluciones que a continuación le propongo.

Enunciados de los ejercicios para practicar

Ejercicio 01: Pasar 5 euros a pesetas

Transformar 5 euros a su equivalencia en pesetas.

PISTA: 1 euro son 166,386 pesetas.
La peseta era la moneda de curso legal en España anterior a la entrada en uso de la moneda europea: el euro.

Ejercicio 02: Pasar de euros a pesetas

Transformar cualquier cantidad en euros a su equivalencia en pesetas.

PISTA: Cuando en el enunciado, el dato de entrada no está especificado, se quiere indicar que el dato, al ser genérico o indefinido, se debe introducir por el usuario del programa a través del teclado. Es decir, con una sentencia "Leer".

Ejercicio 03: Pasar de pesetas a euros

Transformar cualquier cantidad en pesetas a su equivalencia en euros.

PISTA: Para convertir de pesetas a euros, hay que usar la operación matemática contraria a la multiplicación que ha usado en el ejercicio de pasar de euros a pesetas.

Ejercicio 04: Dada una fecha decir cuál es su horóscopo

Dado un mes y un día, introducidos por el usuario, el programa nos debe mostrar el horóscopo al que pertenecen esos datos.

Ejercicio 05: Dada una fecha decir cuál es su horóscopo Celta de Animales

Dado un mes y un día, introducidos por el usuario, el programa nos debe mostrar el horóscopo Celta de Animales al que pertenecen esos datos.

Ejercicio 06: Pirámide de asteriscos

Codifique un algoritmo para dibujar por pantalla la siguiente pirámide de asteriscos:

```
    *
   ***
  *****
 *******
*********
```

PISTA: Contar el número de asteriscos por fila de la pirámide.

Ejercicio 07: Pirámide hueca de asteriscos

Codifique un algoritmo para dibujar por pantalla la siguiente pirámide de asteriscos:

```
    *
   * *
  *   *
 *     *
*********
```

PISTA: Es la misma pirámide que en el ejercicio anterior, pero con huecos por dentro.

Ejercicio 08: Calcular un número elevado a otro

Calcular el valor de un número, introducido por teclado, elevado a otro número también introducido por teclado.

Explicación:

Si se introduce el número 5 y lo elevamos a 2, nos debe mostrar por pantalla el resultado: 25

Ejercicio 09: Imprimir la Tabla de Multiplicar de un número

Devolver la tabla de multiplicar completa, de un número introducido por teclado.

El resultado debe ser:

Tabla de multiplicar del número: 5

0 x 5 = 0
1 x 5 = 5
2 x 5 = 10
3 x 5 = 15
4 x 5 = 20
5 x 5 = 25
6 x 5 = 30
7 x 5 = 35
8 x 5 = 40
9 x 5 = 45
10 x 50 = 50

Ejercicio 10: Realizar la Tabla de Verdad de las páginas 38 y 39

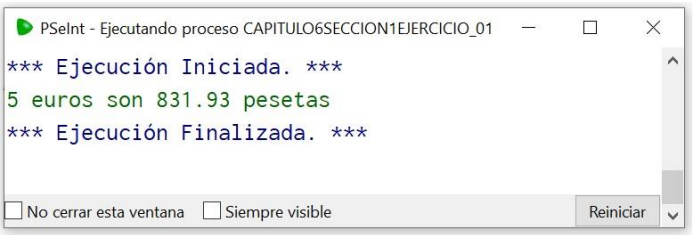
Recordemos que en las páginas 38 y 39 se aclaraba el uso que, los operadores lógicos **Y** y **O**, tenían respecto del idioma castellano. Y para ello, utilizábamos los ejemplos de las frases en castellano "Hoy es lunes **Y** empieza la semana" y "Hoy es lunes **O** empieza la semana".

Soluciones de los ejercicios para practicar

Solución Ejercicio 01: Pasar 5 euros a pesetas

```
Algoritmo Capitulo6Ejercicio_01
        //Para resolver este ejercicio
        //hay que saber que 1 euro equivale a 166,386 pesetas.

        Definir euros Como Real;
        euros = 5; //por el enunciado del ejercicio

        Escribir euros, " euros son ", euros * 166.386, " pesetas";
        //esta sentencia "Escribir" está compuesta de las siguientes secciones:
        //  1.- resolución de la expresión compuesta por la variable "euros".
        //  2.- resolución de la expresión compuesta por el literal " euros son ".
        //  3.- resolución de la expresión compuesta por la operación matemática:
        //        euros * 166.386   donde el asterisco equivale al símbolo de
        //                          multiplicación de las matemáticas.
        //Hay que fijarse que los decimales se escriben en PSeInt
        //con el carácter de punto "." y no con una coma.
        //  4.- resolución de la expresión compuesta por el literal " pesetas".
FinAlgoritmo
```

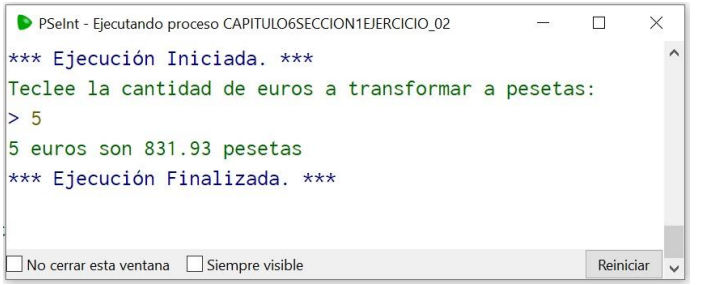
Resultado:



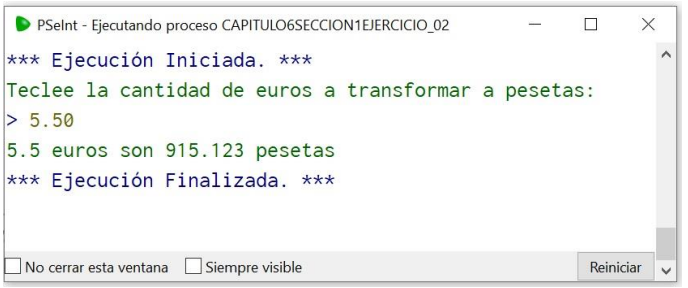
Solución Ejercicio 02: Pasar de euros a pesetas

```
Algoritmo Capitulo6Ejercicio_02
        //Para resolver este ejercicio
        //hay que saber que 1 euro equivale a 166,386 pesetas.

        //Definición de variables utilizadas en el programa.
        Definir euros, pesetas Como Real;
        //Hay que fijarse que ambas variables son de tipo "Real"
        //es decir, pueden trabajar con números decimales o sin decimales: números
        //enteros.

        //Datos de Entrada
        Escribir "Teclee la cantidad de euros a transformar a pesetas: ";
        Leer euros; //por el enunciado del ejercicio
        //Datos de Proceso
        pesetas = euros * 166.386; //hacemos la transformación de euros a pesetas.
        //Datos de Salida
        Escribir euros, " euros son ", pesetas, " pesetas";
        //esta sentencia "Escribir" está compuesta de las siguientes secciones:
        //  1.- resolución de la expresión compuesta por la variable "euros".
        //  2.- resolución de la expresión compuesta por el literal " euros son ".
        //  3.- resolución de la expresión compuesta por la variable "pesetas".
        //  4.- resolución de la expresión compuesta por el literal " pesetas".

        //Hay que fijarse que los literales contemplan los caracteres de los
        //espacios en blanco para separar los resultados numéricos de los textos
        //literales.
FinAlgoritmo
```

Resultado:



Resultado: Introducimos 5 euros con 50 céntimos.

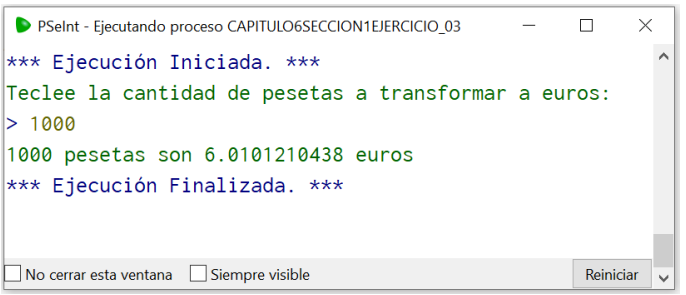


Fíjese cómo PSeInt utiliza los decimales con el carácter de punto “.”
Y cómo elimina el cero a la derecha en la salida del resultado por pantalla, el cero de la derecha de 5 euros con 50 céntimos.

Solución Ejercicio 03: Pasar de pesetas a euros

```
Algoritmo Capitulo6Ejercicio_03
        //Para resolver este ejercicio
        //hay que saber que 166,386 pesetas son 1 euro.

        //Definición de variables utilizadas en el programa.
        Definir euros, pesetas Como Real;
        //Hay que fijarse que ambas variables son de tipo "Real"
        //es decir, pueden trabajar con números decimales o sin decimales (enteros).

        //Datos de Entrada
        Escribir "Teclee la cantidad de pesetas a transformar a euros: ";
        Leer pesetas; //por el enunciado del ejercicio
        //Datos de Proceso
        euros = pesetas / 166.386; //hacemos la transformación de pesetas a euros.
        //Datos de Salida
        Escribir pesetas, " pesetas son ", euros, " euros";
FinAlgoritmo
```

Resultado:



Hago notar que las cifras de miles no se introducen con el punto. Es decir, el dato de entrada no lo escribimos como 1.000 pesetas. Porque el programa PSeInt lo consideraría como un entero 1 más tres ceros de parte decimal, es decir, solamente se consideraría el número entero 1 y se despreciarían los ceros decimales (por estar los ceros a la derecha del punto decimal del número Real).

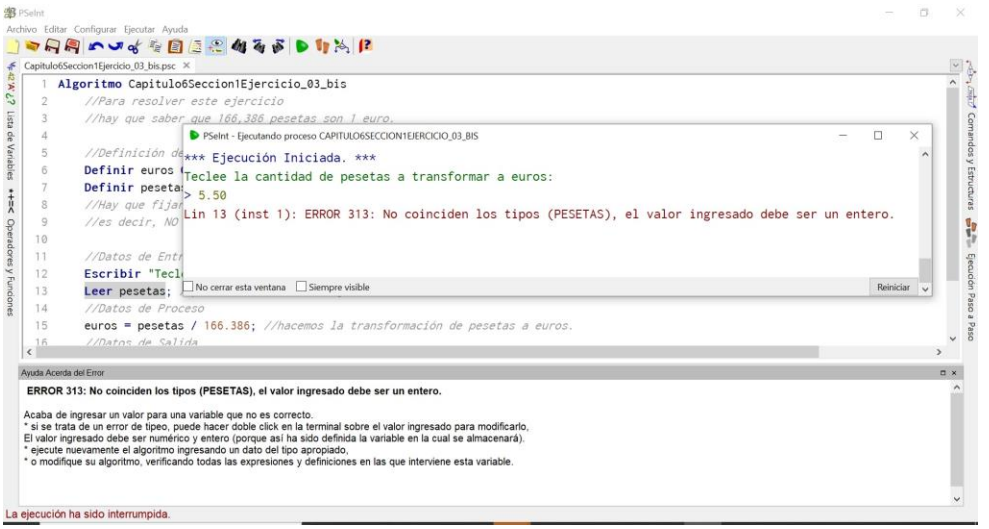
De las pesetas no existen los céntimos de peseta, aunque sí que existieron a lo largo de la historia del uso de la peseta en España. Es decir, no existen: 915,123 pesetas pasarlo a euros. Debemos intentar que este dato no se pueda introducir y que si el usuario lo introduce que PSeInt dé un mensaje de error. Para ello, modificamos el programa de la siguiente forma:

```
Algoritmo Capitulo6Ejercicio_03_bis
        //Para resolver este ejercicio
        //hay que saber que 166,386 pesetas son 1 euro.

        //Definición de variables utilizadas en el programa.
        Definir euros Como Real;
        Definir pesetas Como Entero;
        //Hay que fijarse que ambas variables NO son de tipo "Real"
        //es decir, NO pueden trabajar ambas con números decimales.

        //Datos de Entrada
        Escribir "Teclee la cantidad de pesetas a transformar a euros: ";
        Leer pesetas; //por el enunciado del ejercicio
        //Datos de Proceso
        euros = pesetas / 166.386; //hacemos la transformación de pesetas a euros.
        //Datos de Salida
        Escribir pesetas, " pesetas son ", euros, " euros";
FinAlgoritmo
```

Compruebe el lector que obtenemos el mismo resultado que con el código del programa anterior, para una entrada de 1000 pesetas.

Comprobamos este código con el dato de entrada en pesetas: 5.50

Resultado:



Como puede apreciar, PSeInt nos presenta un error en Tiempo de Ejecución y nos deja señalada, en color gris, la línea del programa en la que se produce el error, en este caso:

Leer pesetas;

Por lo tanto, una variable de tipo Entero no admite decimales (lógicamente, está comprobado porque nos sale un error), pero a la inversa sí que es posible. Es decir, una variable de tipo Real admite números de tipo Entero (lógicamente, porque matemáticamente los números Enteros están incluidos dentro de los números Reales).

Pero, esto no puede quedarse así. No se puede hacer un programa de ordenador que "estalle" en el momento de la ejecución del programa. Es decir, hay que capturar la excepción que provoca PSeInt ante la ejecución del programa al ejecutar la introducción de este dato con decimales y no permitir que el programa PSeInt aborte su ejecución. El lector debe imaginar que, en este caso no tiene consecuencias, pero si estamos haciendo el programa que maneja datos numéricos de los componentes y resultados del control de una Central Nuclear… no podemos dejar que el programa aborte ante un dato inesperado, como es el caso que el usuario no tenga sentido que utilice números en decimales al introducir el valor en pesetas, en decimal. Y el programa "estalle".

Vamos a resolverlo.

Validar los datos de los valores de entrada al programa

Como hemos comprobado en el ejercicio anterior, el programa de ejecución de nuestro algoritmo: PSeInt; no puede abortar su ejecución ante un error inesperado de los datos de entrada. Por lo tanto, debemos validar los datos de los valores de entrada al programa por parte del usuario, en este caso (porque los valores de entrada podrían ser procedentes de otro ordenador y no de un ser humano).

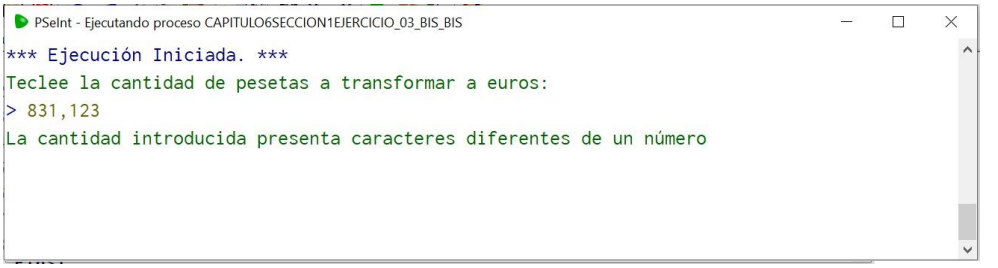
En entornos profesionales de programación, siempre debemos validar los datos de entrada al programa. Y debemos validarlos porque los datos de entrada de un programa pueden haberse generado como datos de salida por parte de otro programa. Es decir, los programas informáticos pueden "hablar" entre sí sin intervención humana (de un usuario humano). Por lo tanto, que hoy sea un humano el que introduzca nuestros datos de entrada al programa, no quiere decir que, a lo largo del tiempo de vida del programa, siempre sea un humano quien nos introduzca los datos. Con el tiempo, se puede sustituir un humano por la salida de datos de otro programa.

Comprobemos el algoritmo de depuración o validación de entrada de los valores de entrada al algoritmo de transformación de pesetas a euros.

```
Algoritmo Capitulo6Ejercicio_03_bis_bis
        //Para resolver este ejercicio hay que saber que 166,386 pesetas son 1 euro.
        //Definición de variables utilizadas en el programa.
        Definir euros Como Real;
        Definir pesetas Como Cadena;
        Definir pesetasEntero, i Como Entero;
        Definir esUnNumero Como Logico;

        //Inicialización de variables
        esUnNumero = VERDADERO; //En principio a valor VERDADERO
        //porque consideramos que el usuario va a teclear bien la entrada:
        //que va a introducir un número Entero.
        i = 1; //El primer carácter de un dato de tipo Cadena
        //empieza en la posición 1.

        //Datos de Entrada
        Escribir "Teclee la cantidad de pesetas a transformar a euros: ";
        Leer pesetas;
```

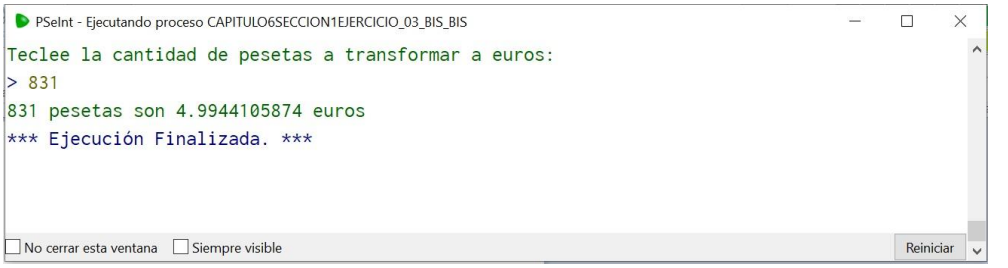
```
        Repetir

                //Escribir "El carácter ", i, " es ", SUBCADENA(pesetas, i, i);
                //Descomentar la anterior línea de código porque es una línea que es un
                //Chivato. Es decir, quitar el comentario inicial de la anterior línea.
                //Un chivato es una salida por pantalla del contenido de alguna variable
                //en tiempo de ejecución. Se usa porque es más rápido que usar el
                //Depurador de PSeInt.
//Inserte la siguiente línea de código "Si" en una sola línea del editor de PSeInt
                Si (SUBCADENA(pesetas, i, i) <> "0")
                  Y (SUBCADENA(pesetas, i, i) <> "1")
                  Y (SUBCADENA(pesetas, i, i) <> "2")
                  Y (SUBCADENA(pesetas, i, i) <> "3")
                  Y (SUBCADENA(pesetas, i, i) <> "4")
                  Y (SUBCADENA(pesetas, i, i) <> "5")
                  Y (SUBCADENA(pesetas, i, i) <> "6")
                  Y (SUBCADENA(pesetas, i, i) <> "7")
                  Y (SUBCADENA(pesetas, i, i) <> "8")
                  Y (SUBCADENA(pesetas, i, i) <> "9") Entonces
                        //Hemos encontrado un carácter diferente de un dígito entre
                        // 0,1,2,3,4,5,6,7,8,9.
                        esUnNumero = FALSO;
                FinSi

                i = i + 1; //Incrementamos el iterador de la cadena

                Si NO (esUnNumero) Entonces
                        //Visualizar error en datos de entrada del usuario
                        Escribir "La cantidad introducida presenta caracteres diferentes”
                        Escribir “ de un número";
                        Esperar 3 Segundos;
                        Borrar Pantalla;
                        esUnNumero = VERDADERO; //Inicializamos el valor lógico.
                        i = 1; //Inicializamos el iterador de la cadena.
                        //Volvemos a solicitar un nuevo número.
                        Escribir "Teclee la cantidad de pesetas a transformar a euros: ";
                        Leer pesetas;
                FinSi

        Hasta Que i > LONGITUD(pesetas) Y esUnNumero
                //Cambiamos los datos de entrada del tipo Cadena al tipo Entero.
        //Esta función no falla porque nos hemos asegurado que "pesetas" contiene
        //solamente dígitos entre el 0 y el 9.
        pesetasEntero = CONVERTIRANUMERO(pesetas);

        //Datos de Proceso
        euros = pesetasEntero / 166.386;
        //hacemos la transformación de pesetas a euros.
        //La operación matemática contraria a la multiplicación, es la división.

        //Datos de Salida
        Escribir pesetas, " pesetas son ", euros, " euros";
FinAlgoritmo
```

Resultado: cuando en los datos de entrada hay algún carácter diferente de un dígito, se muestra un error y se vuelve a pedir un dato de entrada válido.



Después de mostrar este mensaje de error, el programa espera tres segundos y se borra la pantalla. Para, seguidamente, volver a mostrar el mensaje para introducir una nueva cantidad de pesetas.

Resultado: con la introducción de una cadena compuesta únicamente de dígitos.



Pero, de la misma forma que hay que validar los datos de entrada, hay que **formatear los datos de salida** ya que una cajera de un supermercado no puede devolver 4.9944105874 euros. Es decir, la cajera debe devolver 4 euros y 99 céntimos de euros.

Por lo tanto, vamos a formatear los datos de salida para que sean manejables (entendibles) por un ser humano.

Formatear los datos de los valores de salida del programa

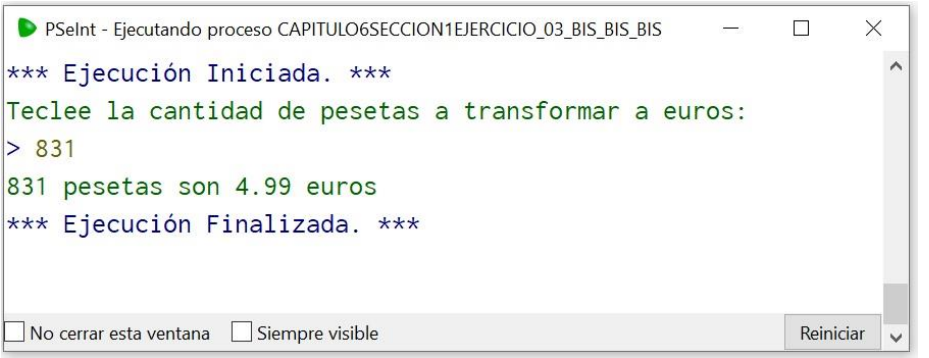
Como hemos comprobado en el ejercicio anterior, el programa debe visualizar los datos de salida de la ejecución del programa (resultado) en un formato legible por el destinatario final de la solución, sea este destinatario un ser humano u otro programa.

```
Algoritmo Capitulo6Ejercicio_03_bis_bis_bis
        //Para resolver este ejercicio hay que saber que 166,386 pesetas son 1 euro.
        //Definición de variables utilizadas en el programa.
        Definir euros Como Real;
        Definir pesetas, eurosCadena Como Cadena;
        Definir pesetasEntero, i Como Entero;
        Definir esUnNumero, esUnDecimal Como Logico;

        //Inicialización de variables
        esUnNumero = VERDADERO; //En principio a valor VERDADERO
        //porque consideramos que el usuario va a teclear bien la entrada:
        //que va a introducir un número Entero.
        i = 1; //El primer carácter de un dato de tipo Cadena
        //empieza en la posición 1.

        //Datos de Entrada
        Escribir "Teclee la cantidad de pesetas a transformar a euros: ";
        Leer pesetas;

        Repetir

                //Escribir "El caracter ", i, " es ", SUBCADENA(pesetas, i, i);
                //Descomentar la anterior línea de código porque es una línea que es un
                //Chivato.
                //Un chivato es una salida por pantalla del contenido de alguna variable
                //en tiempo de ejecución. Se usa porque es más rápido que usar el
                //Depurador de PSeInt.
                Si (SUBCADENA(pesetas, i, i) <> "0")
                   Y (SUBCADENA(pesetas, i, i) <> "1")
                   Y (SUBCADENA(pesetas, i, i) <> "2")
                   Y (SUBCADENA(pesetas, i, i) <> "3")
                   Y (SUBCADENA(pesetas, i, i) <> "4")
                   Y (SUBCADENA(pesetas, i, i) <> "5")
                   Y (SUBCADENA(pesetas, i, i) <> "6")
                   Y (SUBCADENA(pesetas, i, i) <> "7")
                   Y (SUBCADENA(pesetas, i, i) <> "8")
                   Y (SUBCADENA(pesetas, i, i) <> "9") Entonces
```

```
                //Hemos encontrado un carácter diferente de un dígito entre
                // 0,1,2,3,4,5,6,7,8,9.
                esUnNumero = FALSO;
        FinSi

        i = i + 1; //Incrementamos el iterador de la cadena

        Si NO (esUnNumero) Entonces
                //Visualizar error en datos de entrada del usuario
                Escribir "La cantidad introducida presenta caracteres diferentes";
                Escribir " de un número";
                Esperar 3 Segundos;
                Borrar Pantalla;
                esUnNumero = VERDADERO; //Inicializamos el valor lógico.
                i = 1; //Inicializamos el iterador de la cadena.
                //Volvemos a solicitar un nuevo número.
                Escribir "Teclee la cantidad de pesetas a transformar a euros: ";
                Leer pesetas;
        FinSi

Hasta Que i > LONGITUD(pesetas) Y esUnNumero

//Cambiamos los datos de entrada del tipo Cadena al tipo Entero.
//Esta función no falla porque nos hemos asegurado que "pesetas" contiene
//solamente dígitos entre el 0 y el 9.
pesetasEntero = CONVERTIRANUMERO(pesetas);

//Datos de Proceso
euros = pesetasEntero / 166.386; //hacemos la transformación de pesetas a
                                 //euros.
//La operación matemática contraria a la multiplicación, es la división.

//Datos de Salida
//Formateamos la salida a un valor entendible por un ser humano.
//Cambiamos los datos de entrada del tipo Entero al tipo Cadena.
eurosCadena = CONVERTIRATEXTO(euros);

//Estamos seguros que en la variable "euros" hay un número entero
//o bien un número real. Es decir, un número con parte entera
//y parte decimal separada por un carácter de punto: "."
//Nos quedamos con la parte entera y solamente dos decimales
//de la parte después del carácter "."

//Inicializamos las variables
i = 1; //inicializamos a 1 al iterador de la cadena.
esUnDecimal = FALSO; //inicialmente suponemos que es solamente un Entero.

//Escribir "El dígito a tratar es: ", euros;
//Descomentar la anterior línea de código porque es una línea que es un
//Chivato.
//Un chivato es una salida por pantalla del contenido de alguna variable
//en tiempo de ejecución. Se usa porque es más rápido que usar el Depurador
//de PSeInt.
```

```
        Repetir

                //Escribir "El caracter ", i, " es ", SUBCADENA(eurosCadena, i, i);
                //Descomentar la anterior línea de código porque es una línea que es un
                //Chivato.

                //Buscamos el Carácter de la parte decimal que es el carácter PUNTO.
                Si (SUBCADENA(eurosCadena, i, i) = ".") Entonces
                        //Hemos encontrado un carácter PUNTO.
                        esUnDecimal = VERDADERO; //para salir del bucle
                        //Actualizamos el valor de la cadena para quedarnos
                        //con la parte entera y dos decimales.
                        //OJO: el valor de la variable "i" está en la posición
                        //de la cadena que tiene el carácter PUNTO.
                        eurosCadena = SUBCADENA(eurosCadena, 1, i+2 );
                        //Nos quedamos con los dos decimales después del carácter
                        //PUNTO.
                FinSi

                i = i + 1; //Incrementamos el iterador de la cadena

                Si i > LONGITUD(eurosCadena) Entonces
                        //Hemos sobrepasado la longitud de la variable "eurosCadena"
                        //porque la variable "eurosCadena" es un número Entero.
                        //Es decir, no tiene parte decimal.
                        esUnDecimal = VERDADERO; //Para salir del bucle
                FinSi
        Hasta Que esUnDecimal

        //Al salir del bucle, la variable de cadena "eurosCadena"
        //tendrá unos dígitos de la parte entera, si es un número Entero.
        //O bien, tendrá unos dígitos de la parte entera y el carácter PUNTO
        //y solamente dos dígitos de la parte decimal; porque es un número
        //decimal.

        //Procedemos a sacar el resultado por pantalla.
        Escribir pesetas, " pesetas son ", eurosCadena, " euros";
FinAlgoritmo
```

Resultado: formateando la salida para el usuario humano, a un número decimal con dos decimales, solamente.



ojo

Los datos hay que Validarlos a la entrada al programa (comprobar que son correctos o que son los datos con el formato esperado por el programa) y Formatearlos a la salida del programa (adaptarlos a la presentación visual necesaria para su comprensión por el destinatario del programa, sea este un ser humano u otro programa).

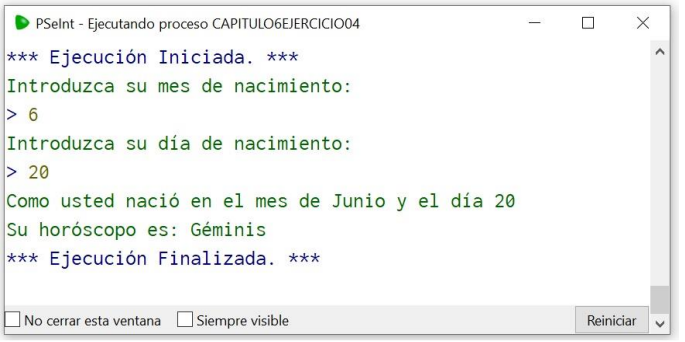
Solución Ejercicio 04: Dada una fecha decir cuál es su horóscopo

```
Algoritmo Capitulo6Ejercicio04

    //Declaramos las variables del programa
    Definir mes, dia Como Entero;
    Definir mesNacimiento, horoscopo Como Cadena;

    //DATOS DE ENTRADA
    Escribir "Introduzca su mes de nacimiento: ";
    Leer mes;
    Escribir "Introduzca su día de nacimiento: ";
    Leer dia;

    //DATOS DE PROCESO
    Segun mes Hacer
```

```
1:
        mesNacimiento = "Enero";
        Si (dia < 21) Entonces
                horoscopo = "Capricornio";
        SiNo
                horoscopo = "Acuario";
        FinSi
2:
        mesNacimiento = "Febrero";
        Si (dia < 20) Entonces
                horoscopo = "Acuario";
        SiNo
                horoscopo = "Piscis";
        FinSi
3:
        mesNacimiento = "Marzo";
        Si (dia < 21) Entonces
                horoscopo = "Piscis";
        SiNo
                horoscopo = "Aries";
        FinSi
4:
        mesNacimiento = "Abril";
        Si (dia < 21) Entonces
                horoscopo = "Aries";
        SiNo
                horoscopo = "Tauro";
        FinSi
5:
        mesNacimiento = "Mayo";
        Si (dia < 20) Entonces
                horoscopo = "Tauro";
        SiNo
                horoscopo = "Géminis";
        FinSi
6:
        mesNacimiento = "Junio";
        Si (dia < 22) Entonces
                horoscopo = "Géminis";
        SiNo
                horoscopo = "Cáncer";
        FinSi
7:
        mesNacimiento = "Julio";
        Si (dia < 22) Entonces
                horoscopo = "Cáncer";
        SiNo
                horoscopo = "Leo";
        FinSi
```

```
            8:
                mesNacimiento = "Agosto";
                Si (dia < 24) Entonces
                    horoscopo = "Leo";
                SiNo
                    horoscopo = "Virgo";
                FinSi
            9:
                mesNacimiento = "Septiembre";
                Si (dia < 23) Entonces
                    horoscopo = "Virgo";
                SiNo
                    horoscopo = "Libra";
                FinSi
            10:
                mesNacimiento = "Octubre";
                Si (dia < 23) Entonces
                    horoscopo = "Libra";
                SiNo
                    horoscopo = "Escorpio";
                FinSi
            11:
                mesNacimiento = "Noviembre";
                Si (dia < 23) Entonces
                    horoscopo = "Escorpio";
                SiNo
                    horoscopo = "Sagitario";
                FinSi
            12:
                mesNacimiento = "Diciembre";
                Si (dia < 21) Entonces
                    horoscopo = "Sagitario";
                SiNo
                    horoscopo = "Capricornio";
                FinSi
        FinSegun

        //DATOS DE SALIDA
        Escribir "Como usted nació en el mes de ", mesNacimiento, " y el día ", dia;
        Escribir "Su horóscopo es: ", horoscopo;

FinAlgoritmo
```

Resultado:



Recuerde que no se deben utilizar tildes en el nombre de las variables y en cualquier identificador del programa fuente (código a ejecutar).

Solución Ejercicio 05: Dada una fecha decir cuál es su horóscopo Celta de Animales

Para responder a este problema debemos buscar en Internet los INTERVALOS DEL ZODIACO CELTA DE ANIMALES

WEB: **https://www.terra.cl/estilo-de-vida/2020/11/30/signos-del-zodiaco-animal-celta-signos-de-arboles-cual-eres-tu-3681.html**

Visitada: 04-08-2022

OBTENEMOS:

1.-Dragón verde / Green Dragon (21 de enero - 17 de febrero)
2.-Caballo de mar / Sea Horse (18 de febrero - 27 de marzo)
3.-El Halcón / Hawk (28 de marzo - 14 de abril)
4.-Serpiente marina / Sea Serpent (15 de abril - 12 de mayo)
5.-Zorro / Fox (13 de mayo - 9 de junio)
6.-Caballo blanco / White Horse (10 de junio - 7 de julio)
7.- Unicornio (8 de julio - 4 de agosto)
8.-Salmón arcoíris / Rainbow Salmon (5 de agosto - 1 de septiembre)
9.-Cisne blanco / White Swan (2 de septiembre - 29 de septiembre)
10.-Mariposa / Butterfly (30 de septiembre - 27 de octubre)
11.-Sabueso blanco / White Hound (28 de octubre - 24 de noviembre)
12.-Caballo negro / Black Horse (25 de noviembre - 23 de diciembre)
13.-Ciervo / Stag (24 de diciembre - 20 de enero)

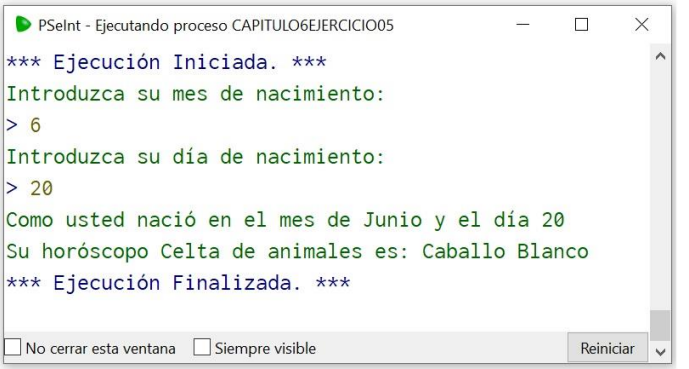
Según la anterior relación de intervalos, el algoritmo se quedaría de la siguiente forma:

```
Algoritmo Capitulo6Ejercicio05

	//Declaramos las variables del programa
	Definir mes, dia Como Entero;
	Definir mesNacimiento, horoscopo Como Cadena;


	//DATOS DE ENTRADA
	Escribir "Introduzca su mes de nacimiento: ";
	Leer mes;
	Escribir "Introduzca su día de nacimiento: ";
	Leer dia;

	//DATOS DE PROCESO
	Segun mes Hacer
		1:
			mesNacimiento = "Enero";
			Si (dia < 21) Entonces
				horoscopo = "Ciervo";
			SiNo
				horoscopo = "Dragón Verde";
			FinSi
		2:
			mesNacimiento = "Febrero";
			Si (dia < 18) Entonces
				horoscopo = "Dragón Verde";
			SiNo
				horoscopo = "Caballo de Mar";
			FinSi
		3:
			mesNacimiento = "Marzo";
			Si (dia < 28) Entonces
				horoscopo = "Caballo de Mar";
			SiNo
				horoscopo = "El Halcón";
			FinSi
		4:
			mesNacimiento = "Abril";
			Si (dia < 15) Entonces
				horoscopo = "El Halcón";
			SiNo
				horoscopo = "Serpiente Marina";
			FinSi
		5:
			mesNacimiento = "Mayo";
			Si (dia < 13) Entonces
				horoscopo = "Serpiente Marina";
			SiNo
				horoscopo = "Zorro";
			FinSi
```

```
6:
        mesNacimiento = "Junio";
        Si (dia < 10) Entonces
                horoscopo = "Zorro";
        SiNo
                horoscopo = "Caballo Blanco";
        FinSi
7:
        mesNacimiento = "Julio";
        Si (dia < 8) Entonces
                horoscopo = "Caballo Blanco";
        SiNo
                horoscopo = "Unicornio";
        FinSi
8:
        mesNacimiento = "Agosto";
        Si (dia < 5) Entonces
                horoscopo = "Unicornio";
        SiNo
                horoscopo = "Salmón Arcoiris";
        FinSi
9:
        mesNacimiento = "Septiembre";
        Si (dia = 1) Entonces
                horoscopo = "Salmón Arcoiris";
        SiNo
                Si (dia < 30) Entonces
                        horoscopo = "Cisne Blanco";
                SiNo
                        horoscopo = "Mariposa";
                FinSi
        FinSi
10:
        mesNacimiento = "Octubre";
        Si (dia < 28) Entonces
                horoscopo = "Mariposa";
        SiNo
                horoscopo = "Sabueso Blanco";
        FinSi
11:
        mesNacimiento = "Noviembre";
        Si (dia < 25) Entonces
                horoscopo = "Sabueso Blanco";
        SiNo
                horoscopo = "Caballo Negro";
        FinSi
12:
        mesNacimiento = "Diciembre";
        Si (dia < 24) Entonces
                horoscopo = "Caballo Negro";
        SiNo
                horoscopo = "Ciervo";
        FinSi
FinSegun
```

```
    //DATOS DE SALIDA
    Escribir "Como usted nació en el mes de ", mesNacimiento, " y el día ", dia;
    Escribir "Su horóscopo Celta de animales es: ", horoscopo;

FinAlgoritmo
```

Resultado:



Felicito al lector porque, en estos dos últimos ejercicios, ha aprendido a preparar los datos para construir las expresiones condicionales de la sentencia SI-ENTONCES y ha practicado el uso de la sentencia condicional SI-ENTONCES-SINO de forma adecuada.

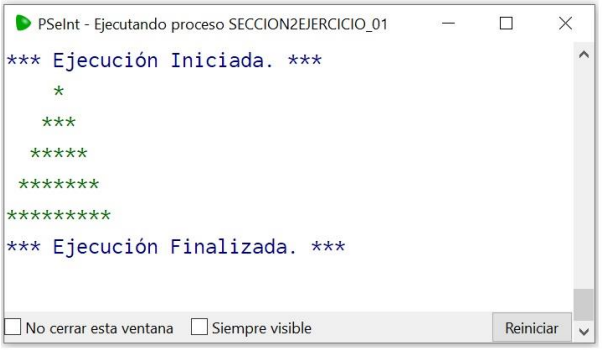
Cuando utilice una sentencia condicional; y, sobre todo si esta anidada (una sentencia dentro de otra); le recomiendo que haga pruebas de cada bifurcación de la sentencia/s SI→ENTONCES y SI→SINO; para comprobar el correcto funcionamiento de cada camino de la sentencia/s condicional/es.

Solución Ejercicio 06: Pirámide de asteriscos

```
Algoritmo Capitulo6Ejercicio_06
    //Para resolver este ejercicio
    //hay que empezar fijándose en la última línea de la pirámide:
    //vemos que son 9 asteriscos y que disminuye en un asterisco
    //por la izquierda y la derecha según vamos ascendiendo desde la
    //base de la pirámide.
    //Solución: La forma más simple es utilizar sentencias "Escribir"
```

```
        Escribir "   *   ";
        Escribir "  ***  ";
        Escribir " ***** ";
        Escribir "*******";
        Escribir "*********";
FinAlgoritmo
```

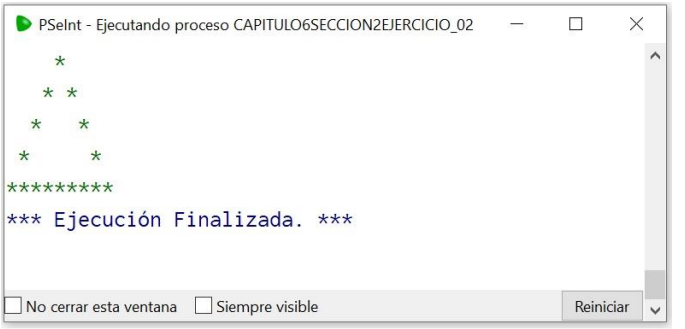
Resultado:



Solución Ejercicio 07: Pirámide hueca de asteriscos

```
Algoritmo Capitulo6Ejercicio_07
        //Para resolver este ejercicio
        //hay que empezar fijándose en la última línea de la pirámide:
        //vemos que son 9 asteriscos y que disminuye en un asterisco
        //por la izquierda y la derecha según vamos ascendiendo desde la
        //base de la pirámide. Pero, la pirámide está hueca por dentro.
        //Solución: La forma más simple es utilizar sentencias "Escribir"

        Escribir "   *   ";
        Escribir "  * *  ";
        Escribir " *   * ";
        Escribir "*     *";
        Escribir "*********";
FinAlgoritmo
```

Resultado:



ojo

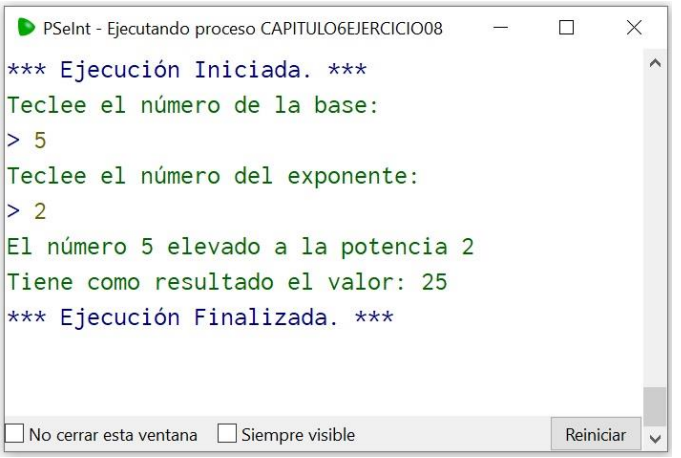
Aunque es cierto que un programador siempre tiene que buscar un algoritmo que resuelva el problema. No siempre es así, como demuestran estos ejercicios. Un programador debe buscar la solución más simple y limpia posible al problema; y, muchas veces se obceca en complicadas soluciones algorítmicas.

Solución Ejercicio 08: Calcular un número elevado a otro

```
Algoritmo Capitulo6Ejercicio08
        //Inicializar Variables
        Definir base, exponente, resultado, i Como Entero;

        //No vamos a hacer Validación de datos de entrada.

        //DATOS DE ENTRADA
        Escribir "Teclee el número de la base: ";
        Leer base;
        Escribir "Teclee el número del exponente: ";
        Leer exponente;
```

```
//DATOS DE PROCESO
resultado = 1;
//Un número multiplicado por cero da como resultado cero.
//Se inicializa a este valor
//porque la primera iteración del bucle "Para"
//se ejecuta la operación aritmética de multiplicación:    resultado * base
//Y si "resultado" tiene el valor cero, dará como salida un valor cero.
//PSeInt inicializa las variables enteras a cero, como dato por defecto.

Para i = 1 Hasta exponente Hacer
        resultado = resultado * base;
FinPara

//DATOS DE SALIDA
Escribir "El número ", base, " elevado a la potencia ", exponente;
Escribir "Tiene como resultado el valor: ", resultado;

FinAlgoritmo
```

Resultado:



Esta es una solución "muy pobre". El programa funciona, pero… no es la solución "ideal".

ojo

Un programador debe tener, o al menos procurar tener, el mayor conocimiento posible de las características y utilidades del lenguaje, su entorno y librerías (funciones predefinidas). Es decir, un programador debe dominar todas las herramientas a su disposición (o, al menos, ir aprendiéndolas con el tiempo).

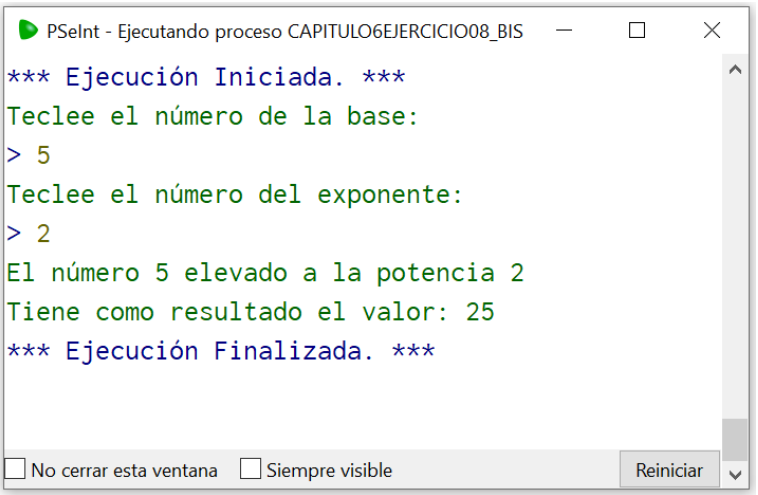
Por lo tanto, ¿ es erróneo el algoritmo ?. No. ¿ Es el resultado más optimo ?. No.

En conclusión, el programador ha resuelto el problema. Pero, ¿habría una forma más simple, limpia, elegante y veloz de resolver el problema ?. **Sí**.

La solución está en conocer bien el lenguaje que utiliza el intérprete de **PSeInt**. Es decir, una solución mejor pasaría por utilizar el **Operador de Potencia**.

```
Algoritmo Capitulo6Ejercicio08_bis
        //Inicializar Variables
        Definir base, exponente, resultado Como Entero;

        //No vamos a hacer Validación de datos de entrada.


        //DATOS DE ENTRADA
        Escribir "Teclee el número de la base: ";
        Leer base;
        Escribir "Teclee el número del exponente: ";
        Leer exponente;

        //DATOS DE PROCESO
        resultado = base ^ exponente;

        //DATOS DE SALIDA
        Escribir "El número ", base, " elevado a la potencia ", exponente;
        Escribir "Tiene como resultado el valor: ", resultado;

FinAlgoritmo
```

Resultado:



Con esta solución hemos ahorrado una variable (con lo que hemos reducido el uso de la memoria del ordenador y, por lo tanto, hemos ahorrado en gastos de los recursos del ordenador) y hemos mejorado la legibilidad del código; al resolver, la fase de procesamiento del algoritmo, en tan solo una línea de código. Por lo tanto, no solamente hemos mejorado en legibilidad (código más fácil de entender por nosotros y por otro programador) sino también en velocidad de ejecución del programa al haber eliminado un bucle.

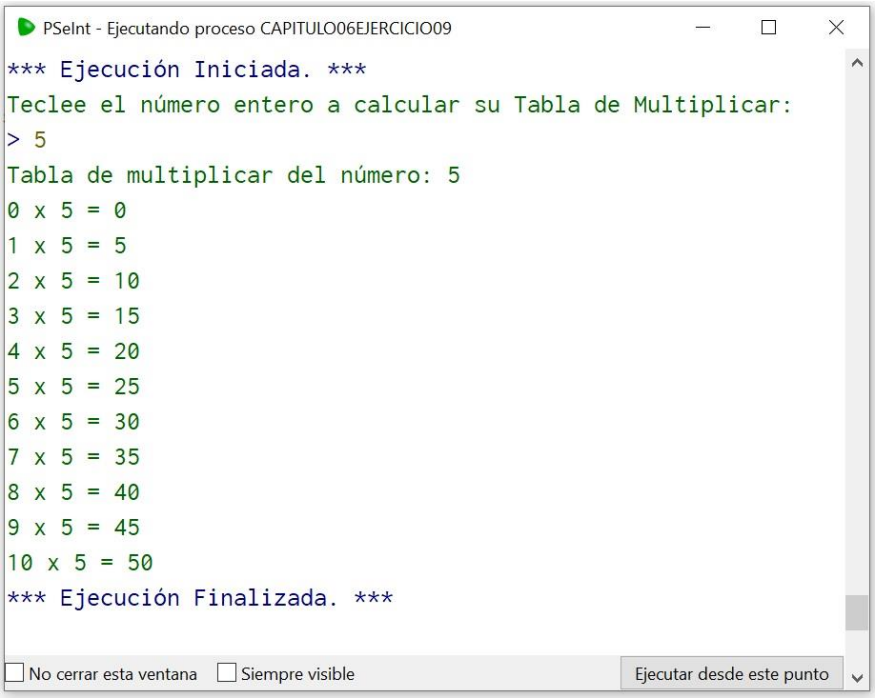
A la hora de programar (codificar un programa) no siempre el primer algoritmo (solución) que se nos viene a la mente, es la mejor solución de implementación posible (la forma de escribirlo para resolverlo).

Solución Ejercicio 09: Imprimir la Tabla de Multiplicar de un número

Solución 1: bucle Para

```
Algoritmo Capitulo06Ejercicio09
        //Definir variables del programa
        Definir numeroEntero, i Como Entero;
        //Entrada de datos
        Escribir "Teclee el número entero a calcular su Tabla de Multiplicar:";
        Leer numeroEntero;
        //Procesar datos y realizar salida
        Escribir "Tabla de multiplicar del número: ", numeroEntero;

        Para i = 0 Hasta 10 Hacer
                Escribir i, " x ", numeroEntero, " = ", i * numeroEntero;
        FinPara

FinAlgoritmo
```

Resultado:



El inconveniente de esta solución es que no es un programa estructurado: todo está "acoplado" dentro del programa. Los algoritmos deben tener un cierto grado de desacoplamiento: estar divididos en módulos reutilizables.

Solución 2: con código reutilizable

Cómo hacerlo:

- utilizar el Procedimiento "escribirTablaMultiplicar(numeroIntroducido)" con un parámetro numérico
- Variable con el texto del encabezado
- Bucle del 0 al 10
- Escribir cada línea de la operación de multiplicación con el resultado obtenido

```
Algoritmo Capitulo06Ejercicio09_bis
        //Definir variables del programa
        Definir numeroEntero Como Entero;
        //Entrada de datos
        Escribir "Teclee el número entero a calcular su Tabla de Multiplicar:";
        Leer numeroEntero;

        //Procesar datos y realizar salida
        EscribirTablaMultiplicar(numeroEntero);

FinAlgoritmo
```

```
SubProceso EscribirTablaMultiplicar(numeroIntroducido)

        //Definir variables
        Definir literalEncabezado Como Cadena;
        Definir i Como Entero;

        //Inicializar datos
        literalEncabezado = "Tabla de multiplicar del número: ";

        //Procesar datos y realizar salida
        Escribir literalEncabezado, numeroIntroducido;

        Para i = 0 Hasta 10 Hacer
                Escribir i, " x ", numeroIntroducido, " = ", i * numeroIntroducido;
        FinPara

FinSubProceso
```

Como puede comprobar en la salida de la consola de PSeInt, obtenemos el mismo resultado.

Quizás este preguntándose: para qué quiero utilizar una variable en la instrucción

```
literalEncabezado = "Tabla de multiplicar del número: ";
```

Esta respuesta requiere tener en mente dos aspectos.

→ Si queremos hacer en el código una gestión autónoma de los mensajes al usuario.

Una decisión de la **Fase de Diseño** de la aplicación supone decantarse por una solución de mensajes "harcodeados" (se lee: "jarcodeados" y viene del verbo "harcodear" y en castellano significa: embebidos) dentro del código o no. Es decir, que los mensajes al usuario estén escritos literalmente o embebidos dentro del código (entre las sentencias del algoritmo) o no. En nuestro caso, al ser poco más de diez líneas de código no tiene importancia que los mensajes al usuario aparezcan dentro del código (entre las líneas del algoritmo) y no estén diferenciados o extraídos del algoritmo. Pero si la aplicación tiene miles de líneas, convendrá el lector conmigo que los mensajes al usuario serán, y conviene, que sean muy numerosos.
Para este caso de muchos mensajes al usuario, conviene hacer una gestión a parte de estos mensajes. Por eso usamos una variable para contener el mensaje al usuario.

La idea es que los mensajes al usuario se encuentren todos en registros de una tabla de la base de datos de la aplicación. Por ejemplo, en la tabla PARAMETROS. Ante un mensaje al usuario, el programa iría a ese registro de la tabla de la base de datos, lo extraería sobre la variable y el mensaje al usuario solamente tiene que mostrar el contenido de esa variable, leída de la base de datos.

Esto tiene otra ventaja, y esta ventaja toma importancia en la **Fase de Mantenimiento** del Software (Mantenimiento de la aplicación). Cuando la aplicación se ha entregado al cliente (usuario) y este desea modificar un mensaje de la aplicación, no tenemos que volver a retocar el código del algoritmo, sino que, únicamente cambiando el registro de la tabla de la base de datos de ese mensaje, lo tendríamos resuelto. Sin necesidad de tocar el código, insisto; y, por lo tanto, sin necesidad de crear un nuevo programa ejecutable, volverlo a probar y distribuirlo a todos los usuarios de la aplicación que pueden ser cientos.

→ Si queremos hacer una aplicación en varios idiomas.

Una vez que tenemos los mensajes al usuario en los registros de una tabla de la base de datos, el paso de la aplicación de unos mensajes al usuario en castellano, a otros idiomas, se hace muy fácilmente. Solamente, añadiendo mensajes al usuario en otros idiomas en la tabla de la base de datos y preguntando en el código fuente (algoritmo) con una sentencia SI-ENTONCES; si el usuario ha seleccionado trabajar en español o inglés, por ejemplo; a la entrada de la aplicación como una variable global al programa que se puede llamar IDIOMAELEGIDO, se le muestra el registro del mensaje en castellano o inglés de la base de datos.

A este proceso de añadir otros idiomas a nuestra aplicación se le denomina: internacionalización de la aplicación. Y puede encontrar más información en Internet buscando el término: **i18n**

Una **Base de Datos** es una recopilación organizada de información o datos estructurados, que normalmente se almacena de forma electrónica en un sistema informático: conjunto de aplicaciones; al que normalmente se le denomina: Sistema de Gestión de Bases de Datos (DBMS-Data Base Management System).

La diferencia, de forma muy simplificada, es que una tabla de una Base de Datos es como una tabla o array de un programa de PSeInt. Pero mientras que una tabla de PSeInt se almacena (reside) en memoria durante la ejecución de la aplicación y, al terminar la ejecución de la aplicación se pierden sus valores porque desaparecen (se eliminan) de la memoria. Las tablas de las Base de Datos residen (se almacenan) en un disco duro que es un dispositivo magnético que guarda de forma permanente y duradera la información de los valores que tiene cada casilla (registro) de una tabla de esa Base de Datos.

Solución Ejercicio 10: Realizar la Tabla de Verdad de las páginas 38 y 39

Recordemos que en las páginas 38 y 39 se aclaraba el uso que, los operadores lógicos **Y** y **O**, tenían respecto del idioma castellano. Y para ello, utilizábamos los ejemplos de las frases en castellano "Hoy es lunes **Y** empieza la semana" y "Hoy es lunes **O** empieza la semana".

Vamos a construir su Tabla de Verdad algorítmica de las anteriores frases.

CASO 1 Frase: Hoy es lunes **Y** empieza la semana

EN PAÍSES DE HABLA HISPANA

Valores que puede tomar la Variable "hoy"	Valores que puede tomar la Constante "EMPIEZALASEMANA"	Expresión algorítmica a evaluar: (hoy = "Lunes") Y (hoy = EMPIEZALASEMANA)	Resultado de evaluar la expresión
Lunes	LUNES	VERDADERO Y VERDADERO	VERDADERO
Martes	LUNES	FALSO Y FALSO	FALSO
Miércoles	LUNES	FALSO Y FALSO	FALSO
Jueves	LUNES	FALSO Y FALSO	FALSO
Viernes	LUNES	FALSO Y FALSO	FALSO
Sábado	LUNES	FALSO Y FALSO	FALSO
Domingo	LUNES	FALSO Y FALSO	FALSO
Cualquier otro valor	LUNES	FALSO Y FALSO	FALSO

EN PAÍSES DE HABLA ANGLOSAJONA

Valores que puede tomar la Variable "hoy"	Valores que puede tomar la Constante "EMPIEZALASEMANA"	Expresión algorítmica a evaluar: (hoy = "Lunes") Y (hoy = EMPIEZALASEMANA)	Resultado de evaluar la expresión
Lunes	DOMINGO	VERDADERO Y FALSO	FALSO
Martes	DOMINGO	FALSO Y FALSO	FALSO
Miércoles	DOMINGO	FALSO Y FALSO	FALSO
Jueves	DOMINGO	FALSO Y FALSO	FALSO
Viernes	DOMINGO	FALSO Y FALSO	FALSO
Sábado	DOMINGO	FALSO Y FALSO	FALSO
Domingo	DOMINGO	FALSO Y VERDADERO	FALSO
Cualquier otro valor	DOMINGO	FALSO Y FALSO	FALSO

A la vista de la Tabla de Verdad de la expresión algorítmica con el operador Y, en países de habla hispana, y; la expresión en idioma castellano es cierta (VERDADERO) solamente cuando las variables "hoy" toma el valor "Lunes" y realmente empieza la semana en "LUNES".
Lo mismo ocurre en la Tabla de Verdad en países de habla anglosajona: el operador Y es correcto en el idioma castellano porque para un país anglosajón nunca sería cierto (es FALSO) que el "Lunes" se empiece la semana. Porque para los países anglosajones, la semana empieza el "Domingo".

Por lo tanto, en el idioma castellano, el operador lógico Y se comporta igual que en un algoritmo computacional, estemos en un país de habla hispana o anglosajona.

Vamos ahora el caso del operador lógico **O** y su uso en el idioma castellano frente a un algoritmo de ordenador.

CASO 2 Frase: Hoy es lunes **O** empieza la semana

EN PAÍSES DE HABLA HISPANA

Valores que puede tomar la Variable "hoy"	Valores que puede tomar la Constante "EMPIEZALASEMANA"	Expresión algorítmica a evaluar: (hoy = "Lunes") O (hoy = EMPIEZALASEMANA)	Resultado de evaluar la expresión
Lunes	LUNES	VERDADERO O VERDADERO	VERDADERO
Martes	LUNES	FALSO O FALSO	FALSO
Miércoles	LUNES	FALSO O FALSO	FALSO
Jueves	LUNES	FALSO O FALSO	FALSO
Viernes	LUNES	FALSO O FALSO	FALSO
Sábado	LUNES	FALSO O FALSO	FALSO
Domingo	LUNES	FALSO O FALSO	FALSO
Cualquier otro valor	LUNES	FALSO O FALSO	FALSO

De la Tabla de Verdad se deduce que los valores de la expresión lógica, con el operador O, coincide con "lo esperado" a nuestra forma de interpretar el habla en el idioma castellano, en países de habla hispana: la semana solamente empieza si es lunes (valor VERDADERO de la expresión evaluada con el operador lógico O).

EN PAÍSES DE HABLA ANGLOSAJONA

Valores que puede tomar la Variable "hoy"	Valores que puede tomar la Constante "EMPIEZALASEMANA"	Expresión algorítmica a evaluar: (hoy = "Lunes") O (hoy = EMPIEZALASEMANA)	Resultado de evaluar la expresión
Lunes	DOMINGO	VERDADERO O FALSO	VERDADERO
Martes	DOMINGO	FALSO O FALSO	FALSO
Miércoles	DOMINGO	FALSO O FALSO	FALSO
Jueves	DOMINGO	FALSO O FALSO	FALSO
Viernes	DOMINGO	FALSO O FALSO	FALSO
Sábado	DOMINGO	FALSO O FALSO	FALSO
Domingo	DOMINGO	FALSO O VERDADERO	VERDADERO
Cualquier otro valor	DOMINGO	FALSO O FALSO	FALSO

De la Tabla de Verdad se ve que hay dos valores VERDADERO, y; se deduce que los valores de la expresión lógica, con el operador O, NO coincide con "lo esperado" a nuestra forma de interpretar el habla en el idioma castellano, en países de habla anglosajona. Ya que estando en "Lunes" también sería cierta (VERDADERO) la expresión lógica con el operador O; y en los países anglosajones una semana NO puede empezar un "Lunes", ya que, para los países anglosajones la semana empieza únicamente el "Domingo".

Conclusión, el operador lógico O NO "funciona" igual a lo "esperado" en el uso que hacemos a la disyunción "o" en el idioma castellano y cómo se comporta el operador lógico O en un algoritmo ejecutado en un ordenador.

Es muy importante que interiorice la diferencia entre la Disyunción "o" en castellano y el operador lógico O en un algoritmo de ordenador ya que, en caso contrario, cometerá errores en los resultados de la ejecución de sus algoritmos al evaluar expresiones lógicas al no distinguir usted cuándo utilizar el operador lógico Y o bien el operador lógico O.

Aplicaciones

Capítulo 7

Aplicaciones

Aprender a Programar | Ejemplos en PSeInt

Vamos a resolver varias aplicaciones informáticas casi reales. Para ello, nos basaremos en los ejemplos de aplicaciones siguientes:

→ Adivina Mi Número

→ Calculadora de varias Operaciones Matemáticas

→ Día de la Semana entre los años 1801 y 2100

→ Cifrado César

→ Contar Palabras Diferentes

→ Maître o Jefe de Sala: asignación de mesa

→ Jugar a la Máquina Tragaperras

Vamos con ellas.

APLICACIÓN

Adivina Mi Número

La aplicación consiste en adivinar un número generado aleatoriamente por el ordenador, entre 1 y 10. El usuario dispone de 5 intentos para adivinarlo y guardará cada uno de los cinco intentos para que, el usuario, se percate del número elegido en cada uno de los intentos anteriores.

CÓDIGO FUENTE

```
// Nombre de la aplicación: Adivina Mi Número
// Autor: José Luis Rodríguez Muñoz
// Fecha Última Actualización: 02-02-2022

Algoritmo AdivinaMiNumero

	//La dimensión de la tabla intentos es 4,
	//porque el máximo de intentos es 5.
	//Por lo tanto, no hace falta guardar el último intento
	//porque o bien acierta el número al azar o no,
	//en el último intento.
	Dimension intentos[4];
	// Llamamos al procedimiento que ejecuta todo el programa
	ProcesarAdivinaMiNumero( intentos )
FinAlgoritmo

SubProceso ProcesarAdivinaMiNumero( intentos )

	Definir  numeroAlAzar, unNumero, numeroIntentos Como Entero;
	Definir  unaCadena Como Cadena;
	Definir primerCaracter Como Caracter;

	MAXIMOINTENTOS = 5; //Definimos una constante

	//Obtenemos el número al azar que debe adivinar el usuario
	//este número estará entre 1 y 10.
	//El número al Azar no puede ser cero porque el cero
	//es el número para terminar el programa.
	Repetir
		numeroAlAzar = AZAR(11);
	Hasta Que numeroAlAzar <> 0
```

```
//Descomentar la siguiente línea de código
//para hacer las pruebas.
//A estas líneas de escritura
//se las denomina: líneas de chivatos.
//Escribir "Numero a adivinar: ", numeroAlAzar;

numeroIntentos = 1;

Escribir "Adivina mi número entre 1 y 10.";

Repetir
        //Mostramos el mensaje de inicio de jugadas
        Escribir "Tiene cinco intentos.”,
                 “ Teclee INTRO para salir del juego.";
        Escribir "Intento ", numeroIntentos,
                 ". Introduzca un número: ";
        Leer unaCadena;

        //Vamos a intentar mitigar que no nos introduzcan
        //una cadena de texto o de símbolos
        //para que no aborte el programa
        primerCaracter = SUBCADENA(unaCadena,1,1);
        Si primerCaracter <> "0" Y primerCaracter <> "1"
          Y primerCaracter <> "2" Y primerCaracter <> "3"
          Y primerCaracter <> "4" Y primerCaracter <> "5"
          Y primerCaracter <> "6" Y primerCaracter <> "7"
          Y primerCaracter <> "8" Y primerCaracter <> "9"
           Entonces
                //Salimos del programa
                //porque nos han introducido una cadena vacía
                //es decir, han pulsado INTRO
                //o bien nos han introducido un primer
                //caracter distinto de "0"-->"9"

                //NO podemos quitar todos los errores
                //si nos introducen la cadena: 2HOLA
                //nos dará un error que abortará el programa
                //porque no es un número
                unNumero = 0;
        SiNo
                unNumero = CONVERTIRANUMERO(
                                        unaCadena);
        FinSi

        Si unNumero = 0 Entonces
                numeroIntentos = MAXIMOINTENTOS;
                //Para salir del bucle
        SiNo
                Si unNumero < 0 O unNumero >= 11 Entonces
                        //El número introducido es inválido: debe
                        //estar entre 1 y 10
                        //El usuario pierde este turno
```

```
                Escribir "Ha perdido un intento. El número ",
                  " unNumero, " es un valor incorrecto.",
                  " El número debe estar entre 1 y 10";
SiNo
                Si unNumero = numeroAlAzar Entonces
                        //HA ganado el usuario
                        Escribir "Enhorabuena,",
                          " has acertado el número buscado";
                        numeroIntentos =
                                MAXIMOINTENTOS;
                        //Para salir del bucle
                SiNo
                        Si numeroIntentos =
                           MAXIMOINTENTOS Entonces
                                //El usuario ha perdido la
                                //partida
                                //ya que hemos llegado al
                                //número máximo
                                //de intentos y el número
                                //introducido
                                //es distinto del número
                                //generado al azar

                                Escribir "Ha ganado el
                                  " ordenador.
                                  " El número a adivinar era: ",
                                  numeroAlAzar;
                        SiNo
                                Si numeroIntentos > 1 Y
                                   NumeroYaIntentado(
                                           unNumero, intentos )
                                   Entonces
                                        //No es el primer intento
                                        //y ya ha introducido ese
                                        //número en otro intento
                                        Escribir "Ha perdido un",
                                          " intento. El número ",
                                          unNumero,
                                        " ya lo había intentado.";
                                SiNo
                                        //Guardamos el número
                                        //intentado en el índice
                                        //del intento de la tabla
                                        intentos[numeroIntentos] =
                                                        unNumero;

                                        //Mostramos el mensaje
                                        //de orientación
                                        //para el siguiente
                                        //intento del usuario
                                        Si unNumero <
                                                numeroAlAzar
                                          Entonces
```

```
                                                    Escribir
                                                    "Número”,
                                                    “ incorrecto. El”,
                                                    “ número a “,
                                                    “ adivinar es”,
                                                    “ mayor...";
                                                SiNo
                                                    Escribir
                                                "Número incorrecto.”,
                                                “ El número a adivinar”,
                                                “ es menor...";
                                                FinSi
                                            FinSi
                                        FinSi
                                    FinSi
                                FinSi
                            FinSi

                            numeroIntentos = numeroIntentos + 1;
                            //escribimos una línea en blanco para separar los
                            //mensaje
                            Escribir "";

        Hasta Que numeroIntentos > MAXIMOINTENTOS;

        Escribir "Bien jugado";
FinSubProceso

Funcion yaIntentado = NumeroYaIntentado( unNumero, intentos )
        Definir i Como Entero;
        Definir intentado Como Logico;
        intentado = FALSO;
        i = 1;

        Repetir

                Si intentos[i] = unNumero Entonces
                        intentado = VERDADERO;
                        i = 4; //Para salir del bucle
                FinSi
                i = i + 1;
        Hasta Que i > 4
        //Devolver el resultado de la función
        yaIntentado = intentado;

FinFuncion
```

DOCUMENTACIÓN

En esta solución hemos visto cómo filtrar los datos de entrada del usuario para no tener una ejecución que produzca un error fatal y se aborte el programa. Con esto, hemos mitigado una interrupción anómala del programa, en la medida de lo posible.

Hemos visto el uso de una **constante** y cómo tiene utilidad. Gracias a esta constante, solamente tenemos que cambiar un valor en un único sitio del programa, para que se modifique todo el código. Esto favorece la legibilidad del código fuente y, sobre todo la mantenibilidad del programa (modificarlo, en el futuro).

APLICACIÓN

Calculadora de varias Operaciones Matemáticas

La aplicación consiste en seleccionar una operación matemática por pantalla y ejecutarla para obtener el resultado de la operación. Las operaciones disponibles serán:
1. Factorial número, 2. Media de números, 3. Potencia, 4. Salir.

CÓDIGO FUENTE

```
// Nombre de la aplicación: Calculadora de varias Operaciones Matemáticas
// Autor: José Luis Rodríguez Muñoz
// Fecha Última Actualización: 02-02-2022

Algoritmo VariasOperaciones
        // Llamamos al procedimiento que ejecuta todo el programa
        ProcesarOpcionMenuPantalla
FinAlgoritmo

// MostrarMenu: procedimiento que muestra la
// opción de pantalla que ha de seleccionar
// el usuario en el menú mostrado en la pantalla
// del ordenador.
SubProceso  MostrarMenu
        // Escribimos por pantalla las opciones de menú
        // para el usuario del programa
        Escribir "1. Factorial número"
        Escribir "2. Media de números"
        Escribir "3. Potencia"
        Escribir "4. Salir"
FinSubProceso

// LeerNumero: función que solicita un número mientras
// el número introducido por el usuario no sea
// positivo o cero
```

```
Funcion unNumero = LeerNumero
        Definir elNumero Como Real
        // Inicialmente la variable de test del bucle Hasta a falso
        correcto = falso
        elNumero = 0

        Repetir
                //Escribir "Introduzca un número: "
                Leer elNumero
                Si elNumero >= 1 Entonces
                        // Solamente se aceptan los números positivos
                        correcto = verdadero
                SiNo
                        // Mensaje de error por pantalla
                        // No se aceptan los números negativos
                        Escribir "Los números deben estar ",
                                 "en el rango mayor que 0 "
                        Esperar 3 Segundos
                FinSi
        Hasta que correcto = verdadero
        // Otra forma de hacer el bucle es prescindir de la variable
        // de test y utilizar la condición del fin del bucle Hasta de
        // la forma: numero >= 0

        // Devolvemos el valor de la variable numero
        // en el parámetro de salida del algoritmo de la función
        unNumero = elNumero
FinFuncion

// LeerOpcion: función que devuelve un número solamente si
// está entre el rango de números del 1 al 4.
Funcion unaOpcion = LeerOpcion
        // Inicialmente la variable del bucle Hasta es falso
        correcto = falso
        laOpcion = 0
        Repetir
                Escribir "Introduzca un número de opción: "
                Leer laOpcion

                Segun laOpcion Hacer
                        1: correcto = verdadero
                        2: correcto = verdadero
                        3: correcto = verdadero
                        4: correcto = verdadero
                        De Otro Modo:
                                // Mensaje de error por pantalla
                                Escribir "El número debe estar entre el 1",
                                         " y el 4"
                                correcto = falso
                                Esperar 3 Segundos
                FinSegun
        Hasta que correcto = verdadero
```

```
        // Otra forma de hacer el bucle es prescindir de la variable
        // de test y utilizar la condición del fin del bucle Hasta de
        // la forma: opcion >= 1 AND opcion <= 4

        // Devolvemos el valor de la variable opcion
        // en el parámetro de salida del algoritmo de la función
        unaOpcion = laOpcion
FinFuncion

// ProcesarOpcionMenuPanttalla: procedimiento que ejecuta
// de las opciones que haya seleccionado el
// usuario hasta que éste selecciona la opción
// de Salir del programa en la pantalla.
SubProceso ProcesarOpcionMenuPantalla
        // Inicializamos el valor de la variable opción que es de tipo
        // entero a un valor diferente de las opciones disponibles.
        laOpcion = 0
        // Llamamos a los diferentes procedimientos que ejecutan
        // cada una de las opciones de usuario que elige en la
        // pantalla.
        Mientras laOpcion <> 4 Hacer
                // Llamamos al procedimiento que visualiza la pantalla
                // de menú para el usuario.
                MostrarMenu()
                laOpcion = LeerOpcion()
                Segun laOpcion Hacer
                        1: ProcesarFactorialNumero
                        2: ProcesarMediaDeNumeros
                        3: ProcesarPotencia
                        4: ProcesarSalir
                        De Otro Modo:
                                // Mensaje de error por pantalla
                                Escribir "Número de opción incorrecto"
                FinSegun
                Esperar 3 Segundos
                Borrar Pantalla
        FinMientras

        // se repite el bucle Mientras que la opción introducida
        // sea distinta de 4
FinSubProceso

// ProcesarSalir: procedimiento que finaliza la ejecución
// del programa
SubProceso ProcesarSalir
        Escribir "Fin del programa"
FinSubProceso
```

```
// ProcesarPotencia: procedimiento que lee dos números
// base y exponente y eleva el número base al número exponente.
SubProceso ProcesarPotencia
	Definir base, exponente, resultado Como Real
	// Declaramos e inicializamos las variables número a usar
	base = 0
	exponente = 0
	resultado = 1
	// resultado se inicializa a 1 porque vamos a realizar
	// una operación de multiplicación. Si inicializamos
	// a cero, entonces "valorDeLaVariable x 0" nos da cero.
	// Que es un error

	// Solicitamos los dos números al usuario
	Escribir "Introduzca el número Base: "
	base = LeerNumero()
	Escribir "Introduzca el número Exponente: "
	exponente = LeerNumero()

	// Ejecutamos el bucle Para para elevar el número
	// base al número exponente. Por ejemplo, Si base=2
	// y Si exponente=0, no entraría en el bucle y se devuelve 1
	// que sería el valor que tendría la variable resultado: 20 = 1
	Para i=1 Hasta exponente Con Paso 1 Hacer
		resultado = resultado * base
	FinPara

	// Mostramos por pantalla al usuario el resultado
	Escribir "El resultado del número ", base, " elevado al ",
		 exponente, " es: ", resultado
FinSubProceso


// ProcesarMediaDeNumeros: procedimiento que realiza
// la Media de una cantidad de números que se solicitan
// al usuario por pantalla.
SubProceso ProcesarMediaDeNumeros
	Definir resultado Como Real
	Definir cantidad, i Como Entero
	Dimension arrayNumeros[10]
	cantidad = 0
	resultado = 0
	// Introducimos la cantidad de números totales sobre los que
	// calcular la Media. La cantidad debe ser entre 1 y 10
	// Siendo 10 el tamaño máximo del Array.
	Repetir
		Escribir "Introduzca el total de números entre 1 y 10: "
		cantidad = TRUNC(LeerNumero())
	Hasta Que cantidad >= 1 Y cantidad <= 10
```

```
        // Rellenamos el Array de números
        Para i=1 Hasta cantidad Con Paso 1 Hacer
                Escribir "El número ", i, " de ", cantidad, " es : "
                arrayNumeros[i] = LeerNumero()
        FinPara

        // Visualizamos al usuario por pantalla los números
        // que ha introducido y hemos almacenado en el Array.
        Escribir "La relación de números introducidos, son: "

        Para i=1 Hasta cantidad Con Paso 1 Hacer
                Escribir "El numero ", i, " es: ", arrayNumeros[i]
                // El resultado de esta sentencia
                // para el valor de i=1 y suponiendo que el número
                // introducido por el usuario por pantalla es 24.
                // el resultado por pantalla de la primera línea
                // del bucle Para sería:
                // El numero 1 es 24
        FinPara

        // Calculamos la Media de los números del Array
        resultado = Media(arrayNumeros, cantidad)
        // Visualizamos por pantalla el resultado obtenido
        Escribir "La Media de los números es: ", resultado
        Esperar 3 Segundos
FinSubProceso

// Media: función que calcula la Media de los parámetros
// que recibe como entrada: un Array y una cantidad total.
Funcion laMedia = Media ( numerosMedia, cantidad)
        Definir resultado Como Real
        suma = 0
        resultado = 0
        // PASO 1: Calcular la suma de los números del Array
        Para i=1 Hasta cantidad Con Paso 1 Hacer
                suma = suma + numerosMedia[i]
        FinPara
        // PASO 2: Calcular la división de la suma entre la
        // cantidad total de números que hay en el Array
        resultado = suma / cantidad
        // La variable cantidad no es nunca cero (que daría error
        // en la división por intento de división por cero).
        // Y nunca será cero porque en el procedimiento
        // de llamada a la función nos hemos cerciorado que
        // la variable cantidad siempre va a estar entre 1 y 10
        // en la condición del bucle Hasta:
        // cantidad >= 1 AND cantidad <= 10

        // Devolvemos el valor de la variable resultado
        // en el parámetro de salida del algoritmo de la función
        laMedia = resultado
FinFuncion
```

```
// ProcesarFactorialNumero: procedimiento que calcula
// el factorial de un número introducido por el usuario.
SubProceso ProcesarFactorialNumero
        elNumero = 0
        resultado = 0
        // Solamente se admite el cálculo del factorial de números
        // positivos y distintos de cero
        Repetir
                Escribir "Introduzca el número a calcular el Factorial: "
                elNumero = TRUNC(LeerNumero())
        Hasta Que elNumero > 0
                // La condición del bucle Hasta elimina el numero cero
                // Calculamos el Factorial del número
        resultado = Factorial(elNumero)
                // Visualizamos por pantalla el resultado obtenido
        Escribir "El Factorial del número ", elNumero, " es: ", resultado
FinSubProceso

// Factorial: función que calcula
// el factorial de un número introducido como parámetro.
Funcion unFactorial = Factorial( elNumero )
        // La variable resultado no puede inicializarse a cero
        // porque vamos a hacer una multiplicación con la variable.
        resultado = 1
        contadorIteraciones = 0
        // Solamente se puede calcular el Factorial de
        // un número positivo y distinto del cero
        Si elNumero <= 0 Entonces
                resultado = 0
        SiNo
                // Calcular el Factorial de forma no recursiva: iterativa
                contadorIteraciones = elNumero
                Para i=1 Hasta elNumero Con Paso 1 Hacer
                        resultado = resultado * contadorIteraciones
                        contadorIteraciones = contadorIteraciones - 1
                FinPara
        FinSi

        // Devolvemos el valor de la variable resultado
        // en el parámetro de salida del algoritmo de la función
        unFactorial = resultado
FinFuncion
```

DOCUMENTACIÓN

En esta solución se ha primado un enfoque modular de las funcionalidades del programa: cada funcionalidad tiene su subrutina dedicada a la misma. Este enfoque de **Programación Estructurada** de la aplicación favorece la reutilización del código (poder utilizar una función determinada en otro programa para otra aplicación que tengamos que hacer).

APLICACIÓN

Día de la Semana entre los años 1801 y 2100

La aplicación consiste en recibir una cadena con una fecha, en un formato determinado; y, devolver el día de la semana que tiene esa fecha en formato texto.

Ejemplo:

Introduzca la fecha para saber el día de la semana.
Por ejemplo: 02-06-1968
- 28-01-2022

El día de la semana es:
- Viernes

ANÁLISIS DEL ALGORITMO DE LA SOLUCIÓN

EJEMPLO: ¿Qué día de la semana es el 3 de abril de 1999?.
SOLUCIÓN:
Localizado el año 1999 en la Tabla <<A>>, se busca en la Tabla <<B>> el número que, encontrándose en la misma fila horizontal que el citado año, corresponde a la columna del mes de Abril (en este caso el 4). Súmese a este número la cifra correspondiente al día buscado (3), y llévese el total (7) a la Tabla <<C>> donde encontramos que es **Sábado;** que es la solución buscada.

TABLA <<C>> DÍAS

D	1	8	15	22	29	36
L	2	9	16	23	30	37
M	3	10	17	24	31	
X	4	11	18	25	32	
J	5	12	19	26	33	
V	6	13	20	27	34	
S	7	14	21	28	35	

TABLA <<B>> MESES

Numero Fila	E	F	M	A	M	J	J	A	S	O	N	D
1	4	0	0	3	5	1	3	6	2	4	0	2
2	5	1	1	4	6	2	4	0	3	5	1	3
3	6	2	2	5	0	3	5	1	4	6	2	4
4	0	3	4	0	2	5	0	3	6	1	4	6
5	2	5	5	1	3	6	1	4	0	2	5	0
6	3	6	6	2	4	0	2	5	1	3	6	1
7	4	0	0	3	5	1	3	6	2	4	0	2
8	5	1	2	5	0	3	5	1	4	6	2	4
9	0	3	3	6	1	4	6	2	5	0	3	5
10	1	4	4	0	2	5	0	3	6	1	4	6
11	2	5	5	1	3	6	1	4	0	2	5	0
12	3	6	0	3	5	1	3	6	2	4	0	2
13	5	1	1	4	6	2	4	0	3	5	1	3
14	6	2	2	5	0	3	5	1	4	6	2	4
15	0	3	3	6	1	4	6	2	5	0	3	5
16	1	4	5	1	3	6	1	4	0	2	5	0
17	3	6	6	2	4	0	2	5	1	3	6	1
18	4	0	0	3	5	1	3	6	2	4	0	2
19	5	1	1	4	6	2	4	0	3	5	1	3
20	6	2	3	6	1	4	6	2	5	0	3	5
21	1	4	4	0	2	5	0	3	6	1	4	6
22	2	5	5	1	3	6	1	4	0	2	5	0
23	3	6	6	2	4	0	2	5	1	3	6	1
24	4	0	1	4	6	2	4	0	3	5	1	3
25	6	2	2	5	0	3	5	1	4	6	2	4
26	0	3	3	6	1	4	6	2	5	0	3	5
27	1	4	4	0	2	5	0	3	6	1	4	6
28	2	5	6	2	4	0	2	5	1	3	6	1

TABLA <<A>> AÑOS

Numero Fila	1801-1900				1901-2000				2001-2100				
1	1	29	57	85		25	53	81		9	37	65	93
2	2	30	58	86		26	54	82		10	38	66	94
3	3	31	59	87		27	55	83		11	39	67	95
4	4	32	60	88		28	56	84		12	40	68	96
5	5	33	61	89	1	29	57	85		13	41	69	97
6	6	34	62	90	2	30	58	86		14	42	70	98
7	7	35	63	91	3	31	59	87		15	43	71	99
8	8	36	64	92	4	32	60	88		16	44	72	00
9	9	37	65	93	5	33	61	89		17	45	73	
10	10	38	66	94	6	34	62	90		18	46	74	
11	11	39	67	95	7	35	63	91		19	47	75	
12	12	40	68	96	8	36	64	92		20	48	76	
13	13	41	69	97	9	37	65	93		21	49	77	
14	14	42	70	98	10	38	66	94		22	50	78	
15	15	43	71	99	11	39	67	95		23	51	79	
16	16	44	72		12	40	68	96		24	52	80	
17	17	45	73		13	41	69	97		25	53	81	
18	18	46	74		14	42	70	98		26	54	82	
19	19	47	75		15	43	71	99		27	55	83	
20	20	48	76		16	44	72	00		28	56	84	
21	21	49	77	00	17	45	73		1	29	57	85	
22	22	50	78		18	46	74		2	30	58	86	
23	23	51	79		19	47	75		3	31	59	87	
24	24	52	80		20	48	76		4	32	60	88	
25	25	53	81		21	49	77		5	33	61	89	
26	26	54	82		22	50	78		6	34	62	90	
27	27	55	83		23	51	79		7	35	63	91	
28	28	56	84		24	52	80		8	36	64	92	

CÓDIGO FUENTE

```
// Nombre de la aplicación: Día de la Semana
// Autor: José Luis Rodríguez Muñoz
// Fecha Última Actualización: 02-02-2022

Algoritmo DiaDeLaSemana

        //Definimos las tablas globales de la aplicación
        Dimension tablaA_1801_1900[28,4];
        Dimension tablaA_1901_2000[28,4];
        Dimension tablaA_2001_2100[28,5];
        Dimension tablaB[28,12];
        Dimension tablaC[7,6];
        Dimension diaSemana[7];

        // Llamamos al procedimiento que ejecuta todo el programa
        ProcesarDiaDeLaSemana( tablaA_1801_1900,
                                        tablaA_1901_2000,
                                        tablaA_2001_2100,
                                        tablaB, tablaC, diaSemana);

FinAlgoritmo

SubProceso ProcesarDiaDeLaSemana( tablaA_1801_1900,
tablaA_1901_2000, tablaA_2001_2100, tablaB, tablaC, diaSemana)

        //Este valor se pasará Por Valor a las subrutinas
        Definir fecha Como Cadena;
        //Variable de control de la condición de fin de bucle
        Definir fechaCorrecta Como Logico;
        //Estos valores se pasarán Por Refencia a las subrutinas
        Definir unDia, unMes, unAnio, filaBuscada, columnaBuscada,
                anioBuscado Como Entero;

        RellenarTablas( tablaA_1801_1900, tablaA_1901_2000,
                                tablaA_2001_2100, tablaB, tablaC, diaSemana);

        Repetir
                //Inicializamos las variables
                fecha = ""; //Asignamos la cadena vacía
                fechaCorrecta = FALSO;

                Escribir "Introduzca la fecha para saber el día de la”,
                        “ semana.";
                Escribir "Por ejemplo: 02-06-1968";
                Escribir "Pulse INTRO para finalizar."
                Leer fecha;
```

```
                Si fecha = "" Entonces
                        //Si nos introducen la cadena vacía
                        //es que han pulsado INTRO sin introducir una
                        //fecha
                        //entonces se finaliza el programa
                        fechaCorrecta = VERDADERO;
                SiNo
                        fechaCorrecta = ValidarFecha(fecha, unDia,
                                                     unMes, unAnio,
                                                     anioBuscado);
                        Si fechaCorrecta Entonces
                                //La fecha está validada en:
                                // --> Formato correcto: está sintácticamente
                                //bien formada
                                // --> Y es una fecha que existe entre los
                                //años 1801 y 2100
                                Escribir "El día de la semana es: ",
                                        EscribirFecha( unDia,
                                                       unMes, unAnio,
                                                       anioBuscado,
                                                       tablaA_1801_1900,
                                                       tablaA_1901_2000,
                                                       tablaA_2001_2100,
                                                       tablaB, tablaC,
                                                       diaSemana);
                        SiNo
                                //Se ha imprimido un Mensaje de Error
                                //Escribir "Fecha errónea.";
                                Esperar 3 segundos;

                                Borrar Pantalla;
                        FinSi
                FinSi

        Hasta Que fechaCorrecta = VERDADERO

FinSubProceso

Funcion fechaValida = ValidarFecha(fecha, unDia Por Referencia,
 unMes Por Referencia, unAnio Por Referencia,
 anioBuscado Por Referencia)

        //No validamos que la fecha esté demimitada
        //por los símbolos de guion. Es decir,
        //vale cualquier carácter que utilice el usuario.
        //Eso, siempre que el día, el mes y el año
        //se encuentren en la posición correcta.

        Si LONGITUD(fecha) <> 10 Entonces
                Escribir "Fecha errónea por no tener diez caracteres";
                esFechaValida = FALSO;
        SiNo
```

```
//Verificamos que nos han tecleado núimeros
esFechaValida = SinErroresSintacticos(fecha);
Si esFechaValida Entonces

//Extraemos el día de la fecha
unDia = CONVERTIRANUMERO(SUBCADENA(fecha,
                                   1,2));
//Escribir "El dia es: ", unDia;
Si unDia >= 1 Y unDia <= 31 Entonces

        //Extraemos el mes de la fecha
        unMes = CONVERTIRANUMERO(
                        SUBCADENA(fecha,
                                  4, 5));
        //Escribir "El mes es: ", unMes;
        Si unMes >= 1 Y unMes <= 12 Entonces

                //Extraemos el año de la fecha
                unAnio = CONVERTIRANUMERO(
                                SUBCADENA(fecha, 7, 10));
                //Escribir "El año es: ", unAnio;
                Si unAnio >= 1801 Y unAnio <= 2100
                         Entonces
                        //Reasignamos el Año para que nos
                        //sirva como valor
                        //de búsqueda en la tabla A. De esta
                        //forma
                        //esta subrutina es la única que trata
                        //con la cadena fecha
                        anioBuscado =
                                CONVERTIRANUMERO(
                                        SUBCADENA(fecha, 9, 10));
                        //Escribir "El número del año buscado
                        //es: ", anioBuscado;
                        Si anioBuscado = 0 Entonces
                                //Por los valores que les
                                //hemos dado a las tablas
                                //el valor 00 de la tabla origen,
                                //corresponde
                                //a un valor 100 en nuestra
                                //tabla de trabajo
                                //de la aplicación.
                                anioBuscado = 100;
                        FinSi
                        esFechaValida = VERDADERO;
                SiNo
                        Escribir "Fecha errónea porque no”,
                                 “ es un año de una fecha.”,
                                  “Años válidos: 1801-->2100";
                        esFechaValida = FALSO;
                FinSi
        SiNo
```

```
                        Escribir "Fecha errónea porque no es un”,
                                 “ mes de una fecha";
                        esFechaValida = FALSO;
                    FinSi
                SiNo
                    Escribir "Fecha errónea porque no es un día de”,
                             “ una fecha";
                    esFechaValida = FALSO;
                FinSi
            FinSi
        FinSi
        //Devolvemos el valor de retorno de la función
        fechaValida = esFechaValida;
FinFuncion

Funcion elDia = EscribirFecha( unDia Por Referencia,
 unMes Por Referencia, unAnio Por Referencia,
 anioBuscado Por Referencia, tablaA_1801_1900,
 tablaA_1901_2000, tablaA_2001_2100, tablaB, tablaC, diaSemana)

        Definir valorAnioBuscado, valorDiaBuscado,
                 filaBuscada Como Entero;

        Si unAnio >= 1801 Y unAnio <= 1900 Entonces
                valorAnioBuscado = BuscarValor ( tablaA_1801_1900,
                                                 28, 4, anioBuscado,
                                                 filaBuscada);
                Si valorAnioBuscado = 999 Entonces
                        Escribir "Error en datos: error en tabla”,
                                 “ tablaA_1801_1900";
                FinSi
        FinSi
        Si unAnio >= 1901 Y unAnio <= 2000 Entonces
                valorAnioBuscado = BuscarValor ( tablaA_1901_2000,
                                                 28, 4, anioBuscado,
                                                 filaBuscada);
                Si valorAnioBuscado = 999 Entonces
                        Escribir "Error en datos: error en tabla”,
                                 “ tablaA_1901_2000";
                FinSi
        FinSi
        Si unAnio >= 2001 Y unAnio <= 2100 Entonces
                valorAnioBuscado = BuscarValor ( tablaA_2001_2100,
                                                 28, 5, anioBuscado,
                                                 filaBuscada);
                Si valorAnioBuscado = 999 Entonces
                        Escribir "Error en datos: error en tabla”,
                                 “ tablaA_2001_2100";
                FinSi
        FinSi
```

```
        Si valorAnioBuscado <> 999 Entonces
                valorDiaBuscado = BuscarValor ( tablaC, 7, 6,
                                                (tablaB[filaBuscada,unMes] +
                                                unDia), filaBuscada);
                Si valorDiaBuscado <> 999 Entonces
                        //Devolvemos el valor de retorno de la función
                        elDia = diaSemana[filaBuscada];
                SiNo
                        Escribir "Error en datos: error en tabla tablaC";
                        elDia = 999;
                FinSi
        FinSi
FinFuncion

Funcion unValor = BuscarValor ( tablaBuscada, limiteFilas, limiteColumnas,
valorABuscar, filaBuscada Por Referencia)
        Definir encontrado Como Logico;
        Definir i, j, elValorBuscado Como Entero;
        encontrado = FALSO;
        i = 1;
        j = 1;

        Mientras i <= limiteFilas Y encontrado <> VERDADERO Hacer
                Mientras j <= limiteColumnas Y
                          encontrado <> VERDADERO Hacer
                Si tablaBuscada[i,j] = valorABuscar Entonces
                        elValorBuscado = tablaBuscada[i,j];
                        filaBuscada = i;
                        encontrado = VERDADERO;
                FinSi
                j = j + 1;
                FinMientras
                i = i + 1;
                j = 1;
        FinMientras

        Si No encontrado Entonces
                Escribir "Error de datos. Póngase en contacto con el",
                         " informático.";
                //Devolvemos el valor de retorno de la función
                unValor = 999;
        SiNo
                //Devolvemos el valor de retorno de la función
                unValor = elValorBuscado;
        FinSi

FinFuncion
```

```
SubProceso RellenarTablas (tablaA_1801_1900, tablaA_1901_2000,
 tablaA_2001_2100, tablaB, tablaC, diaSemana)
        //inicializamos las tablas fijas

        //TABLA_A_1801_1900
        tablaA_1801_1900[1,1]=1;    tablaA_1801_1900[1,2]=29;
        tablaA_1801_1900[1,3]=57;   tablaA_1801_1900[1,4]=85;
        tablaA_1801_1900[2,1]=2;    tablaA_1801_1900[2,2]=30;
        tablaA_1801_1900[2,3]=58;   tablaA_1801_1900[2,4]=86;
        tablaA_1801_1900[3,1]=3;    tablaA_1801_1900[3,2]=31;
        tablaA_1801_1900[3,3]=59;   tablaA_1801_1900[3,4]=87;
        tablaA_1801_1900[4,1]=4;    tablaA_1801_1900[4,2]=32;
        tablaA_1801_1900[4,3]=60;   tablaA_1801_1900[4,4]=88;
        tablaA_1801_1900[5,1]=5;    tablaA_1801_1900[5,2]=33;
        tablaA_1801_1900[5,3]=61;   tablaA_1801_1900[5,4]=89;
        tablaA_1801_1900[6,1]=6;    tablaA_1801_1900[6,2]=34;
        tablaA_1801_1900[6,3]=62;   tablaA_1801_1900[6,4]=90;
        tablaA_1801_1900[7,1]=7;    tablaA_1801_1900[7,2]=35;
        tablaA_1801_1900[7,3]=63;   tablaA_1801_1900[7,4]=91;
        tablaA_1801_1900[8,1]=8;    tablaA_1801_1900[8,2]=36;
        tablaA_1801_1900[8,3]=64;   tablaA_1801_1900[8,4]=92;
        tablaA_1801_1900[9,1]=9;    tablaA_1801_1900[9,2]=37;
        tablaA_1801_1900[9,3]=65;   tablaA_1801_1900[9,4]=93;
        tablaA_1801_1900[10,1]=10;  tablaA_1801_1900[10,2]=38;
        tablaA_1801_1900[10,3]=66;  tablaA_1801_1900[10,4]=94;
        tablaA_1801_1900[11,1]=11;  tablaA_1801_1900[11,2]=39;
        tablaA_1801_1900[11,3]=67;  tablaA_1801_1900[11,4]=95;
        tablaA_1801_1900[12,1]=12;  tablaA_1801_1900[12,2]=40;
        tablaA_1801_1900[12,3]=68;  tablaA_1801_1900[12,4]=96;
        tablaA_1801_1900[13,1]=13;  tablaA_1801_1900[13,2]=41;
        tablaA_1801_1900[13,3]=69;  tablaA_1801_1900[13,4]=97;
        tablaA_1801_1900[14,1]=14;  tablaA_1801_1900[14,2]=42;
        tablaA_1801_1900[14,3]=70;  tablaA_1801_1900[14,4]=98;
        tablaA_1801_1900[15,1]=15;  tablaA_1801_1900[15,2]=43;
        tablaA_1801_1900[15,3]=71;  tablaA_1801_1900[15,4]=99;
        tablaA_1801_1900[16,1]=16;  tablaA_1801_1900[16,2]=44;
        tablaA_1801_1900[16,3]=72;  tablaA_1801_1900[16,4]=0;
        tablaA_1801_1900[17,1]=17;  tablaA_1801_1900[17,2]=45;
        tablaA_1801_1900[17,3]=73;  tablaA_1801_1900[17,4]=0;
        tablaA_1801_1900[18,1]=18;  tablaA_1801_1900[18,2]=46;
        tablaA_1801_1900[18,3]=74;  tablaA_1801_1900[18,4]=0;
        tablaA_1801_1900[19,1]=19;  tablaA_1801_1900[19,2]=47;
        tablaA_1801_1900[19,3]=75;  tablaA_1801_1900[19,4]=0;
        tablaA_1801_1900[20,1]=20;  tablaA_1801_1900[20,2]=48;
        tablaA_1801_1900[20,3]=76;  tablaA_1801_1900[20,4]=0;
        tablaA_1801_1900[21,1]=21;  tablaA_1801_1900[21,2]=49;
        tablaA_1801_1900[21,3]=77;  tablaA_1801_1900[21,4]=100;
        tablaA_1801_1900[22,1]=22;  tablaA_1801_1900[22,2]=50;
        tablaA_1801_1900[22,3]=78;  tablaA_1801_1900[22,4]=0;
        tablaA_1801_1900[23,1]=23;  tablaA_1801_1900[23,2]=51;
        tablaA_1801_1900[23,3]=79;  tablaA_1801_1900[23,4]=0;
        tablaA_1801_1900[24,1]=24;  tablaA_1801_1900[24,2]=52;
```

```
tablaA_1801_1900[24,3]=80;  tablaA_1801_1900[24,4]=0;
tablaA_1801_1900[25,1]=25;  tablaA_1801_1900[25,2]=53;
tablaA_1801_1900[25,3]=81;  tablaA_1801_1900[25,4]=0;
tablaA_1801_1900[26,1]=26;  tablaA_1801_1900[26,2]=54;
tablaA_1801_1900[26,3]=82;  tablaA_1801_1900[26,4]=0;
tablaA_1801_1900[27,1]=27;  tablaA_1801_1900[27,2]=55;
tablaA_1801_1900[27,3]=83;  tablaA_1801_1900[27,4]=0;
tablaA_1801_1900[28,1]=28;  tablaA_1801_1900[28,2]=56;
tablaA_1801_1900[28,3]=84;  tablaA_1801_1900[28,4]=0;


//TABLA_A_tablaA_1901_200
tablaA_1901_2000[1,1]=0;    tablaA_1901_2000[1,2]=25;
tablaA_1901_2000[1,3]=53;   tablaA_1901_2000[1,4]=81;
tablaA_1901_2000[2,1]=0;    tablaA_1901_2000[2,2]=26;
tablaA_1901_2000[2,3]=54;   tablaA_1901_2000[2,4]=82;
tablaA_1901_2000[3,1]=0;    tablaA_1901_2000[3,2]=27;
tablaA_1901_2000[3,3]=55;   tablaA_1901_2000[3,4]=83;
tablaA_1901_2000[4,1]=0;    tablaA_1901_2000[4,2]=28;
tablaA_1901_2000[4,3]=56;   tablaA_1901_2000[4,4]=84;
tablaA_1901_2000[5,1]=1;    tablaA_1901_2000[5,2]=29;
tablaA_1901_2000[5,3]=57;   tablaA_1901_2000[5,4]=85;
tablaA_1901_2000[6,1]=2;    tablaA_1901_2000[6,2]=30;
tablaA_1901_2000[6,3]=58;   tablaA_1901_2000[6,4]=86;
tablaA_1901_2000[7,1]=3;    tablaA_1901_2000[7,2]=31;
tablaA_1901_2000[7,3]=59;   tablaA_1901_2000[7,4]=87;
tablaA_1901_2000[8,1]=4;    tablaA_1901_2000[8,2]=32;
tablaA_1901_2000[8,3]=60;   tablaA_1901_2000[8,4]=88;
tablaA_1901_2000[9,1]=5;    tablaA_1901_2000[9,2]=33;
tablaA_1901_2000[9,3]=61;   tablaA_1901_2000[9,4]=89;
tablaA_1901_2000[10,1]=6;   tablaA_1901_2000[10,2]=34;
tablaA_1901_2000[10,3]=62;  tablaA_1901_2000[10,4]=90;
tablaA_1901_2000[11,1]=7;   tablaA_1901_2000[11,2]=35;
tablaA_1901_2000[11,3]=63;  tablaA_1901_2000[11,4]=91;
tablaA_1901_2000[12,1]=8;   tablaA_1901_2000[12,2]=36;
tablaA_1901_2000[12,3]=64;  tablaA_1901_2000[12,4]=92;
tablaA_1901_2000[13,1]=9;   tablaA_1901_2000[13,2]=37;
tablaA_1901_2000[13,3]=65;  tablaA_1901_2000[13,4]=93;
tablaA_1901_2000[14,1]=10;  tablaA_1901_2000[14,2]=38;
tablaA_1901_2000[14,3]=66;  tablaA_1901_2000[14,4]=94;
tablaA_1901_2000[15,1]=11;  tablaA_1901_2000[15,2]=39;
tablaA_1901_2000[15,3]=67;  tablaA_1901_2000[15,4]=95;
tablaA_1901_2000[16,1]=12;  tablaA_1901_2000[16,2]=40;
tablaA_1901_2000[16,3]=68;  tablaA_1901_2000[16,4]=96;
tablaA_1901_2000[17,1]=13;  tablaA_1901_2000[17,2]=41;
tablaA_1901_2000[17,3]=69;  tablaA_1901_2000[17,4]=97;
tablaA_1901_2000[18,1]=14;  tablaA_1901_2000[18,2]=42;
tablaA_1901_2000[18,3]=70;  tablaA_1901_2000[18,4]=98;
tablaA_1901_2000[19,1]=15;  tablaA_1901_2000[19,2]=43;
tablaA_1901_2000[19,3]=71;  tablaA_1901_2000[19,4]=99;
tablaA_1901_2000[20,1]=16;  tablaA_1901_2000[20,2]=44;
tablaA_1901_2000[20,3]=72;  tablaA_1901_2000[20,4]=100;
tablaA_1901_2000[21,1]=17;  tablaA_1901_2000[21,2]=45;
```

```
tablaA_1901_2000[21,3]=73;  tablaA_1901_2000[21,4]=0;
tablaA_1901_2000[22,1]=18;  tablaA_1901_2000[22,2]=46;
tablaA_1901_2000[22,3]=74;  tablaA_1901_2000[22,4]=0;
tablaA_1901_2000[23,1]=19;  tablaA_1901_2000[23,2]=47;
tablaA_1901_2000[23,3]=75;  tablaA_1901_2000[23,4]=0;
tablaA_1901_2000[24,1]=20;  tablaA_1901_2000[24,2]=48;
tablaA_1901_2000[24,3]=76;  tablaA_1901_2000[24,4]=0;
tablaA_1901_2000[25,1]=21;  tablaA_1901_2000[25,2]=49;
tablaA_1901_2000[25,3]=77;  tablaA_1901_2000[25,4]=0;
tablaA_1901_2000[26,1]=22;  tablaA_1901_2000[26,2]=50;
tablaA_1901_2000[26,3]=78;  tablaA_1901_2000[26,4]=0;
tablaA_1901_2000[27,1]=23;  tablaA_1901_2000[27,2]=51;
tablaA_1901_2000[27,3]=79;  tablaA_1901_2000[27,4]=0;
tablaA_1901_2000[28,1]=24;  tablaA_1901_2000[28,2]=52;
tablaA_1901_2000[28,3]=80;  tablaA_1901_2000[28,4]=0;


//tabla_A_2001_2100
tablaA_2001_2100[1,1]=0;    tablaA_2001_2100[1,2]=9;
tablaA_2001_2100[1,3]=37;   tablaA_2001_2100[1,4]=65;
tablaA_2001_2100[1,5]=93;
tablaA_2001_2100[2,1]=0;    tablaA_2001_2100[2,2]=10;
tablaA_2001_2100[2,3]=38;   tablaA_2001_2100[2,4]=66;
tablaA_2001_2100[2,5]=94;
tablaA_2001_2100[3,1]=0;    tablaA_2001_2100[3,2]=11;
tablaA_2001_2100[3,3]=39;   tablaA_2001_2100[3,4]=67;
tablaA_2001_2100[3,5]=95;
tablaA_2001_2100[4,1]=0;    tablaA_2001_2100[4,2]=12;
tablaA_2001_2100[4,3]=40;   tablaA_2001_2100[4,4]=68;
tablaA_2001_2100[4,5]=96;
tablaA_2001_2100[5,1]=0;    tablaA_2001_2100[5,2]=13;
tablaA_2001_2100[5,3]=41;   tablaA_2001_2100[5,4]=69;
tablaA_2001_2100[5,5]=97;
tablaA_2001_2100[6,1]=0;    tablaA_2001_2100[6,2]=14;
tablaA_2001_2100[6,3]=42;   tablaA_2001_2100[6,4]=70;
tablaA_2001_2100[6,5]=98;
tablaA_2001_2100[7,1]=0;    tablaA_2001_2100[7,2]=15;
tablaA_2001_2100[7,3]=43;   tablaA_2001_2100[7,4]=71;
tablaA_2001_2100[7,5]=99;
tablaA_2001_2100[8,1]=0;    tablaA_2001_2100[8,2]=16;
tablaA_2001_2100[8,3]=44;   tablaA_2001_2100[8,4]=72;
tablaA_2001_2100[8,5]=100;
tablaA_2001_2100[9,1]=0;    tablaA_2001_2100[9,2]=17;
tablaA_2001_2100[9,3]=45;   tablaA_2001_2100[9,4]=73;
tablaA_2001_2100[9,5]=0;
tablaA_2001_2100[10,1]=0;   tablaA_2001_2100[10,2]=18;
tablaA_2001_2100[10,3]=46;  tablaA_2001_2100[10,4]=74;
tablaA_2001_2100[10,5]=0;
tablaA_2001_2100[11,1]=0;   tablaA_2001_2100[11,2]=19;
tablaA_2001_2100[11,3]=47;  tablaA_2001_2100[11,4]=75;
tablaA_2001_2100[11,5]=0;
tablaA_2001_2100[12,1]=0;   tablaA_2001_2100[12,2]=20;
tablaA_2001_2100[12,3]=48;  tablaA_2001_2100[12,4]=76;
```

```
tablaA_2001_2100[12,5]=0;
tablaA_2001_2100[13,1]=0;   tablaA_2001_2100[13,2]=21;
tablaA_2001_2100[13,3]=49;  tablaA_2001_2100[13,4]=77;
tablaA_2001_2100[13,5]=0;
tablaA_2001_2100[14,1]=0;   tablaA_2001_2100[14,2]=22;
tablaA_2001_2100[14,3]=50;  tablaA_2001_2100[14,4]=78;
tablaA_2001_2100[14,5]=0;
tablaA_2001_2100[15,1]=0;   tablaA_2001_2100[15,2]=23;
tablaA_2001_2100[15,3]=51;  tablaA_2001_2100[15,4]=79;
tablaA_2001_2100[15,5]=0;
tablaA_2001_2100[16,1]=0;   tablaA_2001_2100[16,2]=24;
tablaA_2001_2100[16,3]=52;  tablaA_2001_2100[16,4]=80;
tablaA_2001_2100[16,5]=0;
tablaA_2001_2100[17,1]=0;   tablaA_2001_2100[17,2]=25;
tablaA_2001_2100[17,3]=53;  tablaA_2001_2100[17,4]=81;
tablaA_2001_2100[17,5]=0;
tablaA_2001_2100[18,1]=0;   tablaA_2001_2100[18,2]=26;
tablaA_2001_2100[18,3]=54;  tablaA_2001_2100[18,4]=82;
tablaA_2001_2100[18,5]=0;
tablaA_2001_2100[19,1]=0;   tablaA_2001_2100[19,2]=27;
tablaA_2001_2100[19,3]=55;  tablaA_2001_2100[19,4]=83;
tablaA_2001_2100[19,5]=0;
tablaA_2001_2100[20,1]=0;   tablaA_2001_2100[20,2]=28;
tablaA_2001_2100[20,3]=56;  tablaA_2001_2100[20,4]=84;
tablaA_2001_2100[20,5]=0;
tablaA_2001_2100[21,1]=1;   tablaA_2001_2100[21,2]=29;
tablaA_2001_2100[21,3]=57;  tablaA_2001_2100[21,4]=85;
tablaA_2001_2100[21,5]=0;
tablaA_2001_2100[22,1]=2;   tablaA_2001_2100[22,2]=30;
tablaA_2001_2100[22,3]=58;  tablaA_2001_2100[22,4]=86;
tablaA_2001_2100[22,5]=0;
tablaA_2001_2100[23,1]=3;   tablaA_2001_2100[23,2]=31;
tablaA_2001_2100[23,3]=59;  tablaA_2001_2100[23,4]=87;
tablaA_2001_2100[23,5]=0;
tablaA_2001_2100[24,1]=4;   tablaA_2001_2100[24,2]=32;
tablaA_2001_2100[24,3]=60;  tablaA_2001_2100[24,4]=88;
tablaA_2001_2100[24,5]=0;
tablaA_2001_2100[25,1]=5;   tablaA_2001_2100[25,2]=33;
tablaA_2001_2100[25,3]=61;  tablaA_2001_2100[25,4]=89;
tablaA_2001_2100[25,5]=0;
tablaA_2001_2100[26,1]=6;   tablaA_2001_2100[26,2]=34;
tablaA_2001_2100[26,3]=62;  tablaA_2001_2100[26,4]=90;
tablaA_2001_2100[26,5]=0;
tablaA_2001_2100[27,1]=7;   tablaA_2001_2100[27,2]=35;
tablaA_2001_2100[27,3]=63;  tablaA_2001_2100[27,4]=91;
tablaA_2001_2100[27,5]=0;
tablaA_2001_2100[28,1]=8;   tablaA_2001_2100[28,2]=36;
tablaA_2001_2100[28,3]=64;  tablaA_2001_2100[28,4]=92;
tablaA_2001_2100[28,5]=0;

//TABLA_B
tablaB[1,1]=4; tablaB[1,2]=0; tablaB[1,3]=0; tablaB[1,4]=3; tablaB[1,5]=5;
tablaB[1,6]=1; tablaB[1,7]=3; tablaB[1,8]=6; tablaB[1,9]=2; tablaB[1,10]=4;
```

```
tablaB[1,11]=0;        tablaB[1,12]=2;
tablaB[2,1]=5; tablaB[2,2]=1; tablaB[2,3]=1; tablaB[2,4]=4; tablaB[2,5]=6;
tablaB[2,6]=2; tablaB[2,7]=4; tablaB[2,8]=0; tablaB[2,9]=3; tablaB[2,10]=5;
tablaB[2,11]=1;        tablaB[2,12]=3;
tablaB[3,1]=6; tablaB[3,2]=2; tablaB[3,3]=2; tablaB[3,4]=5; tablaB[3,5]=0;
tablaB[3,6]=3; tablaB[3,7]=5; tablaB[3,8]=1; tablaB[3,9]=4; tablaB[3,10]=6;
tablaB[3,11]=2;        tablaB[3,12]=4;
tablaB[4,1]=0; tablaB[4,2]=3; tablaB[4,3]=4; tablaB[4,4]=0; tablaB[4,5]=2;
tablaB[4,6]=5; tablaB[4,7]=0; tablaB[4,8]=3; tablaB[4,9]=6; tablaB[4,10]=1;
tablaB[4,11]=4;        tablaB[4,12]=6;
tablaB[5,1]=2; tablaB[5,2]=5; tablaB[5,3]=5; tablaB[5,4]=1; tablaB[5,5]=3;
tablaB[5,6]=6; tablaB[5,7]=1; tablaB[5,8]=4; tablaB[5,9]=0; tablaB[5,10]=2;
tablaB[5,11]=5;        tablaB[5,12]=0;
tablaB[6,1]=3; tablaB[6,2]=6; tablaB[6,3]=6; tablaB[6,4]=2; tablaB[6,5]=4;
tablaB[6,6]=0; tablaB[6,7]=2; tablaB[6,8]=5; tablaB[6,9]=1; tablaB[6,10]=3;
tablaB[6,11]=6;        tablaB[6,12]=1;
tablaB[7,1]=4; tablaB[7,2]=0; tablaB[7,3]=0; tablaB[7,4]=3; tablaB[7,5]=5;
tablaB[7,6]=1; tablaB[7,7]=3; tablaB[7,8]=6; tablaB[7,9]=2; tablaB[7,10]=4;
tablaB[7,11]=0;        tablaB[7,12]=2;
tablaB[8,1]=5; tablaB[8,2]=1; tablaB[8,3]=2; tablaB[8,4]=5; tablaB[8,5]=0;
tablaB[8,6]=3; tablaB[8,7]=5; tablaB[8,8]=1; tablaB[8,9]=4; tablaB[8,10]=6;
tablaB[8,11]=2;        tablaB[8,12]=4;
tablaB[9,1]=0; tablaB[9,2]=3; tablaB[9,3]=3; tablaB[9,4]=6; tablaB[9,5]=1;
tablaB[9,6]=4; tablaB[9,7]=6; tablaB[9,8]=2; tablaB[9,9]=5; tablaB[9,10]=0;
tablaB[9,11]=3;        tablaB[9,12]=5;
tablaB[10,1]=1;        tablaB[10,2]=4;        tablaB[10,3]=4;
tablaB[10,4]=0;        tablaB[10,5]=2;        tablaB[10,6]=5;
tablaB[10,7]=0;        tablaB[10,8]=3;        tablaB[10,9]=6;
tablaB[10,10]=1;       tablaB[10,11]=4;       tablaB[10,12]=6;
tablaB[11,1]=2;        tablaB[11,2]=5;        tablaB[11,3]=5;
tablaB[11,4]=1;        tablaB[11,5]=3;        tablaB[11,6]=6;
tablaB[11,7]=1;        tablaB[11,8]=4;        tablaB[11,9]=0;
tablaB[11,10]=2;       tablaB[11,11]=5;       tablaB[11,12]=0;
tablaB[12,1]=3;        tablaB[12,2]=6;        tablaB[12,3]=0;
tablaB[12,4]=3;        tablaB[12,5]=5;        tablaB[12,6]=1;
tablaB[12,7]=3;        tablaB[12,8]=6;        tablaB[12,9]=2;
tablaB[12,10]=4;       tablaB[12,11]=0;       tablaB[12,12]=2;
tablaB[13,1]=5;        tablaB[13,2]=1;        tablaB[13,3]=1;
tablaB[13,4]=4;        tablaB[13,5]=6;        tablaB[13,6]=2;
tablaB[13,7]=4;        tablaB[13,8]=0;        tablaB[13,9]=3;
tablaB[13,10]=5;       tablaB[13,11]=1;       tablaB[13,12]=3;
tablaB[14,1]=6;        tablaB[14,2]=2;        tablaB[14,3]=2;
tablaB[14,4]=5;        tablaB[14,5]=0;        tablaB[14,6]=3;
tablaB[14,7]=5;        tablaB[14,8]=1;        tablaB[14,9]=4;
tablaB[14,10]=6;       tablaB[14,11]=2;       tablaB[14,12]=4;
tablaB[15,1]=0;        tablaB[15,2]=3;        tablaB[15,3]=3;
tablaB[15,4]=6;        tablaB[15,5]=1;        tablaB[15,6]=4;
tablaB[15,7]=6;        tablaB[15,8]=2;        tablaB[15,9]=5;
tablaB[15,10]=0;       tablaB[15,11]=3;       tablaB[15,12]=5;
tablaB[16,1]=1;        tablaB[16,2]=4;        tablaB[16,3]=5;
tablaB[16,4]=1;        tablaB[16,5]=3;        tablaB[16,6]=6;
tablaB[16,7]=1;        tablaB[16,8]=4;        tablaB[16,9]=0;
tablaB[16,10]=2;       tablaB[16,11]=5;       tablaB[16,12]=0;
```

```
tablaB[17,1]=3;     tablaB[17,2]=6;     tablaB[17,3]=6;
tablaB[17,4]=2;     tablaB[17,5]=4;     tablaB[17,6]=0;
tablaB[17,7]=2;     tablaB[17,8]=5;     tablaB[17,9]=1;
tablaB[17,10]=3;    tablaB[17,11]=6;    tablaB[17,12]=1;
tablaB[18,1]=4;     tablaB[18,2]=0;     tablaB[18,3]=0;
tablaB[18,4]=3;     tablaB[18,5]=5;     tablaB[18,6]=1;
tablaB[18,7]=3;     tablaB[18,8]=6;     tablaB[18,9]=2;
tablaB[18,10]=4;    tablaB[18,11]=0;    tablaB[18,12]=2;
tablaB[19,1]=5;     tablaB[19,2]=1;     tablaB[19,3]=1;
tablaB[19,4]=4;     tablaB[19,5]=6;     tablaB[19,6]=2;
tablaB[19,7]=4;     tablaB[19,8]=0;     tablaB[19,9]=3;
tablaB[19,10]=5;    tablaB[19,11]=1;    tablaB[19,12]=3;
tablaB[20,1]=6;     tablaB[20,2]=2;     tablaB[20,3]=3;
tablaB[20,4]=6;     tablaB[20,5]=1;     tablaB[20,6]=4;
tablaB[20,7]=6;     tablaB[20,8]=2;     tablaB[20,9]=5;
tablaB[20,10]=0;    tablaB[20,11]=3;    tablaB[20,12]=5;
tablaB[21,1]=1;     tablaB[21,2]=4;     tablaB[21,3]=4;
tablaB[21,4]=0;     tablaB[21,5]=2;     tablaB[21,6]=5;
tablaB[21,7]=0;     tablaB[21,8]=3;     tablaB[21,9]=6;
tablaB[21,10]=1;    tablaB[21,11]=4;    tablaB[21,12]=6;
tablaB[22,1]=2;     tablaB[22,2]=5;     tablaB[22,3]=5;
tablaB[22,4]=1;     tablaB[22,5]=3;     tablaB[22,6]=6;
tablaB[22,7]=1;     tablaB[22,8]=4;     tablaB[22,9]=0;
tablaB[22,10]=2;    tablaB[22,11]=5;    tablaB[22,12]=0;
tablaB[23,1]=3;     tablaB[23,2]=6;     tablaB[23,3]=6;
tablaB[23,4]=2;     tablaB[23,5]=4;     tablaB[23,6]=0;
tablaB[23,7]=2;     tablaB[23,8]=5;     tablaB[23,9]=1;
tablaB[23,10]=3;    tablaB[23,11]=6;    tablaB[23,12]=1;
tablaB[24,1]=4;     tablaB[24,2]=0;     tablaB[24,3]=1;
tablaB[24,4]=4;     tablaB[24,5]=6;     tablaB[24,6]=2;
tablaB[24,7]=4;     tablaB[24,8]=0;     tablaB[24,9]=3;
tablaB[24,10]=5;    tablaB[24,11]=1;    tablaB[24,12]=3;
tablaB[25,1]=6;     tablaB[25,2]=2;     tablaB[25,3]=2;
tablaB[25,4]=5;     tablaB[25,5]=0;     tablaB[25,6]=3;
tablaB[25,7]=5;     tablaB[25,8]=1;     tablaB[25,9]=4;
tablaB[25,10]=6;    tablaB[25,11]=2;    tablaB[25,12]=4;
tablaB[26,1]=0;     tablaB[26,2]=3;     tablaB[26,3]=3;
tablaB[26,4]=6;     tablaB[26,5]=1;     tablaB[26,6]=4;
tablaB[26,7]=6;     tablaB[26,8]=2;     tablaB[26,9]=5;
tablaB[26,10]=0;    tablaB[26,11]=3;    tablaB[26,12]=5;
tablaB[27,1]=1;     tablaB[27,2]=4;     tablaB[27,3]=4;
tablaB[27,4]=0;     tablaB[27,5]=2;     tablaB[27,6]=5;
tablaB[27,7]=0;     tablaB[27,8]=3;     tablaB[27,9]=6;
tablaB[27,10]=1;    tablaB[27,11]=4;    tablaB[27,12]=6;
tablaB[28,1]=2;     tablaB[28,2]=5;     tablaB[28,3]=6;
tablaB[28,4]=2;     tablaB[28,5]=4;     tablaB[28,6]=0;
tablaB[28,7]=2;     tablaB[28,8]=5;     tablaB[28,9]=1;
tablaB[28,10]=3;    tablaB[28,11]=6;    tablaB[28,12]=1;
```

```
        //TABLA_C
        tablaC[1,1]=1; tablaC[1,2]=8; tablaC[1,3]=15;       tablaC[1,4]=22;
        tablaC[1,5]=29;       tablaC[1,6]=36;
        tablaC[2,1]=2; tablaC[2,2]=9; tablaC[2,3]=16;       tablaC[2,4]=23;
        tablaC[2,5]=30;       tablaC[2,6]=37;
        tablaC[3,1]=3; tablaC[3,2]=10;       tablaC[3,3]=17;       tablaC[3,4]=24;
        tablaC[3,5]=31;       tablaC[3,6]=0;
        tablaC[4,1]=4; tablaC[4,2]=11;       tablaC[4,3]=18;       tablaC[4,4]=25;
        tablaC[4,5]=32;       tablaC[4,6]=0;
        tablaC[5,1]=5; tablaC[5,2]=12;       tablaC[5,3]=19;       tablaC[5,4]=26;
        tablaC[5,5]=33;       tablaC[5,6]=0;
        tablaC[6,1]=6; tablaC[6,2]=13;       tablaC[6,3]=20;       tablaC[6,4]=27;
        tablaC[6,5]=34;       tablaC[6,6]=0;
        tablaC[7,1]=7; tablaC[7,2]=14;       tablaC[7,3]=21;       tablaC[7,4]=28;
        tablaC[7,5]=35;       tablaC[7,6]=0;


        //diaSemana:
        //almacena los valores de trabajo
        //que se utilizan en la aplicación
        diaSemana[1] = 'Domingo';
        diaSemana[2] = 'Lunes';
        diaSemana[3] = 'Martes';
        diaSemana[4] = 'Miércoles';
        diaSemana[5] = 'Jueves';
        diaSemana[6] = 'Viernes';
        diaSemana[7] = 'Sábado';

FinSubProceso


Funcion sonSoloNumeros = SinErroresSintacticos(fecha)

        //Verifica que el día, el mes y el año
        //están compuestos de caracteres numéricos.
        //Si no usamos esta función, falla
        //la función predefinida CONVERTIRANUMERO.

        Definir i Como Entero;
        Definir todoNumeros Como Logico;
        Definir unNumero Como Caracter;

        i = 1;
        todoNumeros = VERDADERO;

        //Una fecha correcta está constituida
        //únicamente de números "0"-->"9"
        //y el guion  "-"
```

```
        Mientras i <= 10 Y todoNumeros = VERDADERO Hacer
                unNumero = SUBCADENA(fecha, i, i);

                Si i = 3 O i = 6 Entonces
                        Si unNumero <> "-" Entonces
                                //Los guiones NO están correctamente
                                //situados
                                Escribir "Fecha errónea. El carácter ",
                                    unNumero, " es distinto de un guion";
                                todoNumeros = FALSO;
                        FinSi
                SiNo
                        Si unNumero <> "0" Y unNumero <> "1"
                          Y unNumero <> "2" Y unNumero <> "3"
                          Y unNumero <> "4" Y unNumero <> "5"
                          Y unNumero <> "6" Y unNumero <> "7"
                          Y unNumero <> "8" Y unNumero <> "9" Entonces
                                Escribir "Fecha errónea. El carácter ",
                                        unNumero,
                                        " es distinto de un número";
                                todoNumeros = FALSO;
                        FinSi
                FinSi
                i = i + 1;
        FinMientras
        //Devolvemos el resultado de la función
        sonSoloNumeros = todoNumeros;
FinFuncion
```

DOCUMENTACIÓN

De esta solución merece destacarse la subrutina "RellenarTablas".

Quizás opine: ¡cuánto trabajo para rellenar los valores fijos de las tablas!.

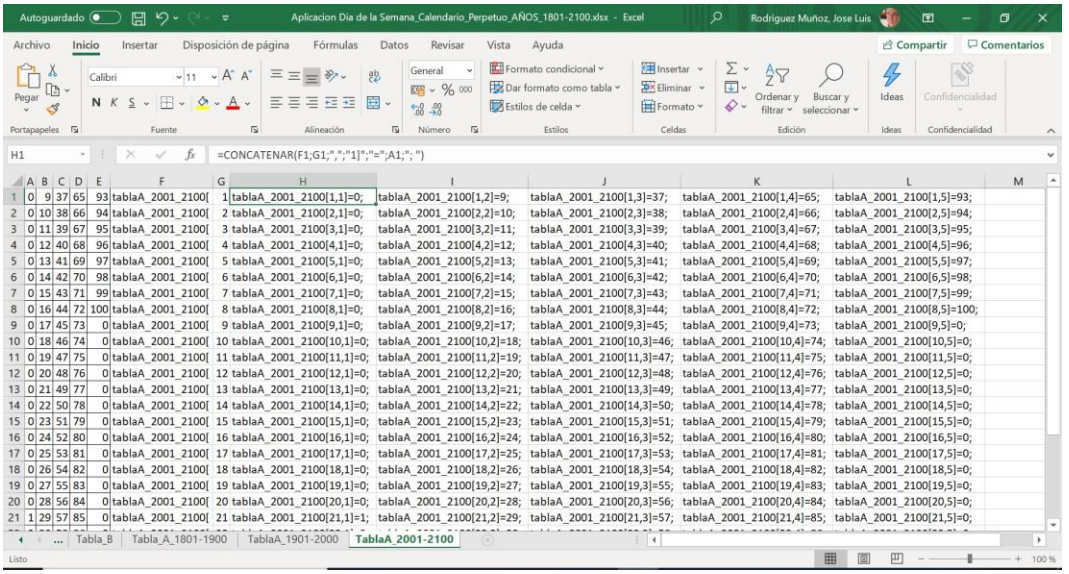
No ha sido tanto trabajo, gracias a un programa de Hoja de Cálculo; que en mi caso he utilizado Microsoft Excel.

Para rellenar los datos, hemos analizado los datos de las tablas fijas y nos hemos percatado que la "Tabla <<A>> AÑOS" se puede rellenar con un algoritmo. Pero, es más rápido y, por lo tanto, más barato utilizar una hoja de cálculo para rellenar las tablas de Años; y, no hacer un algoritmo, probarlo, verificarlo, etcétera.

Para rellenar la Hoja de Cálculo hemos utilizado la fórmula:

```
=CONCATENAR(F1;G1;",";"1]";"=";A1;"; ")
```

En el siguiente ejemplo hemos rellenado con esa fórmula la tabla A:



El valor original "00" de las tablas se ha transformado en el valor "100" y, los valores vacíos de las tablas originales se han rellenado con ceros.

Las demás tablas se rellenan con la misma lógica.

Otra decisión que hemos tomado, en el Diseño de la solución, es que no se devuelve, como resultado de las funciones, el valor de la variable **filaBuscada**, sino que se devuelve el valor buscado o el valor 999 en caso de error. Esta devolución de un valor 999, es una rudimentaria gestión de errores, pero eficaz.

Esta aplicación es un ejemplo de programa cuyo destinatario es otro programa. Es decir, es un programa que se puede utilizar para su uso por otro programa al formar parte de una librería de subrutinas de nuestra empresa.

Recomiendo al lector que aprenda a utilizar un programa de Hoja de Cálculo. Si sigue este consejo, creo que le sacará de muchos apuros.

APLICACIÓN

Cifrado César

La aplicación consiste en recibir una cadena con un texto, cifrarlo con la encriptación César, imprimirlo por pantalla, y desencriptarlo para volverlo a imprimir en texto sin cifrar.

ANÁLISIS DEL ALGORITMO DE LA SOLUCIÓN

En **criptografía**, el cifrado César, también conocido como cifrado por desplazamiento, código de César o desplazamiento de César, es una de las técnicas de **cifrado** más simples y más usadas. Es un tipo de **cifrado por sustitución** en el que una letra en el texto original es reemplazada por otra letra que se encuentra un número fijo de posiciones más adelante en el **alfabeto**. Por ejemplo, con un desplazamiento de 3, la A sería sustituida por la D (situada 3 lugares a la derecha de la A), la B sería reemplazada por la E, etc. Este método debe su nombre al emperador romano **Julio César**, que lo usaba para comunicarse con sus **generales**.

El cifrado César muchas veces puede formar parte de sistemas más complejos de codificación, como el **cifrado Vigenère**, e incluso tiene aplicación en el sistema **ROT13**. Como todos los cifrados de sustitución alfabética simple, el cifrado César se descifra con facilidad y en la práctica no ofrece mucha seguridad en la comunicación.

Fuente Wikipedia: **https://es.wikipedia.org/wiki/Cifrado_C%C3%A9sar**

EJEMPLO de prueba a utilizar: En un lugar de la Mancha, de cuyo nombre no quiero acordarme

SOLUCIÓN:

Por cada carácter del texto, utilizaremos un desplazamiento de 3 letras en el código de caracteres ASCII que utiliza PSeInt.

Restricción: se respetarán las mayúsculas y minúsculas del texto original. Es decir, si la letra es mayúscula, se desplazarán 3 caracteres de letras mayúsculas en el código ASCII. Se actuará igual con las letras minúsculas.

DISEÑO DEL ALGORITMO DE LA SOLUCIÓN

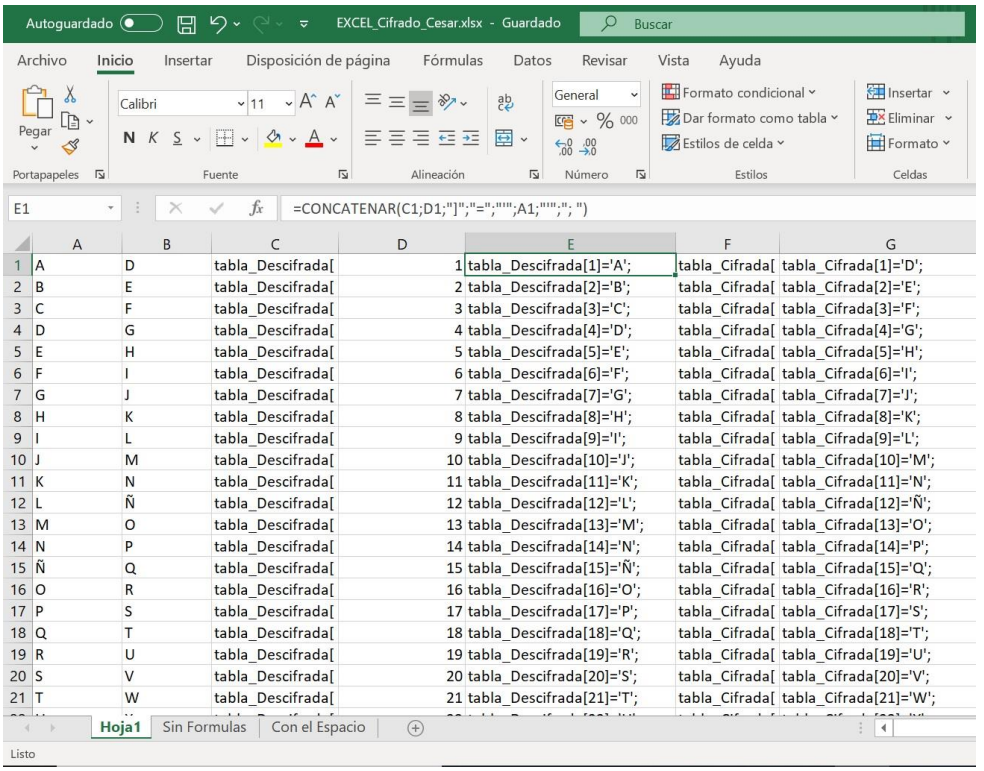
En la etapa del Diseño del algoritmo nos hemos dado cuenta de que, el programa PSeInt no dispone de una función predefinida que dada una letra, por ejemplo, "A" la transforme a su equivalente en código ASCII, se le sume 3 al código ASCCII y se le convierta, el código ASCII resultante en un carácter alfanumérico.

Ante esta imposibilidad de sumar un 3 a cada carácter del texto a encriptar y, luego, restarle un 3 a cada carácter encriptado para hacer la desencriptación; no hay más remedio que utilizar tablas de encriptación.

Para ello, se ha utilizado una tabla de Cifrado y una tabla de Descifrado, de forma que la posición 1 de la tabla corresponde a la letra "A", y la letra cifrada en la otra tabla corresponde a la letra "D" en la posición del índice 1 de esa otra tabla.

En el análisis que se ha hecho, se debe contemplar, en principio a las letras mayúsculas y minúsculas y signos de puntación y, también, números. Las tablas se transforman en una tabla de 77 posiciones cada una.

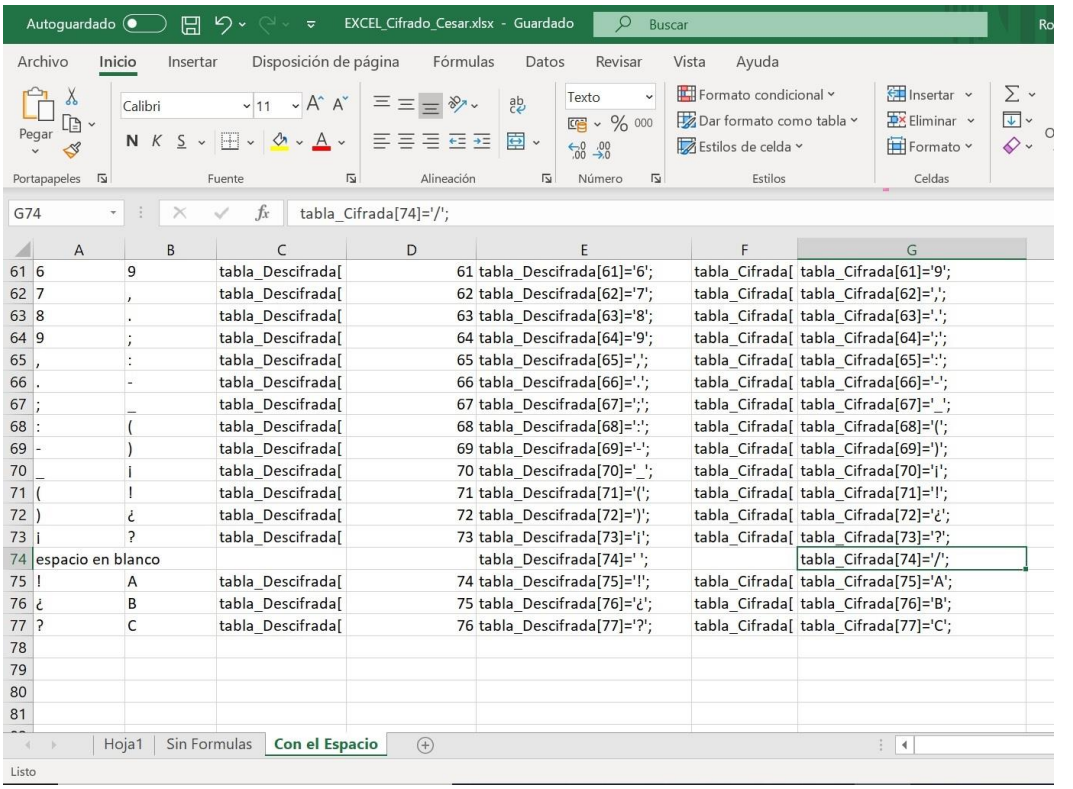
Veamos cómo se han inicializado las tablas en una hoja de cálculo de Microsoft Excel, en mi caso.



	A	B	C	D	E	F	G
1	A	D	tabla_Descifrada[	1	tabla_Descifrada[1]='A';	tabla_Cifrada[	tabla_Cifrada[1]='D';
2	B	E	tabla_Descifrada[	2	tabla_Descifrada[2]='B';	tabla_Cifrada[	tabla_Cifrada[2]='E';
3	C	F	tabla_Descifrada[	3	tabla_Descifrada[3]='C';	tabla_Cifrada[	tabla_Cifrada[3]='F';
4	D	G	tabla_Descifrada[	4	tabla_Descifrada[4]='D';	tabla_Cifrada[	tabla_Cifrada[4]='G';
5	E	H	tabla_Descifrada[	5	tabla_Descifrada[5]='E';	tabla_Cifrada[	tabla_Cifrada[5]='H';
6	F	I	tabla_Descifrada[	6	tabla_Descifrada[6]='F';	tabla_Cifrada[	tabla_Cifrada[6]='I';
7	G	J	tabla_Descifrada[	7	tabla_Descifrada[7]='G';	tabla_Cifrada[	tabla_Cifrada[7]='J';
8	H	K	tabla_Descifrada[	8	tabla_Descifrada[8]='H';	tabla_Cifrada[	tabla_Cifrada[8]='K';
9	I	L	tabla_Descifrada[	9	tabla_Descifrada[9]='I';	tabla_Cifrada[	tabla_Cifrada[9]='L';
10	J	M	tabla_Descifrada[	10	tabla_Descifrada[10]='J';	tabla_Cifrada[	tabla_Cifrada[10]='M';
11	K	N	tabla_Descifrada[	11	tabla_Descifrada[11]='K';	tabla_Cifrada[	tabla_Cifrada[11]='N';
12	L	Ñ	tabla_Descifrada[	12	tabla_Descifrada[12]='L';	tabla_Cifrada[	tabla_Cifrada[12]='Ñ';
13	M	O	tabla_Descifrada[	13	tabla_Descifrada[13]='M';	tabla_Cifrada[	tabla_Cifrada[13]='O';
14	N	P	tabla_Descifrada[	14	tabla_Descifrada[14]='N';	tabla_Cifrada[	tabla_Cifrada[14]='P';
15	Ñ	Q	tabla_Descifrada[	15	tabla_Descifrada[15]='Ñ';	tabla_Cifrada[	tabla_Cifrada[15]='Q';
16	O	R	tabla_Descifrada[	16	tabla_Descifrada[16]='O';	tabla_Cifrada[	tabla_Cifrada[16]='R';
17	P	S	tabla_Descifrada[	17	tabla_Descifrada[17]='P';	tabla_Cifrada[	tabla_Cifrada[17]='S';
18	Q	T	tabla_Descifrada[	18	tabla_Descifrada[18]='Q';	tabla_Cifrada[	tabla_Cifrada[18]='T';
19	R	U	tabla_Descifrada[	19	tabla_Descifrada[19]='R';	tabla_Cifrada[	tabla_Cifrada[19]='U';
20	S	V	tabla_Descifrada[	20	tabla_Descifrada[20]='S';	tabla_Cifrada[	tabla_Cifrada[20]='V';
21	T	W	tabla_Descifrada[	21	tabla_Descifrada[21]='T';	tabla_Cifrada[	tabla_Cifrada[21]='W';

Existe un carácter especial que hay que considerar a parte: el carácter de espacio en blanco. En mi caso, he decidido colocarlo después del cifrado de los signos de puntuación y, por supuesto, antes de que la tabla "de la vuelta": si se avanzan tres caracteres hacia adelante, al final de la tabla, nos quedamos sin caracteres de conversión; que, en nuestro caso, son los caracteres: A, B, C. Ya que a la letra "A" le corresponde de cifrado "D", porque el análisis de la solución decía que se debía "avanzar" 3 caracteres del alfabeto.

A continuación, vemos cómo se contempla el carácter especial de espacio en blanco y la "vuelta" de los caracteres en la tabla.



	A	B	C	D	E	F	G
61	6	9	tabla_Descifrada[	61	tabla_Descifrada[61]='6';	tabla_Cifrada[	tabla_Cifrada[61]='9';
62	7	,	tabla_Descifrada[	62	tabla_Descifrada[62]='7';	tabla_Cifrada[	tabla_Cifrada[62]=',';
63	8	.	tabla_Descifrada[	63	tabla_Descifrada[63]='8';	tabla_Cifrada[	tabla_Cifrada[63]='.';
64	9	;	tabla_Descifrada[	64	tabla_Descifrada[64]='9';	tabla_Cifrada[	tabla_Cifrada[64]=';';
65	,	:	tabla_Descifrada[	65	tabla_Descifrada[65]=',';	tabla_Cifrada[	tabla_Cifrada[65]=':';
66	.	-	tabla_Descifrada[	66	tabla_Descifrada[66]='.';	tabla_Cifrada[	tabla_Cifrada[66]='-';
67	;	_	tabla_Descifrada[	67	tabla_Descifrada[67]=';';	tabla_Cifrada[	tabla_Cifrada[67]='_';
68	:	(	tabla_Descifrada[	68	tabla_Descifrada[68]=':';	tabla_Cifrada[	tabla_Cifrada[68]='(';
69	-	)	tabla_Descifrada[	69	tabla_Descifrada[69]='-';	tabla_Cifrada[	tabla_Cifrada[69]=')';
70	_	¡	tabla_Descifrada[	70	tabla_Descifrada[70]='_';	tabla_Cifrada[	tabla_Cifrada[70]='¡';
71	(	!	tabla_Descifrada[	71	tabla_Descifrada[71]='(';	tabla_Cifrada[	tabla_Cifrada[71]='!';
72	)	¿	tabla_Descifrada[	72	tabla_Descifrada[72]=')';	tabla_Cifrada[	tabla_Cifrada[72]='¿';
73	¡	?	tabla_Descifrada[	73	tabla_Descifrada[73]='¡';	tabla_Cifrada[	tabla_Cifrada[73]='?';
74	espacio en blanco				tabla_Descifrada[74]=' ';		tabla_Cifrada[74]='/';
75	!	A	tabla_Descifrada[	74	tabla_Descifrada[75]='!';	tabla_Cifrada[	tabla_Cifrada[75]='A';
76	¿	B	tabla_Descifrada[	75	tabla_Descifrada[76]='¿';	tabla_Cifrada[	tabla_Cifrada[76]='B';
77	?	C	tabla_Descifrada[	76	tabla_Descifrada[77]='?';	tabla_Cifrada[	tabla_Cifrada[77]='C';
78							
79							
80							
81							

Como se aprecia en la imagen, el carácter de espacio en blanco se ha decidido sustituir en la encriptación por el carácter de la barra inclinada. Por supuesto, este carácter no es uno de los símbolos a detectar para ser encriptado, porque en otro caso, habría "colisiones" en los caracteres a cifrar y descifrar.

CÓDIGO FUENTE

```
// Nombre de la aplicación: Cifrado Cesar
// Autor: José Luis Rodríguez Muñoz
// Fecha Última Actualización: 24-04-2022

Algoritmo CifradoCesar

        Definir unTexto Como Cadena;
        Definir unaLetra Como Caracter;
        //Longitud de caracteres de la cadena de entrada
        Definir limite Como Entero;
        Definir i, j Como Entero; //Contador del bucle
        Definir sinError, noEncontrado Como Logico; //Fin de bucle
        Definir textoCifrado, textoDescifrado Como Cadena;
        //Definimos las tablas globales de la aplicación
        Dimension tabla_Descifrada[77]; //tabla desencriptada
        Dimension tabla_Cifrada[77]; //tabla encriptada

        //Entrada de datos
        Escribir "Teclee el texto a cifrar: "
        Leer unTexto;

        //Inicialización
        //El Espacio en Blanco está en la posición del índice 74
        tabla_Descifrada[1]='A';        tabla_Descifrada[2]='B';
        tabla_Descifrada[3]='C';        tabla_Descifrada[4]='D';
        tabla_Descifrada[5]='E';        tabla_Descifrada[6]='F';
        tabla_Descifrada[7]='G';        tabla_Descifrada[8]='H';
        tabla_Descifrada[9]='I';        tabla_Descifrada[10]='J';
        tabla_Descifrada[11]='K';       tabla_Descifrada[12]='L';
        tabla_Descifrada[13]='M';       tabla_Descifrada[14]='N';
        tabla_Descifrada[15]='Ñ';       tabla_Descifrada[16]='O';
        tabla_Descifrada[17]='P';       tabla_Descifrada[18]='Q';
        tabla_Descifrada[19]='R';   tabla_Descifrada[20]='S';
        tabla_Descifrada[21]='T';       tabla_Descifrada[22]='U';
        tabla_Descifrada[23]='V';   tabla_Descifrada[24]='W';
        tabla_Descifrada[25]='X';       tabla_Descifrada[26]='Y';
        tabla_Descifrada[27]='Z';       tabla_Descifrada[28]='a';
        tabla_Descifrada[29]='b';       tabla_Descifrada[30]='c';
        tabla_Descifrada[31]='d';       tabla_Descifrada[32]='e';
        tabla_Descifrada[33]='f';       tabla_Descifrada[34]='g';
        tabla_Descifrada[35]='h';       tabla_Descifrada[36]='i';
        tabla_Descifrada[37]='j';       tabla_Descifrada[38]='k';
        tabla_Descifrada[39]='l';       tabla_Descifrada[40]='m';
        tabla_Descifrada[41]='n';       tabla_Descifrada[42]='ñ';
        tabla_Descifrada[43]='o';       tabla_Descifrada[44]='p';
        tabla_Descifrada[45]='q';       tabla_Descifrada[46]='r';
        tabla_Descifrada[47]='s';       tabla_Descifrada[48]='t';
```

```
tabla_Descifrada[49]='u';       tabla_Descifrada[50]='v';
tabla_Descifrada[51]='w';       tabla_Descifrada[52]='x';
tabla_Descifrada[53]='y';       tabla_Descifrada[54]='z';
tabla_Descifrada[55]='0';       tabla_Descifrada[56]='1';
tabla_Descifrada[57]='2';       tabla_Descifrada[58]='3';
tabla_Descifrada[59]='4';       tabla_Descifrada[60]='5';
tabla_Descifrada[61]='6';       tabla_Descifrada[62]='7';
tabla_Descifrada[63]='8';       tabla_Descifrada[64]='9';
tabla_Descifrada[65]=',';       tabla_Descifrada[66]='.';
tabla_Descifrada[67]=';';       tabla_Descifrada[68]=':';
tabla_Descifrada[69]='-';       tabla_Descifrada[70]='_';
tabla_Descifrada[71]='(';       tabla_Descifrada[72]=')';
tabla_Descifrada[73]='i';       tabla_Descifrada[74]=' ';
tabla_Descifrada[75]='!';       tabla_Descifrada[76]='¿';
tabla_Descifrada[77]='?';

tabla_Cifrada[1]='D';   tabla_Cifrada[2]='E';
tabla_Cifrada[3]='F';   tabla_Cifrada[4]='G';
tabla_Cifrada[5]='H';   tabla_Cifrada[6]='I';
tabla_Cifrada[7]='J';   tabla_Cifrada[8]='K';
tabla_Cifrada[9]='L';   tabla_Cifrada[10]='M';
tabla_Cifrada[11]='N';  tabla_Cifrada[12]='Ñ';
tabla_Cifrada[13]='O';  tabla_Cifrada[14]='P';
tabla_Cifrada[15]='Q';  tabla_Cifrada[16]='R';
tabla_Cifrada[17]='S';  tabla_Cifrada[18]='T';
tabla_Cifrada[19]='U';  tabla_Cifrada[20]='V';
tabla_Cifrada[21]='W';  tabla_Cifrada[22]='X';
tabla_Cifrada[23]='Y';  tabla_Cifrada[24]='Z';
tabla_Cifrada[25]='a';  tabla_Cifrada[26]='b';
tabla_Cifrada[27]='c';  tabla_Cifrada[28]='d';
tabla_Cifrada[29]='e';  tabla_Cifrada[30]='f';
tabla_Cifrada[31]='g';  tabla_Cifrada[32]='h';
tabla_Cifrada[33]='i';  tabla_Cifrada[34]='j';
tabla_Cifrada[35]='k';  tabla_Cifrada[36]='l';
tabla_Cifrada[37]='m';  tabla_Cifrada[38]='n';
tabla_Cifrada[39]='ñ';  tabla_Cifrada[40]='o';
tabla_Cifrada[41]='p';  tabla_Cifrada[42]='q';
tabla_Cifrada[43]='r';  tabla_Cifrada[44]='s';
tabla_Cifrada[45]='t';  tabla_Cifrada[46]='u';
tabla_Cifrada[47]='v';  tabla_Cifrada[48]='w';
tabla_Cifrada[49]='x';  tabla_Cifrada[50]='y';
tabla_Cifrada[51]='z';  tabla_Cifrada[52]='0';
tabla_Cifrada[53]='1';  tabla_Cifrada[54]='2';
tabla_Cifrada[55]='3';  tabla_Cifrada[56]='4';
tabla_Cifrada[57]='5';  tabla_Cifrada[58]='6';
tabla_Cifrada[59]='7';  tabla_Cifrada[60]='8';
tabla_Cifrada[61]='9';  tabla_Cifrada[62]=',';
tabla_Cifrada[63]='.';  tabla_Cifrada[64]=';';
tabla_Cifrada[65]=':';  tabla_Cifrada[66]='-';
tabla_Cifrada[67]='_';  tabla_Cifrada[68]='(';
tabla_Cifrada[69]=')';  tabla_Cifrada[70]='i';
tabla_Cifrada[71]='!';  tabla_Cifrada[72]='¿';
```

```
tabla_Cifrada[73]='?';  tabla_Cifrada[74]='/';
tabla_Cifrada[75]='A';  tabla_Cifrada[76]='B';
tabla_Cifrada[77]='C';

limite = LONGITUD(unTexto);
textoCifrado = "";
i = 1;
sinError = VERDADERO;

Mientras i <= limite Y sinError Hacer

        unaLetra = SUBCADENA( unTexto, i, i );
        j = 1;
        noEncontrado = VERDADERO;

        Mientras j <= 77 Y noEncontrado Hacer
                Si unaLetra = tabla_Descifrada[j] Entonces
                        noEncontrado = FALSO;
                        textoCifrado = CONCATENAR( textoCifrado ,
                                                   tabla_Cifrada[j]);
                FinSi
                j = j + 1; //incremento del contador
        FinMientras
        Si noEncontrado Entonces
                sinError = FALSO;
        FinSi
        i = i + 1; //incremento del contador
FinMientras

Si sinError Entonces

        //Inicialización
        textoDescifrado = "";

        Para i = 1 Hasta limite Hacer

                //Inicialización
                unaLetra = SUBCADENA( textoCifrado, i, i );
                j = 1;
                noEncontrado = VERDADERO;

                Mientras j <= 77 Y noEncontrado Hacer
                        Si unaLetra = tabla_Cifrada[j] Entonces
                                noEncontrado = FALSO;
                                textoDescifrado = CONCATENAR(
                                                  textoDescifrado,
                                                  tabla_Descifrada[j]);
                        FinSi
                        j = j + 1; //incremento del contador
                FinMientras
```

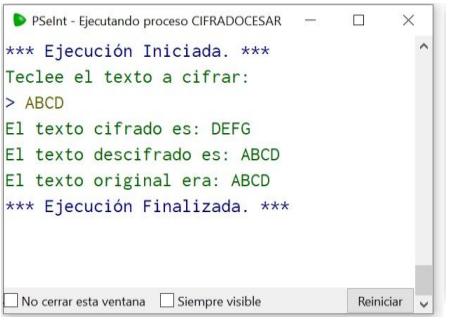
```
                    Si noEncontrado Entonces
                        //Salida de ERROR del programador
                        //Porque no se ha podido descifrar un carácter
                        //de la tabla encriptada
                        Escribir "ERROR en la tabla de ciframiento: ",
                            "tabla_Cifrada";
                        Escribir "Consulte al Administrador de la aplicación";
                        Escribir "ERROR 0001: No se ha podido descifrar por no",
                            " encontrarse el carácter: ", tabla_Cifrada[j - 1];
                        Escribir "dentro de los caracteres cifrados por la tabla, ",
                            "en la posición del índice:", j - 1;
                    FinSi
                FinPara

                //Primera salida con el texto cifrado
                Escribir "El texto cifrado es: ", textoCifrado;

                //Segunda salida con el texto descifrado
                Escribir "El texto descifrado es: ", textoDescifrado;

                //Salida de comprobación
                Escribir "El texto original era: ", unTexto;
            SiNo
                //Salida de ERROR
                Escribir "ERROR en el texto original: ", unTexto;
                Escribir "No se ha podido cifrar por no encontrase el carácter: ",
                    unaLetra , " dentro de los caracteres permitidos";
            FinSi

FinAlgoritmo
```

PRUEBAS DEL ALGORITMO DE LA SOLUCIÓN

Debemos probar el código.

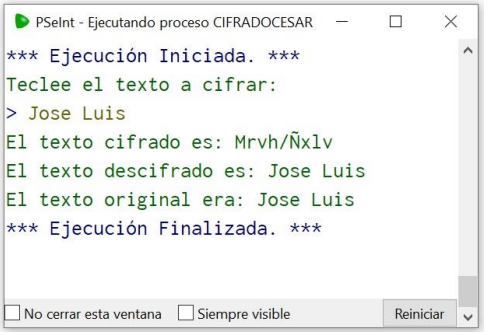
Seguidamente reflejamos el resultado de las ejecuciones de las pruebas realizadas del código fuente con los resultados obtenidos.

PRUEBA 01: Las tablas de cifrado y descifrado están correctamente inicializadas y son accesibles y manejables por el código fuente.



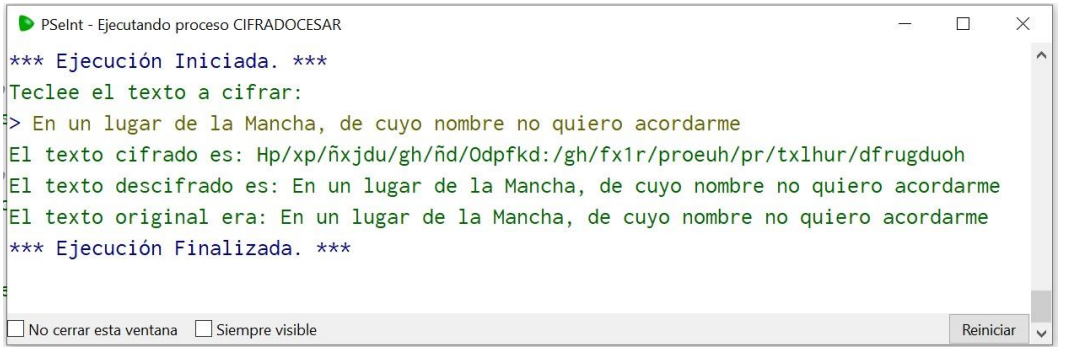
RESULTADO DE LA PRUEBA 01: OK.
La introducción del texto ABC se cifra de forma correcta y se descifra correctamente, según el enunciado del problema: la letra "A" se transforma en "D" y así sucesivamente.

PRUEBA 02: Comprobar que se detecta, se cifra y se descifra correctamente el carácter del espacio en blanco.



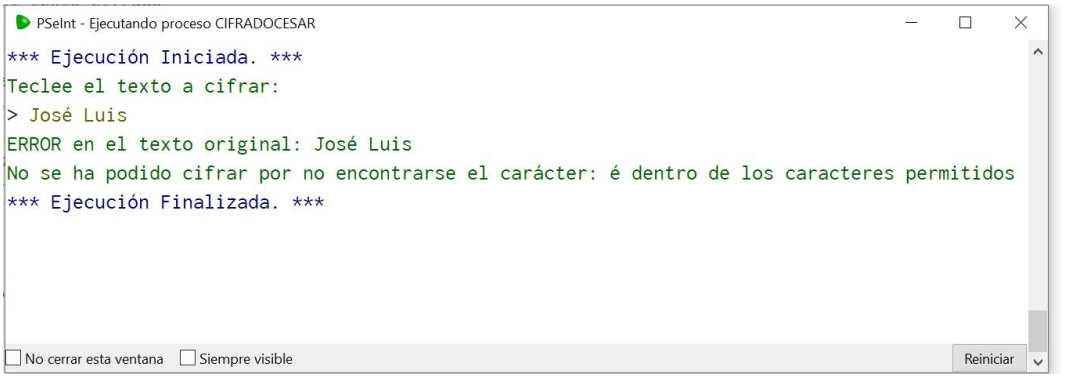
RESULTADO DE LA PRUEBA 02: OK.
El carácter del espacio en blanco se cifra de forma correcta y se descifra correctamente.

PRUEBA 03: Se prueba el Caso de Prueba con el que tiene que funcionar el programa: el Caso Base; que son los Valores Esperados.



RESULTADO DE LA PRUEBA 03: OK.
El texto de prueba asignado se ha cifrado y descifrado correctamente.

PRUEBA 04: Comprobar que se detecta, y no se puede cifrar un carácter del texto de entrada porque no se encuentra dentro de los caracteres permitidos o contemplados en las tablas de encriptación y desencriptación.



RESULTADO DE LA PRUEBA 04: OK.
Se ha detectado el carácter que no se puede cifrar porque no se encuentra reflejado en las tablas de cifrado y descifrado. En el caso de prueba se ha detectado el carácter: "é".

MEJORAS DEL ALGORITMO DE LA SOLUCIÓN

Para futuras versiones del programa del Cifrado César se podría hacer:

→ Para que el programa pueda mejorar su nivel de dificultad de desencriptación, solamente se necesita invertir el orden del resultado del texto cifrado, de manera que el texto: ABC; apareciera encriptado de la forma: FED.

PISTA: solamente hay que invertir el orden de dos elementos de asignación de las variables en dos funciones predefinidas.

→ También podría mejorarse su nivel de vulnerabilidad en la facilidad de desencriptar el texto, si los números en vez de sumarse 3 posiciones se sustituyeran por otro número cualquiera no consecutivo, o incluso, mezclar la sustitución de números y símbolos ortográficos de escritura unos con otros. Esto lo podemos hacer fácilmente gracias a que hemos utilizado dos estructuras de datos: dos tablas. Solamente cambiando el valor de la posición 25, por ejemplo, en la tabla "tabla_Cifrada" por otro símbolo, ya habríamos mejorado el nivel de vulnerabilidad del descifrado al haber aumento su nivel de dificultad para descifrar el código de cifrado que se recoge en la tabla de Cifrado: "tabla_Cifrada".

APLICACIÓN

Contar Palabras Diferentes

La aplicación consiste en recibir una cadena con un texto, y listar el conjunto de palabras diferentes, junto con la cantidad de ocurrencias de cada una de las palabras diferentes que aparecen en el texto de entrada.

ANÁLISIS DEL ALGORITMO DE LA SOLUCIÓN

EJEMPLO de prueba a utilizar: En un lugar de la Mancha, de cuyo nombre no quiero acordarme
SOLUCIÓN:

Para la resolución del ejercicio se utilizará una tabla de dos dimensiones, donde cada fila estará compuesta por una palabra diferente en la columna primera; y, en la segunda columna estará formada por un contador numérico que se irá incrementando en cada aparición de la palabra en el texto de entrada.

Para rellenar la tabla, el algoritmo deberá normalizar el texto de entrada de forma que se eliminen los caracteres de los signos de puntuación ortográficos y los espacios en blanco y estos se transformen por el carácter "/". Además, se transformarán los caracteres válidos a minúsculas para que la palabra "Hola" sea igual que la palabra "HOLA" o "hola". En una segunda pasada de tratamiento del programa, los caracteres "/" servirán para delimitar cada palabra que deberá estar comprendida entre dos caracteres "/".

DISEÑO DEL ALGORITMO DE LA SOLUCIÓN

En la etapa del Diseño del algoritmo nos hemos dado cuenta que, el programa PSeInt no dispone de un tipo de dato Tabla que permita contener tipos de datos diferentes. Es decir, durante el Análisis deberíamos usar una tabla de varias filas y dos columnas. En la primera columna aparecen las palabras diferentes que son de tipo Cadena. Y en la segunda columna aparece un contador del número de ocurrencias de esa palabra en el texto que es de tipo Entero. El programa PSeInt no permite dos tipos de datos diferentes en la misma tabla. Por lo tanto, se opta por utilizar dos tablas. Una tabla de Palabras diferentes de tipo Cadena y otra tabla, del mismo tamaño que la anterior, pero de tipo Entero con cada una de las ocurrencias de cada palabra de la otra tabla en el texto de entrada.

Durante la segunda fase del algoritmo (en la segunda pasada del tratamiento del texto de entrada, que será ya un texto "normalizado"), se irá comprobando si el carácter de inicio del texto pertenece a una palabra delimitada por los caracteres "/". Eso nos permitirá ir formando las palabras e irlas insertando en la tabla de palabras diferentes, en el caso de que, al buscarla en la tabla de palabras, no se haya encontrado previamente en el texto; en cuyo caso se deberá incrementar el contador de ocurrencias de esa palabra en la tabla de contador de palabras. Una vez procesado cada carácter del texto normalizado, ese carácter se irá eliminando del texto normalizado que queda por procesar.

Se ha decidido ir eliminando el carácter de inicio del texto normalizado, porque otra solución sería, una vez que tenemos el texto normalizado (delimitadas las palabras por los caracteres "/") se podría obtener la primera palabra, insertarla en la tabla de palabras diferentes y recorrer todo el texto normalizado hasta el final buscando si se encuentra esa palabra en el texto normalizado, incrementado su ocurrencia. Pero este algoritmo es muy costoso de implementar y por cada palabra nos obliga a recorrer el texto normalizado desde la posición de cada palabra hasta el final de las palabras del texto. Es decir, es un algoritmo complicado de hacer y muy costoso en tiempo de ejecución: tarda mucho en ejecutarse (es muy lento y, por lo tanto, ineficiente).

CÓDIGO FUENTE

```
// Nombre de la aplicación: Contar Palabras Diferentes
// Autor: José Luis Rodríguez Muñoz
// Fecha Última Actualización: 26-04-2022

Algoritmo ContarPalabrasDiferentes

        Definir unTexto Como Cadena;
        Definir unaLetra Como Caracter;
        Definir textoNormalizado Como Cadena;
        Definir unaPalabra Como Cadena;
        Definir i, j Como Entero; //Contador del bucle
        //Fin de bucle
```

```
Definir noFinTexto, noFinPalabra, palabraEncontrada Como Logico;

//Definimos las tablas globales de la aplicación
//Contiene el conjunto de palabras diferentes en cada elemento
Definir LIMITE_TABLA Como Entero;   //constante para delimitar las tablas
LIMITE_TABLA = 50; //Tamaño máximo de las tablas
//Longitud de palabras diferentes de la cadena de entrada
//para las tablas de Palabras y contadores de palabras
Definir limitePalabrasDiferentes Como Entero;
Dimension tabla_Palabras[LIMITE_TABLA]; //tabla de palabras.Tabla Dinámica
//Contiene un contador del número de ocurrencias de la palabra
//de cada elemento en la misma posición que la anterior tabla: tabla_Palabras
Dimension tabla_ContadorPalabras[LIMITE_TABLA]; //tabla con el contador de
                                                // las palabras

//Entrada de datos
Repetir

        Escribir "Teclee el texto a contar sus palabras diferentes: "
        Leer unTexto;

Hasta Que LONGITUD(unTexto) > 0

//PASO 1: normalizamos el texto introducido
//Sustituimos los caracteres de espacio en blanco
//por el carácter "/"
//Eliminamos los signos de puntuación: ",", ";", ".", ":", "¿", "?", "¡", "!"
//y pasamos las letras a minúscula

Para i = 1 Hasta LONGITUD(unTexto) Hacer

        unaLetra = SUBCADENA( unTexto, i, i );
        Si unaLetra = ' ' Entonces
                textoNormalizado = CONCATENAR( textoNormalizado, '/' );
        SiNo
                Si NO(unaLetra = ',' O unaLetra = ';' O unaLetra = '.'
                        O unaLetra = ':' O unaLetra = '¿' O unaLetra = '?'
                        O unaLetra = 'i' O unaLetra = '!' ) Entonces
                        //El caracter que estamos tratando es una letra del
                        // alfabeto o un número
                        //y lo transformamos en minúscula para que la palabra:
                        // Hola
                        //sea igual que la palabra: HOLA; e igual que la palabra:
                        // hola
                        textoNormalizado = CONCATENAR( textoNormalizado,
                                                MINUSCULAS(unaLetra) );
                FinSi
        FinSi

FinPara

//Primera salida con el texto normalizado
Escribir "El texto normalizado es: ", textoNormalizado;
```

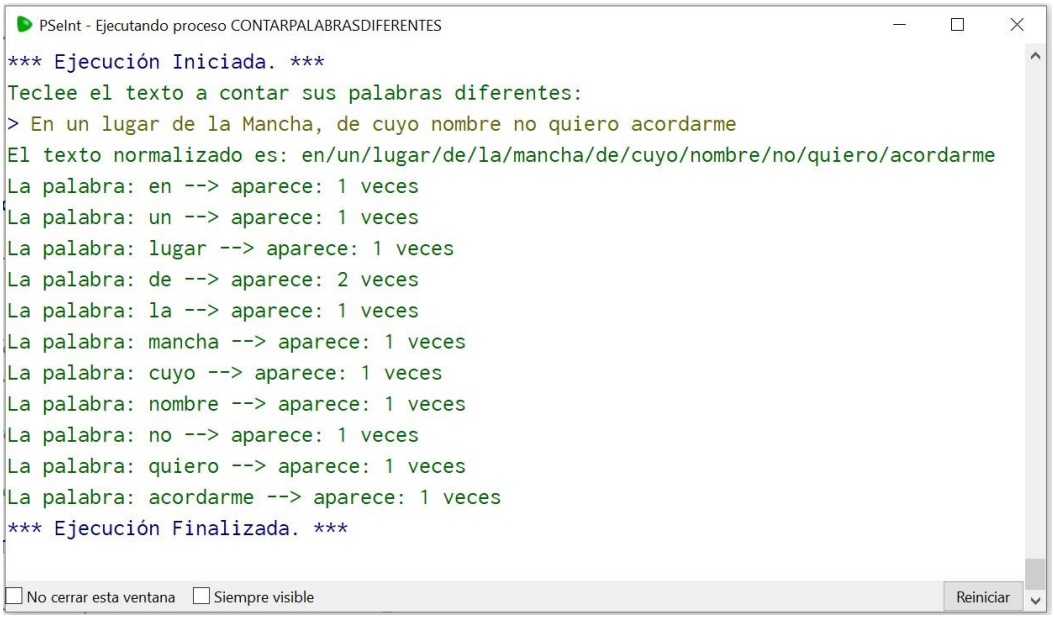
```
//PASO 2: Contamos la ocurrencia de cada palabra diferente

//Inicialización
noFinTexto = VERDADERO;
limitePalabrasDiferentes = 0;

Mientras noFinTexto Hacer

        //PASO 2.1: formamos la palabra hasta el carácter "/"
        noFinPalabra = VERDADERO;
        unaPalabra = "";

        Mientras noFinPalabra Hacer

                //Obtenemos el primer caracter del texto normalizado que queda
                unaLetra = SUBCADENA( textoNormalizado, 1, 1 );

                Si LONGITUD(textoNormalizado) = 1 Entonces
                        noFinPalabra = FALSO; //porque se ha acabado la
                                              // palabra
                        noFinTexto = FALSO; //porque se ha acabado el texto

                        Si unaLetra <> '/' Entonces
                                //La variable unaLetra tiene la última letra del
                                // texto normalizado
                                unaPalabra = CONCATENAR( unaPalabra,
                                                         unaLetra );
                        FinSi

                SiNo
                        Si unaLetra = '/' Entonces
                                noFinPalabra = FALSO; //porque se ha acabado
                                                      // la palabra
                        SiNo
                                //NO se ha llegado al final de la palabra
                                //entonces añadimos la letra a la palabra
                                //que estamos construyendo
                                //NOTA: el caracter "/" no forma parte de la
                                // palabra
                                unaPalabra = CONCATENAR( unaPalabra,
                                                         unaLetra );
                        FinSi
                FinSi

                //CHIVATO PARA PRUEBAS: Escribir "Palabra: ", unaPalabra;

                //Eliminamos del texto el caracter que hemos tratado
                textoNormalizado = SUBCADENA( textoNormalizado, 2,
                                        LONGITUD(textoNormalizado) );
                //CHIVATO PARA PRUEBAS: Escribir "Texto restante: ",
                // textoNormalizado;

        FinMientras
```

```
//PASO 2.2: Contamos la ocurrencia de la palabra en la tabla
Si LONGITUD(unaPalabra) >= 1 Entonces
        //Si hay palabra que contabilizar entonces se entra al SI
        Si limitePalabrasDiferentes = 0 Entonces
                //La tabla de palabras está vacía entonces insertamos la
                // palabra
                limitePalabrasDiferentes = 1;
                tabla_Palabras[limitePalabrasDiferentes] = unaPalabra;
                //Asignamos la palabra
                tabla_ContadorPalabras[limitePalabrasDiferentes] = 1;
                //Asignamos el contador de palabras
        SiNo
                //Buscamos la palabra en la tabla
                palabraEncontrada = FALSO;
                j = 1;
                Mientras j <= limitePalabrasDiferentes  Y NO
                        palabraEncontrada Hacer
                        Si tabla_Palabras[j] = unaPalabra Entonces
                                //palabra encontrada en la tabla de
                                //palabras
                                tabla_ContadorPalabras[j] =
                                tabla_ContadorPalabras[j] + 1;
                                //incrementamos el contador de palabras
                                palabraEncontrada = VERDADERO;
                                //Salimos del bucle
                        FinSi
                        j = j + 1; //incrementamos el contador
                FinMientras
                Si NO palabraEncontrada Entonces
                        //Insertamos una nueva palabra en la tabla
                        Si limitePalabrasDiferentes < LIMITE_TABLA
                         Entonces
                                //incrementamos el contador de palabras
                                // diferentes
                                limitePalabrasDiferentes =
                                 limitePalabrasDiferentes + 1;
                                tabla_Palabras[limitePalabrasDiferentes]
                                 = unaPalabra; //Asignamos la palabra
                                tabla_ContadorPalabras[
                                limitePalabrasDiferentes] = 1;
                                 //Asignamos el contador de palabras
                        SiNo
                                Escribir "ERROR INTERNO: ",
                                        " Desbordamiento de tabla";
                                Escribir "El texto tiene demasiadas ",
                                        "palabras diferentes. El límite ",
                                        "son: ", LIMITE_TABLA,
                                        " palabras diferentes";
                        FinSi
                FinSi
        FinSi
FinSi
FinMientras
```

```
        //Segunda salida con los resultados
        Si limitePalabrasDiferentes > 0 Entonces
                //Si hay palabras en la tabla de palabras entonces se entra en el SI

                Para i = 1 Hasta limitePalabrasDiferentes Hacer
                        Escribir "La palabra: ", tabla_Palabras[i], " --> aparece: ",
                                tabla_ContadorPalabras[i], " veces";
                FinPara

        FinSi
FinAlgoritmo
```

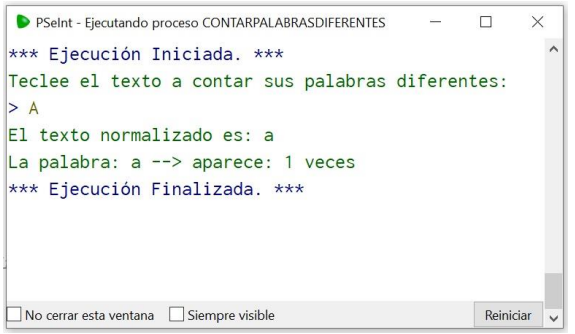
PRUEBAS DEL ALGORITMO DE LA SOLUCIÓN

PRUEBA 01: El programa funciona con el ejemplo base de la Fase de Análisis:



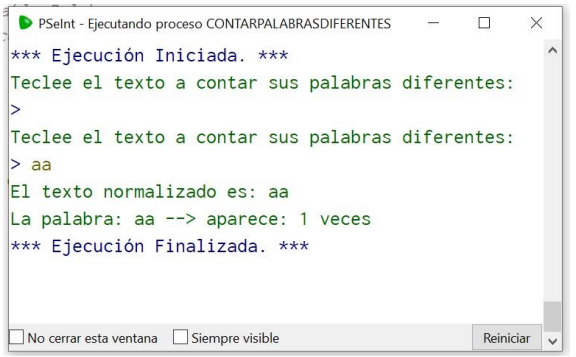
RESULTADO DE LA PRUEBA 01: OK.
Se ha detectado el carácter “,” y los espacios en blanco se han transformado correctamente en caracteres “/”. Se han transformado los caracteres en minúsculas y se han contado correctamente el número de palabras diferentes. También se han reconocido todas las palabras diferentes del texto de entrada.

PRUEBA 02: El programa funciona con una única letra en el texto de entrada



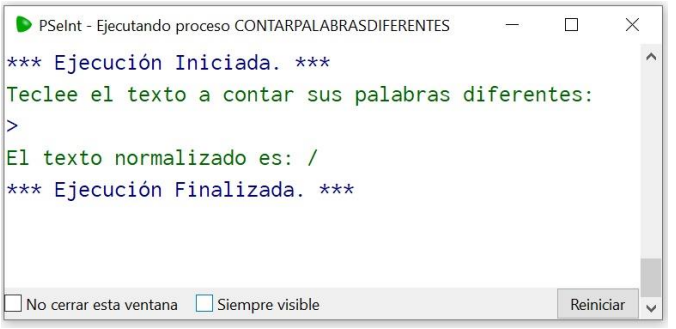
RESULTADO DE LA PRUEBA 02: OK.
Se ha detectado el carácter "A" correctamente y se ha contabilizado bien.

PRUEBA 03: El programa funciona sin ninguna letra en el texto de entrada



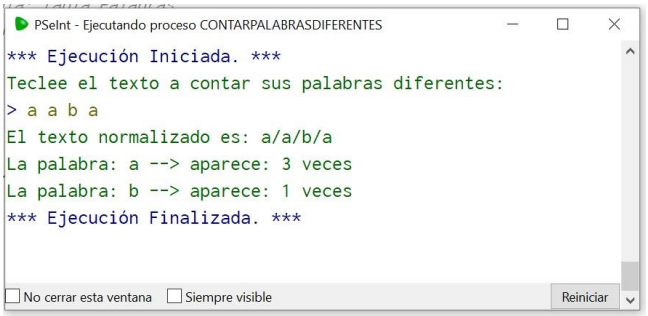
RESULTADO DE LA PRUEBA 03: OK.
Se ha detectado la ausencia de caracteres en el texto de entrada, porque se ha vuelto a solicitar el mensaje de Entrada de Datos al usuario. Se ha detectado y procesado correctamente una palabra de más de un carácter. La palabra "aa" se ha contabilizado bien.

PRUEBA 04: El programa funciona con un único carácter " ": espacio en blanco; como texto de entrada



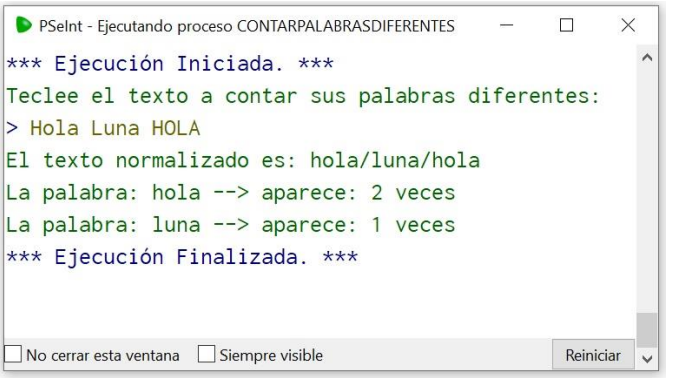
RESULTADO DE LA PRUEBA 04: OK.
Se ha detectado el carácter " " en el texto de entrada y se ha sustituido correctamente por el carácter "/" y el programa ha funcionado correctamente únicamente con este carácter como texto de entrada.

PRUEBA 05: El programa detecta varias palabras iguales e individuales en el texto de entrada



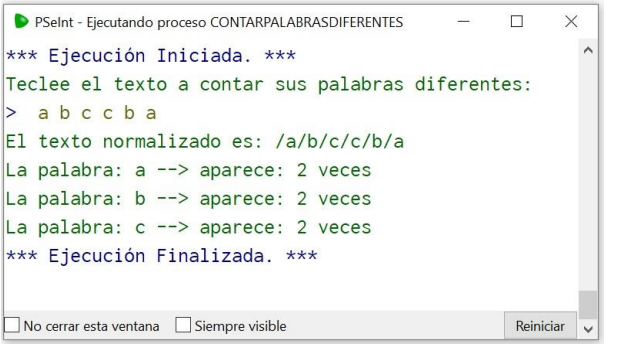
RESULTADO DE LA PRUEBA 05: OK.
Se han contado las palabras iguales que tienen una sola letra.

PRUEBA 06: El programa detecta varias palabras iguales de varias letras e individuales en el texto de entrada



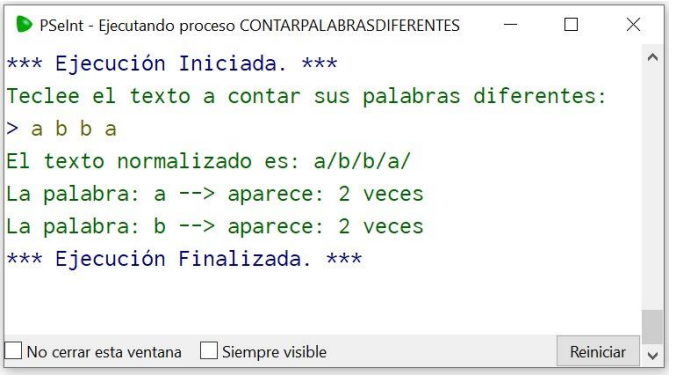
RESULTADO DE LA PRUEBA 06: OK.
Se han contado las palabras iguales que tienen varias letras.

PRUEBA 07: El programa detecta que el texto de entrada empieza por un carácter " ": espacio en blanco.



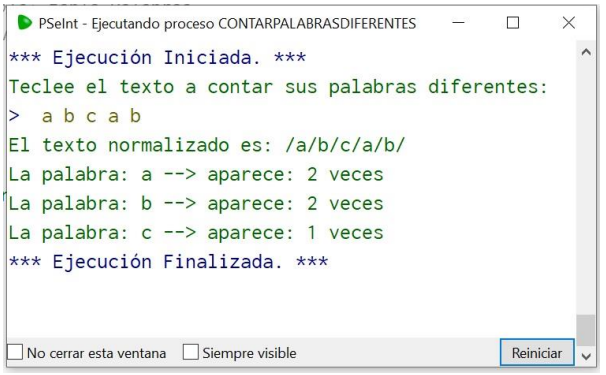
RESULTADO DE LA PRUEBA 07: OK.
Se han contado las palabras iguales que tienen una letra cuando el texto empieza por un espacio en blanco.
Hay que fijarse que el carácter " " se transforma por un carácter "/" antes de la primera palabra del texto.

PRUEBA 08: El programa detecta que el texto de entrada termina por un carácter " ": espacio en blanco.



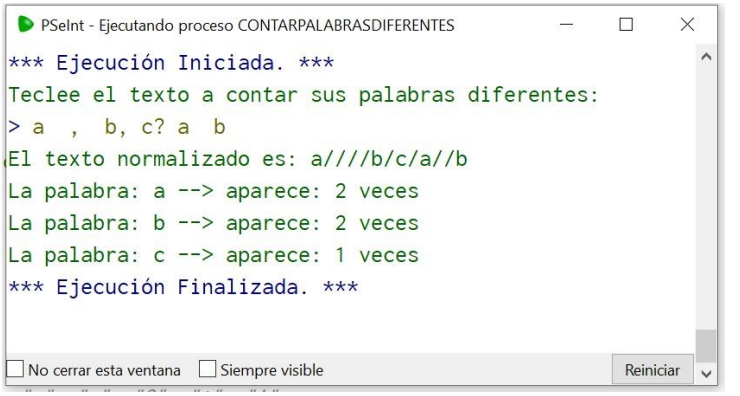
RESULTADO DE LA PRUEBA 08: OK.
Se han contado las palabras iguales que tienen una letra cuando el texto termina por un espacio en blanco.
Hay que fijarse que el carácter " " se transforma por un carácter "/" después de la última palabra del texto.

PRUEBA 09: El programa detecta que el texto de entrada empieza y termina por un carácter " ": espacio en blanco.



RESULTADO DE LA PRUEBA 09: OK.
Se han contado las palabras iguales que tienen una letra cuando el texto empieza y termina por un espacio en blanco.
Hay que fijarse que el carácter " " (es el carácter espacio en blanco) se transforma por un carácter "/" antes de la primera y después de la última palabra del texto.

PRUEBA 10: El programa detecta que el texto de entrada tiene varios caracteres " ": espacio en blanco; seguidos, y, signos de puntuación.



RESULTADO DE LA PRUEBA 10: OK.
Se han contado las palabras iguales que tienen una letra cuando el texto contiene varios espacios en blanco consecutivos.

PRUEBA 11: El programa muestra un error cuando hay un desbordamiento de intento de asignar una nueva palabra más allá del límite del tamaño de las tablas de palabras y contadores de palabras.

Para realizar esta prueba, se ha modificado el código de la constante LIMITE_TABLA y se le ha asignado el valor 5, para no tener que teclear más de 50 palabras ya que ese es el tamaño prefijado para las tablas del programa.

Por lo tanto, solamente hemos modificado la línea de código siguiente:

LIMITE_TABLA = 5;

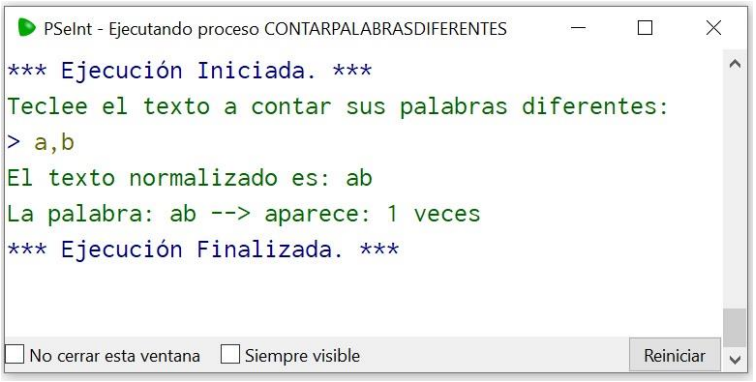
El resultado de esta prueba es:

```
PSeInt - Ejecutando proceso CONTARPALABRASDIFERENTES
*** Ejecución Iniciada. ***
Teclee el texto a contar sus palabras diferentes:
> 1 2 3 4 5 6 7 8 9
El texto normalizado es: 1/2/3/4/5/6/7/8/9
ERROR INTERNO: Desbordamiento de tabla
El texto tiene demasiadas palabras diferentes. El límite son: 5 palabras diferentes
ERROR INTERNO: Desbordamiento de tabla
El texto tiene demasiadas palabras diferentes. El límite son: 5 palabras diferentes
ERROR INTERNO: Desbordamiento de tabla
El texto tiene demasiadas palabras diferentes. El límite son: 5 palabras diferentes
ERROR INTERNO: Desbordamiento de tabla
El texto tiene demasiadas palabras diferentes. El límite son: 5 palabras diferentes
La palabra: 1 --> aparece: 1 veces
La palabra: 2 --> aparece: 1 veces
La palabra: 3 --> aparece: 1 veces
La palabra: 4 --> aparece: 1 veces
La palabra: 5 --> aparece: 1 veces
*** Ejecución Finalizada. ***
```

RESULTADO DE LA PRUEBA 11: OK.
Se ha verificado que no falla la aplicación ya que se muestra el mensaje de error, previniendo que se pueda producir un error por un desbordamiento de índice de tabla por intento de asignar una nueva palabra más allá de la posición de índice 5 que es el límite de la tabla modificado en el código fuente.

MEJORAS DEL ALGORITMO DE LA SOLUCIÓN

Para futuras versiones del programa de Contar Palabras Diferentes se podría mejorar lo siguiente.
No se detecta el caso de PRUEBA: a,b
Ya que no hay un espacio en blanco entre el carácter "," y "b".
Veamos el resultado de este caso.



Esta propuesta de mejora está solucionada en los programas contenidos en la web del libro.

APLICACIÓN

Maître o Jefe de Sala: asignación de mesa

La aplicación consiste en la simulación de una asignación de mesa en un local de comida rápida, que está funcionando en ese momento, por parte de un maître o jefe de sala. La aplicación debe recibir un número entero que representa la cantidad de comensales que ocuparán la mesa y devolverá la mesa asignada o bien que no puede dar la mesa a la persona o grupo.

Diccionario: Un jefe de sala es un camarero especializado en restaurantes y hoteles, generalmente de alta posición.

Restricciones:
→ El número de comensales admitidos por mesa, no puede superar el número de 4.
→ Las mesas se deben asignar de la primera de la lista a la última, siendo su orden: mesa número 1, mesa número 2, etcétera.

→ El local solamente dispone de 5 mesas para ocupar por los comensales.

ANÁLISIS DEL ALGORITMO DE LA SOLUCIÓN

Documento de Especificación de Requisitos: Un local de comida rápida nos ha encargado una aplicación para colocar a los clientes en sus mesas. En una mesa se pueden sentar de 0 (mesa vacía) a 4 comensales (mesa llena). Cuando llega un cliente se le pregunta cuántos son y se les asigna la primera mesa vacía más cerca de la puerta (mesa 1). La aplicación también debe permitir dar de baja (eliminar) un determinado número de ocupantes de una mesa. Cada vez que se sientan nuevos clientes, o se van, se debe mostrar el estado de las mesas. Los grupos no se pueden dividir, aunque haya huecos sueltos suficientes en varias mesas.

DISEÑO DEL ALGORITMO DE LA SOLUCIÓN

Restricciones de la fase de Diseño:
→ Como el local se supone que está en funcionamiento en ese momento, se va a simular el estado inicial de las mesas con la asignación de un número aleatorio de ocupantes por mesa entre 0 y 4 ocupantes.

→ El programa deberá solicitar el número de ocupantes a asignar mesa (para el alta de clientes). Para la baja de clientes (cuando abandonan la mesa), el programa solicitará el número de clientes que se van y el número de la mesa que dejan.

PROGRAMACIÓN DEL ALGORITMO (CONSTRUCCIÓN) DE LA SOLUCIÓN

Al codificar el programa, el programador se da cuenta que el código fuente se simplificaría mucho y sería más manejable el programa y más intuitivo para el usuario de la aplicación, si el alta de clientes se introdujera con un número positivo (sumar clientes al local) y si es una baja de clientes se introdujera con un número negativo (restar clientes al local). Esto permitiría que el usuario de la aplicación tuviera que leer menos mensajes en pantalla de introducir número de clientes o de seleccionar opciones de si el cliente era un alta o una baja de clientes.

Por lo tanto, se modifica el análisis del algoritmo y se acuerda que el alta de clientes se hará introduciendo una cantidad positiva, que es el número de clientes que solicitan mesa y la baja de clientes es una cantidad negativa, que es el número de clientes que dejan una mesa.

Con este cambio de la Fase del Análisis en la fase de Construcción (implementación), el programa queda como sigue:

CÓDIGO FUENTE

```
Algoritmo Maitre

    // Nombre de la aplicación: Maitre
    // Autor: José Luis Rodríguez Muñoz
    // Fecha Última Actualización: 08-08-2022

    //Declaración de variables del programa
    Dimension mesa[5]; //El local dispone de 5 mesas
    Definir i, MESASMAXIMO, OCUPACIONMAXIMA Como Entero;
    //Número de clientes que llegan al local buscando mesa o se van
    Definir clientes Como Entero;

    //Inicializar las variables y constantes
    MESASMAXIMO = 5;
    OCUPACIONMAXIMA = 4;
```

```
        //Al tratarse de una aplicación de "Ejemplo"
        //vamos a simular la ocupación del restaurante
        //en un momento dado. Para simular el restaurante
        //obtenemos el número al azar que contiene cada una
        //de las mesas del local. Este número estará entre 0 y 4.
        //El número al Azar puede ser cero porque el cero
        //es el número para indicar que la mesa está vacía.
        Para i = 1 Hasta MESASMAXIMO Hacer
                mesa[i] = AZAR( OCUPACIONMAXIMA + 1 );
        FinPara

        //El anterior bucle habría que eliminarlo del código
        //en el momento de la entrega de la aplicación
        //al cliente. Porque este bucle de asignación de
        //ocupaciones al azar, lo necesitamos solamente a efectos
        //de poder desarrollar y probar el código fuente.

        Repetir

                MostrarOcupacionMesas(mesa);
                //Introducir los datos de entrada
                Escribir "¿Cuántos son? (Introduzca 0 para salir del programa): "
                Leer clientes;

                Si clientes < 0 Entonces
                        //Una entrada de número negativo implica que se quiere
                        //eliminar ese número de clientes introducidos
                        //de una determinada mesa cuyo número se le pedirá
                        //en la subrutina.
                        Eliminar(clientes, MESASMAXIMO, mesa);
                FinSi
                Si clientes > 0 Entonces
                        //Una entrada de número positivo implica que se quiere
                        //añadir ese número de clientes introducidos
                        //en la mesa que quepan. Se asignará primero la mesa
                        //que esté vacía, aunque el grupo de clientes quepa
                        //en una mesa ocupada previamente en una mesa más
                        //cercana a la puerta. La mesa más cercana a la puerta
                        //del local es la mesa 1, luego la mesa 2, etc.
                        Si clientes > OCUPACIONMAXIMA Entonces
                                Escribir "ERROR DE DATOS DE ENTRADA: El local";
                                Escribir " solamente admite grupos de ";
                                Escribir OCUPACIONMAXIMA, " comensales como";
                                Escribir " máximo por mesa.";
                        SiNo
                                Nuevos(clientes, MESASMAXIMO, OCUPACIONMAXIMA,
                                        mesa);
                        FinSi
                FinSi

        Hasta Que clientes = 0
        Escribir "Fin del programa";
FinAlgoritmo
```

```
Subproceso MostrarOcupacionMesas(mesa)
	// Muestra el estado de ocupación de las mesas en el local.
	Definir i Como Entero;

	Escribir "|-----------|-----|-----|-----|-----|-----|";
	Escribir Sin Saltar "|  Mesa nº: ";

	Para i = 1 Hasta 5 Hacer
		Escribir Sin Saltar "|  ", i, "  ";
	FinPara
	Escribir "|"; //Salto de línea
	Escribir "|-----------|-----|-----|-----|-----|-----|";
	Escribir Sin Saltar "| Ocupación ";

	Para Cada mesita De mesa Hacer
		Escribir Sin Saltar "|  ", mesita, "  ";
	FinPara

	Escribir "|"; //Salto de línea
	Escribir "|-----------|-----|-----|-----|-----|-----|";
FinSubProceso

Subproceso Eliminar(clientes, MESASMAXIMO, mesa Por Referencia)

	Definir numeroMesa Como Entero;

	Escribir "¿En qué mesa se van los clientes?:"
	Leer numeroMesa;

	Si numeroMesa > MESASMAXIMO O numeroMesa <= 0 Entonces
		Escribir "ERROR DE DATOS DE ENTRADA: Mesa inexistente."
	SiNo
		clientes = ABS(clientes); //Pasamos a número positivo para poder hacer
					  //operaciones matemáticas.
		//El número de mesa es un valor válido entre 1 y 5
		Si (mesa[numeroMesa] - clientes) < 0 Entonces
			//si no existen ese número de comensales en la mesa
			//introducida
			Escribir "ERROR DE DATOS DE ENTRADA:El número de clientes”;
			Escribir “ no son de la mesa: ", numeroMesa;
			Escribir "SOLUCIÓN: Elija otra mesa u otro número de clientes.";
		SiNo
			//Borramos los clientes tecleados de ese número de mesa
			mesa[numeroMesa] = mesa[numeroMesa] - clientes;
			Escribir "Gracias por su visita.";
		FinSi
	FinSi
FinSubProceso
```

```
Subproceso Nuevos(clientes, MESASMAXIMO, OCUPACIONMAXIMA, mesa Por Referencia)
        // Buscar una mesa para los nuevos clientes.

        Definir i, mesaVacia, mesaHueco Como Entero;
        Definir hayMesaVacia, hayHueco Como Logico;

        //Primero buscamos la que esté vacía
        //más cerca de la puerta: mesa = 1.
        i = 1;
        mesaVacia = 0;
        hayMesaVacia = FALSO;

        Mientras (i <= MESASMAXIMO) Y NO(hayMesaVacia) Hacer
                Si mesa[i] = 0 Entonces
                        mesaVacia = i;
                        hayMesaVacia = VERDADERO;
                FinSi
                i = i + 1;
        FinMientras

        Si hayMesaVacia Entonces
                mesa[mesaVacia] = clientes;
                Escribir "Por favor, siéntense en la mesa número ", mesaVacia;
        SiNo
                //Buscar un hueco en la mesa en la que quepa
                //el grupo de clientes.
                i = 1;
                mesaHueco = 0;
                hayHueco = FALSO;

                Mientras (i <= MESASMAXIMO) Y NO(hayHueco) Hacer
                        Si clientes <= (OCUPACIONMAXIMA - mesa[i]) Entonces
                                mesaHueco = i;
                                hayHueco = VERDADERO;
                        FinSi
                        i = i + 1;
                FinMientras

                Si hayHueco Entonces
                        //Colocamos a los clientes en el primer hueco disponible
                        mesa[mesaHueco] = mesa[mesaHueco] + clientes;
                        Escribir "Tendrán que compartir mesa. Por favor, siéntense en”;
                        Escribir “ la mesa número ", mesaHueco;
                SiNo
                        Escribir "Lo siento, no queda sitio. Si quieren, pueden esperar”;
                        Escribir “ en la barra del bar.";
                FinSi
        FinSi
FinSubProceso
```

APLICACIÓN

Jugar a la Máquina Tragaperras

La aplicación consiste en la simulación de una Máquina Tragaperras. Una máquina tragaperras es:

Ilustración 07.01 | Imagen de la Máquina Tragaperras: Mister Gadget CMMG2000 Hucha MÁQUINA TRAGAPERRAS, Multicolor, 13,50 x 9,60 x 18,40 cm

La aplicación elegirá una secuencia de 5 casillas (en vez de tres) cuyos valores se encuentran entre el 1 y 3. Es decir, en vez de jugar con el valor "Bar", "7", "Campanilla", etcétera; se elegirá entre cinco casillas de números con valores comprendidos entre el 1 y el 3. La aplicación asignará una elección aleatoria de las cinco casillas como elección ganadora.
El jugador jugará contra la máquina. Es decir, habrá una elección aleatoria de valores ganadores por parte de la máquina tragaperras (el ordenador) y habrá dos jugadores:

→ El primer jugador será el ordenador que elegirá aleatoriamente su jugada: las opciones numéricas de los 5 valores de las casillas.

→ El segundo jugador será el usuario que elegirá, según los datos introducidos por teclado, los cinco valores de las casillas numéricas.

El jugador gana cuando alcanza más aciertos uno que otro.
Se deberá informar el ganador, ya sea el ordenador o el usuario; y, si ha habido empate.
Se deberá informar al jugador el número de aciertos del ordenador y del usuario (el jugador).
Se deberá aceptar solamente valores de las casillas entre 1 y 3, emitiéndose un mensaje de error en el caso que el usuario no introduzca un 1, un 2 o bien un 3.
El juego se termina o bien por un error de datos del usuario o bien porque el usuario (el jugador) no quiere seguir jugando.

Restricciones:
→ El jugador puede contestar a la pregunta de si quiere seguir jugando con un "Si" o solamente con una "s"; y, todas sus posibles combinaciones de respuesta: "Si", "sI", "si", etcétera.

Ejemplos de resultados de la ejecución de la aplicación.

SALIDA DE JUGADA 1: Ha habido empate

```
PSeInt - Ejecutando proceso MAQUINATRAGAPERRAS
*** Ejecución Iniciada. ***
¿Quiere jugar? (Si/No): > s
Teclee su apuesta de cinco números, con valores entre el 1 y el 3.
Numero 1 : > 1
Numero 2 : > 1
Numero 3 : > 1
Numero 4 : > 1
Numero 5 : > 1

|--------------------|-----|-----|-----|-----|-----|
| Máquina Tragaperras |  3  |  1  |  2  |  1  |  2  |
|--------------------|-----|-----|-----|-----|-----|
| Apuesta Ordenador  |  2  |  1  |  1  |  2  |  2  |
|--------------------|-----|-----|-----|-----|-----|
| Apuesta Usuario    |  1  |  1  |  1  |  1  |  1  |
|--------------------|-----|-----|-----|-----|-----|

Atención. Ha habido empate

Pulse una tecla para continuar...
```

SALIDA DE JUGADA 2: Ha ganado el ordenador

```
PSeInt - Ejecutando proceso MAQUINATRAGAPERRAS
¿Quiere jugar? (Si/No): > si
Teclee su apuesta de cinco números, con valores entre el 1 y el 3.
Numero 1 : > 1
Numero 2 : > 2
Numero 3 : > 3
Numero 4 : > 1
Numero 5 : > 2

|--------------------|-----|-----|-----|-----|-----|
| Máquina Tragaperras |  3  |  2  |  2  |  3  |  1  |
|--------------------|-----|-----|-----|-----|-----|
| Apuesta Ordenador  |  3  |  1  |  1  |  3  |  3  |
|--------------------|-----|-----|-----|-----|-----|
| Apuesta Usuario    |  1  |  2  |  3  |  1  |  2  |
|--------------------|-----|-----|-----|-----|-----|

Vaya. Ha ganado el ordenador, con un resultado de 2 números acertados; frente a 1 de sus aciertos

Pulse una tecla para continuar...
```

SALIDA DE JUGADA 3: Ha ganado el usuario

```
PSeInt - Ejecutando proceso MAQUINATRAGAPERRAS
¿Quiere jugar? (Si/No): > sI
Teclee su apuesta de cinco números, con valores entre el 1 y el 3.
Numero 1 : > 2
Numero 2 : > 1
Numero 3 : > 2
Numero 4 : > 3
Numero 5 : > 3

|--------------------|-----|-----|-----|-----|-----|
| Máquina Tragaperras |  3  |  1  |  3  |  3  |  3  |
|--------------------|-----|-----|-----|-----|-----|
| Apuesta Ordenador  |  1  |  2  |  2  |  2  |  1  |
|--------------------|-----|-----|-----|-----|-----|
| Apuesta Usuario    |  2  |  1  |  2  |  3  |  3  |
|--------------------|-----|-----|-----|-----|-----|

Enhorabuena. Ha ganado usted, con un resultado de 3 números acertados; frente a 0 aciertos del ordenador

Pulse una tecla para continuar...
```

SALIDA DE JUGADA 4: Fin de la partida porque no ha respondido "si", "s" o sus equivalencias a la pregunta de si el jugador quiere jugar.

```
PSeInt - Ejecutando proceso MA...
*** Ejecución Iniciada. ***
¿Quiere jugar? (Si/No): > hola
Fin juego
*** Ejecución Finalizada. ***
```

SALIDA DE JUGADA 5: Mensaje de error por haber introducido unos valores de las casillas a jugar, diferentes de 1, 2 ó 3.

```
PSeInt - Ejecutando proceso MAQUINATRAGAPERRAS
¿Quiere jugar? (Si/No): > Si
Teclee su apuesta de cinco números, con valores entre el 1 y el 3.
Numero 1 : >
Numero 2 : > 1
Numero 3 : > 1
Numero 4 : > 2
Numero 5 : > 2
ERROR: En su apuesta de los cinco números, todos sus valores no se encuentran entre el 1 y el 3
Fin juego
*** Ejecución Finalizada. ***
```

NOTA: Para reproducir este error, el usuario ha pulsado sobre la tecla "ENTER", también llamada tecla "INTRO":

cuando tenía que introducir el "Numero 1", y no ha realizado la introducción de ningún valor para el valor de entrada correspondiente al "Numero 1".

CÓDIGO FUENTE

```
// Nombre de la aplicación: Jugar a la Máquina Tragaperras
// Autor: José Luis Rodríguez Muñoz
// Fecha Última Actualización: 15-11-2022

Algoritmo MaquinaTragaperras

        Dimension secuenciaElegida[5];
        Dimension apuestaOrdenador[5];
        Dimension apuestaUsuario[5];

        // Llamamos al procedimiento que ejecuta todo el programa
        ProcesarMaquinaTragaperras(
                        secuenciaElegida,apuestaOrdenador,apuestaUsuario )
FinAlgoritmo

SubProceso ProcesarMaquinaTragaperras( secuenciaElegida, apuestaOrdenador,
apuestaUsuario )

        Definir numeroAlAzar, numero1, numero2, numero3, numero4, numero5, i,
aciertosOrdenador, aciertosUsuario Como Entero;
```

```
        //Guardar si el usuario desea seguir jugando o quiere terminar el juego
        //los valores que contendrá serán: SI, S, NO, N
        Definir deseoJugar Como Cadena;
        Definir quiereJugar Como Logico;
        Definir LIMITETABLA Como Entero;

        //Definimos las constantes del programa.
        LIMITETABLA = 5;


Repetir

        //ENTRADA DE DATOS

        //Preparación del entorno de ejecución del programa
        //con los datos internos de la máquina Tragaperras.

        //Cargamos los datos aleatorios a adivinar
        //por el ordenador y por el usuario
        //de la Máquina Tragaperras.
        Para i = 1 hasta LIMITETABLA Hacer
                //Obtenemos el número al azar que elige el ordenador
                //este número estará entre 1 y 3.
                //El número al Azar no puede ser cero.
                Repetir
                        numeroAlAzar = AZAR(4);
                Hasta Que numeroAlAzar <> 0
                //Asignamos el número al azar en cada casilla
                //de la Máquina Tragaperras
                secuenciaElegida[i] = numeroAlAzar;
        FinPara
        //Cargamos los datos aleatorios que el ordenador elige
        //para adivinar los número elegidos por
        //la Máquina Tragaperras.
        Para i = 1 hasta LIMITETABLA Hacer
                //Obtenemos el número al azar que debe adivinar el usuario
                //este número estará entre 1 y 3.
                //El número al Azar no puede ser cero.
                Repetir
                        numeroAlAzar = AZAR(4);
                Hasta Que numeroAlAzar <> 0
                //Asignamos el número al azar en cada casilla
                //de la apuesta que hace el ordenador.
                apuestaOrdenador[i] = numeroAlAzar;
        FinPara

        //Inicialmente el usuario no desea seguir jugando.
        quiereJugar = FALSO;
        Escribir Sin Saltar "¿Quiere jugar? (Si/No): ";
        Leer deseoJugar;

        deseoJugar = MAYUSCULAS(deseoJugar);
```

```
Segun deseoJugar Hacer
        "SI", "S":
                quiereJugar = VERDADERO;
        De Otro Modo:
                quiereJugar = FALSO;
FinSegun

Si quiereJugar Entonces
        //Mostramos el mensaje de inicio de jugada.
        Escribir "Teclee su apuesta de cinco números, con valores entre el 1”;
        Escribir “ y el 3.";
        //Inicializamos las variables de lectura al valor por defecto.
        numero1=0; numero2=0; numero3=0; numero4=0; numero5=0;
        //Leemos la apuesta del usuario con la introducción
        //de sus cinco números.
        Escribir Sin Saltar "Número 1 : "; Leer numero1;
        Escribir Sin Saltar "Número 2 : "; Leer numero2;
        Escribir Sin Saltar "Número 3 : "; Leer numero3;
        Escribir Sin Saltar "Número 4 : "; Leer numero4;
        Escribir Sin Saltar "Número 5 : "; Leer numero5;

        //Usamos un chivato de código.
        //Escribir "N1=", numero1,"N2=", numero2 ,"N3=", numero3,
        //"N4=", numero4,"N5=", numero5;

        Si (numero1 = 1 O numero1 = 2 O numero1 = 3)
        Y (numero2 = 1 O numero2 = 2 O numero2 = 3)
        Y (numero3 = 1 O numero3 = 2 O numero3 = 3)
        Y (numero4 = 1 O numero4 = 2 O numero4 = 3)
        Y (numero5 = 1 O numero5 = 2 O numero5 = 3) Entonces
                quiereJugar = VERDADERO;
                //Guardamos la elección de la jugada del usuario.
                apuestaUsuario[1] = numero1;
                apuestaUsuario[2] = numero2;
                apuestaUsuario[3] = numero3;
                apuestaUsuario[4] = numero4;
                apuestaUsuario[5] = numero5;
        SiNo
                quiereJugar = FALSO;
                //Mostramos el mensaje de ERROR y se finaliza el juego.
                Escribir "ERROR: En su apuesta de los cinco números,”;
                Escribir “ todos sus valores no se encuentran entre el 1 y el 3";
        FinSi

        Si quiereJugar Entonces
                //Ya tenemos las apuesta elegida por la Máquina Tragaperras
                //la apuesta elegida por el ordenador
                //y la apuesta elegida por el usuario.
                //Por lo tanto, podemos procesar el resultado de la jugada.
                aciertosOrdenador = 0;
                aciertosUsuario = 0;
```

```
//PROCESAR LOS DATOS
Para i = 1 Hasta LIMITETABLA Hacer
        //Al ser 5 opciones, cada acierto
        //aumenta la suma de aciertos en 1 punto más.

        Si secuenciaElegida[i] = apuestaOrdenador[i] Entonces
                aciertosOrdenador = aciertosOrdenador + 1;
        FinSi

        Si secuenciaElegida[i] = apuestaUsuario[i] Entonces
                aciertosUsuario = aciertosUsuario + 1;
        FinSi
FinPara

//SALIDA DATOS

//Visualizamos la salida de los datos de la partida.
Escribir "  "; //dejamos una línea intencionadamente en blanco.
Escribir "|--------------------|-----|-----|-----|-----|-----|";
Escribir "| Máquina Tragaperras |  ", secuenciaElegida[1], "  |  ";
Escribir secuenciaElegida[2], "  |  ", secuenciaElegida[3], "  |  ";
Escribir secuenciaElegida[4], "  |  ", secuenciaElegida[5], "  |";
Escribir "|--------------------|-----|-----|-----|-----|-----|";
Escribir "| Apuesta Ordenador   |  ", apuestaOrdenador[1];
Escribir "  |  ", apuestaOrdenador[2], "  |  ";
Escribir apuestaOrdenador[3], "  |  ", apuestaOrdenador[4];
Escribir "  |  ", apuestaOrdenador[5], "  |";
Escribir "|--------------------|-----|-----|-----|-----|-----|";
Escribir "| Apuesta Usuario     |  ", apuestaUsuario[1], "  |  ";
Escribir apuestaUsuario[2], "  |  ", apuestaUsuario[3], "  |  ";
Escribir apuestaUsuario[4], "  |  ", apuestaUsuario[5], "  |";
Escribir "|--------------------|-----|-----|-----|-----|-----|";
Escribir "  "; //dejamos una línea intencionadamente en blanco.

//Obtenemos quién ha ganado,
//si el ordenador o el usuario; o ha habido empate.
Si aciertosOrdenador = aciertosUsuario Entonces
        Escribir "Atención. Ha habido empate";
SiNo
        Si aciertosOrdenador > aciertosUsuario Entonces
                Escribir "Vaya. Ha ganado el ordenador, con un”;
                Escribir “ resultado de ", aciertosOrdenador;
                Escribir " números acertados; frente a ";
                Escribir aciertosUsuario, " de sus aciertos";
        SiNo
                Si aciertosUsuario = 5 Entonces
                        Escribir "Estupendo.”;
                        Escribir “ Ha ganado con todos”;
                        Escribir ” los aciertos posibles";
                SiNo
```

```
                                        Escribir "Enhorabuena.”;
                                        Escribir “ Ha ganado usted,”;
                                        Escribir “ con un resultado”;
                                        Escribir “ de ";
                                        Escribir aciertosUsuario;
                                        Escribir " números acertados; frente a ";
                                        Escribir aciertosOrdenador;
                                        Escribir " aciertos del ordenador";
                                FinSi

                        FinSi
                FinSi

                //Preparamos la nueva jugada.
                Escribir "  "; //dejamos una línea intencionadamente en blanco.
                Escribir "Pulse una tecla para continuar...";
                Esperar Tecla;
                Borrar Pantalla;

        FinSi
    FinSi
Hasta Que NO(quiereJugar)

Escribir "Fin del juego";

FinSubProceso
```

DOCUMENTACIÓN

Al haber usado la **función predefinida**: MAYUSCULAS nos evitamos poner todas las opciones posibles que puede teclear el usuario para jugar: "Si", "S", "s", "si", "sI", "SI". Con el uso de esta función, reducimos el número de opciones posibles a considerar en el código, lo que nos evita el error de codificación de que se nos olvide alguna de las posibles opciones válidas de confirmación de que queremos jugar.

Además, hemos comprobado que, si el usuario nos introduce un carácter entero, es decir, un 1, un 2 ó un 3; la función predefinida MAYUSCULAS no produce un mensaje de error, ya que la mayúscula de “1” es el “1”, etcétera.
También, hemos comprobado que la función MAYUSCULAS no produce un error al introducir un número decimal, por ejemplo: 3.4

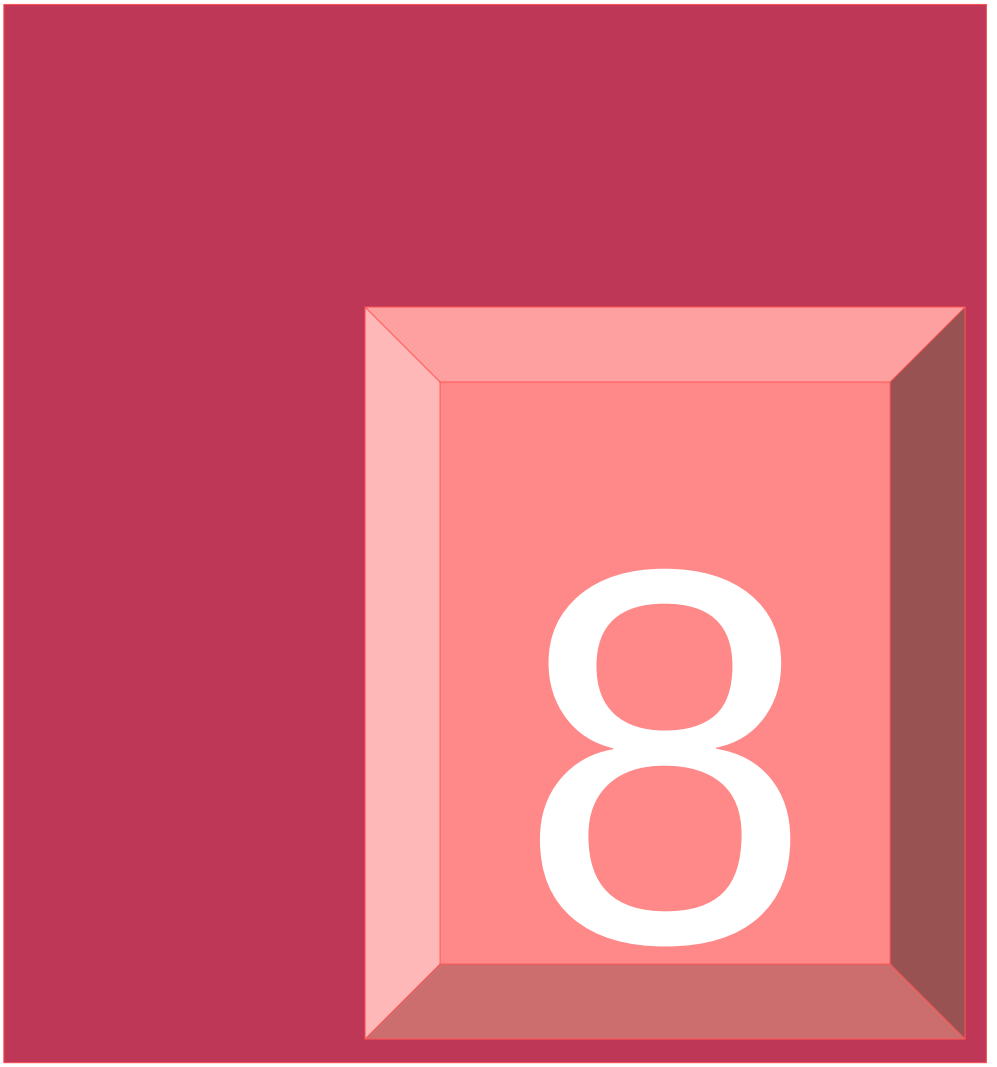
En cuanto al rendimiento del algoritmo: la cantidad de tiempo que tarda en ejecutarse el código; por el uso de la función MAYUSCULAS, no creemos que se ralentiza la ejecución y obtenemos dos resultados valiosos para la **etapa de Mantenimiento** del Código (cuando se entrega la aplicación al usuario y hay que hacerla modificaciones y mejoras):

→ El código es más fácil de entender, al tener que validar menos opciones de respuesta del usuario a la respuesta de si quiere jugar. Es decir, mejoramos la eficacia de la Fase de Mantenimiento.

→ Al ser más rápido de entender el código (algoritmo), mejoramos la eficacia de la **etapa de Mantenimiento** (necesitamos menos tiempo en la etapa de comprensión del código, al ser más simple), y también la eficiencia porque, al ser más fácil de entender, se tarda menos tiempo en modificar el código. Lo que redunda en un menor coste económico de la etapa de Mantenimiento del Software.

Cuando estamos codificando un algoritmo (escribiéndolo), suele ser válido el axioma: "***menos es más***"; la mayoría de las veces.

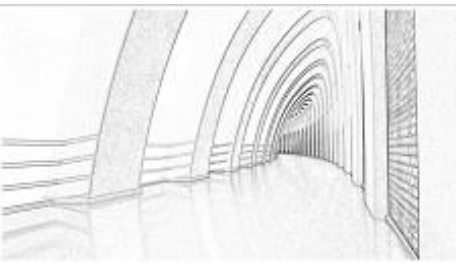
Anexo



Capítulo 8

Anexo

Aprender a Programar | Ejemplos en PSeInt



En este anexo, se han incorporado diferentes apartados que quedaban inconexos en la exposición de anteriores capítulos.

Instalación de PSeInt

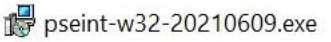
Vamos a ver la instalación del programa informático **PSeInt** en un ordenador con el sistema operativo **Microsoft Windows**.

Para obtener el programa PSeInt, buscamos en nuestro navegador favorito y desde la página de búsqueda que prefiramos, buscamos el concepto:

sourceforge pseint

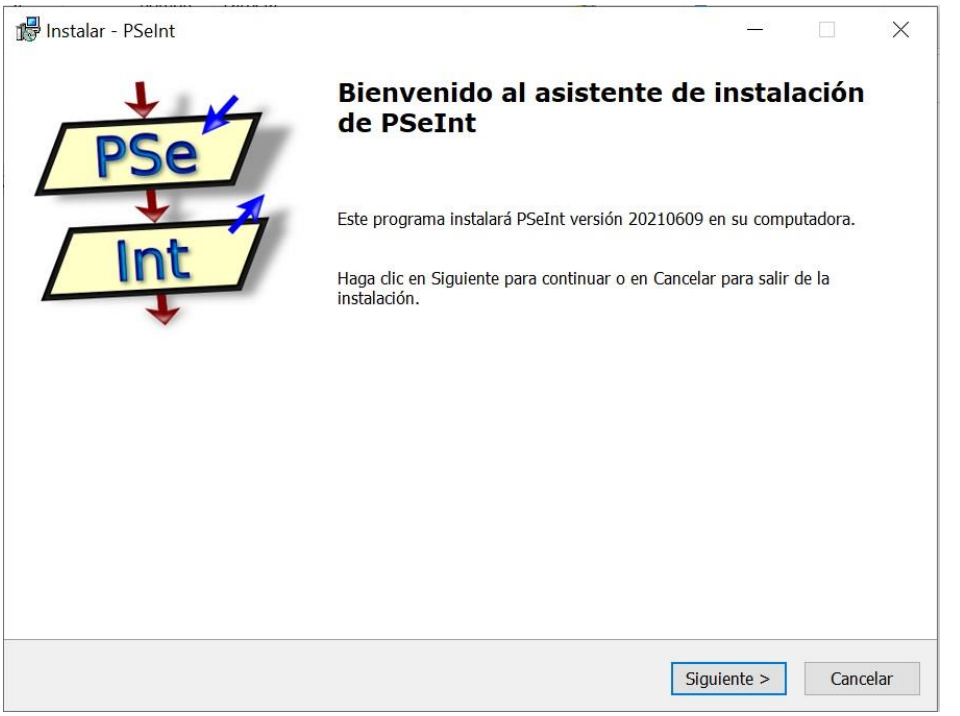
Descargamos desde la página web oficial de "sourceforge", el ejecutable para Windows de PSeInt.

Ejecutamos, con el perfil de Administrador del Equipo, el programa ejecutable con doble clic del botón izquierdo del ratón.



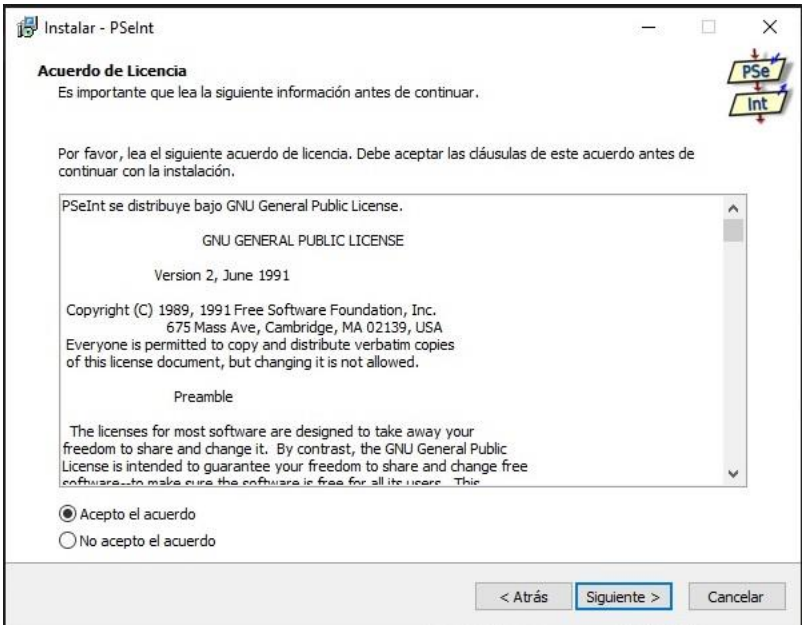
En mi caso, el programa ejecutable es el de la versión 20210609 (que significa una fecha al revés: 09-06-2021)

Y nos aparece la primera pantalla del asistente de instalación del programa PSeInt.

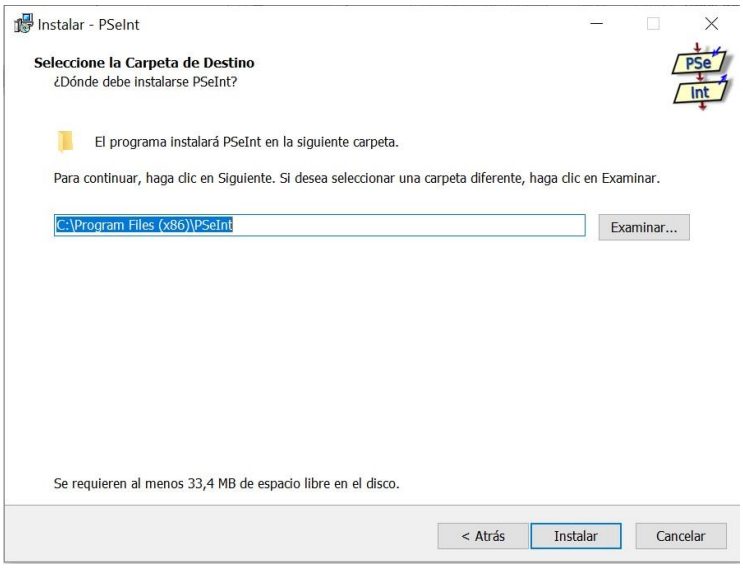


Pulsamos sobre el botón “Siguiente”.

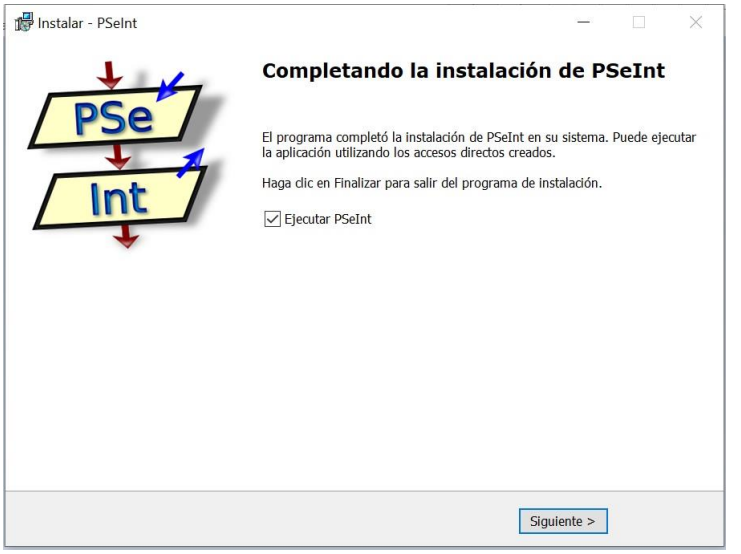
Y nos aparece la pantalla de aceptación de los Derechos de Autor y la exculpación al autor de los daños que pueda ocasionar el programa en nuestro ordenador.



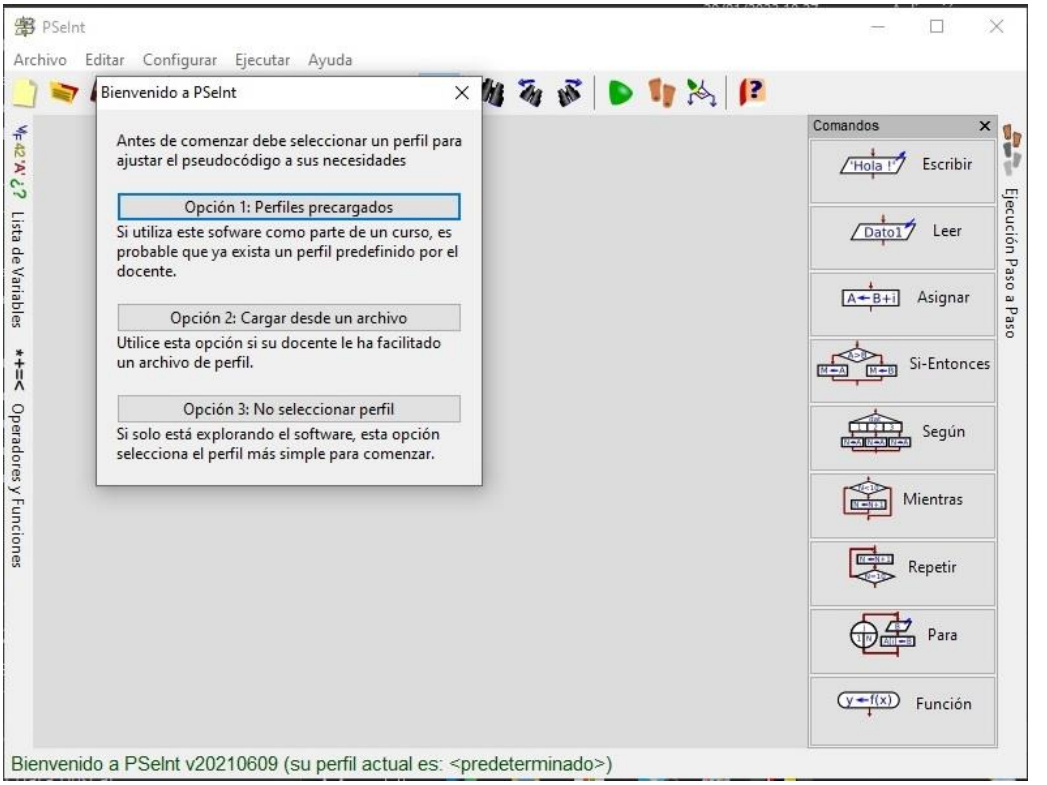
En esa pantalla, hacemos clic en el botón de radio: "Acepto el acuerdo". Y pasamos a la siguiente pantalla.



Presionamos con un clic sobre el botón "Instalar". Y nos aparece la pantalla del final de la instalación.

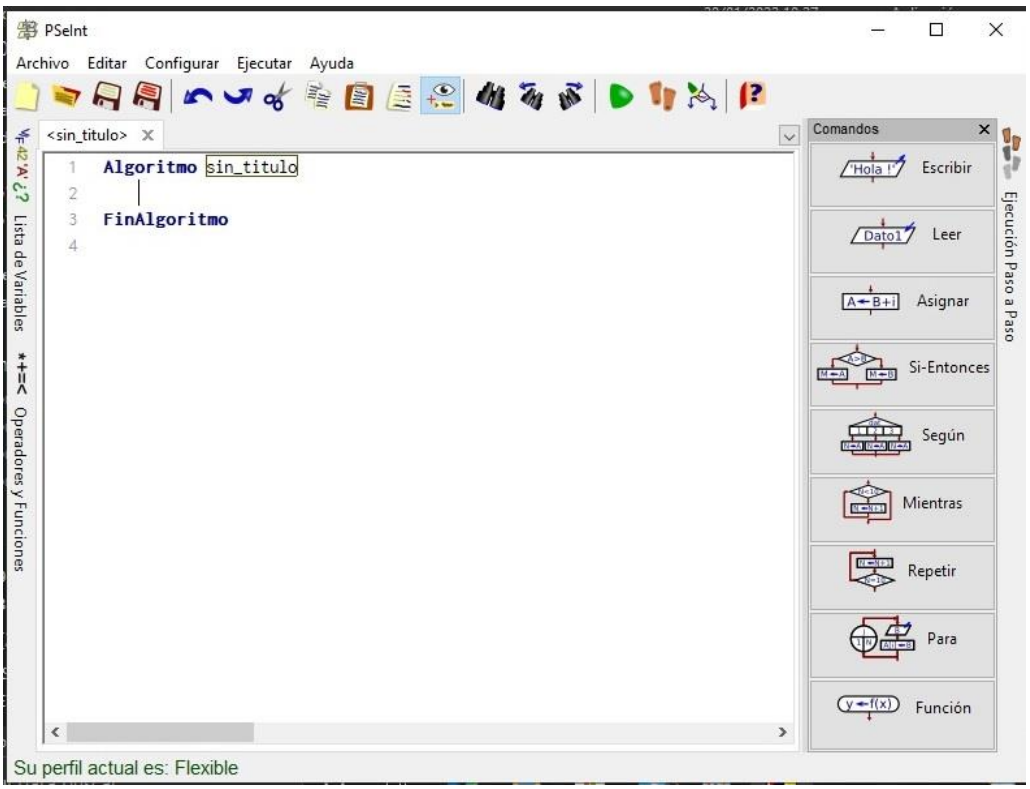


Al pulsar sobre el botón "Siguiente", se arranca el programa.



Si es la primera vez que utilizamos el programa, nos aparece una pantallita de elección entre tres opciones. Debemos elegir la "Opción 3: No seleccionar perfil"; que nos fijará el Perfil de Usuario en "**Flexible**".

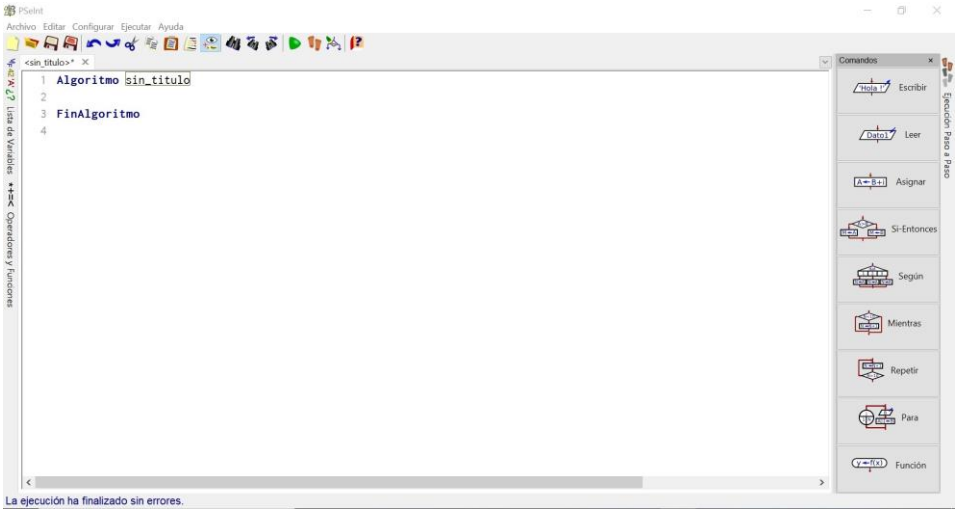
El resultado de la elección de la "**Opción 3: No seleccionar perfil**" será haber seleccionado el Perfil de Usuario "Flexible", como se muestra en la parte inferior de la pantalla siguiente.



Fíjese que, al final de la pantalla, aparece en letras de color verde el mensaje de PSeInt: "Su perfil actual es: Flexible"

Ahora ya podemos empezar a utilizar el programa PSeInt.

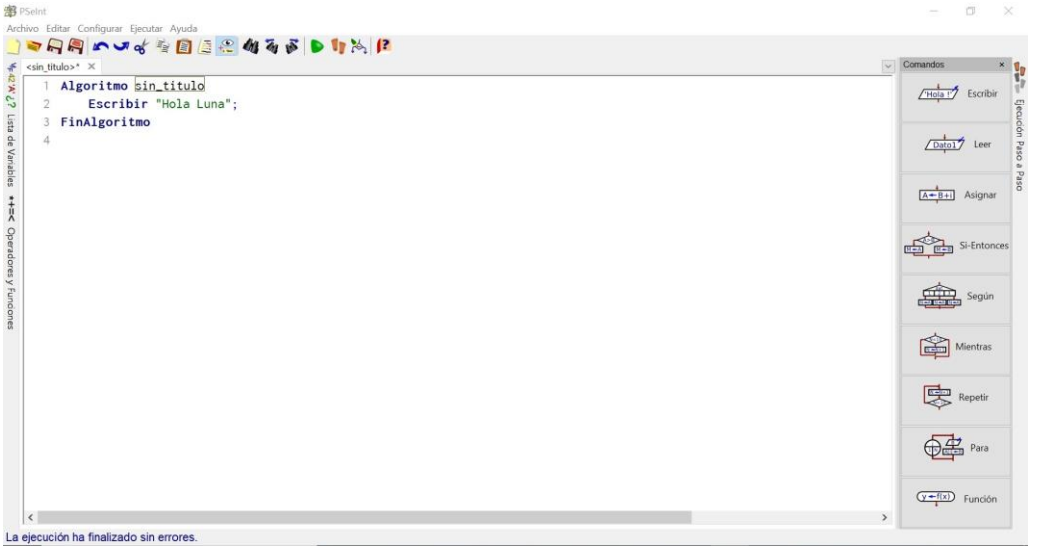
En la pantalla que nos ha aparecido:



Escribimos en la pantalla en blanco, el siguiente texto, después de la primera línea:
Escribir "Hola Luna";

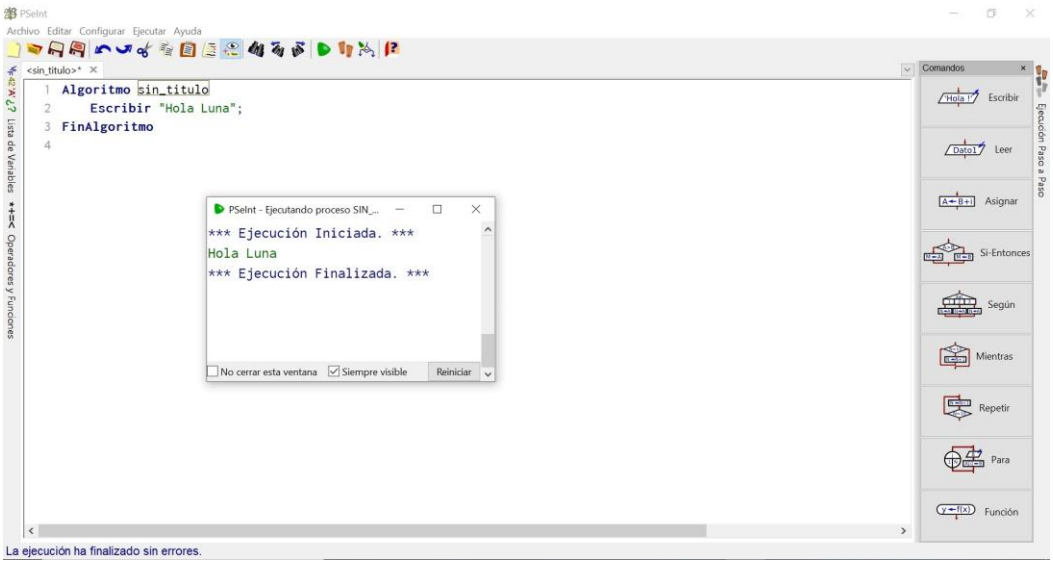
No olvidar terminar la instrucción con un ;

Y la pantalla queda como se muestra seguidamente.



Ahora, pulsamos sobre el Icono del Triángulo en Verde, para ejecutar la instrucción que hemos escrito: **Escribir "Hola Luna";**

Nos aparecerá la ejecución de la sentencia en la ventana de la Consola de Ejecución. Esta ventana es el aspecto que nos aparecería en el ordenador si ejecutáramos el programa que hemos creado con la instrucción: **Escribir "Hola Luna";**

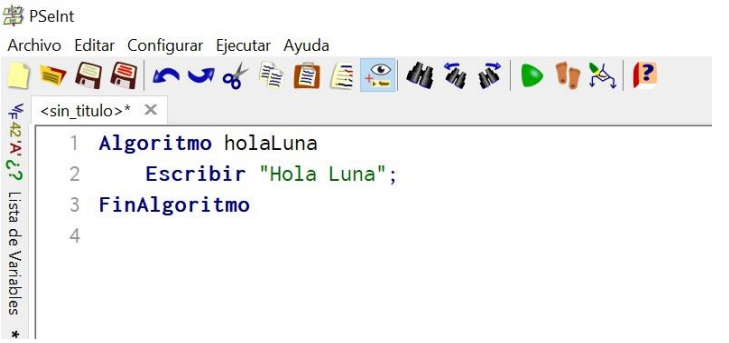


Enhorabuena, acaba de instalar y ejecutar su primer programa con **PSeInt**.

Cómo guardar un programa (código fuente)

Para guardar (archivar) el programa, primero debemos darle un nombre. Escriba el nombre del Algoritmo con la palabra: holaLuna

Tal y como se muestra en la siguiente pantalla.

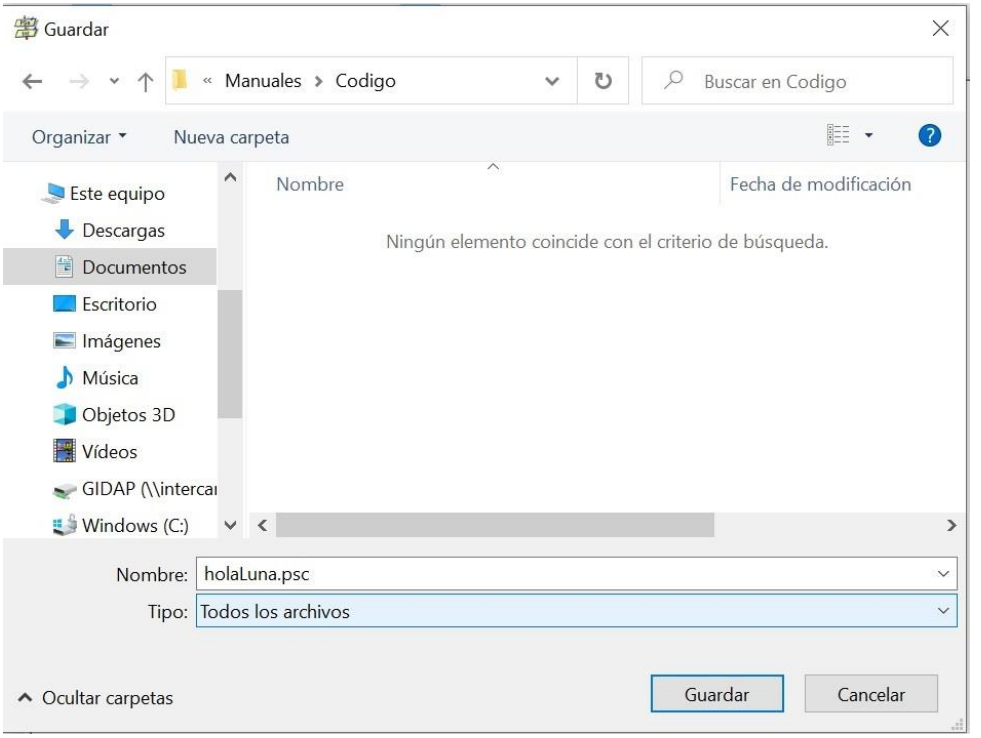


ojo

No hemos visualizado toda la pantalla en la ilustración anterior, porque el resto de la pantalla no aporta información útil para guardar el programa (código fuente).

Para almacenar el código fuente del programa, hacemos clic en el icono del Disquete con el Texto en Gris

Su función es guardar el código fuente del programa en un archivo del ordenador, para ello va a mostrar un Cuadro de Diálogo del sistema operativo, para elegir la ruta y el nombre del fichero a guardar. Este cuadro de diálogo solamente aparecerá la primera vez que pulsemos y elijamos un nombre para el fichero del código fuente del programa, ya que, en el resto de las ocasiones, en las que pulsemos sobre el icono del Disquete con el Texto en Gris, no nos aparecerá el cuadro de diálogo de ficheros porque el archivo ya existirá con un nombre.



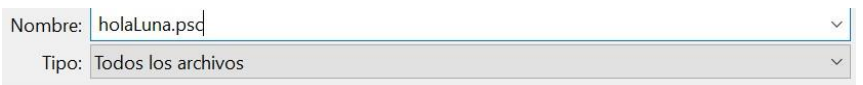
De la pantalla anterior se deduce que la ruta del Árbol de Directorios de mi ordenador es:

Manuales\Codigo

ojo

Estas carpetas de: Manuales\Codigo las habremos creado previamente desde el Sistema Operativo Windows, antes de elegirla, como ruta destinataria, para almacenar el fichero.

De la parte de debajo de la pantalla,



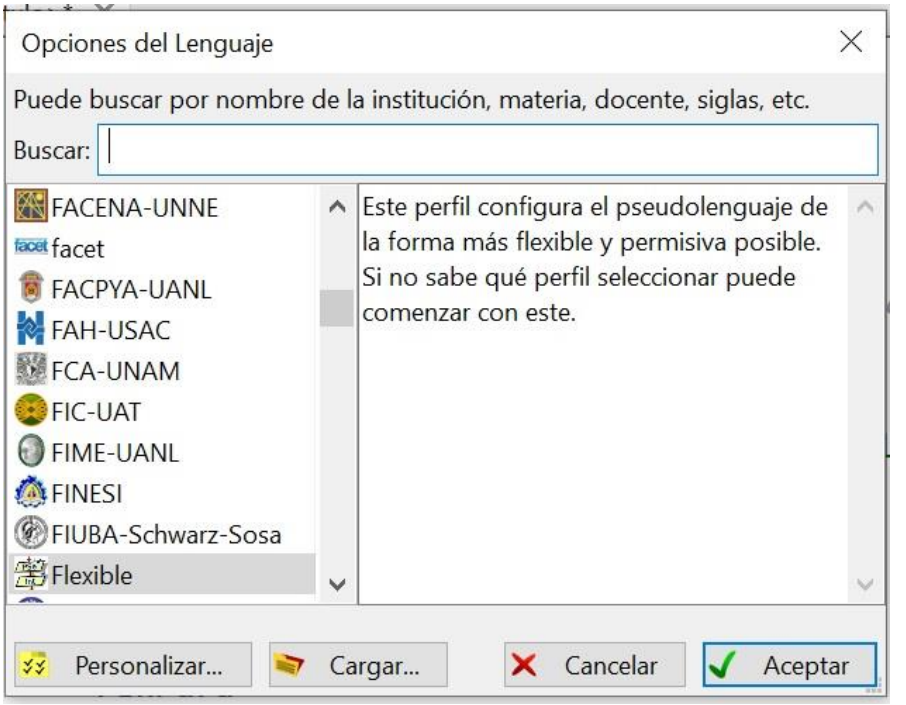
se aprecia que el nombre del fichero, que contenderá el código fuente del programa, se llamará:

holaLuna.psc

Donde las letras "psc" es la extensión del nombre del fichero que utiliza el programa PSeInt. Es decir, el programa PSeInt solamente puede cargar (visualizar) en su pantalla de edición, ficheros que haya creado PSeInt con la extensión "psc".

Configurar PSeInt

Para configurar PSeInt con el Perfil de Usuario que utilizamos en esta obra, seleccionaremos el menú "**Configurar**" y elegiremos la opción "**Opciones del Lenguaje (perfiles)...**". Entonces seleccionamos la opción "**Flexible**" y pulsamos sobre el botón "**Aceptar**", como se muestra en la siguiente pantalla.



Justificación del uso de PSeInt

Qué **características** ventajosas tiene el uso **de PSeInt** como entorno de desarrollo de software:

→ Permite editar simultáneamente varios programas a la vez en pantalla

→ Se puede transformar el código escrito a su equivalente Diagrama de Fujo pulsando únicamente un botón

→ Nos permite utilizar el diagrama Nassi-Shneiderman de nuestro pseudocódigo

→ Es capaz de reescribir el pseudocódigo de nuestro programa a diferentes lenguajes "reales" de programación: seleccionamos el menú **Archivo** y elegimos su opción **Exportar**; y nos aparecen los siguientes lenguajes.

→ Nos ayuda en la escritura de la sintaxis del pseudocódigo mientras lo estamos escribiendo, ofreciendo el detalle de las causas del error sintáctico y aportando las soluciones más frecuentes

→ Cuando ejecutamos el algoritmo, nos permite:

>> Modificar el algoritmo y ver los cambios en la ejecución de forma inmediata, sin necesidad de reingresar los datos

>> Modificar uno o más datos, según queramos, de una ejecución ya finalizada para observar cómo varían los resultados

>> Deshacer una ejecución para reiniciarla o repetirla desde un punto arbitrario que nosotros elijamos

>> Ejecutar el algoritmo paso a paso, controlando la velocidad de ejecución e ir inspeccionando las variables y la evaluación de las expresiones

>> Confeccionar automáticamente una batería de pruebas mediante una tabla de prueba de escritorio

Buenas prácticas en la escritura de programas

Para que el código fuente sea legible (se entienda) por todos los programadores, y por lo tanto modificable (también en la **Fase de Mantenimiento**), se recomienda usar determinadas reglas en su escritura, independientemente de la sintaxis del lenguaje; es decir, se trate de un lenguaje en **pseudocódigo** o de un lenguaje "real": Java, PHP, Python, etcétera.

Un ejemplo de **Normas de Escritura de Código** del programa sería:

Longitud de línea

→ Evitar las líneas que sobrepasen el límite de una pantalla de ordenador. Si no es posible reducir la línea escrita, hay que dividir la instrucción en varias líneas (romper la línea).

Longitud de procedimiento y/o función

→ Evitar que las líneas de instrucciones se extiendan a más número de líneas de las que caben en una pantalla de ordenador. Si no es posible reducir el número de las líneas escritas (simplificar el algoritmo), hay que preguntarse si la funcionalidad del procedimiento o función está bien repartida. Es decir, preguntarse si el procedimiento o función no tiene asignado la resolución de demasiadas funcionalidades o acciones. En caso afirmativo, dividir la funcionalidad en varios procedimientos y/o funciones diferentes.

Sangría entre líneas

→ La indentación o sangrado permite que el código fuente sea más legible.

Ejemplo:

```
Algoritmo Bienvenida
    //Esta línea de comentario está indentada para que quede dentro del
    //conjunto o grupo de instrucciones delimitadas por el bloque de
    //de la sentencia "Algoritmo—FinAlgoritmo"
    Escribir "Hola PseInt";
    Escribir "Buenos días";
FinAlgoritmo
```

→ Se usan varios espacios como unidad de indentación. Aunque es recomendable usar la tecla de tabulación 

como indentador.

→ El posicionamiento del texto en todo el código debe ser predecible. Por ejemplo, las instrucciones del cuerpo de un bucle están indentadas dentro de las palabras reservadas que delimitan el bucle.

→ Los **comentarios** deben respetar la indentación: los comentarios deben situarse a la altura de las sentencias del bloque de instrucciones a las que está documentando.

Rompiendo líneas

Cuando una instrucción sea superior a una línea de la pantalla, hay que romperla de acuerdo con estos criterios:

→ Romper la instrucción después de una coma.

→ Romper la instrucción antes de un operador.

→ Alinear la nueva línea con el comienzo de la instrucción que nos queda por escribir, al mismo nivel que la línea anterior.

ojo

El programa PSeInt no permite la interpretación de una sentencia en varias líneas de código. Pero estas normas de escritura que se exponen, para romper líneas, le servirán para Entornos de Desarrollo Integrados (IDE) de los lenguajes "reales" de programación.

Uso de líneas en blanco

→ Las líneas en blanco mejoran la legibilidad del programa escrito, mediante la activación de secciones o grupos lógicos de código que están lógicamente relacionados.

Documentar nuestros programas

Documentar una aplicación informática nos va a servir de guía para, después de un tiempo, leer nuestro propio código fuente (programa). Nos va a ahorrar tiempo a nosotros mismo y también a nuestros compañeros de trabajo.

Es un buen hábito que debemos adoptar tanto personalmente como de forma grupal. Además, consultar la documentación, es esencial en la **Fase de Mantenimiento** de una aplicación. De hecho, la Fase de Mantenimiento dura mucho más tiempo que todas las anteriores fases.

Las razones por las que debemos documentar nuestro código fuente son:

→ Nos ayuda a entender mejor nuestro código fuente (programa).

→ Ayuda a los demás compañeros a entender nuestro código fuente.

→ Nos ayuda a corregir errores fácilmente.

→ Nos ayuda a mantener claro el objetivo de lo que ha pedido el usuario.

→ El código fuente se vuelve reutilizable (se puede utilizar en otra aplicación diferente).

→ Una aplicación sin documentación es inmantenible: es muy difícil de modificar en la Fase de Mantenimiento, llegando a veces a ser imposible su modificación por falta de conocimiento sobre lo que hace la aplicación y por qué lo hace como lo hace (cómo y por qué se llegó a implementar ese código fuente de esa forma).

Cómo es un día de trabajo de un programador informático

Hemos estado viendo las características y herramientas de las que dispone un programador informático para realizar su trabajo. Pero, cómo las emplea en su día a día en una jornada laboral de desarrollo de software.

Si tomamos como referencia una jornada estándar o típica de un intervalo de tiempo de una semana de trabajo, un programador informático invertirá sus horas laborales en cinco actividades "principales": programación, investigación-planificación-aprendizaje, reuniones, descansos y temas urgentes.

Actividad de Programación

Es la labor principal de un programador de software: programar código. Pero un programador no solamente programa su código fuente, sino que también, tiene que leer e interpretar, corregir y/o continuar el trabajo de codificación de otro compañero.
Los programadores trabajan en equipos de desarrollo de software y es habitual que un programador se atasque o bloquee, y pida ayuda a un compañero porque "cuatro ojos ven más que dos". Por lo tanto, hay que aprender de otras formas de desarrollar el código ya que cada programador tiene su forma de "ver" la codificación que depende mucho de la experiencia, el aprendizaje y conocimientos de cada uno.

Un programador suele pasar entre el 45% y el 60% (a veces hasta el 90%) de su jornada laboral en la actividad de Programación

Actividad de Investigación-Planificación-Aprendizaje

Para acometer un nuevo proyecto, se deben tomar muchas decisiones técnicas. Por ejemplo, qué metodología de desarrollo de software se va a utilizar, en que lenguaje, que herramientas de desarrollo se van a utilizar, que librerías o utilidades vamos a emplear. Estas decisiones entrarían es la fase de Investigación del arranque de un proyecto.

Otras investigaciones son, por ejemplo, qué Patrones de Diseño vamos a usar y cuándo los vamos a incorporar para ir planificando su uso. Y, por supuesto, está la actividad de Aprendizaje. Un programador debe dedicar del orden de 2 horas a la semana a estudiar y conocer nuevas técnicas, herramientas y destrezas de un desarrollador de software.

Un programador suele pasar un 25% de su tiempo semanal en estas actividades

Actividad de Reuniones

Las reuniones no suelen gustar a casi nadie, pero son necesarias. Por ejemplo, si trabajas en un entorno de desarrollo *Agile* vas a tener una reunión de 15 o 20 minutos diarios con tus compañeros de equipo en los que cada uno va a informar lo que hizo ayer en su jornada laboral, lo que tiene planificado hacer hoy y si tiene algún impedimento o bloqueo para llevarlo a cabo. Otro tipo de reuniones son las de Toma de Requisitos o, incluso, las de formación a los usuarios finales o clientes.

Se suele emplear en reuniones el 10% del tiempo semanal

Actividad de Descansos

Los descansos son imprescindibles y algunos son obligatorios que los tomes como es la pausa de media mañana. Hay que darse un paseíto y estirar los músculos cada hora o como mucho cada dos horas de trabajo sedentario. Puede ser para ir al aseo o para comentar, con un compañero, el partido de ayer o qué película viste el fin de semana.

Conviene utilizar un 5% de tu tiempo en "recargar las pilas" y "oxigenarse"

Actividad de Temas Urgentes

El tiempo de dedicación a esta actividad, depende de varios factores. El pasarte el día "apagando fuegos" como un bombero, depende de los proyectos asignados, por ejemplo, que tengas varios **proyectos en mantenimiento** (ya entregados y haya que hacer evolutivos, actualizaciones tecnológicas o correcciones de errores) o que el proyecto en el que estás trabajando tenga errores en las Pruebas de Validación.

El tiempo de incidencias no debería de exceder del 15% de tu jornada, en un "mundo ideal"

Bien es cierto que estas cantidades de tiempo asignadas dependen de lo avanzado del estado del desarrollo del proyecto y, por supuesto, también influye del nivel de responsabilidad del programador. Es decir, no tiene el mismo porcentaje de dedicación diaria, que se ha expuesto, un programador junior o recién llegado, que un jefe de equipo de programadores. En concreto, un programador junior invertirá más cantidad de tiempo en la etapa de aprendizaje y un jefe de equipo invertirá más tiempo en la etapa de planificación y en reuniones, por mencionar solamente dos ejemplos.

Ética del programador

Cuando actualizo mi currículum profesional, les paso una copia a mis compañeros y colegas de profesión para obtener su punto de vista de si lo que expongo, se ajusta a la realidad de mi trabajo y "encaja" con mi personalidad y perfil profesional. Pues bien, una vez añadí, en mi currículum, la siguiente frase: "*interés y gusto por el trabajo bien hecho*". Entonces le pasé el currículum a un excompañero de universidad, que trabaja en una de las mayores empresas privadas de informática en España, y me contestó: "*José Luis, esa frase no se ajusta a la realidad empresarial. En el ámbito de las empresas privadas, muchas veces, tengo la impresión de que lo único importante es cobrar el trabajo y, no si el trabajo está bien o mal hecho*".

Con esta anécdota personal quiero exponer que: una cosa es cómo eres y cómo trabajas y, otra cosa, es cómo se trabaja en según tu tipo de empresa en particular.

Para una relación de los principios éticos y deontológicos en el campo de la Ingeniería Informática, consulte la web de España:

https://ccii.es/CodigoDeontologico

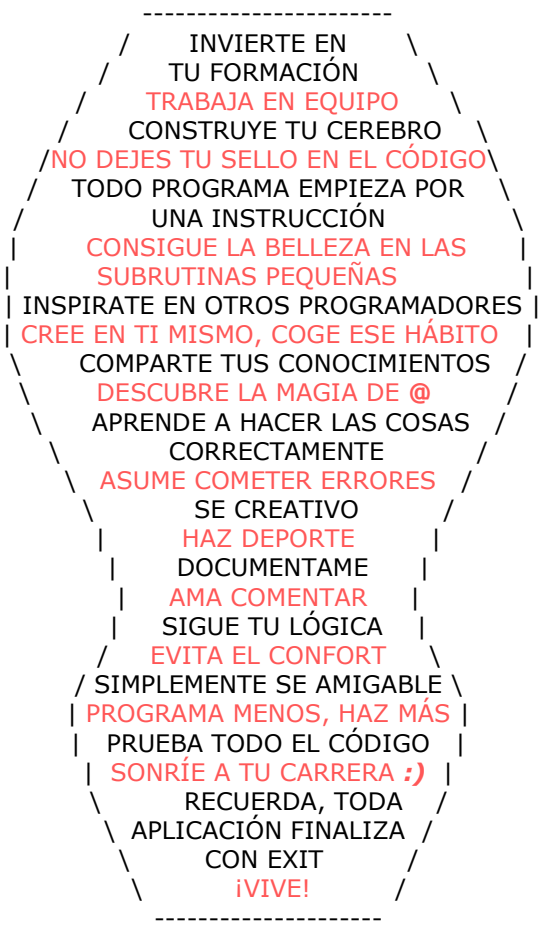
Pero, como la realidad supera a la ficción. Le recomiendo que, tanto a nivel profesional como personal, se debería guiar por el Código del Cowboy (entre otros "marcos" éticos posibles que existen, como es el del "ideario" Cristiano, por ejemplo).

Código del Cowboy (Código del Oeste)		
1	Live each day with courage	Vive cada día con coraje
2	Take pride in your work	Ten orgullo de tu trabajo
3	Always finish what you start	Termina siempre lo que empieces
4	Do what has to be done	Haz lo que tengas que hacer
5	Be tough, but fair	Se duro, pero justo
6	When you make a promise, keep it	Cuando hagas una promesa, cúmplela
7	Ride for de brand	Cabalga a través de las marcas (Vive de acuerdo a las normas)
8	Talk less and say more	Habla menos y di más (Habla menos y haz más)
9	Remember that some things aren't for sale	Recuerda que algunas cosas no están en venta (Recuerda que algunas cosas no se venden)
10	Know where to draw the line	Conoce dónde está el límite (Reconoce tu responsabilidad) (Reconoce dónde está tu responsabilidad)

Hay una cuestión que no suele tenerse presente a la hora de elegir una profesión y es la Salud Laboral. Antes de elegir una profesión, hay que saber si estamos capacitados para desempeñarla y si seremos capaces de mantener nuestra ocupación laboral con el paso del tiempo.
Para saber si estamos capacitados para el trabajo con ordenadores y las consecuencias sobre nuestra salud que nos traerá este trabajo, propongo que lea el siguiente enlace:

http://programacion.net/articulo/enfermedades_derivadas_de_la_programacion_1039

Seguidamente, presento al lector un decálogo de principios y valores que pueden guiarle en el desempeño de su labor como Programador y/o Desarrollador de Software (programas), que recojo en el siguiente Manifiesto.


INVIERTE EN
TU FORMACIÓN
TRABAJA EN EQUIPO
CONSTRUYE TU CEREBRO
NO DEJES TU SELLO EN EL CÓDIGO
TODO PROGRAMA EMPIEZA POR
UNA INSTRUCCIÓN
CONSIGUE LA BELLEZA EN LAS
SUBRUTINAS PEQUEÑAS
INSPIRATE EN OTROS PROGRAMADORES
CREE EN TI MISMO, COGE ESE HÁBITO
COMPARTE TUS CONOCIMIENTOS
DESCUBRE LA MAGIA DE @
APRENDE A HACER LAS COSAS
CORRECTAMENTE
ASUME COMETER ERRORES
SE CREATIVO
HAZ DEPORTE
DOCUMENTAME
AMA COMENTAR
SIGUE TU LÓGICA
EVITA EL CONFORT
SIMPLEMENTE SE AMIGABLE
PROGRAMA MENOS, HAZ MÁS
PRUEBA TODO EL CÓDIGO
SONRÍE A TU CARRERA :)
RECUERDA, TODA
APLICACIÓN FINALIZA
CON EXIT
¡VIVE!

Palabras reservadas de PSeInt

Las palabras reservadas son las palabras que están restringidas para su uso exclusivo por el intérprete de la sintaxis del código fuente de **PSeInt**. Al ser estas palabras de uso restringido por **PSeInt**, no las puede utilizar el programador para un nombre de una variable, por ejemplo, o bien para un nombre de subrutina, una constante, etcétera.

Pero cuáles son las palabras reservadas de PSeInt. Vamos a establecer el siguiente criterio de nomenclatura.

```
Definir nombreVariable Como Entero;
```

Donde, la palabra Definir nos vamos a referir a ella como **palabra reservada** porque esa palabra es necesaria escribirla para que PSeInt "entienda" que queremos definir una variable.

La palabra "nombreVariable" es un **identificador** que nombra única y exclusivamente a ese nombre de variable como variable de tipo entero y es exclusivo para el programa y es un nombre único en ese programa porque no se puede definir una constante con la palabra "nombreVariable" porque ya definimos una variable entera con ese identificador; y, no se pueden llamar igual una variable y una constante en el mismo programa, tampoco se puede llamar igual una subrutina.

La palabra Como es una **cláusula** porque es una especie de palabra reservada que es una palabra "comodín". Es decir, se puede usar en varios contextos. Por ejemplo, podemos utilizar: Como Entero, Como Real, Como Logico, etc. Otras veces, una cláusula es una combinación de palabras reservadas que son opcionales: pueden aparecer o no. Como es el caso de la cláusula SiNo, Con Paso, etcétera.

La palabra Entero es una palabra reservada, ya que es necesaria que aparezca para que el intérprete sepa (reconozca) de qué tipo de dato es la variable del identificador "nombreVariable".

En **PSeInt** las palabras reservadas son:

Nombre Reservado	Tipo Reserva (Palabra/ Cláusula)	Definición y/o ejemplo de código fuente
Algoritmo	Palabra	Algoritmo <identificador> <**Sentencias**> FinAlgoritmo
FinAlgoritmo	Palabra	
Definir	Palabra	Definir <identificador> Como Entero Definir <Identificador1>, <identificador2> Como Entero Definir <Identificador1>, <identificadorN> Como Entero
Como	Cláusula	
Entero	Palabra	
Real	Palabra	
Cadena	Palabra	
Carácter	Palabra	
Lógico	Palabra	
Si	Palabra	Si <condición> Entonces <**Sentencias**> FinSi Si <condición> Entonces <**Sentencias**> SiNo <**Sentencias**> FinSi
Entonces	Palabra	
SiNo	Cláusula	
FinSi	Palabra	

Según	Palabra	Segun <identificador> Hacer
		  1: <**Sentencias**>
		  2,3,4: <**Sentencias**>
		  De Otro Modo <**Sentencias**>
		FinSegun

De Otro Modo	Cláusula	
FinSegun	Palabra	
Mientras	Palabra	Mientras <condición> Hacer <**Sentencias**> FinMientras
FinMientras	Palabra	

Nombre Reservado	Tipo Reserva (Palabra/ Cláusula)	Definición y/o ejemplo de código fuente
Para	Palabra	Para <identificador> = <numero1> Hasta <numero2> Con Paso <numero3> Hacer <**Sentencias**> FinPara Para <identificador1> = <identificador2> Hasta <identificador3> Con Paso <identificador4> Hacer <**Sentencias**> FinPara NOTA: el <numero3> puede ser positivo o negativo. Al igual que el <identificador4> NOTA: Se puede realizar cualquier combinación entre <numero> e <identificador>
Con Paso	Cláusula	
Cada elemento De	Palabra	
Hacer	Palabra	

FinPara	Palabra	RECORRER CADA ELEMENTO
		DE UNA TABLA:
		Dimension **tabla**[10];
		//recorre los 10 elementos y
		//va asignándoles enteros aleatorios
		
		Para Cada **elemento** De **tabla** Hacer
		
		//**elemento** toma el contenido
		//de cada posición del arreglo
		//y si se modifica **elemento**
		//se modifica el arreglo
		
		elemento = AZAR(100);
		
		FinPara

Nombre Reservado	Tipo Reserva (Palabra/ Cláusula)	Definición y/o ejemplo de código fuente
Repetir	Palabra	Repetir **<Sentencias>** Hasta Que <condición>
Hasta Que	Palabra	
Escribir	Palabra	Escribir <cadena> Escribir <identificador> Escribir <cadena>, <identificador> Escribir <identificador>, <cadena> NOTA: Se puede realizar cualquier combinación y cantidad entre <cadena> e <identificador>. Donde <cadena> es una serie de caracteres encerrados entre comillas dobles, por ejemplo: "Hola"
Leer	Palabra	Leer <identificador>
Funcion	Palabra	Funcion <identificadorRetorno> = <identificadorFuncion> **<Sentencias>** FinFuncion Funcion <identificadorRetorno> = <identificadorFuncion>(<parametro1>, <parametroN>) **<Sentencias>** FinFuncion
FinFuncion	Palabra	
SubProceso	Palabra	SubProceso <identificadorSubproceso> **<Sentencias>** FinSubproceso SubProceso <identificadorSubproceso> (<parametro1>, <parametroN>) **<Sentencias>** FinSubproceso
FinSubproceso	Palabra	

Nombre Reservado	Tipo Reserva (Palabra/ Cláusula)	Definición y/o ejemplo de código fuente
Esperar	Palabra	Esperar <número> segundos Esperar Tecla Esperar <número> milisegundos NOTA: Donde <número> es el valor de un número entero
Tecla	Palabra	
Segundos	Palabra	
Milisegundos	Palabra	
Borrar Pantalla	Palabra	Borrar Pantalla
Escribir Sin Saltar	Palabra	Escribir Sin Saltar "Escribe sin saltar de línea"
Dimension	Palabra	Dimension <identificadorTabla>[<tamaño>] NOTA: Donde <tamaño> es un valor de un número entero o bien el nombre de una variable que tiene un valor de un número entero NOTA: Para declarar una tabla o arreglo de 2 dimensiones, hay que declararlo: Dimension <identificadorTabla>[<tamaño1>, <tamaño2>] NOTA: <tamaño1> indica las filas. <tamaño2> indica las columnas.

Conocimientos matemáticos de un programador informático

En realidad, para trabajar programando no se necesita ser un experto en matemáticas. Para la programación habitual (es decir, la alejada del Big Data, Machine Learning y los campos de la aplicación de la teoría estadística, parábolas de misiles, etcétera), es suficiente saber lo siguiente:

→ Suma, resta, multiplicación y división: Y en realidad, sólo hay que saber cuándo se necesita hacer estas operaciones, porque es el ordenador el que va a hacer los cálculos por usted.

→ Cómo obtener el porcentaje de un número. Por ejemplo, para calcular el IVA (Impuesto del Valor Añadido) o el tanto por ciento de ganancia de una venta.

→ Saber la regla de los signos en las operaciones con números enteros: Ejemplo: si la variable **A** tiene el valor negativo **-5**. La operación **-A** da como resultado un valor entero positivo de un **5** (la evaluación de la expresión **-A** significa: menos por menos, es más; es decir, el resultado de la evaluación es un valor numérico positivo).

→ Saber qué es un sistema de coordenadas cartesianas. En programación, el origen (0,0) es la esquina superior izquierda de la pantalla o ventana, y el eje X aumenta conforme avanzamos de izquierda a derecha de la pantalla, y; el eje Y aumenta al bajar por la pantalla. Es decir, en los ejes X e Y de los cuadrantes de coordenadas, la pantalla del ordenador pertenecería al cuadrante cuarto del eje de coordenadas X e Y.

→ Saber el teorema de Pitágoras, ya que lo tendrá que utilizar para encontrar la distancia entre dos puntos en un sistema de coordenadas cartesianas. Esto se utilizaría en aplicaciones para móviles con geolocalización de su ubicación y, últimamente, también en aplicaciones web cuando el usuario del navegador acepta la opción "Permitir el acceso a su ubicación".

→ Dominar los sistemas de numeración binario (de ceros y unos), decimal (significado y utilización del punto decimal y notación científica) y hexadecimal (muy utilizado en la asignación de los colores de los datos visualizados en pantalla. Por ejemplo, esta obra está basada en el color negro, y, en el color rojo representado por el color RGB en hexadecimal: FF5757).

→ Teoría de Conjuntos para trabajar con Bases de Datos, que sirve para programar en el lenguaje SQL (Lenguaje de Consulta Estructurada, en inglés: Structured Query Language) que es el mayoritario en el uso de datos en Bases de Datos Relacionales.

Si, además, está interesado en las matemáticas para computación, visite la página web del **Proyecto Euler**:

https://projecteuler.net/

En ella podrá encontrar desafíos matemáticos que se resuelven sabiendo programar.

Traducir otros idiomas al castellano

Por fin llegamos al tema central de la cuestión, cuando se trabaja con lenguajes de programación "reales" y temas de informática en general: la documentación y los textos están en otros idiomas. En nuestro caso, hay que manejar mucha documentación que es la información que debemos consultar y traducir de la lengua inglesa. Quizás este apartado debería haber sido el primero en aparecer en esta obra: traducir del inglés al español.

Es obvio que, la mejor manera de traducir del inglés al castellano es dominar la lengua inglesa. Pero, si no es nuestro caso; aquí dejo unas recomendaciones y puntos de vista sobre este tema esencial: traducir del inglés.

Traducción de páginas web

La opción más fácil y cómoda es utilizar la funcionalidad de traducción de textos de páginas web del navegador de Google.

Últimamente están saliendo extensiones o Plug-in (en inglés) de aplicacioncitas, que se incrustan e instalan en tu navegador, que hacen las funciones de traductor de los textos de las páginas web. Personalmente, creo que son un peligro para la seguridad e integridad de su ordenador. Yo no los he instalado nunca, por una cuestión de Seguridad Informática.

> La seguridad es cosa de todos, pero empieza por uno mismo.

Traducción de textos y archivos

La página del Traductor de Google es una opción asequible y cómoda de utilizar. Pero no ofrece los resultados de calidad, que nos gustaría, en las traducciones.

Yo recomiendo el traductor online gratuito: Deepl. Para acceder a su página web, hay que buscar en el navegador:

Traductor deepl

Y buscar también un manual en castellano para aprender a utilizarlo (aunque es muy similar a la página del Traductor de Google):

Como usar el traductor deepl

Dentro de las funcionalidades de **la versión gratis** del Traductor Deepl, comprobaremos que puede traducir textos, introducidos directamente en la página web. Y, también, puede traducir **documentos** con un **tamaño máximo de 5M**. Es decir, 5 MegaByte que son 5000K (5000 KiloByte). Y con la restricción de **solamente** poder traducir **3 documentos al mes**.

Si lo prefiere, dispone de una aplicación de ordenador y/o móvil para su descarga desde la página web de Deepl.

ojo

Si quiere traducir un documento en formato PDF de más de 5M (mayor a 5000K), puede buscar en Internet páginas web para que le ayuden a "eliminar páginas en PDF" de un documento con este formato, hasta que el tamaño del archivo sea menor de 5M.

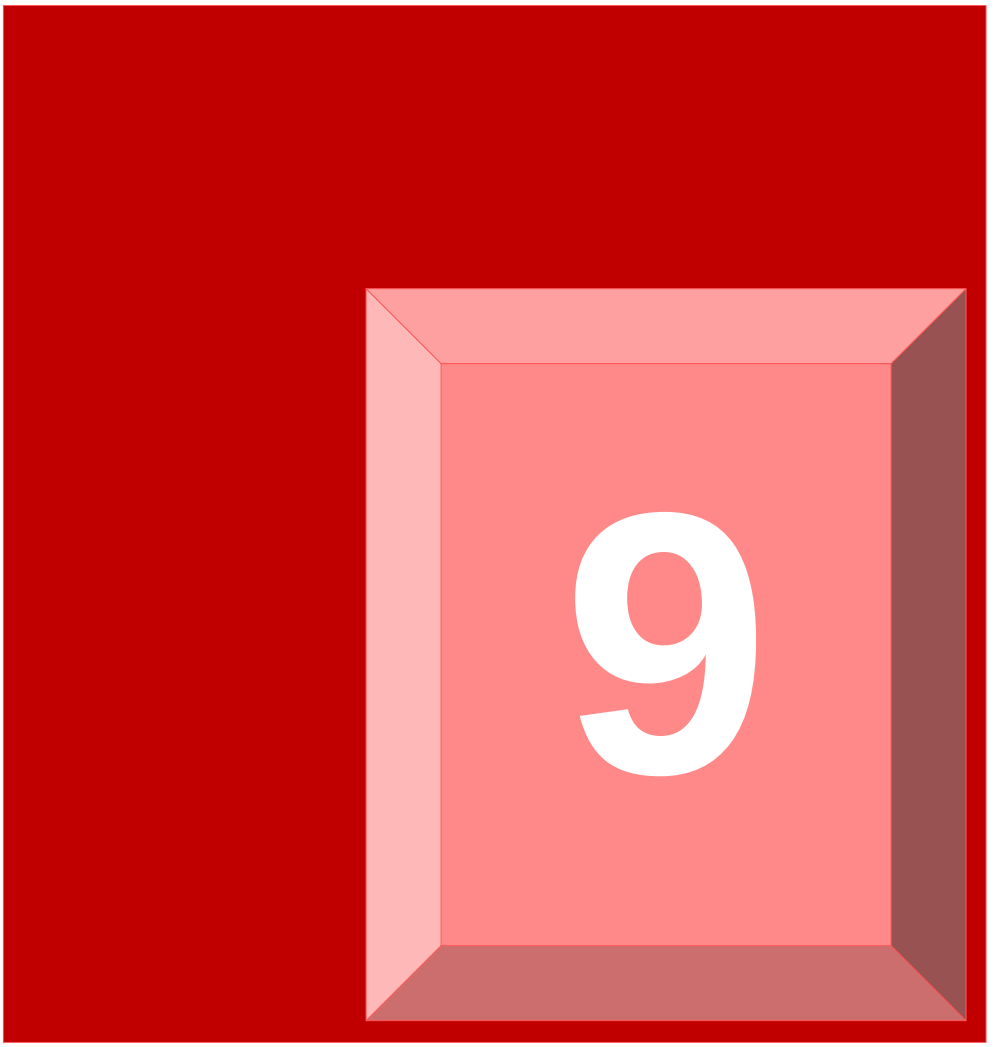
Traducción de vídeos

En lo referente a la traducción de vídeos, recomiendo utilizar la aplicación móvil: SayHi.

Que se encuentra gratuitamente (esta vez es gratuita completamente) en la tienda de aplicaciones de Amazon: Appstore Amazon.

Por lo tanto, para un teléfono Android, primero deberá descargarse la aplicación móvil de la tienda de aplicaciones de Amazon y, en la tienda, buscar la aplicación SayHi para su descarga e instalación.

Despedida



Capítulo 9

Despedida

Aprender a Programar | Ejemplos en PSeInt

Y, ¡eso es todo por ahora!. Pero… espere, aún hay más que decir.

Agradecimientos

La mayoría de los libros acaban con los agradecimientos para reconocer el esfuerzo de todas las personas que han rodeado al autor para que pudiera hacer realidad su obra.

Yo no iba a ser menos, en mostrar mi agradecimiento a mi esposa *Belén* y a mis hijos *Adrián* y *Miguel* por aguantar las largas horas de dedicación, al margen del trabajo cotidiano de un esposo y padre de familia.

También quería agradecer, en primer lugar, a mi hermano mayor *Juan Carlos* que me enseñó y me guio en este campo de la programación; precisamente en la creación de mis primeros algoritmos en pseudocódigo. La programación que tanto me ilusionaba, se convirtió en mi pasión y posteriormente en mi profesión.

Quiero agradecer su labor, esfuerzo, profesionalidad y ejemplo a mi primer profesor en la Universidad Politécnica de Madrid en España, en la disciplina de Programación: *José Gabriel Pérez Díez*; que fue el que, inicialmente, me enseñó esta profesión del Desarrollo de Software.

Quiero agradecer al creador y mantenedor del programa **PSeInt**:

Pablo Novara

Por su acierto en la creación de este proyecto como materia de *Programación 1* de la carrera de *Ingeniería en Informática* de la Facultad de Ingeniería y Ciencias Hídricas de la Universidad Nacional del Litoral, cuando era estudiante y, su posterior esfuerzo y sacrificio en mantenerlo, actualizarlo, darle soporte al foro de la comunidad y seguir mejorando el programa periódicamente.

Quiero dar las gracias a aquellos que realizaron trabajos sobre **PSeInt** y que me han ayudado a la hora de organizar mi contenido:

→ *Sara Milena López Ordóñez* en su documento "Manual de Programación en PSeInt"
→ *Rommel Castillo Suazo*; junto con *Alejandro Caro* en su documento "Manual de PSeInt"

Para finalizar, sobre todo quiero agradecer al lector el haber utilizado esta obra para su aprendizaje que espero y deseo sea de su utilidad

Dónde Seguir Aprendiendo

Ahora le recomiendo zambullirse en la ayuda de **PSeInt** y practicar con los ejemplos de algoritmos que se disponen en la misma. Ya que visita el sistema de Ayuda de **PSeInt**, le animo a seguir practicando algoritmos y que aprenda a utilizar, con la Ayuda de **PSeInt**, el módulo o funcionalidad de Ejecución Paso a Paso de un programa. La experiencia, en el manejo de este módulo, le ayudará en la utilización de lenguajes "reales" en sus respectivos editores.

Pero si ya está dispuesto a subir un escaloncito en el nivel de dificultad. El siguiente paso es aprender un lenguaje de programación "real". Para ello le recomiendo la página web de consulta de los 24 mejores sitios web gratis para aprender programación.

https://www.hostinger.es/tutoriales/mejores-sitios-para-aprender-a-programar-gratis

Si me permite una recomendación personal, investigue los cursos de programación en los siguientes sitios web gratuitos. Gratis y de pago, en mayor o menor grado.

Gratis, hasta cierto punto	
Code Academy	https://www.codecademy.com/
University of the People	https://www.uopeople.edu/
Saylor Academy	https://www.saylor.org/
FreeCodeCamp	https://www.freecodecamp.org/
Fundación Accenture . Apartado *Fundaula*	https://www.accenture.com/es-es/about/company/fundacion
Fundación Telefónica	https://www.fundaciontelefonica.com/
Miriadax	https://miriadax.net/
Coursera	https://es.coursera.org/
edX	https://www.edx.org/
De pago	
Bitdegree	https://es.bitdegree.org/learning
Udemy	https://www.udemy.com/
Platzi	https://www.platzi.com/

Con el aprendizaje de un lenguaje "real" (lenguaje que sirve para desarrollar cualquier tipo de aplicación funcional), podrá adentrarse en nuevos conocimientos como son los Algoritmos y Estructuras de datos que todo programador debe conocer. Un posible itinerario es: Algoritmo de la Burbuja, Algoritmo Quicksort, Algoritmo Merge Sort, Algoritmo de Búsqueda Binaria, Tablas de Hash, Pilas, Colas, Listas Enlazadas, Árboles, Grafos, Árboles Binarios, Árboles Heaps, Algoritmo BFS, Algoritmo DFS, Algoritmo de Dijkstra, Algoritmos de Optimización. Y si quiere seguir profundizando… ya pone el límite usted mismo: Algoritmo de Big Data: Apache MapReduce, Algoritmo de Blockchain: Árbol de Merkle, Algoritmos de Inteligencia Artificial, Realidad Aumentada, etcétera.

En el aprendizaje de un lenguaje "real", le será de utilidad compartir su experiencia y conocimiento en el mismo; y, sobre todo poder resolver sus propias dudas, visitando la página web profesional:

En su versión en castellano: https://es.stackoverflow.com/
En su versión en inglés: https://stackoverflow.com/

En cuanto a las dudas, puede que sea de su utilidad hacer uso de las páginas web de documentación de herramientas y productos siguientes:

https://devdocs.io/
https://overapi.com/

Para obtener una descripción (explicación) de los errores que nos aparecen en los lenguajes "reales" a la hora de programar; es muy habitual utilizar la página web del buscador de Google:

https://www.google.com

Pero, permítame indicarle una alternativa a Google, entrando en la página web:

https://www.you.com/code

Y teclee, por ejemplo, la búsqueda:

java loop

Si prefiere buscar en castellano, pruebe con la traducción de la anterior búsqueda:

bucle java

Y si lo que pretende es seguir aprendiendo como aficionado a los algoritmos, visite esta página.

http://rosettacode.org

En ella encontrará infinidad de algoritmos y propuestas de soluciones a los mismos, en casi cualquier lenguaje de programación "real".

Si desea prepararse para aumentar su lógica de programación, visite la página siguiente.

https://retosdeprogramacion.com/

Y si está decidido a emprender una carrera profesional en desarrollo de software, visite las páginas web.

http://www.leetcode.com
https://www.hackerrank.com

En estas páginas podrá prepararse para una entrevista laboral técnica como desarrollador informático profesional.

Habilidades técnicas para llegar a ser un programador profesional

Los siguientes conocimientos de habilidades y técnicas, solamente se adquieren con el ejercicio profesional de un trabajo profesional en una empresa; y, se necesitan varios años para dominarlos. La mayoría de estos temas aquí expuestos no los encontrará en los libros de informática y están reservados al ámbito profesional.

- Familiarizarse con los componentes de un ordenador, su función y funcionalidad.
- Aprender a usar la Línea de Comandos.
- Realizar descargas e instalación de aplicaciones.
- Conocer un IDE (Entorno de Desarrollo Integrado) profesional.
- Aprender un lenguaje de programación productivo ("real"): algoritmos y estructuras de datos, patrones de diseño, desarrollo dirigido por test (TDD).
- Aprender a programar: programación imperativa y procedimental utilizando la consola (es la técnica que hemos utilizado en la realización de este libro), profundizar en la programación estructurada, programación orientada a objetos, programación de aplicaciones con interfaz gráfica; y, quizás programación funcional y, programación concurrente (paralela). Además, podría adentrarse en la programación web, la programación de dispositivos móviles e, incluso, la programación IoT (Internet of Things - Internet de las Cosas).
- Manejo de errores y excepciones del programa.
- Uso de Internet: consultas en la web, consulta de documentación online, uso de foros técnicos.
- Aprender a probar aplicaciones: casos de prueba, pruebas de rendimiento.
- Comprender los roles y perfiles de los usuarios de la aplicación, y a contemplarlos en la creación del código de la aplicación.
- Formatear los datos de salida del programa.
- Validar los datos de entrada al programa.
- Almacenar los datos en ficheros externos y en una base de datos relacional.
- Repasar la Teoría de Conjuntos y aprender el lenguaje SQL.
- Dotar de seguridad informática a las aplicaciones desarrolladas.
- Utilizar un gestor de pantallas para construir la interfaz visual de la aplicación para los usuarios.
- Aprender sobre Experiencia de Usuario (UX) y sobre Interfaces de Usuario (UI).
- Aplicar la Usabilidad y el Diseño Centrado en el Usuario.
- Aprender a generar informes con una herramienta específica para confeccionarlos, saber representar información gráficamente, crear cuadros de mando (dashboard), exportar información a Microsoft Office y/o LibreOffice.
- Aprender a documentar el código fuente con comentarios y a documentar: cómo se ha desarrollado la aplicación y a confeccionar los manuales de usuario.
- Dar la formación a los usuarios: elaborar la documentación técnica de la formación, preparar la presentación, impartir las jornadas de formación.
- Hacer la instalación y la distribución de la aplicación al usuario final.
- Conocer la legislación aplicable en la profesión informática y en al ámbito de la aplicación concreta que va a desarrollar: Licencias de distribución del software, uso del software desarrollado.
- Saber estimar el tiempo que va a costarle terminar una de las anteriores tareas; y/o la tarea en la que está trabajando o a punto de empezar a comenzar.

Una vez que haya obtenido la experiencia suficiente, podrá probar a trabajar sin fronteras, en la web de ofertas laborales de teletrabajo: **http://remoteok.com**

Sobre el autor

José Luis Rodríguez Muñoz es padre de dos hijos y esposo. Ha estudiado **Diplomado en Informática** por la Universidad Politécnica de Madrid en España (1995-Título universitario de habilitación a nivel español) y el **Grado en Ingeniería Informática** por la Universidad de León en España (2013-Título universitario de habilitación a nivel europeo). Además, ha cursado el **Certificado de Aptitud Pedagógica** (CAP) en la Universidad Alfonso X El Sabio de Madrid en España (2007), cuyo título le acredita como profesor de Enseñanza Secundaria para poder ejercer la docencia preuniversitaria (nivel educativo de Secundaria y Formación Profesional) que es una habilitación para todo el estado español.

Empezó trabajando de programador y ha realizado todos los trabajos de desarrollador de software hasta el de jefe de proyectos.
Actualmente, trabaja en desarrollo de software en el ***Organismo Autónomo Informática del Ayuntamiento de Madrid (IAM)*** donde supervisa, a empresas privadas externas, los encargos de desarrollos web de aplicaciones de gestión municipal de la ciudad de Madrid.

Se considera una persona familiar y amante de la informática. Además de este libro de programación que está leyendo; es un poeta aficionado con varios libros publicados en Amazon.

Estaría muy agradecido si me enviara un correo electrónico, indicándome sus impresiones sobre esta obra y las formas en las que se podría mejorar

Contacto

Mi correo electrónico es:

elpuma1968@gmail.com

El acceso a la página web del libro, se encuentra en la dirección de Internet:

https://elpuma1968.weebly.com

Donde podrá copiar el código fuente de los ejemplos, ejercicios, soluciones y las aplicaciones presentadas.

Hoja de Ruta (Roadmap) de este libro para aprender a programar

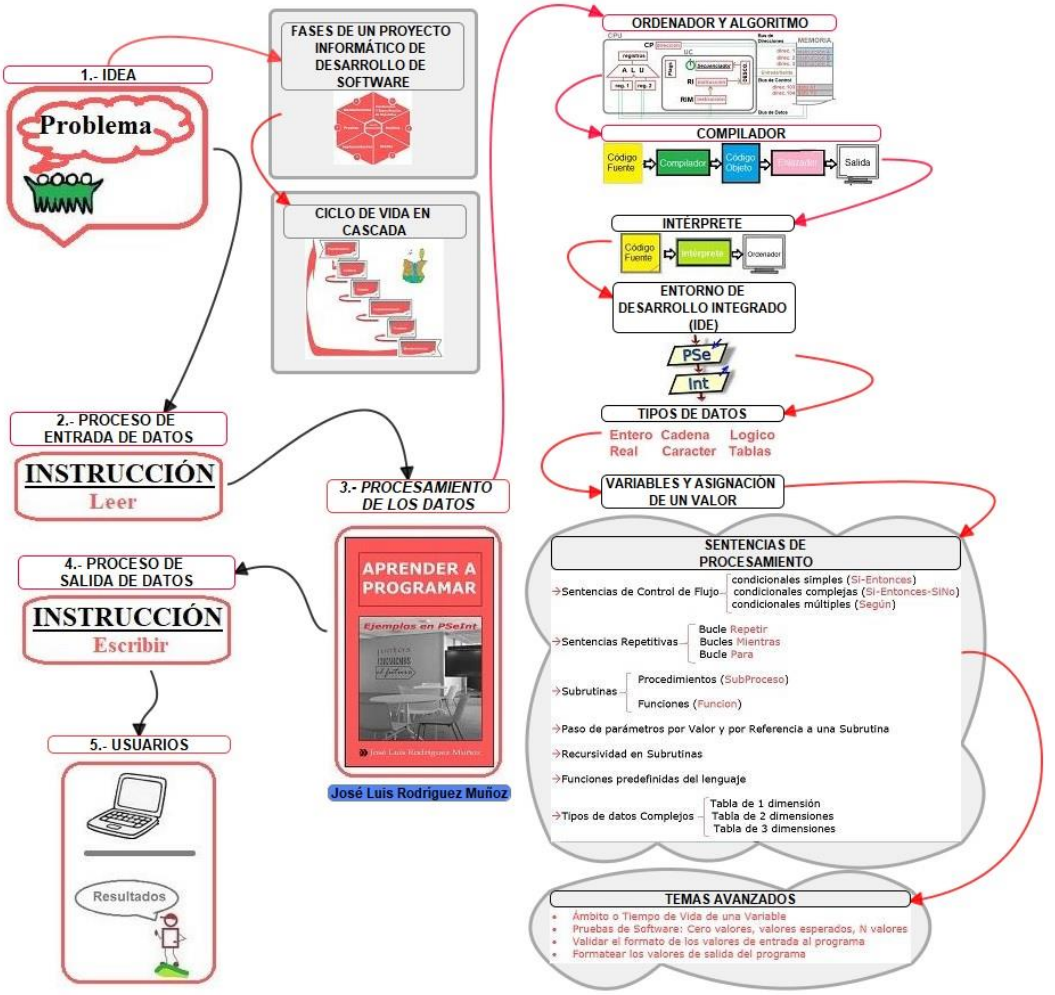
Una vez completado el libro, detallo los puntos tratados en el mismo de forma que pueda autoevaluarse para saber si entendió, recuerda y/o debe profundizar en algún concepto presentado.

- → Familiarizarse con la resolución de problemas
- → Concepto básico de "Ciclo de Vida del Desarrollo de Software": Fases de un Proyecto Informático de Software, Desarrollo de un Proyecto en Cascada
- → Ordenador: funcionamiento de la CPU en la Arquitectura Von Neumann
- → Qué es un Algoritmo: Programa de ordenador
- → Tipos de Lenguajes de Programación: Compiladores e Intérpretes
- → Estudio de un IDE (Integrated Development Environment - Entorno de Desarrollo Integrado): instalación, interfaz gráfica, editar Código Fuente
- → Tipos de Datos Simples: Entero, Real, Cadena, Caracter, Logico
- → Variables y Asignación de un Valor
- → Tipos de Sentencias: de Entrada (instrucción Leer), de Salida (instrucción Escribir), de Procesamiento (el resto del tipo de sentencias)
- → Sentencias de Control de Flujo: condicionales simples (Si-Entonces), condicionales complejas (Si-Entonces-SiNo), condicionales múltiples (Según)
- → Sentencias Repetitivas: Bucle Repetir, Bucle Mientras, Bucle Para
- → Subrutinas: Procedimientos (Subproceso) y Funciones
- → Paso de parámetros por Valor y por Referencia a una Subrutina
- → Recursividad en Subrutinas
- → Funciones predefinidas del lenguaje
- → Tipos de datos Complejos: Tabla de 1 dimensión, Tabla de 2 dimensiones
- → Ámbito o Tiempo de Vida de una Variable
- → Pruebas de Software: Cero valores, valores esperados y Valores Límites, N valores
- → Validar el formato de los valores de entrada al programa
- → Formatear los valores de salida del programa

Próximos pasos (cuando conozca un lenguaje de programación "real"):

- → Almacenar datos con la gestión de la información en ficheros externos
- → Almacenar datos usando Bases de Datos Relacionales
- → Aprender el lenguaje SQL

Mapa Mental de la hoja de ruta (Roadmap) de este libro para aprender a programar



Glosario de Términos

J

M

O

P

R

S

T

U

V

Este libro se encuentra publicado en papel en la web de Amazon