Programación estructurada: Parte 2

Lenguaje de programación C++

4. Introducción a C ++

El lenguaje de programación C++ es un lenguaje potente, complejo y muy extenso. Sus estructuras son similares a las de C, Java, PHP y muchos otros lenguajes. Por lo que su aprendizaje puede resultar de aplicación a otros lenguajes distintos. Además existen compiladores para la mayoría de los sistemas operativos.

La estructura general de un programa escrito en C++ es:

a) **// Comentario** Es un comentario que no afecta al comportamiento del programa, sirve para ayudar al programador a para recordar para lo que sirve el programa o para orientar a otros que quieran modificarlo. Los comentarios breves empiezan por // y acaban al final de la línea. Si se desea que ocupen varias líneas , empezarían por **/*** y terminarían por ***/**.

b) **#include <isostream>** Debe aparecer al principio de cualquier programa que escriba algo por pantalla o lea algo desde el teclado.

c) **int main()** Indica que lo que aparece a continuación corresponde con el cuerpo del programa. También deberá parecer prácticamente siempre.

d) **{}** son las llaves e indican el principio y final de un bloque, por ejemplo el principio y final de un programa.

e) **std::cout << " Hola, mundo";** Es una orden cuya función es escribir en pantalla lo que se indica entre comillas dobles.

f) **return 0;** sirve para indicar que el programa ha terminado sin errores. Debe incluirse siempre, si queremos que el programa funcione con compiladores modernos y antiguos.

Para probar un programa en C++ hay que escribir el programa en un editor, como gedit y guardarlo con extensión **cpp** (por ejemplo prog1.cpp) en la carpeta Documentos/Nombres/2eval/c++.A continuación se deberá abrir un terminal y teclear **g++ Documentos/nombre/2eval/c++/prog1.cpp -o prog1** (donde **g++** es el nombre del compilador, **prog1.cpp** es el nombre de nuestro programa y **-o prog1** es lo que permite indicar el nombre que tendrá el ejecutable)

Si no aparece ningún mensaje de error el último paso será lanzarlo con **./prog1**

prog1.cpp

Escribe el siguiente código en gedit y compílalo según se ha indicado anteriormente.

```
//prog1
#include <iostream>
int main()
{
        std::cout <<"Hola, mundo";
        return 0;
}
```

prog2.cpp

Crea un programa que escriba tu nombre en pantalla y pruébalo.

prog3.cpp

Puedes usar cout para escribir varios textos. Crea un programa que escriba Hola; luego un espacio en blanco; y , finalmente, tu nombre y pruébalo.

5. Cálculos en C++

Los principales operadores para hacer cálculos son +,-,*,/ y %(Suma, resta o negación,multiplicación, división y resto.

Si queremos escribir varios mensajes con una única orden **cout**, solo es necesario que cada texto se preceda por dos símbolos de ”menor que”, y todo aparecerá escrito en la misma línea.

Si se desea que un texto aparezca en una línea otro en la linea siguiente, será necesario enviar a cout un símbolo especial denominado std::endl(abreviatura de end of line, “final de línea”)

Programas ejemplos: Los guardamos en una carpeta que se llame **ejemplosc++**
suma.cpp (2.1)
resto.cpp ((2.1)
variostextos.cpp(2.2)
variaslíneas.cpp(2.3)

prog4.cpp

Crea un programa que calcule la diferencia entre 102 y 37 y pruébalo.

prog5.cpp

Crea un programa que calcule el producto de 84 y -23 y pruébalo.

6. Pedir datos al usuario

Un programa en el que los datos están prefijados no tiene mucho sentido. Lo habitual es que esos datos sean introducidos por el usuario o se lean desde un fichero.

Aprenderemos a leer datos. Para ello se emplea la orden **std::cin >>,**y será necesario disponer de un lugar donde guardarlo, un espacio de memoria al que se le dará un nombre, es lo que se conoce como variable. Al principio del programa se deberán declarar las variables que éste va a emplear, detallando el tipo de datos que se van a guardar, por ejemplo, **int,** para indicar que será un número entero, sin cifras decimales y el nombre de esa variable. Si las variables son del mismo tipo se pueden declarar en la misma línea.

int x,y,s;

Programas ejemplo:
triple.cpp (se pide u número a un usuario y cuando éste se introduce, calcula su triple)2.4
sumanumeros.cpp(se piden dos números y se calcula la suma)2.4

prog6.cpp

Realiza un programa que pida dos números enteros y calcule la división del primero entre el segundo y pruébalo.

6.1. Evitar escribir std::

En un programa muy extenso puede resultar engorroso escribir **std::** delantes de cada **cout** y de cada **cin**, por lo que existe una sencilla alternativa, añadir **using namespace std**, al principio del programa.

6.2. Números con decimales

Cuando se realizan operaciones con números enteros en C++, el resultado también es un número entero, pero esto no es apropiado a veces, por ejemplo, al dividir 5 entre 2, el resultado sería 2, en vez de 2,5.
Lo que se debe hacer es usar otro tipo de datos, unos que si permitan guardar números reales. Para ello hay que sustituir **int** por **float** ó double.

Programas ejemplo:
coutfloat.cpp(2.6)

7. Funciones matemáticas

Las principales funciones matemáticas incorporadas en C++ son:

- Raiz cuadrada**: sqrt(x)**
- x elevado a y : **pow(x,y)**
- Coseno: **cos(x)** (en las funciones trigonométricas el ángulo es en radianes)
- Seno: **sin(x)**
- Tangente: **tan(x)**
- Exponencial de x:(e elevado a x): **exp(x)**
- Logaritmo neperiano : **log(x)**
- Logaritmo en base 10: **log10(x)**
  Para usarlas hay que incluir el código **#include <cmath>**

Programas ejemplo
mate.cpp (2.7)

+prog7.cpp
Realiza un programa que calcule la raíz cuadrada del número que elija el usuario y pruébalo.

+prog8.cpp
Realiza un programa calcule la raíz cúbica de un número introducido por el usuario y pruébalo. Eleva el número a 1/3.

+prog9.cpp
Realiza un programa que pida cinco números reales y calcule su media y pruébalo.

+prog10.cpp
Realiza un programa que calcule el seno de un ángulo introducido en grados y pruébalo.

+prog11.cpp
Si se ingresan E euros en un banco con un interés R (en tanto por 1) durante N periodos de tiempo, el dinero que se obtendrá finalmente vendrá dado por $E*(1+R)^N$. Crea un programa que calcule el interés para los datos introducidos por el usuario.

7. Toma de decisiones

7.1. If

Para comprobar si se cumple una determinada condición e indicar qué pasos se deben dar en dicho caso se emplea un if que tiene la forma siguiente:

if (condición)
{ sentencias que se cumplen si la condición es cierta);
}

En las condiciones se suelen utilizar distintos operadores relacionales:< , >, <= ,>= , !=(distinto), ==(igual).Si la condición es cierta realizará las sentencias del if, si es falsa no ejecutará dichas sentencias y el programa seguirá su curso. Si sólo se pone una sentencia se pueden quitar las llaves.

Programas ejemplo
progif.cpp (3.2)

7.2. El caso contrario: else

También es habitual indicar lo que debe hacer el programa si no se cumple una determinada condición, para lo cual se añade la cláusula else, cuya forma es:

if (condición)
{ sentencias que se cumplen si la condición es cierta;
}
else { sentencias que se cumplen si la condición es falsa;
}

Si sólo se pone una sentencia se pueden quitar las llaves.

Programas ejemplo
progifelse.cpp (3.4)

*prog12.cpp

Realiza un programa que pida un número entero al usuario e indique si es par o impar y pruébalo. (Nota: Para ello deberá comprobar si el resto de dividir dicho número entre 2 es cero.

*prog13.cpp

Realiza un programa que pida al usuario dos números enteros y diga cuál es mayor y pruébalo.

*prog14.cpp

Elabora un programa que pida dos números al usuario. Si el segundo no es cero, mostrará la división entre ambos; en caso contrario, aparecerá un mensaje de error.

*prog15.cpp

Elabora un programa que pida dos números al usuario, e indique si el primero es múltiplo del segundo.

7.3. Utilizar los operadores lógicos: Y, O y No

Las condiciones se pueden encadenar como si de oraciones subordinadas se tratara:

- Con una **Y** en caso de que dos condiciones se tengan que cumplir a la vez para obtener verdadero. Su símbolo es **&&**
- Con una **O** en caso de que de dos condiciones, sea suficiente con una condición que sea verdadera, para que el resultado de verdadero. Su símbolo es ||
- Con un **No**, sería verdadero cuando la condición sea falsa.Su símbolo es **!**

Programas ejemplo
logica.cpp

7.4. Utilizar el operador condicional ?

Existe otra forma de asignar un valor a una variable en función de si cumple o no una condición. Se trata del< operador condición> u <operador ternario>. Se escribe así:

Variable= condición ? ValorSisecumple:ValorSinosecumple

Y equivale a decir que la Variable **(**se llamará de una forma, x,y …) toma el primer valor si se cumple la condición y el segundo si no se cumple.

Programas ejemplo
interrogacion.cpp

**prog16.cpp

Elabora un programa que pida al usuario dos números enteros y diga:" Uno de los números es positivo", Los dos números son positivos", o bien "Ninguno de los dos números es positivo", según corresponda y pruébalo. **(Utilizar operadores lógicos)**

***prog17.cpp

Elabora un programa que pida tres números reales al usuario, e indique el valor numérico del mayor de ellos, no su posición ya que algún número puede estar repetido. Pruébalo.**(Utilizar operadores lógicos)**

***prog18.cpp

Diseña un programa, utilizando if, que pida al usuario un número del 1 al 10 y diga si es múltiplo de 3 o no.

7.5. Switch

Para analizar posibles valores de una misma variable, se puede utilizar una orden **switch**. La variable por analizar se indica entre paréntesis, y cada uno de los posibles valores se indica tras la palabra **case**, seguido por un símbolo de dos puntos. Las órdenes a ejecutar en cada caso deben terminar con **break**. También se puede indicar qué hacer si no se da ninguno de los casos preestablecidos usando un bloque **default** opcional. Su forma general es:

```
switch (n)
{
 case 1: cout  << "Uno";
    break;
 case 2: cout  << "Dos";
    break;
      case 7:
      case 8: cout  << "Notable";
    break;
default: cout << "Valor incorrecto";
}
```

Programas ejemplo

switch.cpp(3.7)

**prog19.cpp

Diseña un programa, utilizando switch, que pida al usuario un número del 1 al 10 y diga si es múltiplo de 3 ó no.

**Prog20.cpp

Diseña un programa, utilizando switch, que pida al usuario un número del 1 al 10 y escriba el nombre de la nota correspondiente.

**Prog21.cpp

Diseña un programa, utilizando switch, que pida al usuario un número del 1 al 12 y escriba el mes correspondiente.

8. Bucles

Con frecuencia, las condiciones no se deberán comprobar una sola vez, sino de forma repetida, al igual que instrucciones que se deban repetir. Para ello en C++ tenemos varias soluciones: **while, do-while y for.**

8.1. while

Si se desea que una sección de un programa se repita mientras que se cumpla una cierta condición, se deberá emplear **while**. Su forma general es:

```
 while ( condición)
{
Órdenes que se repiten si la condición es cierta
}
```

Se puede también usar un while para crear un contador, es decir, una variable que aumente de un valor a otro para descubrir cuantas veces ha ocurrido algo. Su forma general sería:

```
  int n= valor;
```

while (condición)
{
Órdenes que se repiten si la condición es cierta
Actualización del contador, por ejemplo n=n+1
}

Las variables admiten cambios de valor, utilizando operaciones aritméticas, así si:

- Queremos sumar un valor a la variable podemos utilizar n = n+3 ó n += 3
- Queremos restar un valor a la variable podemos utilizar n = n-9 ó n -= 9
- Queremos multiplicar por un valor a la variable podemos utilizar n = n*3 ó n *= 3
- Queremos dividir por un valor a la variable podemos utilizar n = n/3 ó n /= 3

Programas ejemplo
while.cpp(4.1)
contador.cpp(4.3) Hacer en la pizarra el desarrollo del bucle.

8.1. do-while

Una alternativa a **while** es comprobar la condición tras dar una primera pasada. Para ello se emplea la construcción **do-while**, que tiene esta forma general:

do
{
Órdenes que se repetirán si se cumple la condición del while
}
while(condición)

Programas ejemplo
dowhile.cpp (4.2)

Prog22a.cpp

Diseña un programa, utilizando do-while, que pida al usuario su contraseña, tantas veces como sea necesario, hasta que introduzca el número 7890.Cada vez que se equivoque informarle que la contraseña es incorrecta.

Prog22b.cpp

Diseña un programa, utilizando while, que pida al usuario su contraseña, tantas veces como sea necesario, hasta que introduzca el número 7890.Cada vez que se equivoque informarle que la contraseña es incorrecta.

Prog23.cpp

Diseña un programa, utilizando do-while, que escriba en pantalla de mayor a menor, los números pares del 26 al 10.Utiliza una construcción que reste 2 a 26 de forma sucesiva.

Prog24.cpp

Crea un programa que pida números positivos al usuario y calcule la suma de todos ellos. Utiliza do-while y dos variables, una llamada suma (cuyo valor inicial será cero) donde se guardarán las sumas parciales que se mostrarán por pantalla con un SUMA: resultado y otra llamada n donde se guardará el número que dé el usuario. Indica al usuario que dejará de sumar si introduce un cero ó un número negativo. Utiliza do-while.

Prog25.cpp
Elabora un programa que pida al usuario su código de usuario y su contraseña , y que no le permita finalizar hasta que introduzca el código de usuario 1024 y la contraseña 7890. Utiliza do-while.

Prog26.cpp
Elabora un programa que dé al usuario la oportunidad de adivinar un número del 1 al 100, en un máximo de 6 intentos. Encada intento, el programa deberá avisar al usuario de si se ha pasado o se ha quedado corto.
Ayuda:

- Utiliza tres variables, una para el número que da el usuario, otra que gurade el número que elijas, que debe averiguar y otra que guarde los intentos.
- Utiliza un do-while.
- Utiliza if, else cuando el usuario haya agotado los intentos ó haya averiguado el número.

******8.2. for**

Cuando se desea crear un contador, existe otra orden que agrupa los tres pasos (valor inicial, incremento y comprobación de finalización)y que, por ello, puede resultar más legible. Se trata de la orden **for**. Su forma general es:

```
int contador;
 for( inicialización del contador; condición; actualización del contador)
 {
 órdenes que se ejecutan si se cumple la condición
 }
```

Inicialización de la variable: El contador se suele iniciar en cero(**n=0**). Si queremos declarar la variable dentro del for y que se destruya automáticamente cuando termina el for tendríamos que escribir **int n=0.**
Condición: La condición suele ser una comparación, como por ejemplo: n<=10, se debe cumplir para que se entre al bucle y se realicen las órdenes.
Actualización del contador: n=n+1 ó también se pone n++, si queremos incrementar el contador. Si lo queremos disminuir: n=n-1 ó n--.

Programas ejemplo
for.cpp(4.5)
fordeclarardentro.cpp(4.3)

Prog27.cpp
Crea un programa que con un for escriba los múltiplos de 3 menores de 100.

Prog28.cpp
Crea un programa que con un for escriba los múltiplos de 3 menores de 100 y los sume mostrando el resultado final por pantalla.

Prog29.cpp
Crea un programa que con un for y un if, que pida al usuario un número y saque sus divisores (recuerda que si un número se divide entre su divisor da cero).