

PRÁCTICA 1: Introducción al entorno de trabajo Dev-C++.

Funciones.

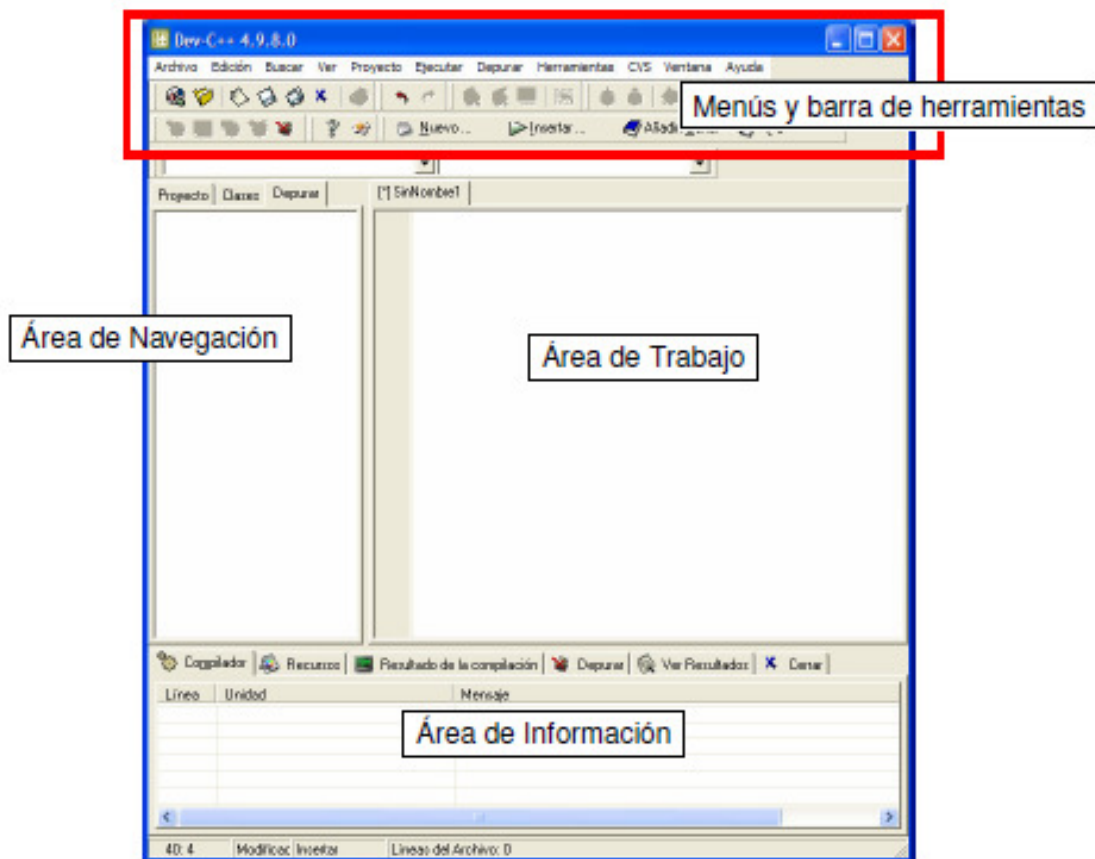
Objetivos de la práctica:

Objetivos de la práctica:

- Introducción al entorno de trabajo Dev C++: edición y compilación de programas en C++
- Uso básico del depurador
- Resolución de ejercicios básicos en lenguaje c++
- **Crear funciones (FUNDAMENTAL).**

1.- Descripción de las 4 zonas en el entorno:

- 1.1.- Menú y barra de herramientas: todas las opciones están en el menú, la barra de herramientas contiene accesos rápidos (atajos) a las funciones.
- 1.2.- Área de navegación (proyecto, clases, depurar).
- 1.3.- Área de trabajo (edición de los archivos abiertos)
- 1.4.- Área de información (compilador, recursos, resultado de la compilación, depurar, ver resultados)



2.- Trabajar con un programa en C++ de ejemplo:

Descarga los archivos del blog de la asignatura y guárdalos en una carpeta.

2.1.- Cargar el archivo, menú "Archivo", opción "Abrir proyecto o archivo": abrir el archivo "ej1practical.cpp".

2.2.- Estructura de programa en C++. Un programa consta de:

a) Inclusión de archivos de cabecera necesarios para la utilización de determinadas funciones propias del lenguaje. P.ej., la entrada y salida con <iostream>

b) Una función llamada main(), que especifica que el conjunto de órdenes del programa propiamente dicho. La estructura típica de esta función es la siguiente:

- Declaración de la información a utilizar en el programa (constantes y variables)
- Entrada de datos necesarios para resolver el problema
- Procesamiento de la información
- Salida de resultados

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

using namespace std;

int main ()
{
//DECLARACIÓN DE VARIABLES
int anyo_actual = 2010;
int anyo;
int edad;

//ENTRADA DE DATOS
cout << "Indica tu año de nacimiento: ";
cin >> anyo;

//PROCESAMIENTO DE INFORMACION
edad = anyo_actual - anyo;

//SALIDA DE RESULTADOS
cout << "Tienes " << edad << " años." << endl;
if (edad >= 18)
cout << ", deberías disimular esas arrugas !!!" << endl;
else
cout << ", eso no se lo cree nadie !!!" << endl;
getch();
return 0;
}
```

2.3.- Compilar el programa: Menú “Ejecutar”, opción “Compilar”. Lo que se está haciendo es comprobar

la corrección del código escrito y después traducir a código ejecutable. En el cuadro de diálogo de la compilación se muestran los siguientes aspectos: “Status” (fase del proceso en que se encuentra), “Errors” (número de errores detectados), “Warnings” (número de avisos generados). Un aviso indica, generalmente, un futuro error de funcionamiento.

Como el programa está bien escrito el proceso de compilación finaliza sin errores ni avisos, cerrar el diálogo. Sólo es preciso compilar el programa cuando se haya modificado alguna parte del código desde la última vez que se compiló. Comprobar mediante el explorador de archivos que la compilación crea un archivo ejecutable de manera “permanente”.

2.4.- Ejecutar (hacer funcionar) el programa: Menú “Ejecutar”, opción “Ejecutar”. Se abre una ventana donde se pone en funcionamiento el programa. Cuando el programa finaliza (en el ejemplo hace falta un “enter” para acabar) la ventana se cierra.

2.5.- La opción “Ejecutar”>> “Compilar y

Ejecutar” realiza los dos procesos anteriores en un único paso. Verificar que esto es así.

2.6.- Verificación de lo que significa compilar el programa y que el archivo que se ejecuta no es el texto que vemos en pantalla sino el resultado de la compilación.

a) Proceder a realizar una modificación en el código fuente. Por ejemplo, comentar las líneas que solicitan y leen el año (líneas 13 y 14).

b) A continuación ir directamente a ejecutar el programa (SIN COMPILAR). Se comprueba que a pesar de haber eliminado esas líneas el programa sigue pidiendo el año, puesto que se está ejecutando el resultado de la última compilación.

c) Ahora, compilar y a continuación ejecutar para comprobar que los cambios se han hecho efectivos.

d) Eliminar los comentarios de las líneas 13 y 14 y volver a comprobar que el funcionamiento es el inicial.

2.7.- Comprensión de los mensajes de error que genera el compilador:

a) Cerrar el archivo actual (“Archivo” >> “Cerrar”)

b) Cargar el programa “ej2practica1.cpp”.

c) Compilar el programa. Ahora el diálogo de compilación no da opción a continuar, salta directamente al editor marcando la primera línea de error. Se debe aprender a reconocer esa marca y a recurrir a la zona de información (zona inferior del entorno) para identificar los mensajes que el compilador ha generado (pestañas “Compilador” y “Resultado de la compilación”). La pestaña “Compilador” permite saltar en la zona de edición a las líneas de error sin más que pinchar 2 veces encima del mensaje. La pestaña “Resultado de la compilación” da información más completa sobre cada uno de los errores.

El compilador indica el punto donde se da cuenta de que hay un error, lo cual no significa que el origen del error esté siempre en la línea que se indica. La causa de un error puede estar en alguna de las líneas anteriores a donde ha sido detectado. Hay que saber leer código e interpretar los errores.

d) Modificar el programa para corregir los errores y verificar el funcionamiento.

3.- Funciones (IMPORTANTE)

Cuando en clase hemos visto la librería `<math.h>` o `<cmath>`, hemos usado varias funciones que pertenecen a esa librería como por ejemplo la función `pow(a,b)` te devuelve el resultado de elevar a, b veces. Para usar dicha función `pow(a,b)` lo hacíamos de la siguiente manera:

```
int valor; //declaramos una variable entera en el caso de hacer la función con enteros
valor=pow(2,3); // en valor tenemos el resultado de elevar 2 al cubo.
```

Lo mismo ocurría con la función `sqrt(a)` que nos devuelve la raíz de a y se usaba de la siguiente manera:

```
double raiz; //declaramos una variable real
raiz=sqrt(25); // en la variable raiz tendremos el resultado de realizar la raíz cuadrada de 25.
```

Nosotros podemos también realizar nuestras propias librerías con nuestras propias funciones. Vamos a ir viendo como se hace y explicandolo.

Vamos a partir de un programa que suma dos números:

```
#include <iostream>
#include <conio.h>
using namespace std;
int main(){
    double a,b; //son los numeros que vamos a sumar
    cout<<"Introduzca un dato ";
    cin>>a;
    cout<<"Introduzca otro dato ";
    cin>>b;
    cout<<"La suma es "<<a+b;
    getch();
    return 0;
}
```

Vemos este programa sencillo y la verdad es que no nos sirve para mucho. Si observáis el programa del menú, si pulsamos 1 nos hace la suma, y prácticamente tenemos que copiar el ejercicio entero. Lo que nos interesaría construir es una **FUNCION** que la pudiéramos usar cuando querremos sumar dos números.

¿Cómo lo hacemos?

Tenemos que construir una función del siguiente tipo: `suma(a,b)`, esta función nos tiene que hacer la suma de dos números por ejemplo cuando pongamos:

```
int a; //declaramos una variable
a= suma(8,10); //en la variable a tenemos que tener el valor 18.
```

Vamos a construirlo y lo explicamos:

Después de las librerías y antes del `int main(){` es la zona donde se programan estas funciones.

```
#include <iostream>
#include <conio.h>
using namespace std;

//función que nos suma dos números
double suma(double a,double b){ //esta es la cabecera de la función
    return a+b; //este es el valor que devuelve la función
```

```

}
int main(){ //comenzamos el programa principal
    cout << suma(100,100); //usamos nuestra función suma
} //fin del programa principal

```

Vamos a explicarlo todo con detalle:

Cabecera de la función **double suma(double a, double b)**

El primer tipo de dato nos indica que el dato que nos va a devolver la función es de tipo double.

suma es el identificador de la función. Que es el nombre que se va a usar para su llamada.

Lo que hay dentro de los paréntesis son los **argumentos** como dijimos al principio suma tendría dos argumentos por que diseñamos nuestra función como la suma de dos números tal que así suma(a,b). podríamos usar los argumentos que queramos. A los argumentos hay que definirles el tipo.

En este caso nosotros hemos llamado a los argumento a y b de tipo real. El **return a+b es lo que nos devuelve la función**. Queremos que la función nos devuelva la suma de los argumentos, los cuales hemos llamado a y b.

Cuando ya tenemos la función programada antes del main, ya la tenemos lista para usar en el programa principal tal y como lo veis. Nuestro programa inicial muestra por pantalla la suma de 100+100, por tanto 200. Vamos hacer un program que devuelva la suma de dos números usando nuestra nueva función.

```

#include <iostream>
#include <conio.h>
using namespace std;

```

```

//función que nos suma dos números
double suma(double a, double b){ //esta es la cabecera de la función
    return a+b; //este es el valor que devuelve la función
}

```

```

int main(){ //comenzamos el programa principal
    double sum1, sum2;
    cout << "introduce un valor";
    cin >> sum1;
    cout << "Introduce un valor ";
    cin >> sum2;
    cout << "la suma es " << suma(sum1, sum2);
}

```

```

} //fin del programa principal

```

Preguntar al profesor para resolver las dudas y si no ha quedado claro.

4.-Ejercicios: (Apoyarse en los apuntes de teoría)

Los ejercicios se entregaran mediante correo electrónico a la dirección javi@agustinosgranada.es antes del domingo 9 de mayo a las 23:59 h. Sólo se entregarán los archivos con extensión cpp.

1.-Realizar un programa que dado dos números nos diga si el primero es divisible por el segundo.

2.- Realizar un programa que dado un número nos diga si es par o impar.

3.- Realizar un programa que nos muestre los factores primos de un número.

4.- Realizar un programa que nos calcule la media de tres notas introducidas por pantalla. El programa nos tiene que mostrar también si el alumno tiene INSUFICIENTE, SUFICIENTE, BIEN, NOTABLE, SOBRESALIENTE y MATRICULA. Siendo:

INSUFICIENTE $0 > \text{nota} < 4,5$
SUFICIENTE $4,6 > \text{nota} < 5,99$
BIEN $6 > \text{nota} < 6,99$
NOTABLE $7 > \text{nota} < 8,5$
SOBRESALIENTE $8,6 > \text{nota} < 9,99$
MATRICULA $\text{nota} = 10$;

Si el usuario introduce alguna de las tres notas mayor que 10 y menor que 0 el programa debe de enviar por pantalla un mensaje de error y acabar.

5.- Crear un programa que de cambio. Para ello tendremos un valor en céntimos introducido por el usuario y un menu de precios de la siguiente manera.

- 1.- Fanta 1,25 €**
- 2.- CocaCola 2,25 €**
- 3.- Acuaris 4,50 €**
- 4.- Cerveza 0,75 €**

El programa al pulsar una opción nos deberá de decir el cambio en las mínimas monedas existentes posibles. Y en el caso de que el valor introducido fuese menor que el precio, nos tiene que enviar un mensaje advirtiéndonos. Introduce las mejoras que creas conveniente.

6.- Crear un programa que introduciendo dos números, muestre un menu de la siguiente manera:

- 1.- Suma**
- 2.- Resta**
- 3.- Multiplicación**
- 4.- División**

El programa deberá realizar la operación elegida. Debes obligatoriamente usar **FUNCIONES** para los distintos apartados. (Consultar el punto 3 del guión y al profesor). Introducir las mejoras que se deseé.