Escuela Politécnica Superior de Elche

FUNDAMENTOS DE INFORMÁTICA
1º Ingeniería Industrial

PRÁCTICA 1:
Introducción al entorno de programación Dev-C++

CURSO 2007-2008

División de Ingeniería de Sistemas y Automática

1. Objetivos

Los objetivos de esta práctica son los siguientes:

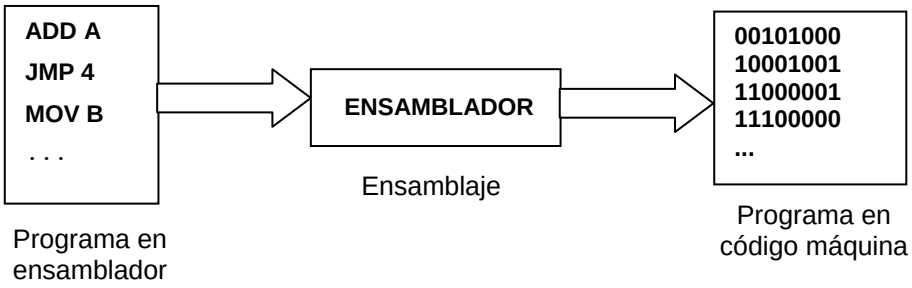
- Comprender el proceso de escritura, compilación, enlace y ejecución de un programa. Conocer qué es un compilador.
- Introducir el entorno de programación que se va a utilizar en las prácticas de la asignatura: Dev-C++. Se aprenderá a crear un proyecto y los programas fuente dentro del mismo y a compilar, enlazar y ejecutar el proyecto para obtener un programa ejecutable.
- Aprender la estructura básica de un programa escrito en el lenguaje C. Utilizar los conceptos elementales de programación.

2. Algunos conceptos previos.

Los programas de un computador se realizan utilizando lenguajes de programación. Un lenguaje de programación es un conjunto de reglas sintácticas y semánticas que sirven para escribir algoritmos para resolver un problema concreto en un computador. Los lenguajes de programación pueden clasificarse en distintos tipos, dependiendo de su cercanía al lenguaje directamente "entendible" por la máquina. Estos tipos son los siguientes:

- *Lenguaje máquina*. Es el lenguaje usado directamente por el computador y compuesto de instrucciones codificadas en binario. En este lenguaje una instrucción es una cadena de unos y ceros que permite a la unidad de control reconocer una operación elemental y ejecutarla. Viene dado por el fabricante del computador.

- *Lenguaje ensamblador*. Es un lenguaje de bajo nivel (cercano a la máquina) en el que se utilizan nemotécnicos para representar las instrucciones del lenguaje máquina para un computador concreto. Un *ensamblador* es un programa que lee, como datos de entrada, un programa escrito en lenguaje ensamblador y produce, como resultado, un programa en lenguaje máquina. En la siguiente figura se muestra este proceso de ensamblaje.

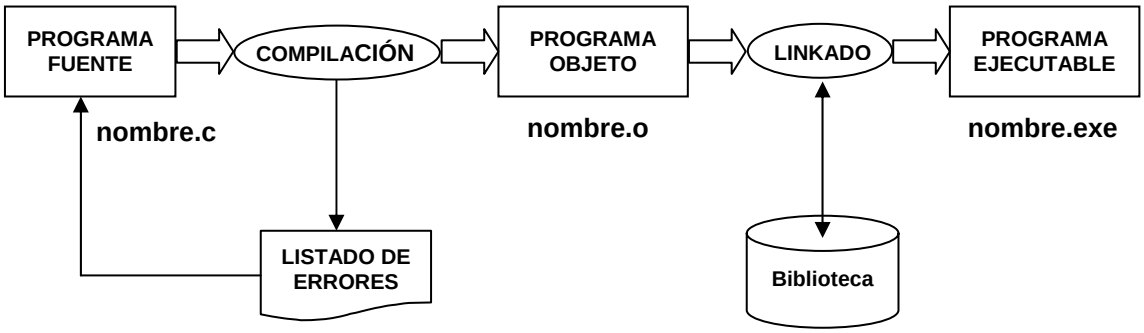


El desarrollo de los ensambladores supuso un paso importante para programar de manera intuitiva un computador, pero los programadores estaban aún forzados a pensar en términos de instrucciones máquina individuales. El lenguaje máquina y el lenguaje ensamblador se denominan *lenguajes de bajo nivel*, porque son dependientes de la arquitectura del computador que los soporta.

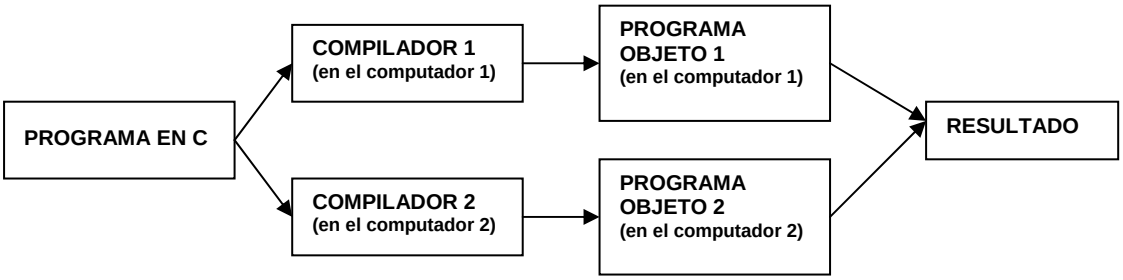
- *Lenguajes de alto nivel*. Son lenguajes de programación más cercanos a los naturales (tales como el inglés), y no tan dependientes de las características de los computadores. Algunos ejemplos de estos lenguajes son C, C++ (que los estudiaremos durante este curso), Pascal, Basic, COBOL, FORTRAN, etc. Los programas escritos en estos lenguajes se traducen a instrucciones en lenguaje máquina mediante programas especiales llamados *compiladores* e *intérpretes*.

  Un *intérprete* es un programa que analiza y ejecuta un programa sentencia a sentencia, sin obtener una traducción completa del mismo en lenguaje máquina.

  Por el contrario, un *compilador* analiza el programa, comprobando su sintaxis e indicando los errores, si existen, y luego genera el programa en lenguaje máquina. Se denomina *programa fuente* al programa escrito en el lenguaje de alto nivel, y *programa objeto* al programa en lenguaje máquina generado por el compilador a partir del programa fuente. El programa objeto necesita otro proceso adicional al de compilación, que se denomina *enlazado* ("*linkado*"). En la figura siguiente se muestra un esquema del proceso de compilación y enlazado.



Una ventaja importante del uso de compiladores es que si se escribe un programa en un lenguaje de alto nivel, es posible ejecutarlo en cualquier computador que disponga del compilador adecuado.



En las prácticas de la asignatura Fundamentos de Informática vamos a utilizar un entorno de programación de libre distribución llamado Dev-C++, que incluye un compilador de C y C++, junto con herramientas que facilitan la tarea de programación, compilación, enlazado y ejecución. Dicho entorno

de programación se puede instalar accediendo a la página web http://www.bloodshed.net/devcpp.html.

3. Cómo iniciar el entorno de desarrollo de Dev-C++

Para lanzar el programa Dev-C++, hacer click en el elemento “Programas” del menú ‘Inicio’ y elegir en la lista de programas, “Bloodshed Dev-C++” y de la lista que aparece en pantalla “Dev-C++”.

También es posible lanzar el programa desde el acceso directo que aparece en el escritorio. Una vez lanzada la aplicación Dev-C++, aparece en pantalla la ventana principal del entorno de programación.

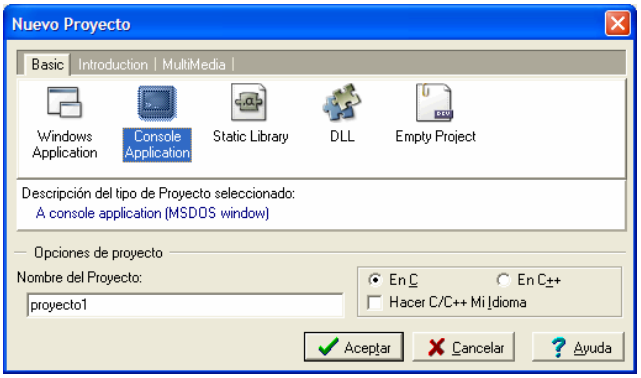
4. Cómo crear un Proyecto en Dev-C++

Para escribir un programa en Dev-C++ debe crearse lo que se denomina un *proyecto*. Un proyecto es una configuración y un grupo de ficheros que juntos producen un programa o fichero ejecutable (aplicaciones). Existen muchos tipos de proyecto, pero nosotros sólo vamos a utilizar el tipo más simple.

➢ Para crear un nuevo proyecto :

1. En el menú principal elegir la opción “***Archivo***”. En el menú desplegable que aparece seleccionar la opción “***Nuevo***” y a continuación "***Proyecto***". Aparecerá una ventana de diálogo para seleccionar el tipo de proyecto

2. Seleccionar el tipo de proyecto. En nuestro caso seleccionaremos la opción "***Console Application***". Este tipo de proyectos es el más simple, que produce una aplicación de tipo consola. Debe marcarse la opción "***En C***".



3. Debe escribirse el nombre del proyecto nuevo en el campo "***Opciones del Proyecto***", "***Nombre***". Vamos a denominar a nuestro primer proyecto "proyecto1" (en minúsculas y sin espacios en blanco).

4. Hacer click sobre el botón "***Aceptar***". Aparecerá un cuadro de diálogo para seleccionar la carpeta en la que se desea guardar el proyecto (un fichero con extensión .dev).

5. Una vez guardado el proyecto aparecerá un esquema de programa en C:

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

int main(int argc, char *argv[])
{
     system("PAUSE");
     return 0;
}
```

Modificar el esquema anterior para que sea como aparece a continuación:

```
#include <stdio.h>

void main( )
{
      printf("Hola mundo\n");
      system("PAUSE");
}
```

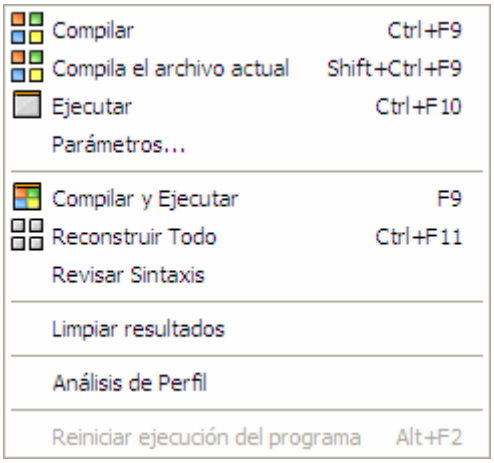
El programa anterior muestra por pantalla (por la consola) el mensaje **Hola mundo**, seguido de un salto de línea. La función *printf* imprime literalmente lo que aparece entre comillas. El símbolo '\n' es un carácter especial que indica un salto de línea en pantalla. La instrucción system("PAUSE"); es necesaria para que el programa de consola se quede esperando a la pulsación de una tecla una vez que finalice el programa. Es conveniente incluir esta instrucción al final de cada programa en las prácticas.

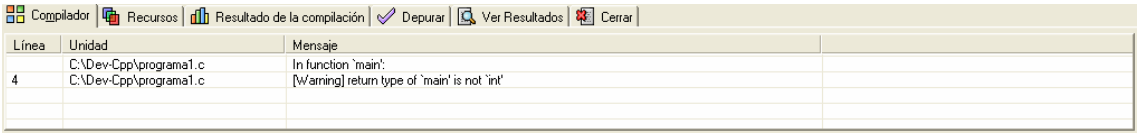
A continuación debe guardarse el fichero fuente anterior. Para ello, seleccionar la opción *Archivo → Guardar como*. Aparecerá una ventana de diálogo en la que debe introducirse el nombre con el que se desea guardar el fichero. Escribir el nombre "programa1.c", dentro de la carpeta en la que se creó el proyecto.

5. Cómo compilar y enlazar el fichero fuente

El siguiente paso será compilar y enlazar el fichero fuente de forma que obtengamos un fichero ejecutable ".exe". Al compilar el fichero fuente se obtiene un fichero objeto, con extensión ".o". Tras enlazar los ficheros objeto del proyecto obtenemos el fichero ejecutable.

Para realizar estas operaciones hay que seleccionar la opción "***Ejecutar → Compilar***" del menú principal dependiendo de si tenemos el programa instalado en inglés o en castellano. Aparecerá una ventana indicándonos que el proceso de compilado y enlazado ha terminado, como se muestra en la figura siguiente:





En dicha ventana podemos observar que se ha realizado la compilación del fichero programa1.c y se ha obtenido el fichero ejecutable proyecto1.exe, con 0 errores (errors) y 1 aviso (warning). Este aviso es debido a que el compilador Dev-C++ está diseñado para que su función main devuelva un entero. Por lo general debemos hacer caso omiso al mensaje.

Si se hubiera producido algún tipo de error en la compilación obtendríamos el mensaje correspondiente en la ventana de salida. Vamos a introducir intencionadamente un error en el programa para comprobar cómo lo muestra el compilador. ***Eliminar el ';' que aparece al final de la línea del 'printf', y comprobar el mensaje de error que se muestra.***

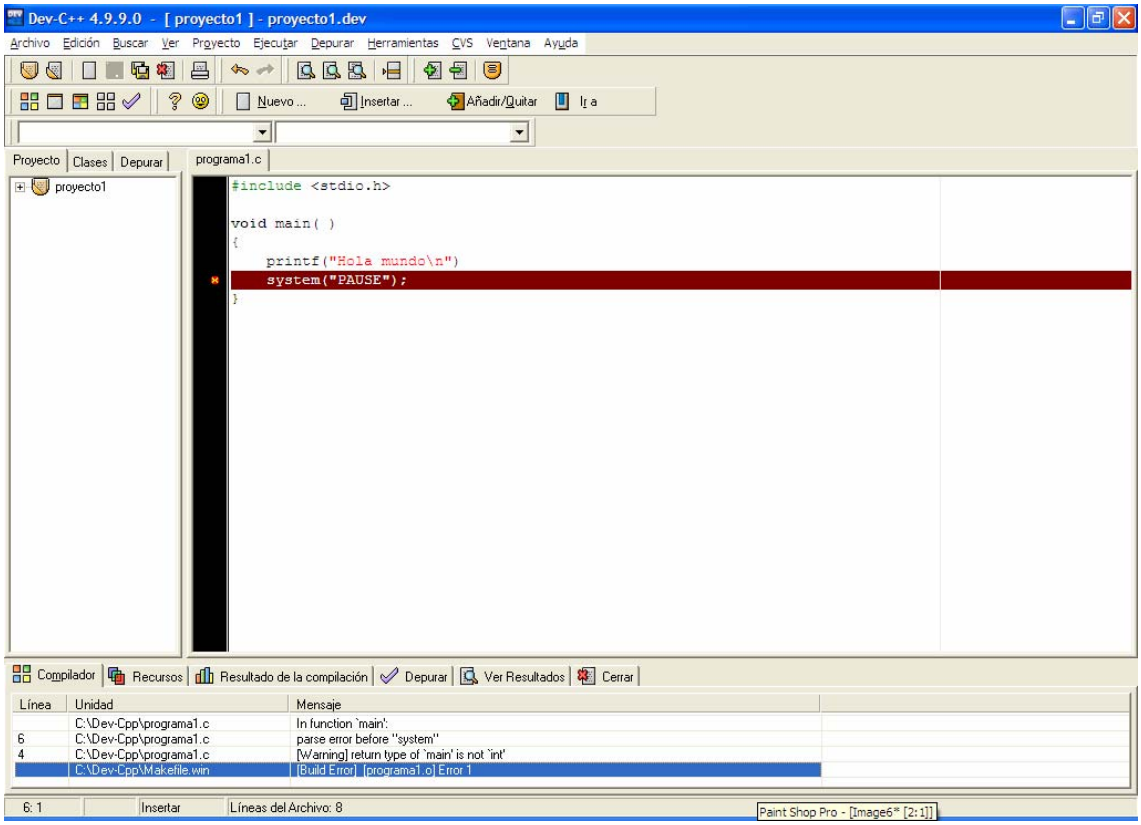
Para ejecutar un programa hay que seleccionar la opción "***Ejecutar → Ejecutar***" o bien la combinación de teclas rápidas Ctrl+F10 o bien mediante el icono correspondiente de la barra de herramientas.

El resultado de la ejecución del programa anterior es el mensaje impreso en una ventana de consola (ventana de MS-DOS), esperando que pulsemos cualquier tecla para continuar. Al pulsar cualquier tecla, el programa finaliza y la ventana de consola desaparece, devolviendo el control al entorno de Dev-C++.

6. La ventana de salida

La ventana de salida situada en la parte inferior de la ventana principal nos sirve para visualizar diferentes tipos de información. Posee varias pestañas: *Compilador*, *Recursos*, *Resultado de la compilación, Depurar, Ver Resultados y Cerrar*. Cuando se pulsa sobre cada una de ellas se nos muestra la información referente a cada una de estas acciones. Si seleccionamos la pestaña *Compilador*, la ventana de salida mostrará los diferentes procesos y mensajes de error que se producen en la fase de compilación y enlazado. Esta lista incluye todos los errores que se producen junto con el nombre del fichero en el que se ha producido el error, el número de línea y el número de error, así como un pequeño mensaje que lo identifica.

Para acceder directamente a la posición en la que se ha producido el error en nuestro código fuente haremos doble clic sobre la línea que nos informa del error. El resultado de estas acciones es una marca en la ventana de edición del fichero que posee el error indicándonos la línea en la que se ha producido.



El resto de solapas de la ventana de salida se explicarán en prácticas posteriores. La solapa ***Cerrar*** se utiliza para cerrar dicha ventana.

7. Ejercicios

Vamos a crear varios proyectos y programas fuente, siguiendo el mismo procedimiento de las secciones anteriores.

➢ Proyecto 2

Crear un proyecto con el nombre *proyecto2*. Crear un fichero fuente con el nombre *programa2*, e introducir el siguiente código:

```
#include <stdio.h>

void main(void)
{
      int numero;

      // Se inicializa la variable numero a 2
      numero = 2;
      printf("Hola mi numero es:%d \n", numero);

      system("PAUSE");
}
```

Compilar, enlazar y ejecutar el proyecto. Comprobar la salida que se obtiene y escribirla en el cuadro siguiente:

Salida obtenida en el proyecto 2:

En este programa aparece un sentencia de asignación (símbolo =), de forma que se le asigna a la variable entera *numero* el valor 2. Posteriormente se imprime un mensaje de texto que incluye el valor asignado a esta variable, mediante el uso del código %d en el lugar exacto donde se desea que aparezca este valor.

➢ Proyecto 3

Crear un proyecto con el nombre *proyecto3*. Crear un fichero fuente con el nombre *programa3*, e introducir el siguiente código:

```
#include <stdio.h>

void main(void)
{
      printf("Division entera:5/4 es %d\n",5/4);
      printf("Division entera:6/3 es %d\n",6/3);
      printf("Division entera:7/4 es %d\n",7/4);
      printf("Division flotante:7./4. es %f\n",7./4.);
      printf("Division mixta:7./4 es %f\n",7./4);
}
```

Compilar, enlazar y ejecutar el proyecto. Comprobar la salida que se obtiene y escribirla en el cuadro siguiente:

Salida obtenida en el proyecto 3:

➢ Proyecto 4

Crear un proyecto con el nombre *proyecto4*. Crear un fichero fuente con el nombre *programa4*, e introducir el siguiente código:

```
void main(void)
{
      int seg, min, resto;
      int segMin = 60;
      printf("Introduzca número de segundos:\n");
      scanf("%d", &seg);
      min = seg/segMin;
      resto = seg%segMin;
      printf("%d seg. Son %d min. y %d seg.\n", seg,min,resto);
}
```

Este programa lee un número expresado en segundos y lo transforma en un número expresado en minutos y segundos. La sentencia *scanf("%d", &seg);* sirve para leer por teclado un número entero (*int*) y guardar el valor tecleado en la variable *seg*.

Compilar, enlazar y ejecutar el proyecto. Comprobar la salida que se obtiene para las entradas 60, 129 y 230 y escribirla en el cuadro siguiente:

Salida obtenida en el proyecto 4:

➢ Proyecto 5

Crear un proyecto con el nombre *proyecto5*. Crear un fichero fuente con el nombre *programa5*, e introducir el siguiente código.

```
main()
{
      printf("Entero: %d\n", sizeof(int));
      printf("Float: %d\n", sizeof(float));
      printf("Double: %d\n", sizeof(double));
      printf("Caracter: %d\n", sizeof(char));

}
```

Compilar, enlazar y ejecutar el proyecto. Comprobar la salida que se obtiene y escribirla en el cuadro siguiente, comentando el resultado obtenido:

Salida obtenida en el proyecto 5:

➢ Proyecto 6

Crear un proyecto con el nombre *proyecto6*. Crear un fichero fuente con el nombre *programa6*, e introducir el siguiente código:

```
main()
{
      printf("%d  %o  %x\n", 336, 336, 336);
      printf("%d  %u\n", -336, -336);
      printf("%c  %d\n", 'A', 'A');
      printf("%d  %c\n", 80, 80);
      printf("%d  %c\n", 336, 336);
}
```

El significado de los códigos de formato es el siguiente:

Código de formato	Significado
%d	número entero con signo expresado en decimal (base 10)
%o	número entero con signo expresado en octal (base 8)
%x	número entero con signo expresado en hexadecimal (base 16)
%u	número entero sin signo expresado en base 10
%c	carácter

Compilar, enlazar y ejecutar el proyecto. Comprobar la salida que se obtiene y escribirla en el cuadro siguiente:

Salida obtenida en el proyecto 6:

➢ Proyecto 7

Crear un proyecto con el nombre *proyecto7*. Crear un fichero fuente con el nombre *programa7*, e introducir el siguiente código:

```
main()
{
      printf("%2d\n",336);
      printf("%5d\n",336);
      printf("%3.2f\n",134.745);
      printf("%7.1f\n",134.745);
      printf("%7.1f\n",-134.745);
      printf("%10.3e\n",1234.56);
}
```

El significado de los códigos de formato es el siguiente:

Código de formato	Significado
%A.Bf	número en coma flotante, donde A es el número total de caracteres y B es el número de decimales
%Ad	número entero con signo
%A.Be	número en coma flotante, donde A es el número total de caracteres y B es el número de decimales, expresado en notación científica

Compilar, enlazar y ejecutar el proyecto. Comprobar la salida que se obtiene y escribirla en el cuadro siguiente:

Salida obtenida en el proyecto 7:

➢ Proyecto 8

Crear un proyecto con el nombre *proyecto8*. Crear un fichero fuente con el nombre *programa8*, e introducir el siguiente código.

```
main()
{
      int i = 1;

      printf("i = %d\n", i);
      printf("i = %d\n", ++i);
      printf("i = %d\n", i);

      i = 1;
      printf("i = %d\n", i);
      printf("i = %d\n", i++);
      printf("i = %d\n", i);

}
```

Compilar, enlazar y ejecutar el proyecto. Comprobar la salida que se obtiene y escribirla en el cuadro siguiente, comentando el resultado obtenido:

Salida obtenida en el proyecto 8:

➢ Proyecto 9

Escribir un programa que lea una cantidad de euros (entera) y diga cuantos billetes de 50, 20, 10 y 5 euros, y cuantas monedas de 2 y 1 euro componen esa cantidad de dinero.
A continuación se muestra un ejemplo de ejecución del programa. En negrita se muestra el dato introducido por el usuario.

```
Introduzca una cantidad de euros: 48
Dicha cantidad consta de:
0 billetes de 50 euros
2 billetes de 20 euros
0 billetes de 10 euros
1 billetes de 5 euros
1 monedas de 2 euro
1 monedas de 1 euro
```

Programa:

➢ Proyecto 10

Escribir un programa que pida las notas de un alumno correspondientes a los tres trimestres del curso. A continuación, debe mostrar las notas introducidas y la nota media, con una precisión de 1 decimal.
A continuación se muestra un ejemplo de ejecución del programa. En negrita se muestran los datos introducidos por el usuario.

Introduzca la primera nota: **4.8**
Introduzca la segunda nota: **7.56**
Introduzca la tercera nota: **9**

Nota 1: 4.8
Nota 2: 7.6
Nota 3: 9.0

La nota media es: 7.1

Programa: