Práctica 1: Empezando a programar en C++

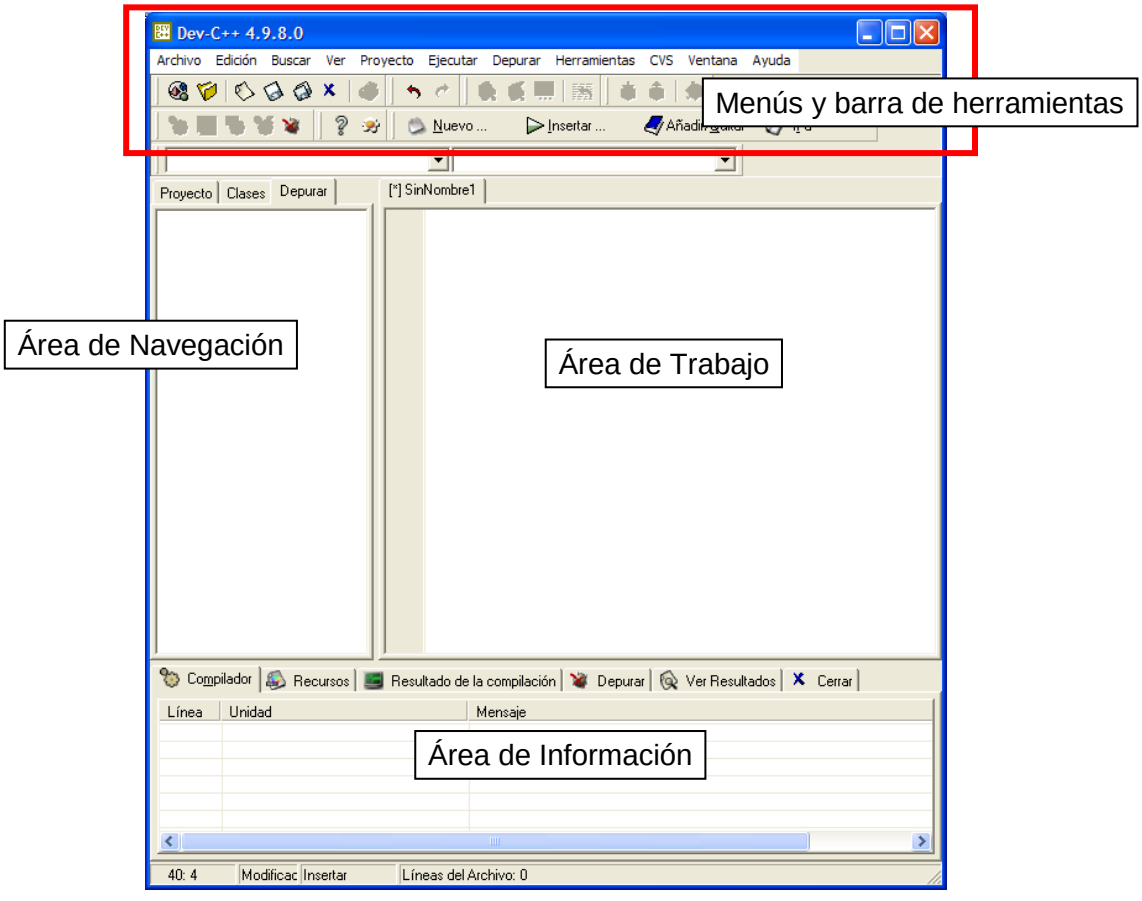
Objetivos

- Conocer el entorno de desarrollo de software DevC++. Edición y compilación de programas. Uso básico del depurador.
- Describir la estructura general de sencillos programas en C++.
- Conocer los operadores de Entrada/Salida de C++.
- Aprender a utilizar apropiadamente los tipos de datos.
- Especificar mediante pseudocódigo o diagrama de flujos un algoritmo.
- Documentar un programa usando Doxygen.

PARTE 1: Conceptos Básicos del Entorno de Desarrollo DevC++

Descripción del entorno:

- **Menú y barra de herramientas.** Todas las opciones están en el menú. La barra de herramientas contiene accesos rápidos (atajos) a las funciones.
- **Área de navegación** (proyecto, clases, depurar).
- **Área de trabajo**: área de edición de los archivos abiertos.
- **Área de información** (compilador, errores resultado de la compilación, depurar).



Trabajar con un programa en DevC++:

Descargar los archivos de ejemplo en un directorio de trabajo. Cargar el archivo `ejemplo1.cpp`.

Estructura de programa en C++.

Un programa en C++ consta de:

a) **Inclusión de archivos** de cabecera necesarios para la utilización de funciones propias del lenguaje. Por ejemplo, para poder leer desde la entrada y salida estándar incluimos <iostream>.

b) **Una función main()** que especifica el conjunto de órdenes del programa. La estructura típica es la siguiente:

- Declaración de las variables y constantes.
- Entrada de datos por teclado.
- Procesamiento de la información.
- Salida de resultados por pantalla.

Inclusión de archivos

Programa

```
#include <iostream>
#include <string>

using namespace std;

int main ()
{
    //DECLARACIÓN DE VARIABLES
    const int anyo_actual = 2021;
    string nombre;
    int anyo;
    int edad;

    //ENTRADA DE DATOS
    cout << "Indica tu nombre: ";
    getline(cin, nombre);
    cout << "Indica tu anyo de nacimiento: ";
    cin >> anyo;

    //PROCESAMIENTO DE INFORMACION
    edad = anyo_actual - anyo;


    //SALIDA DE RESULTADOS
    cout << "Tienes " << edad << " anyos." << endl;
    if (edad >= 18)
        cout << nombre << ", deberias disimular esas arrugas !!!" << endl;
    else
        cout << nombre << ", eso no se lo cree nadie !!!" << endl;


    return 0;
}
```

Declaración de información

Entrada de datos

Procesamiento de información

Salida de resultados

Compilar un programa: Menú `Ejecutar>>Compilar`. Se realiza la comprobación de que el código es correcto y después se traduce a código ejecutable. En el cuadro de dialogo de la compilación se muestra: "Status" (fase del proceso en que se encuentra), "Errors" (número de errores detectados), "Warnings" (número de avisos generados). Un aviso indica, generalmente, un posible error de funcionamiento cuando se ejecute. 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 ha creado un archivo ejecutable `ejemplo1.exe`.

Ejecutar un programa: Menú `Ejecutar>>Ejecutar`. Se abre una ventana desde donde se introducen datos y se muestran resultados. Cuando el programa finaliza (hace falta apretar una tecla) la ventana se cierra. La opción `Ejecutar>>Compilar y Ejecutar` realiza los dos procesos.

Interpretación de los mensajes de error que genera el compilador. Abrir el fichero `ejemplo2.cpp` y compilarlo. Este ejemplo contiene dos errores. El diálogo de compilación salta directamente al editor marcando la primera línea de error. En la zona inferior de información se muestran los mensajes. Se puede saltar a las líneas de error pinchando dos veces encima del mensaje. Se da información detallada sobre cada uno de los errores, **incluso sugerencias para corregirlo**. Se indica la línea y columna donde se ha producido. Cuidado, porque no significa que el origen esté siempre en la línea y columna que se indica. La causa puede estar en varias líneas anteriores.

Corregir los errores y verificar el correcto funcionamiento.

Para depurar el programa es mejor instalar la versión **`Dev-C++ 5.10 (Orwell).`**
Es la versión que se encuentra instalada en los ordenadores de los laboratorios y la que usareís en la asignatura de Programación.

https://sourceforge.net/projects/orwelldevcpp/files/Setup%20Releases/Dev-Cpp%205.10%20TDM-GCC%204.8.1%20Setup.exe/download

Depurar un programa: Mediante el depurador se puede analizar paso a paso lo que hace el programa, lo que permite **sistematizar la búsqueda de errores**. El depurador es de una gran ayuda para el programador.
Abrir `ejemplo3.cpp`. Compilar y ejecutar el programa para comprobar su funcionamiento. Para utilizar el depurador es necesario haber compilado previamente con la opción de generación de código de depuración activado. En `Herramientas>>Opciones del compilador>>Configuración>>Linker` comprobar que está activada la opción `Generar información de Debug`. Pinchar en la pestaña inferior "Depurar".

Fijar algunos puntos de ruptura (puntos en los que se interrumpe la ejecución y podemos visualizar el contenido de las variables en ese momento). Por ejemplo, dos puntos de ruptura en las líneas 40 y 42. Se establecen pulsando sobre la barra a la izquierda del código (se

pueden insertar y eliminar) o desde el menú `Debug>>Toggle BreakPoint`. Pulsar en el menú `Depurar`. Cuando el programa pare en un punto de ruptura, visualizar el contenido de algunas de las variables. Para ello, añade un `watch` para: `a, b, c, discr, x1, x2`. Las variables tienen valores arbitrarios hasta que empieza su cálculo y se actualizan. `Next Step`, ejecuta instrucción a instrucción.

PARTE 2: Documentar un programa con la herramienta Doxygen

La herramienta Doxygen puedes encontrarla en la página:
`http://www.doxygen.nl/download.html`

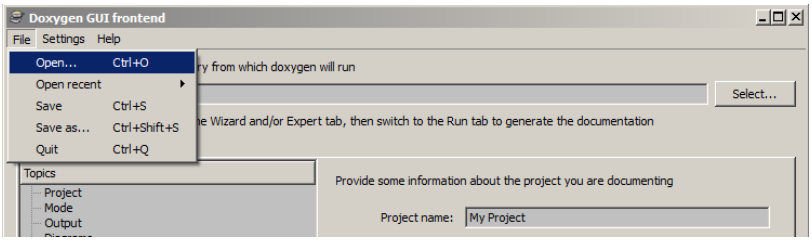
En los ordenadores de los laboratorios está instalada la distribución"**doxygen-1.8.7-setup.exe**". La versión actual es la **1.8.16** y no hay grandes diferencias entre esta versión y la 1.8.7. También existen diferentes distribuciones para otros sistemas operativos.

Al instalar el software se crea un nuevo grupo de programas en el botón de inicio '**doxygen**' en el que podemos encontrar diferentes elementos que permiten la generación automática de documentación a partir de programas fuente debidamente documentados. Utilizaremos '**doxywizard**'. En este curso nos limitaremos a cargar un fichero de configuración previamente diseñado, ajustar el nombre del proyecto de documentación que queremos generar y ejecutar el programa doxygen.

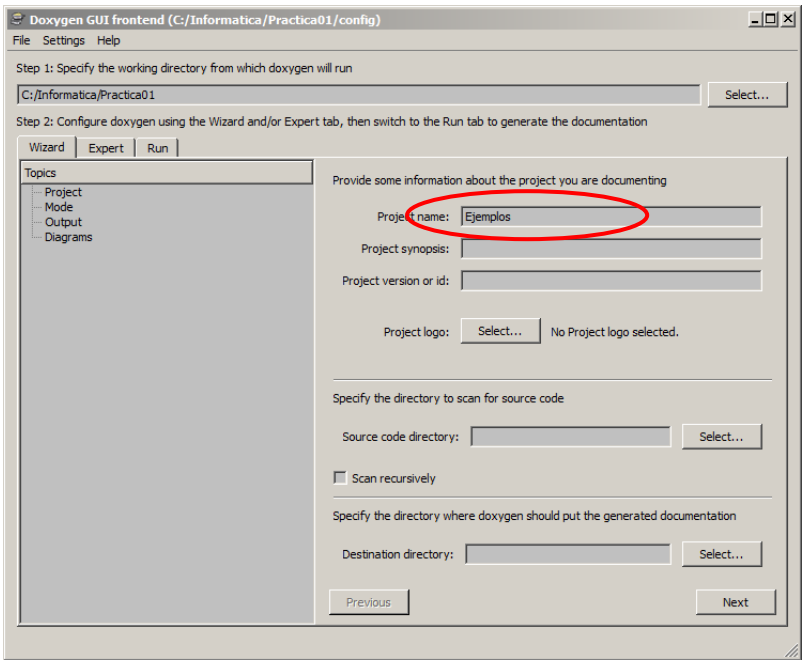
Fichero de configuración:
Descarga el fichero `config` que se encuentra en el Aula Virtual y guardarlo en el directorio de trabajo donde tienes el fichero ejercicio1.cpp de C++. Este fichero ya se proporciona comentado con Doxygen, con los comandos utilizados más habitualmente para generar la documentación.

En la guía de estilo de C++ proporcionada en la asignatura se encuentran los comandos básicos. Una lista más detallada se puede obtener en:
`https://www.doxygen.nl/manual/commands.html`

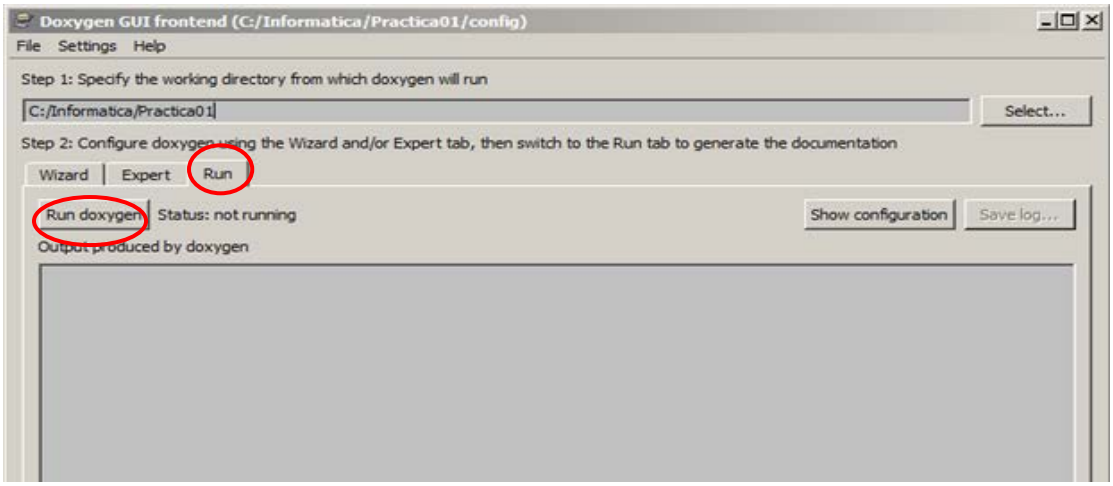
A continuación, abre el programa '**doxywizard**' y carga el fichero `config` copiar (menú "File", opción "Open...").



Una vez cargado, cambia el nombre del proyecto de documentación que vas a generar, que por defecto está en '*Ejemplos*' por el que creas conveniente (por ejemplo, '*Programas de ejemplo practica1*').



Generación de la documentación:
Selecciona la pestaña “Run” y dentro de esta pestaña el botón “Run doxygen”.



Si todos los comentarios son correctos, se mostrará el mensaje "*** `Doxygen has finished`" en la ventana "`Output produced by doxygen`". En caso contrario, se mostrará un mensaje de error informando del tipo de error y la línea en la que se ha producido.

Comprobación de la documentación generada:
En el directorio de trabajo se debe haber creado una nueva carpeta 'html'. Entra en la carpeta y pulsa dos veces sobre el fichero "index.html" para visualizar la documentación generada. Se habrá generado la documentación de todos los archivos fuentes existentes en el directorio de trabajo.

PARTE 3: Conceptos Básicos de C++

Tipos de datos simples en C++

Nombre del Tipo	Memoria utilizada	Rango	Precisión
bool	1 byte	{true, false}	No aplica
char	1 byte	-128…127	No aplica
unsigned char	1 byte	0…255	No aplica
short	2 bytes	-32768…32767	No aplica
int	4 bytes	-2147483648…2147483647	No aplica
long	4 bytes	-2147483648…2147483647	No aplica
unsigned short	2 bytes	0…65535	No aplica
unsigned int	4 bytes	0…4294967295	No aplica
unsigned long	4 bytes	0…4294967295	No aplica
float	4 bytes	10^{-38}…10^{38}	aprox. 7 dígitos
double	8 bytes	10^{-308}…10^{308}	aprox. 15 dígitos

Los valores de rango son orientativos. Pueden variar en función del compilador y de la arquitectura del computador (32-bits, 64-bits).

Enteros: `short, int, long, unsigned short, unsigned int, unsigned long`
Uso: para almacenar números sin decimales.
Ejemplos: número de personas, veces que se repite alguna operación.

De tipo carácter: `char, unsigned char`
Uso: para almacenar caracteres alfabéticos.
Ejemplos: letras y símbolos ortográficos.
Atención: en realidad lo que se guarda es un número (el código ASCII) asociado al carácter

Booleano: `bool`
Uso: para almacenar datos lógicos (verdadero o falso).
Ejemplos: el resultado de una comparación.
Atención: aunque sólo pueden tomar 2 valores (true o false) ocupan 1 byte entero. *Cualquier valor distinto de 0 es interpretado como true, y 0 es false.*

Reales: `float, double`
Uso: para almacenar números con decimales
Ejemplos: longitudes, pesos, velocidades, etc.

Operadores

Los operadores asociados a los tipos de datos simples se muestran en la tabla siguiente junto con su orden de precedencia (de mayor a menor):

Tipo de Operador	Orden de prioridad
Unarios	`!, -, + (signos)`
Multiplicativos	`*, /, %`
Aditivos	`+, -`
Inserción y Extracción de Flujo	`<<  >>`
Relacionales	`>, <, >=, <=`
Igualdad	`==, !=`
And	`&&`
Or	`||`
Asignación	`=`

Si se desea cambiar la precedencia y forzar operaciones en un cierto orden se utilizan paréntesis. Se debe tener en cuenta la **asociatividad a izquierdas** (o a derechas) en el caso de que tengamos el mismo operador dos veces o dos operadores con igual prioridad. La mayoría de los operadores de la tabla anterior son asociativos a izquierdas. Excepciones:

- El operador asignación es asociativo a derechas y los operadores unarios también.
- Los operadores relacionales y de igualdad no son asociativos.

Los operadores se definen de forma distinta para cada tipo de dato. Por ejemplo, la división de dos enteros es un entero. Para evitar errores podemos recurrir a la conversión explicita de tipos poniendo delante del dato el tipo al cual queremos convertirlo. Por ejemplo: `float(9).`

Los operadores de entrada y salida estándar en C++

Para realizar operaciones de entrada y salida se necesita la biblioteca `#include <iostream>`. Los operadores de extracción >> y de inserción << de flujo se utilizan para leer desde el teclado (entrada estándar) y mostrar en pantalla (salida estándar). Algunos ejemplos:

`cin >>  numAlumnos;`	Leemos un dato introducido por teclado y lo guardamos en la variable `numAlumnos`.
`cout << "He comprado dulces";`	Muestra por pantalla la cadena entre comillas: `He comprado dulces`
`cout << "En clase hay " << numAlumnos << " estudiantes matriculados";`	Muestra por pantalla las cadenas que están entre comillas y el valor de la variable `numAlumnos`. Si su valor es 60, mostrará: `En clase hay 60 estudiantes matriculados`
`cout << "Hola" << endl;`	Añade un salto de línea al final del mensaje: `Hola`
`cin.get (c1)`	Leemos un carácter introducido por teclado y lo guardamos en la variable `c1`.

BLOQUE DE EJERCICIOS

Ejercicio 1 (mi primer programa, ejercicio1.cpp). El programa ejercicio1.cpp calcula la superficie de un cubo. El valor del lado (en metros) se introduce como dato de entrada. El programa calcula la superficie y se muestra por pantalla. Modifícalo para que calcule también el volumen y lo muestre por pantalla.

Ejercicio 2 (corrigiendo errores, ejercicio2.cpp). Dado el siguiente código, complétalo añadiendo los `includes` necesarios y la función `main`. Compílalo, ejecútalo y comenta lo que ocurre. Corrige los posibles errores para obtener el resultado correcto. **Para ello, usa el depurador**. **Añade varios puntos de ruptura y visualiza el contenido de todas las variables que aparecen en el programa**.
¿De qué tipo de error se trata (sintáctico, lógico, léxico o de ejecución)?

Documenta el programa usando los comandos de Doxygen y genera la documentación.

```
/*completar */
 float c1 = 0.1000000003;
 float c2 = 0.1000000005;
 float dif;

 dif = c2 - c1;
 cout << "La diferencia es" << dif << endl;

 /*completar */
```

Ejercicio 3 (ejercicio3.cpp). Escribe un programa en C++ para el control de un robot de cocina. Se introduce por el teclado el número de comensales. El programa debe mostrar por pantalla cuántos **litros** de caldo y el tiempo en minutos de cocción para preparar una sopa. Se necesitan 300 ml de caldo por cada comensal y 1 minuto por cada 100 ml de caldo. **Indica el pseudocódigo (no olvides la cabecera) o el diagrama de flujo.**

Por ejemplo: `4 comensales, caldo 1.2 litros, tiempo de cocción 12 minutos.`

Documenta el programa usando los comandos de Doxygen y genera la documentación.

Ejercicio 4 (ejercicio4.cpp). Una institución benéfica ha recibido tres donaciones: una en libras esterlinas, otra en dólares y otra en euros. El dinero recolectado será repartido en tres partidas: 60% para un centro de salud, 35% para un comedor de niños y el resto para gastos administrativos. Realiza un programa en C++ que pida la cantidad de cada una de las donaciones por teclado y muestre por pantalla la donación **(en euros y céntimos de euros)**

para cada una de las partidas. **Indica el pseudocódigo (no olvides la cabecera) o el diagrama de flujo.**
Ayuda: `funciones floor() y ceil() de la librería <cmath>.`

Por ejemplo:

```
Cantidad total recibida mediante donaciones: 15638 euros, 59 céntimos.
Desglose por partidas:
----------------------------------------
Centro de salud: 9383 euros y 15 céntimos
Comedor: 5473 euros y 51 céntimos
Gastos Administrativos: 781 euros y 93 céntimos
----------------------------------------
```

Documenta el programa usando los comandos de Doxygen y genera la documentación.

IMPORTANTE:

- Todos los programas deben seguir la Guía de Estilo de C++ e incluir los comandos de Doxygen (pdf en Aula Virtual) para generar la documentación.
- Subir a Aula Virtual todos los archivos .cpp (de uno en uno) sin comprimir (enunciado y adicionales). Sólo los sube un miembro de la pareja.
- Fecha de entrega: durante la sesión de vuestro grupo de lab.
- Realizar una foto del pseudocódigo y/o el diagrama de flujo. Subir el archivo junto con los ficheros .cpp.