Práctica Nº 2: Estructura general de un programa en C/C++. Introducción a las funciones de Entrada y salida en C++ (cin y cout) sin formato.

Objetivos de la práctica:
- Presentar la estructura general de un programa en C/C++, incluyendo los elementos más básicos.
- Introducir al alumno en las funciones básicas de entrada y salida de C++ (`cin` y `cout`) sin formato.

El aspecto del C.

Un primer ejemplo de programa en C:

ejemplo1:

```
#include <iostream.h>
#include <stdlib.h>

/* Mi primer programa en C */

int main(void)
{
     cout << "Hola Mundo\n";
     system("pause");
}
```

Se pueden resaltar varias cosas del ejemplo:
Primero, que es fácil de entender y de imaginar que hará cuando se ejecute.
Segundo, se distinguen un conjunto de palabras que deben tener un significado especial para el compilador de C, como son *include*, *void*, *main* y *printf*. A este conjunto de palabras se le llaman palabras reservadas del C.
Tercero, se distinguen además un conjunto de símbolos extraños como # , /* , */ , < , > , { , } , ; , (,) .

Los elementos más básicos del C/C++

Dentro de cada programa, hasta en el más simple, se distinguen una serie de elementos que suelen estar siempre presentes.

Elemento	Propósito
#include <iostream.h>	Incluir el archivo de salida/entrada estándar.
main	Nombre de la función principal. Su presencia es obligatorio en cualquier programa C y la ejecución del programa comienza dentro de ella.
void	Tipo de dato general.
( )	Paréntesis que acompañan a las funciones, dentro se colocan su conjunto de argumentos
/* */	Símbolos de comienzo y final de comentario. Los comentarios son ignorados por el compilador y se utilizan para ayudar a clarificar el código del programa.
//	Símbolo de comentario en C++. Nos indica que desde ese punto y hasta final de línea existe un comentario.
;	El punto y coma marca el final de una sentencia (una orden).
{ }	Agrupan un bloque de sentencias pertenecientes a una función o a una sentencia de control

El C es un lenguaje estructurado

Cuando pretendemos resolver un problema, existen varias formas de hacerlo: bien se puede hacer de una manera organizada, o por el contrario, se puede resolver de cualquier manera. Organizar un programa con una estructura clara y modulada suele tener muchísimas ventajas, como por ejemplo, que sea más fácil de comprender, que sea más fácil de depurar y que sea más fácil de mantener y actualizar. Un programa no estructurado o poco estructurado suele implicar todo lo contrario, suele ser difícil de comprender y suele ser difícil de modificar cualquier parte del mismo.

El C es un lenguaje estructurado que se distingue por el uso de bloques. Un bloque no es más que un conjunto de sentencias relacionadas de forma lógica.

Tipos de bloques.

Los tres tipos básicos son:

El bloque secuencial: es el más simple, está formado por un conjunto de sentencia que se ejecutan una tras otra de forma secuencial.

El bloque repetitivo: está formado por un conjunto de sentencia que se ejecutan una o varias veces antes de continuar la ejecución de otros bloques.

El bloque selectivo: formado por un conjunto de sentencias que se ejecuta sólo en el caso de que se cumpla una condición. Dentro del bloque, se permite, además, definir un conjunto de sentencias alternativo que se ejecute en el caso de que no se hubiera cumplido la anterior condición.

Normalmente agrupamos uno o varios de estos bloques formando **módulos** independientes denominadas **funciones**, diseñados para realizar una tarea muy específica. La función es la unidad fundamental de un programa C y todo programa en C se estructura como una colección de ellas.

Cuestión de estilo:

Existen infinidad de criterios para mantener una programación bien estructurada, casi tantos como programadores existen, pero vamos a recomendar algunas reglas que pueden ayudar en las prácticas del laboratorio:

En cuanto a la cabecera del programa: Es conveniente que las primeras líneas de un programa se dediquen a un bloque de comentarios donde aparezca el nombre de los programadores, la fecha de la última actualización y un comentario sobre el objetivo del programa.

En cuanto a los bloques: Se debe dejar una separación de una o varias líneas en blanco entre bloques diferentes, así como poner un pequeño comentario al comienzo del bloque explicando el objetivo del mismo.

También es importantísimo sangrar, en columnas diferentes, las instrucciones de un bloque contenido entre llaves, para dejar claro el comienzo y el final del bloque.

En cuanto a las funciones: Las funciones se deben separar, unas de otras, por líneas en blanco y además se debe de poner antes de la cabecera, un comentario explicando la tarea que realiza así como el sentido de los parámetros y del valor que retorna.

***Estructura general de un programa C*:** La estructura general de un programa se debería adaptar más o menos a la siguiente estructura simplificada:

```
Includes de funciones de librería.
Definiciones de Constantes.
Declaraciones de variables GLOBALES.
Declaración de los prototipos de las funciones.
Cabecera de la Función 1
     Declaración de variables de la Función 1
```

```
     Bloque de sentencias de la Función 1
          ..................
          ..................
     Cabecera de la Función N
     Declaración de variables de la Función N

     Bloque de sentencias de la Función N
```

Por último, señalar que en todo programa C debe existir una función que llamaremos "main", y será la función principal, encargada de iniciar la ejecución del programa.
Que no se pueden anidar funciones en C. Y que las funciones pueden llamarse una a otras o así mismas.

Veamos un ejemplo sencillo de un programa en C bien estructurado:

```
/*  Programa realizado por Fulanito y por Menganito.
Última actualización: el 1 de Enero de 2001.
Objetivo: Este programa es un ejemplo simple e ilustrativo
de un programa bien estructurado en C.
El programa sirve para calcular el área de un circulo */

// Bloque de includes de las librerías externas
#include <iostream.h>
#include <stdlib.h>

/* Bloque de definición de Ctes. */
const float PI= 3.141592;

// Bloque de declaración de variables Globales
float y;

// Bloque de definición de prototipos de funciones
float area(float );

/* Ahora viene la definición del conjunto de funciones  que forman el
programa*/

/* Funcion que calcula el area de un circulo de radio x */
/* Parametros: x - radio del circulo.*/
/* Retorna:  el valor del área    */
float area (float x)                  /* cabecera de la función */
{
    // Bloque de declaración de variables locales
     float resultado;

     /* Bloque de instrucciones de la función area */
     resultado=PI*x*x;        /* guarda en resultado el producto de Pi
                              por x por x */
    return(resultado);        /* valor devuelto por la función */
}
```

```
/* Función principal */
int main (void)
{
     /* bloque de declaración de variables locales a la función main*/
    float z;

    /* Bloque secuencial de presentación  */
    cout << "Este programa es un ejemplo simple" <<
     " de lo que es un programa C bien estructurado";
    cout << endl << "Calcula el área de un circulo"<< endl;

     /* Bloque repetitivo de entrada de datos.
    Se repite hasta que introduzca  como radio un valor positivo */
    do
    {
        cout << "Introduce el valor del radio del circulo " << endl;
        cin >> z;
    } while (z<0);

    /*bloque secuencial para el
     calculo del área y presentación delresultado por pantalla */

     y=area(z);    /* Llamada a la función area */
    cout << "El área vale " << y << endl;

    system("pause");
    return 0;
}
```

Introducción a las funciones de entrada y salida estandar en C++

Un programa en C++ puede realizar operaciones de entrada y salida de varias formas distintas: con la vieja biblioteca de C, para la que hay que incorporar el fichero de cabeceras *''stdio.h''* (este tipo de funciones de entrada y salida lo veremos más adelante cuando expliquemos el concepto de puntero), y con la nueva biblioteca de C++, que requiere incorporar el fichero ''iostream.h''.

A continuación describiremos lo que se conoce como flujos. Un flujo de entrada no es más que la serie de entradas que alimentan un ordenador para que el programa las utilice, y un flujo de salida es la serie de salidas que el programa genera. En esta sección supondremos que las entradas provienen del teclado, y las salidas se envían a la pantalla de un terminal.

Salidas con cout:

`cout << " He comprado dulces";`	Muestra por pantalla la cadena escrita entre comillas
`cout << " He comprado " << num_dulces << "dulces ";`	Muestra por pantalla las cadenas que están entre comillas y el valor de la variable num_dulces
`cout << " Hola \n Hola \n";`	Muestra por pantalla: `Hola`

	Hola Realiza un salto de línea entre los dos Holas.
`cout << " Hola" << endl << "Hola" << endl;`	Muestra por pantalla: Hola Hola Realiza un salto de línea entre los dos holas
`cout << " Hola\tHola" << endl;`	Muestra por pantalla: Hola Hola Deja un espacio de tabulación entre los dos holas

En los ejemplos anteriores hemos usado los caracteres especiales '\n' y '\t', que se denominan **secuencias de escape.** La secuencia de escape '\n' le indica al ordenador que salte a una nueva línea en la salida, y la secuencia '\t' que salte a la siguiente tabulación horizontal.
También hemos visto en los ejemplos otra forma de enviar un salto de línea, la utilización de **endl.**

Entradas con cin:

`cin >> num_pasteles;`	Leemos un dato introducido por teclado y lo guardamos en la variable *'num_pasteles'*.