SISTEMAS OPERATIVOS EN RED

2º SMR
Profesora: Noelia Huguet Chacón
Centro de Estudios Tobalcaide
Curso 2.020 – 2.021

INTRODUCCIÓN

- C es un lenguaje de programación con una amplia trayectoria e influencia histórica sobre otros lenguajes (C++, Java, C#, etc.).
- Este curso permite aprender los aspectos fundamentales de C++ y crear pequeños programas con este lenguaje.
- **C++** es una mejoría sobre muchas de las características de C, y proporciona capacidades de POO (**programación orientada a objetos**) que promete mucho para incrementar la productividad, calidad y reutilización del software.
- En **C,** la unidad de programación es la **función**, con lo cual, se trata de una programación orientada a la acción.
- En **C++,** la unidad de programación es la **clase** a partir de la cual, los objetos son producidos. Se trata , pues, de una programación orientada al objeto.

INTRODUCCIÓN

- Para realizar un programa en C necesitamos un compilador de lenguaje C instalado en nuestro ordenador.
- Un **compilador** es un programa capaz de traducir nuestras órdenes en órdenes comprensibles por el PC, es decir, un programa que nos permite crear programas.
- Nuestras órdenes estarán escritas en lo que conocemos como lenguaje de programación (C, en este caso), las órdenes comprensibles por el PC se llaman código máquina, son ceros y unos, y se guardan en ficheros con extensión ".exe", que es lo que llamamos ficheros ejecutables o programas.
- Los compiladores suelen incorporar un entorno de desarrollo con ventanas, en el que podremos escribir nuestro programa, además de otra serie de herramientas y facilidades para el programador (nosotros).

INSTRUCCIÓN

- **INCLUDE:** es una palabra clave que hace referencia a una instrucción al preprocesador que está presente en la gran mayoría de lenguajes de alto y medio nivel, de forma genérica se usa para adicionar un archivo al código, como por ejemplo la llamada a una biblioteca de funciones en C++ :

  #include <iostream>

- Antes del proceso de compilación, el preprocesador es llamado primero a ejecutarse y buscar llamadas de instrucción al pre-procesador, la instrucción **include** le indica al preprocesador que cuando este se ejecute, el compilador debe incluir un archivo en el código.
- Deberá aparecer al principio de cualquier programa que escriba cosas en pantalla o lea cosas desde teclado.

BIBLIOTECAS

- **stdio.h** (que significa "standard input-output header" (cabecera estándar E/S)) es el archivo de cabecera que contiene las definiciones de las macros, las constantes, las declaraciones de funciones de la biblioteca estándar del lenguaje de programación C para hacer operaciones, estándar, de entrada y salida, así como la definición de tipos necesarias para dichas operaciones.
- **iostream** es un componente de la biblioteca estándar (STL) del lenguaje de programación C++ que es utilizado para operaciones de entrada/salida. Su nombre es un acrónimo de Input/Output Stream.

TIPOS PRIMITIVOS

- La función **main** es imprescindible en cualquier programa C/C++ representa el punto de inicio de su ejecución.
- Debe tener uno de los siguientes formatos:
  - ***int main () {*** *cuerpo* ***}***
  - ***int main (int*** *argc*, ***char ****argv[]****) {*** *cuerpo* ***}***

- La sentencia **return** se emplea para salir de la secuencia de ejecución de las sentencias de un método y, opcionalmente, devolver un valor. Tras la salida del método se vuelve a la secuencia de ejecución del programa al lugar de llamada de dicho método.
- Para finalizar ***int main*** utilizaremos ***return 0***.

COMENTARIOS

- Cuando se escriben programas es muy útil agregar comentarios que ayuden a explicar lo que realiza un programa.
- En C++ se pueden utilizar tres tipos de comentarios:
  - al estilo **C**, se caracterizan por lo siguiente: comenzar el "bloque" de comentarios con /* y terminar dicho "bloque" de comentarios con */

```
/*
Este es un comentario al estilo C.
Todo lo escrito dentro de las etiquetas de apertura y cierre es un comentario.
A estos comentarios se le llaman multilinea, lógicamente
por el hecho de permitir varias lineas de comentarios.
 */
```

  - al estilo **C++,** se empiezan los comentarios con //

```
// Esto es un comentario al estilo C++
```

  - Otra posible forma de comentar código es usando el **preprocesador**.

SALIDA DE TEXTO POR PANTALLA

- Para imprimir una salida de texto en C++ se hace uso de la instrucción ***cout***, junto con **<<.**
- Es importante tener en cuenta que la instrucción ***cout*** siempre va acompañada de **<<** para controlar el flujo de datos que sale.

```
cout << "Hola Mundo" ;
```

- También se puede utilizar en C++ la instrucción **printf()**.

```
printf ("Hola Mundo") ;
```

SALIDA DE TEXTO POR PANTALLA

***printf()*.**

- La instrucción ***printf*** sirve para presentar texto por pantalla, y que el texto que se presenta es lo que está en comillas.
- Si se escribe el siguiente programa:

```
#include <stdio.h>
int main()
{
	printf ("Hola Tobalcaide" );
	return 0;
}
```

- Al compilar, saldría por nuestra pantalla lo siguiente:

```
Hola Tobalcaide
--------------------------------
Process exited after 0.1061 seconds with return value 0
Presione una tecla para continuar . . .
```

SALIDA DE TEXTO POR PANTALLA

cout<<

- La instrucción ***printf*** sirve para presentar texto por pantalla, y que el texto que se presenta es lo que está en comillas.
- Si se escribe el siguiente programa:

```
#include <iostream>
using namespace std;
int main()
{
cout << "Hola Tobalcaide";
return 0;
}
```

- Al compilar, saldría por nuestra pantalla lo siguiente:

```
Hola Tobalcaide
--------------------------------
Process exited after 0.1061 seconds with return value 0
Presione una tecla para continuar . . .
```

SALIDA DE TEXTO POR PANTALLA

- Hay constante para poder espaciar el texto que se escribe, como tabular, o cambiar de línea.
- Las más utilizadas son:
  - **\a** carácter de alarma
  - **\n** nueva línea ; ***<<endl*** (también sirve)
  - **\t** tabulador horizontal
  - **** poder escribir la diagonal invertida
  - **\'** poder escribir el apóstrofe
  - **\"** poder escribir las comillas

SALIDA DE TEXTO POR PANTALLA

- Si se escribe el siguiente programa:

```
#include <stdio.h>
int main()
{
	printf ("Hola Noelia. \n Bienvenida a Tobalcaide." );
	return 0;
}
```

- Al compilar, saldría por nuestra pantalla lo siguiente:

```
Hola Noelia
Bienvenida a Tobalcaide
--------------------------------
Process exited after 0.1061 seconds with return value 0
Presione una tecla para continuar . . .
```

TIPOS DE DATOS

- Los tipos de datos de un lenguaje son, como su propio nombre indica, los tipos de datos con los que se puede trabajar en un lenguaje de programación.
- El lenguaje C++ ofrece una colección de tipos de datos bastante limitada, aunque no por ello poco funcional.
- Dicha colección se compone de los siguientes tipos:
  - **Números enteros** definidos con la palabra clave ***int***
  - **Letras o caracteres** definidos con la palabra clave ***char***
  - **Números reales** o en coma flotante definidos con las palabras claves ***float*** o ***double***

TIPOS DE DATOS

TIPO	EJEMPLO	BYTES	RANGO
char	'C'	1	0 a 255
string	Noelia Huguet		Cadena de carácteres
short	-15	2	-128 a 127
int	1024	2	-32768 a 32767
unsigned int	42345	2	0 a 65535
long	262144	4	-2147483648 a 2147483637
float	10,45	4	3,4*(10 e -38) a 3,4*(10 e 38)
double	0,00000000045	8	1,7*(10 e -308) a 1,7*(10 e 308)
long double	1e-8	8	1,7*(10 e -308) a 1,7*(10 e 308)
bool	True o false	1	0 o 1

CONSTANTES Y VARIABLES

- Se define un dato **constante** como un dato de un programa cuyo valor no cambia durante la ejecución.
- Por el contrario, un dato **variable** es un dato cuyo valor sí cambia en el transcurso del programa.

Identificador	¿es válido?
x	Sí
5x	No, porque no empieza por una letra
x5	Sí
pepe	Sí
_pepe	No, porque no empieza por una letra
pepe_luis	Sí
pepe!luis	No, porque contiene caracteres especiales
raiz	No, porque coincide con la función raíz(x)

OPERACIONES

- Las operaciones básicas son:
  - **Suma (+)**
  - **Resta (-)**
  - **Multiplicación (*)**
  - **División (/)**
  - **Módulo (%):** Es para obtener el resto de una división. Si por ejemplo hacemos 8/2 el resto sería 0 porque 8 es múltiplo de 2. Si hacemos 8/3 el resto sería 2 porque 8 no es múltiplo de 3.
  - **Elevado**: la instrucción es **pow(a,b)**, a elevado a b. Hay que incluir la librería **math.h.**

FUNCIONES

- Además de todas las operaciones, los lenguajes de programación disponen de mecanismos para realizar operaciones más complejas con los datos, como por ejemplo:
  - **abs(x)**. Calcula el valor absoluto de x.
  - **sen(x)**, **cos(x).** Calcula el seno y el coseno de x, respectivamente.
  - **ln(x).** Calcula el logaritmo neperiano de x.
  - **log10(x).** Calcula el logaritmo decimal de x.
  - **redondeo(x).** Redondea el número x al valor entero más próximo.
  - **raiz(x).** Calcula la raíz de x.
  - **cuadrado(x).** Calcula el cuadrado de x.
  - **aleatorio(x).** Genera un número al azar entre 0 y x.

OPERACIONES

- Hay otra operación que es muy frecuente cuando se crean programas, pero que no tiene un símbolo específico para representarla en matemáticas.
- Es incrementar o reducir el valor de una variable en una unidad:

```
a = a+1; //Es lo mismo que poner a++
a = a-1; //Es lo mismo que poner a--
```

- Se puede distinguir entre "preincremento" y "postincremento".
- En C es posible hacer asignaciones como:

```
//si inicialmente a vale 2 y b no tiene asignación, si ponemos:
b=a++;
//Lo que esta instrucción hace es dar a "b" el valor de "a" y aumentar el valor de "a".
//Por tanto, al final tenemos que b=2 y a=3 (postincremento: se incrementa "a" tras asignar su valor).
//En cambio, si escribimos:
b=++a;
// "a" valía 2, primero aumentamos "a" y luego los asignamos a "b" (preincremento), de modo que a=3 y b=3.
```

- También podemos distinguir **postdecremento** (a--) y **predecremento** (--a).

TIPOS DE DATOS

```
#include <iostream>
using namespace std;

int main()
{
   int num; //Declaramos el entero en una línea

   num = 5; //Le asignamos un valor en otra línea

   int num2 = 8; //Asignación y declaración al tiempo

   // podríamos definir varias variables con una sola instrucción (int num=5, num2=8;)

   float numero; //Un numero decimal
   numero = 3.5; //Le asignamos un valor al decimal

   float res = numero + num2; //Sumamos dos variables y las asignamos a res (3.5 + 8 = 11.5)
   res = res + num; //Al valor actual de res le sumamos el valor de num (11.5 + 5 = 16.5)

   res = res*2; //Duplicamos el valor de res 16.5*2 = 33
   cout << res << endl; //Mostramos el valor de res por pantalla

   cout << "La suma de 5+7 es igual a " << 5+ 7;  //Muestra el resultado de la suma de 5+7, a continuación de la frase.

   return 0;
}
```

DATOS INTRODUCIDOS POR EL USUARIO

- Ese lugar donde guardar los datos será un espacio de memoria al que daremos un nombre. Esto es lo que se conoce como **una "variable"**.
- Una variable necesita dos datos: el nombre con el que nos referiremos a ella y el tipo de datos que queremos guardar en ella.
- Usaremos la orden "**cin**" para leer datos, y deberemos indicar dónde queremos guardarlos.
- Un programa que te calcula el doble de un número introducido por el usuario.

```
#include <iostream>
using namespace std;
int main ()
{
    int x;
    cout << "Dime un número: ";
    cin >> x;
    cout << "El doble de tu número es " << x*2;
    return 0;
}
```

DATOS INTRODUCIDOS POR EL USUARIO

- Se pueden introducir varios datos seguidos con la misma instrucción (**cin >>**).

```
int x, y,z;
    cout << "Dime tres números: ";
    cin >> x >> y >> z;
    cout << “La suma de los 3 números es: " << x+y+z;
```

- Se pueden introducir una cadena de caracteres con(**cin.getline**).

```
char b[256];
    cout << "Dime tu nombre completo";
    cin.getline (b,256);
```

OPERADORES

- Los **operadores relacionales:**

OPERADOR	OPERACIÓN
<	Menor que
>	Mayor que
<=	Menor o igual que
>=	Mayor o igual que
==	Igual a
!=	No igual a (distinto que)

- Los operadores lógicos

OPERADOR		OPERACIÓN
&&	**and**	Y
||	**or**	O
!		NO

NÚMERO ALEATORIO

- Para indicar que el programa nos de un número aleatorio, hay que incluir dos librerías en él.
  - #include<stdlib.h>
  - #include<time.h>
- Luego inicializar los números aleatorios incluyendo esto:
  - **srand(time(NULL));**
- Y para que proporcione un número aleatorio entre los números que queremos de forma general sería:
  - **variable = limite_inferior + rand() % (limite_superior +1 - limite_inferior) ;**
- Por ejemplo, queremos que sea entre 0 y 100 el número aleatorio:
  - X = 0 + rand()% (100 +1 – 0);

IF

- La primera construcción que usaremos será "si ... entonces ...".
- El formato equivalente en C++ es:

```
if (condición) sentencia;
```

- Un ejemplo para entenderlo mejor:

```
#include <iostream>
using namespace std;

int main()
{
    int x;

    cout << "Escribe un numero: ";
    cin >> x;
    if (x>0)  cout << "El numero es mayor que cero.";

    return 0;
}
```

IF

- La "sentencia" que se ejecuta si se cumple la condición puede ser una sentencia simple o una compuesta.
- Las sentencias compuestas se forman agrupando varias sentencias simples entre llaves ({ y }):

```
#include <iostream>
using namespace std;
int main()
{
  int x;
  cout << "Escribe un numero: ";
  cin >> x;
  if (x>0)
    {
    cout << "El numero es positivo." << endl;
    cout << “También puedes usar negativos." << endl;
    } // Aqui acaba el "if"
  return 0;
} // Aqui acaba el cuerpo del programa
```

IF … ELSE

- Se puede indicar lo que queremos que ocurra en caso de que no se cumpla la condición, usando la orden "**else**" (en caso contrario), así:

```
#include <iostream>
using namespace std;
int main()
{
  int x;
  cout << "Escribe un x: ";
  cin >> x;
  if (x>0) cout << "El numero es positivo.";
    else cout << "El numero es cero o negativo.";
  return 0;
}
```

IF … ELSE

- Se puede enlazar los "**if**" usando "**else**", para decir "si no se cumple esta condición, mira a ver si se cumple esta otra":

```
#include <iostream>
using namespace std;
int main()
{
  int x;
  cout << "Escribe un x: ";
  cin >> x;
  if (numero < 0)
    cout << "El numero es negativo.";
  else
    if (numero == 0)
      cout << "El numero es cero.";
    else
      cout << "El numero es positivo.";
  return 0;
}
```

IF…ELSE

- Utilizar los operadores lógicos:

```
if ((x==1) && (z==2)) ...
//Si x es igual a 1 y z es igual a 2, haría la condición que se escribiría después. Tiene que cumplir las dos condiciones para ejecutarse.

if ((x==1) || (z==3)) ...
//Si x es igual a 1 o z es igual a 3, haría la condición que se escribiría después. Con que cumpla una de las condiciones se ejecutaría.
```

IF…ELSE

- Para poner más condiciones en el **if**, se puede utilizar ***else if*** hasta el final.

```
float x, y;
 cout << "Escribe la nota de Empresas ";
 cin >> x;
 cout << "Escribe la nota de SOR ";
 cin >> y;
 if ((x>=5) && (y>=5))
   cout << "Has aprobado las dos asignaturas";
 else if ((x<5)||(y<5))
   cout << "Has suspendido una asignatura";
 else
      cout << "Al menos una asignatura esta suspendida";
```

OPERADOR CONDICIONAL

- Operador condicional (**?**) `condicion ? x : y;`
- Equivale a decir "si se cumple la **condición**, toma el valor **x**; si no, toma el valor **y**".
- Un ejemplo de cómo podríamos usarlo sería:

```
#include <iostream>
using namespace std;
int main()
{
  int a, b, m;
  cout << "Escriba un numero: ";
  cin >> a;
  cout << "Escriba otro: ";
  cin >> b;
  m = (a>b) ? a : b; // Si a es mayor que b, escogería 'a' sino escogería 'b'.
  cout << "El mayor de los números es " << m;
return 0;
}
```

SWITCH

- Cuando hay que poner varios "if" seguidos o encadenados puede ser muy lioso, para ello hay una alternativa, es la orden "**switch**", cuya sintaxis es:

```
switch (expresión)
{
  case valor1: sentencia1;
    break;
  case valor2: sentencia2;
    sentencia2b;
    break;
  ...
  case valorN: sentenciaN;
    break;
  default:
    otraSentencia;
}
```

SWITCH

- Ejemplo:

```
#include <iostream>
using namespace std;
int main()
{
  int x;
  cout << "Introduce un numero del 1 al 5: ";
  cin >> x;
  switch (x)
  {
    case 1: cout << "Uno";
            break;
    case 2: cout << "Dos";
            break;
    case 3: cout << "Tres";
            break;
    case 4: cout << "Cuatro";
            break;
    case 5: cout << "Cinco";
            break;
    default:  cout << "Valor incorrecto!";
  }
  return 0;
}
```

SWITCH

- Ejemplo 2:

```
#include <iostream>
using namespace std;
 int main()
{
  int x;
   cout << "Introduce un numero del 1 al 5: ";
  cin >> x;
  switch (x)
  {
    case 1:
    case 3:
    case 5:
            cout << "Impar";
            break;
    case 2:
    case 4:
            cout << "Par";
            break;
    default:
            cout << "Valor incorrecto!";
  }

  return 0;
}
```