Programación en C/C++

Elva Rego García

Índice

- ¿Por qué aprender C/C++?
  - Es un lenguaje muy potente que permite hacer todo tipo de programas, desde aplicaciones para móviles a sistemas de control de centrales nucleares.
  - Es muy flexible. Permite bajar a niveles muy bajos de programación o abstraernos usando librerías.
  - En C/C++ tenemos la posibilidad de hacer programación orientada a objetos o no.
  - Hay infinidad de tutoriales y ayuda en internet.
  - Es muy didáctico. Una vez sabemos programar en C/C++ podemos programar en cualquier otro lenguaje estructurado sin mucho esfuerzo.
  - Muy utilizado en entornos industriales.
  - Ideal para comprender cómo funciona la memoria y el procesador.

"Si dominas C o C++, entiendes cómo piensa la máquina."

- Diferencias fundamentales respecto a Matlab:
  - C/C++:
    - Es un lenguaje de más bajo nivel por lo que no es tan sencillo.
    - El tiempo de ejecución es muchísimo menor!!!
    - Podemos seleccionar la plataforma de ejecución.
    - Es gratis.

  - Matlab:
    - Es un lenguaje sencillo que permite realizar pruebas de concepto con mucha rapidez.
    - Está orientado a matrices y es muy rápido visualizar resultados.
    - Tiene infinidad de librerías de fácil acceso para el procesado se señales y simulaciones de todo tipo.
    - Normalmente se ejecuta dentro de la propia aplicación.
    - Es de pago.

"*MATLAB* es ideal para prototipar; C/C++ para producción."

- Diferencias fundamentales respecto a Java:
  - C++:
    - Permite programación orientada a objetos o no.
    - Tiene el control del procesador.
    - Es multiplataforma. El compilador genera código máquina para una plataforma concreta a elección del programador.
    - El programador controla cuando se libera la memoria (delete).
    - Existen punteros que permiten al programador explorar la memoria física de la máquina.
  - Java:
    - Solo programación orientada a objetos. Cualquier función en Java debe pertenecer a una clase.
    - Se ejecuta sobre un procesador virtual (Java Virtual Machine).
    - Siempre se ejecuta en el JVM.
    - Se libera memoria de forma automática. ¿Esto es bueno o malo?
    - Los objetos son accedidos mediante referencias, por lo que todo son punteros, pero no podemos usarlos para acceder a la memoria física.

    "C/C++ ofrece control total; Java apuesta por portabilidad y simplicidad."

- Como otros lenguajes de programación el programador escribe un código fuente y un compilador lo transforma en código máquina. El compilador es el traductor entre humanos y máquinas.
- Funciones principales:
  - Analizar el código y detectar errores de sintaxis
  - Optimizar el programa para que sea más rápido y eficiente
  - Generar el archivo ejecutable
- Hay varios compiladores disponibles:
  - GCC (Linux y multiplataforma)
  - MSVC integrado en Visual Studio (Microsoft)
  - Clang/LLVM
  - Compiladores específicos por plataforma.
    - MPLAB (Microchip)
    - Tia Portal (Siemens)

"Sin compilador, tu código sería solo un conjunto de ideas sin voz para la máquina."

- Los IDE (Integrated Development Environment) son aplicaciones que reúnen en un solo lugar todas las herramientas necesarias para programar, facilitando el desarrollo de software:
  - Edición de código: Resaltan en diferentes colores las variables, funciones, tipos de variables, …
  - Autocompletado de variables, funciones, etc…
  - Informe de errores en el código.
  - Depuración de programas:
    - Es una herramienta fundamental para corregir errores en el código.
    - Ejecución paso a paso.
    - Visualización y modificación del contenido de las variables durante la ejecución.
    - Alterar el orden de ejecución del código.
- En este curso vamos a usar el IDE de Microsoft: **Microsoft Visual Studio**

"El IDE es tu taller: un solo lugar para escribir, probar y pulir tu código."

- Ejercicio práctico:

- Crear un nuevo proyecto de Visual Studio del tipo “Aplicación de consola” llamado *EjerciciosC++*

- Crear archivo *EjerciciosC++*.h y escribir en él #include <iostream> dentro de la carpeta “Archivos de encabezado”

- Eliminar el #include <iostream> del *EjerciciosC++*.cpp.

- Hacer el include al archivo *EjerciciosC++*.h en *EjerciciosC++*.cpp

Variables:

- Se crean nuevas variables escribiendo

tipoVariable nombreVariable;

int a, b;

float x, y, z;

char c = 'p'; (Se pueden inicializar en su declaración)

bool b = true;

double d = 3.21;

- Pueden ser locales (válida dentro de la función donde se crea) o globales (declaradas en el archivo .h y válidas en los .cpp que las incluyan)

- Podemos crear nuestros propios tipos de variables creando estructuras:

```
struct tipoVariable
{
    tipoVariable nombre1;
    tipoVariable nombre2;
•    ……
    tipoVariable nombreN;
};
```

Operaciones:

Operadores	Descripción	Sintaxis
++	postincremento	lvalue + +
++	preincremento	+ + lvalue
- -	postdecremento	lvalue - -
- -	predecremento	- - lvalue
*	multiplicación	expr * expr
/	división	espr / expr
%	resto	expr % expr
+	suma	expr + expr
-	resta	expr - expr
<	menor que	expr < expr
< =	menor o igual que	expr < = expr
>	mayor que	expr > expr
> =	mayor o igual que	expr > = expr
= =	igual que	expr = = expr
! =	distinto a	expr ! = expr

Funciones:

Las funciones se declaran en los archivos .h y se definen en los .cpp. La declaración de la función indica al compilador las variables de entrada y salida de dicha función.

La declaración en el .h sería:

tipoSalida nombreFuncion(tipo(s)Entrada) ;

int Mayor(int, int);

La definición en el .cpp sería:

```
tipoSalida nombreFuncion(tipo(s)Entrada variable(s)Entrada)
{
    …
    return (salida);
}
int Mayor(int a, int b)
{
    if (a >= b)
        return a;
    else
        return b;
}
```

Selección:

```
if (condicion1)
{
        …
}
else if (condicion2)
{
        …
}
else
{
        …
}
```

Ejemplos de condiciones:

- **(A == B)**
- **(A != B)**
- **!(A > B)**
- **(A == B) && (A >= C)**
- **(A == B) || (A == C)**

Selección:

```
switch (expresión)
{
        case cte1:
                ….
                break;
        case cte2:
                …
                 break;
        case cte3:
        case cte4:
                …
                 break;
        default:
                …
                 break;
}
```

Bucles:

```
for (inicio; condición fin; incremento)
{
        …
}

while (condición)
{
        …
}

do
{
        …
} while (condición)
```

Entrada y salida de consola y de fichero:

- Para entrada y salida de información por consola:

printf() **cout << "Hola“ << endl << “mundo" << endl**

getchar() **cin >> variable**

- Para entrada y salida de información por fichero podemos crear archivos de texto:

Archivos de texto - escritura:

```
#include <fstream>
ofstream fw("fichero.txt");
fw << "Hola“ << endl << “mundo" << endl;
fw.close();
```

Entrada y salida de consola y de fichero:

Archivos de texto - lectura:

```
#include <string>
ifstream fr("fichero.txt");
string linea;
while (getline(fr, linea)
        cout  << linea << endl;
fr.close();
```

- Ejercicios prácticos:

1. Escribir un programa que muestre una lista de números del 1 al 20, indicando a la derecha de cada uno si es divisible por 3 o no.

2. Escribir el programa anterior, pero usando una función para verificar si el número es divisible por tres, un bucle de tipo while, y los números en orden inverso.

3. Escribir un programa que muestre una salida de 10 líneas de este tipo:

1

1 2

1 2 3

1 2 3 4

...

4. Hacer una función que calcule el factorial de un número introducido por consola.