Clase #4

El lenguaje C++

En esta clase introduciremos el lenguaje de programación C++. Comenzaremos por dar una visión general del lenguaje y después trataremos de forma práctica todos los conceptos estudiados en el bloque anterior, viendo como se implementan en el C++.

CONCEPTOS BÁSICOS

Comenzaremos estudiando el soporte del C++ a la programación imperativa, es decir, la forma de definir y utilizar los tipos de datos, las variables, las operaciones aritméticas, las estructuras de control y las funciones. Es interesante remarcar que toda esta parte está heredada del C, por lo que también sirve de introducción a este lenguaje.

Estructura de los programas

El mínimo programa de C++ es:

```
main() { }
```

Lo único que hemos hecho es definir una función (`main`) que no tiene argumentos y no hace nada. Las llaves { } delimitan un bloque en C++, en este caso el cuerpo de la función `main`. Todos los programas deben tener una función `main()` que es la que se ejecuta al comenzar el programa.

Un programa será una secuencia de líneas que contendrán sentencias, directivas de compilación y comentarios.

Las sentencias simples se separan por punto y coma y las compuestas se agrupan en bloques mediante llaves.

Las directivas serán instrucciones que le daremos al compilador para indicarle que realice alguna operación antes de compilar nuestro programa, las directivas comienzan con el símbolo `#` y no llevan punto y coma.

Los comentarios se introducirán en el programa separados por `/*` y `*/` o comenzándolos con `//`. Los comentarios entre `/*` y `*/` pueden tener la longitud que queramos, pero no se anidan, es decir, si escribimos `/* hola /* amigo */ mío */`, el compilador interpretará que el comentario termina antes de mío, y dará un error. Los comentarios que comienzan por `//` sólo son válidos hasta el final de la línea en la que aparecen.

Un programa simple que muestra todo lo que hemos visto puede ser el siguiente:

```
/*
Este es un programa mínimo en C++, lo único que hace es
escribir una frase en la pantalla
*/

#include <iostream>
 using namespace std;

int main()
{
      cout << "Hola mundo\n"; // imprime en la pantalla la
      frase "hola mundo"
}
```

La primera parte separada entre `/*` y `*/` es un comentario. Es recomendable que se comenten los programas, explicando que es lo que estamos haciendo en cada caso, para que cuando se lean sean más comprensibles.

La línea que empieza por `#` es una directiva. En este caso indica que se incluya el fichero "iostream.h", que contiene las definiciones para entrada/salida de datos en C++.

En la declaración de `main()` hemos incluido la palabra `int`, que indica que la función devuelve un entero. Este valor se le entrega al sistema operativo al terminar el programa. Si no se devuelve ningún valor el sistema recibe un valor aleatorio.

La sentencia separada ente llaves indica que se escriba la frase "Hola mundo". El operador `<<` ("poner en") escribe el segundo argumento en el primero. En este caso la cadena `"Hola mundo\n"` se escribe en la salida estándar (`cout`). El carácter `\` seguido de otro carácter indica un solo carácter especial, en este caso el salto de línea (`\n`).

Veremos el tema de la entrada salida estándar más adelante. Hay que indicar que las operaciones de E/S se gestionan de forma diferente en C y C++, mientras que el C proporciona una serie de funciones (declaradas en el fichero "stdio.h"), el C++ utiliza el concepto de *stream*, que se refiere al flujo de la información (tenemos un flujo de entrada que proviene de `cin` y uno de salida que se dirige a `cout)` que se maneja mediante operadores de E/S.

Por último, hay que señalar que debemos seguir ciertas reglas al nombrar tipos de datos, variables, funciones, etc. Los identificadores válidos del C++ son los formados a partir de los caracteres del alfabeto (el inglés, no podemos usar ni la ñ ni palabras acentuadas), los dígitos (0..9) y el subrayado (_), la única restricción es que no podemos comenzar un identificador con un dígito (es así porque se podrían confundir con literales numéricos). Hay que señalar que el C++ distingue entre mayúsculas y minúsculas, por lo que `Hola` y `hola` representan dos cosas diferentes. Hay que evitar el uso de identificadores que sólo difieran en letras mayúsculas y minúsculas, porque inducen a error.

Tipos de datos y operadores

Los tipos elementales definidos en C++ son:

`char`, `short`, `int`, `long`, que representan enteros de distintos tamaños (los caracteres son enteros de 8 bits)

`float`, `double` y `long double`, que representan números reales (en coma flotante).

Para declarar variables de un tipo determinado escribimos el nombre del tipo seguido del de la variable. Por ejemplo:

```
int i;

double d;

char c;
```

Sobre los tipos elementales se pueden emplear los siguientes operadores aritméticos:

```
+     (más, como signo o como operación suma)
-     (menos, como signo o como operación resta)
*     (multiplicación)
/     (división)
%     (resto)
```

Y los siguientes operadores relacionales:

```
==    (igual)
!=    (distinto)

<     (menor que)
>(mayor que)

<=    (menor o igual que)

>=    (mayor o igual que)
```

El operador de asignación se representa por `=`.

En la bibliografía del C++ se suelen considerar como tipos *derivados* los construidos mediante la aplicación de un operador a un tipo elemental o compuesto en su declaración. Estos operadores son:

```
*     Puntero
&     Referencia
[]    Vector (Array)

()    Función
```

Los tipos compuestos son las estructuras (`struct`), las uniones (`unión`) y las clases (`class`).

Estructuras de control

Como estructuras de control el C++ incluye las siguientes construcciones:

***condicionales**:*

`if`	instrucción de selección simple
`switch`	instrucción de selección múltiple

bucles:

`do-while`	instrucción de iteración con condición final
`while`	instrucción de iteración con condición inicial
`for`	instrucción de iteración especial (similar a las de repetición con contador)

***de salto**:*

`break`	instrucción de ruptura de secuencia (sale del bloque de un bucle o instrucción condicional)
`continue`	instrucción de salto a la siguiente iteración (se emplea en bucles para saltar a la posición donde se comprueban las condiciones)
`goto`	instrucción de salto incondicional (salta a una etiqueta)
`return`	instrucción de retorno de un valor (se emplea en las funco

Ingresar dos números enteros y mostrar su suma por pantalla.

```
#include<iostream>
using namespace std;
int main()
  {
    int a,b,c;
    cout<<"Ingrese un numero\n";
    cin>>a;
    cout<<endl<<"Ingrese otro\n";
    cin>>b;
    c=a+b;
    cout<<endl<<"Suma="<<c;
  }
```