**Introducción a la programación**

**Características:** finito, legible, modificable, eficiente, modulable y estructurado.

**Diagramas de flujo**

****

**Lenguajes compilados:** son aquellos lenguajes que utilizan un compilador para poder traducirlo al lenguaje máquina (para un programa escrito en C# usaríamos la máquina virtual CLR.

**Lenguaje interpretado:** no genera un programa escrito en una máquina virtual. Efectúa directamente la traducción y ejecución simultáneamente para cada una de las setencias. Un intérprete verifica cada línea del programa cuando se escribe. La ejecución es más lenta debido a esa traducción simultánea.

**Tipos de variable**

**Operadores**



**Conversión implícita:** La que se hace automáticamente, sin necesidad de que el programador incluya ningún tipo de parámetro. Tener cuidado con no perder información.

int a = 10; double b = 9.2; b = a; Salida por pantalla: 10

**Conversión explicita:** Lo que se llama hacer un casting, o castear.

double x = 123.1; int a = (int)x; Salida por pantalla: 123

Edsger Dijkstra comprobó que todo programa se puede escribir utilizando únicamente **tres instrucciones de control**:

* Secuencia de instrucciones.
* Instrucción condicional. (if/If...else)
* Iteración o bucle de instrucciones. (While/Do...While/for)

**Tipos enumerados**: ofrecen la posibilidad de definir nuevos tipos simples. Estos tipos se declaran en el apartado "tipos" de un programa de la siguiente forma:

enum Dias {Domingo,Lunes,Martes,Miercoles,Jueves,Viernes,Sabado};

**Arrays:** int [] v = new int [10]; int[] array1 = new int[] { 1, 3, 5, 7, 9, 11, 13, 15, 17, 19 }; v[0]… para acceder.

**Matrices:** int [,] matriz = new bool[2,3];

**Cadenas:** string.

**Métodos string**

****

**Depuración de errores**

**Errores de compilación o sintáxis:** errores en el código.

**Errores en tiempo de ejecución:** los que producen un fallo a la hora de ejecutar el programa. Se trata de fragmentos de código que parecen estar correctos y que no tienen ningún error de sintaxis, pero que no se ejecutarán. Por ejemplo, podría escribir correctamente una línea de código para abrir un archivo, pero si el archivo está dañado, la aplicación no se puede llevar a cabo la apertura de ese archivo y se detiene la ejecución.

**Errores lógicos:** son aquellos que aparecen cuando la aplicación está en uso. Son la mayoría de los resultados erróneos, diferentes a los esperados o no deseados, a menudo en respuesta a las acciones del usuario. Son los más difíciles de corregir, ya que no siempre está claro dónde se originan.

|  |
| --- |
| **FUNCIÓN**  |
| Ámbito de la declaración Tipo Función Nombre\_función (parámetros) { //declaración de variables locales //Instrucciones//retorno del tipo}  | Utilizamos la palabra reservada *function* con una lista de parámetros (variables) que vamos a utilizar en ellasCuando la función llega a su fin, retornará un valor del mismo tipo de la función con la directiva *return*  |
| PROCEDIMIENTO  |
| Ámbito de la declaración void Nombre\_procedimento{ // Instrucciones }  | Un procedimiento no devuelve un valor y por tanto no incluye la directiva return  |

**Parámetro real (o actual):** Variable con la que llamamos a la función.

**Parámetro formal:** Solo tiene validez dentro de la función.

**Paso por valor:** Se realiza una copia del parámetro de la función que se ha pasado, y nunca se modifica el original.

**Paso por referencia:** Se trabaja siempre con los originales, por lo tanto todas aquellas modificaciones que se realicen en la función van a afectar a sus parámetros.

**Concepto de librerías:** Cuando hablamos de librerías nos referimos a archivos que nos permiten llevar a cabo diferentes acciones y tareas sin necesidad de que el programador se preocupe de cómo están desarrolladas, solo debe entender cómo utilizarlas.

**La recursividad tiene la siguiente estructura:**
• Un caso base que permita la finalización del programa.

• Casos recursivos, que son los que se van a encargar de que la función vuelva a ejecutarse, pero acercándose cada vez más al caso base.

**Los ficheros o archivos** son una secuencia de bits (0 y 1) que se almacenan en un dispositivo de almacenamiento secundario, por lo que la información va a permanecer a pesar de que se cierre la aplicación que los utilice.

**Ruta absoluta o completa:** se le indica el camino de directorio desde el comienzo. Si es en el sistema operativo Linux o, empezará por la raíz (/), en caso contrario si el sistema operativo es Windows debe de empezar por el nombre de la unidad en cuestión.

**Ruta relativa:** se le indica el camino de directorio desde la posición actual.

**Tipos de fichero**

**Secuencial:** en los ficheros secuenciales los registros se van almacenando en posiciones consecutivas de manera que cada vez que queramos acceder a ellos tendremos que empezar desde el primero e ir recorriéndolos de uno en uno.

Solo se puede realizar una operación de lectura o escritura a la vez. Por ejemplo, si estamos leyendo el fichero, no podemos realizar ninguna operación de escritura sobre él hasta que termine de ser leído y viceversa.

**Aleatorio o directo (No)**: en los ficheros aleatorios o directos podemos acceder a un registro concreto del mismo indicando una posición perteneciente a un conjunto de posiciones posibles.

Debido a que los registros están organizados, estos pueden ser leídos o escritos en cualquier orden, ya que se accede a cada uno a través de su posición. Cuando queremos realizar una operación, basta con colocar el puntero que maneja el fichero justo antes de éste.

**Secuencial indexado**: los ficheros indexados poseen un campo clave para ser identificados.

Permiten el acceso secuencial y aleatorios a un fichero de la siguiente forma:

1) Primero busca el índice en una tabla.

2)Una vez que lo encuentra, el acceso al fichero es directo, ya que solo tenemos que acceder a la posición indicada por el índice.

**Operaciones sobre ficheros**

Cuando abrimos el fichero estamos relacionando un objeto de nuestro programa con un archivo.

Los diferentes modos en los que se puede abrir un fichero son los siguientes:

* ***Lectura***: solamente va a realizar operaciones de lectura en el fichero.
* ***Escritura***: realiza operaciones de escritura en el fichero. Si ya existía, lo sobrescribe.
* ***Añadir***: permite realizar la operación de escritura en un fichero que ya existía añadiéndole a lo anterior.

**Los flujos** (también llamado **Stream**) de datos son las estructuras o pasarelas que tenemos para acceder a los datos de un fichero, de una forma consistente y fiable, desde un código fuente en un cualquier lenguaje de programación (**FileAccess.Read/Write/ReadWrite**; **StreamReader/Writer**).

**BinaryReader y BinaryWriter.**