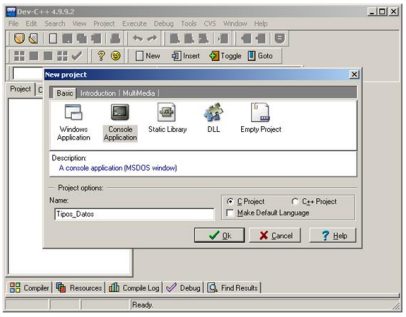
There are no translations available.



El primer objetivo de un programa informático es el manejo de datos. Un dato es toda aquella información que se puede tratar en un programa informático.

En este artículo vamos a conocer los tipos de datos que podemos manejar programando en C.

Existen unos datos de entrada y unos datos de salida.

Los datos de entrada se manipulan en el programa produciendo unos datos de salida.

Un dato dentro de un programa se caracteriza por llevar asociado un identificador, un tipo y un valor.

- Identificador: Nombre para referenciar al dato dentro del programa
- Tipo: el tipo de un dato determina el rango de valores que puede tomar el dato y su ocupación en memoria durante la ejecución del programa
- Valor: Sera un elemento determinado dentro del rango de valores permitidos por el tipo de dato definido.

Algunos ejemplos de datos son: la edad, el saldo de una cuenta bancaria, el nombre de una persona, la letra del piso de una dirección, etc.

A continuación vamos a describir los distintos tipos de datos que existen.

TIPOS DE DATOS

Hay dos clases de tipos de datos: tipos fundamentales y tipos derivados.

Únicamente vamos a ver los tipos de datos fundamentales.

Tipo entero: representa números enteros con o sin signo, que estarán compuestos por los dígitos del 0 al 9, pudiendo ser precedidos por los signos + o -.

Algunos ejemplo de datos enteros son: 0, 23, -176, -1, etc.

Para definir variables en C se antepone la palabra reservada del tipo al identificador de la variable. El tipo de datos entero se define en el lenguaje de programación C por la palabra reservada *int.*

Para definir un tipo de dato en C se escribe lo siguiente:

int nombre_variable = valor;

No es necesario que la variable tenga un valor predeterminado. Se puede definir sin asignarle ningún valor.

Si tenemos varios datos que son del mismo tipo, se pueden definir todas en la misma línea de código escribiendo un único *int*, separando el nombre de las variables por “,”. Una vez que se haya acabado de definir variables, se cierra la línea de código con “;”,

Por ejemplo: int edad = 24;

int edad;

int edad, num, contador;

Tipo real: Se emplean para representar números reales (con decimales).

Para definir datos reales se antepone la palabra reservada *float* al identificador de la variable.

float identificador = valor;

Por ejemplo: float numero1, numero2;

float numero3 = 123.43;

float numero3;

Tipo carácter: Este tipo de datos se emplea para representar un carácter perteneciente a un determinado código utilizado por el ordenador (normalmente el código ASCII).

Para representar este tipo de dato se antepone la palabra reservada *char* al identificador de la variable.

Char identificador = ‘valor’;

Una constante tipo char se representa como un solo carácter encerrado entre comillas simples.

Por ejemplo: char letra, letra2;

char letra=’a’;

Tipo cadena de caracteres: una cadena de caracteres es un número de caracteres consecutivos (incluso ninguno) encerrado entre unos delimitadores determinados, que en el lenguaje C son las comillas dobles.

Para definir variables de tipo cadena, estas se definen como vectores de caracteres, esto es, anteponiendo la palabra reservada *char* al identificador de la variable, y después entre corchetes la longitud máxima de cadena.

Char identificador[cantidad] = “ mensaje ”;

Por ejemplo: char cadena[20];

char cadena[20] = “Hola mundo”;

char cadena[] = “HOLA”;

En la siguiente tabla se hace un resumen de los distintos tipos de datos:

Tipo de dato	**Palabra reservada**	**Ejemplo**
Entero	Int	Int numero=0;

Real

Float

Float numero=12.2;

Carácter

Char

Char letra = ‘a’;

Cadena de carácter

Char

Char palabra[10] = “HOLA”;

Lectura y Escritura de Datos

Para poder leer y escribir datos en el lenguaje de programación C existen una serie de funciones agrupadas en un conjunto de librerías de código objeto, que constituyen la llamada biblioteca estándar del lenguaje. En el caso concreto de las funciones de entrada y salida (lectura y escritura), su archivo de cabecera es *stdio.h.*

Existen varias funciones que realizan la entrada y salida de datos en el lenguaje de programación C, pero nos vamos a centrar solamente en dos de ellas: *printf()* y *scanf().*

Funcion printf()

La función *printf()* sirve para escribir datos en la pantalla con un formato determinado.

El prototipo de esta función es la siguiente:

int printf(const char *formato, arg1, arg2, …, argn)

donde:

formato: hace referencia a una cadena de caracteres que contiene información sobre el formato de salida.

Arg1, arg2, …, argn: argumentos que representan los datos de salida

La cadena de control puede constar de hasta tres tipos de información:

- texto: que será mostrado por pantalla tal cual este escrito

- Secuencias de escape: secuencias reservadas propias del lenguaje. Van precedidas por '' y las más importantes son 'n' que introduce un salto de línea, y 't' que introduce un tabulador.

- Especificadores de formato: son ciertos caracteres precedidos del carácter tanto por ciento '%'.

A continuación se muestra una tabla con estos especificadores de formato dependiendo del tipo de dato a los que representan:

Tipo de dato	**Especificadores de formato**	**Comentario**
Int	%d	Entero con signo
Float	%f	

Real con signo

Char

%c

carácter

Char [n]

%s

Cadena de caracteres

Ejemplos:

printf(“n Valores: %d, %f”, entero, real);

Salida por pantalla:

(Retorno de carro)Valores: 2, 2.3

printf(“%st%dn%f”, concepto, num, coste);

Salida por pantalla:

Cremallera 1234

12.345

Función scanf()

La función *scanf()* se pude considerar de alguna manera como la inversa de la función *printf(),* pues sirve para introducir datos desde el teclado con un formato determinado.

El prototipo de esta función es el siguiente:

int scanf(const char *formato, arg1, arg2, …, argn);

donde:

- formato: hace referencia a una cadena de caracteres (cadena de control) que contiene información sobre el formato de los datos que se van a introducir mediante el teclado.
- arg1, arg2, …, argn: son los argumentos que representan los datos de entrada.

La cadena de control puede constar de:

- Especificadores de formato, que son ciertos caracteres precedido por el carácter tanto por ciento '%'.
- Caracteres de espacio
- Otros caracteres no espacios, para entradas con formato

A continuación se muestra una tabla con estos especificadores de formato dependiendo del tipo de dato a los que representan:

Tipo de dato	**Especificadores de formato**	**Comentario**
Int	%d	Entero con signo
Float	%f	

Real con signo

Char

%c

carácter

Char [n]

%s

Cadena de caracteres

Ejemplos:

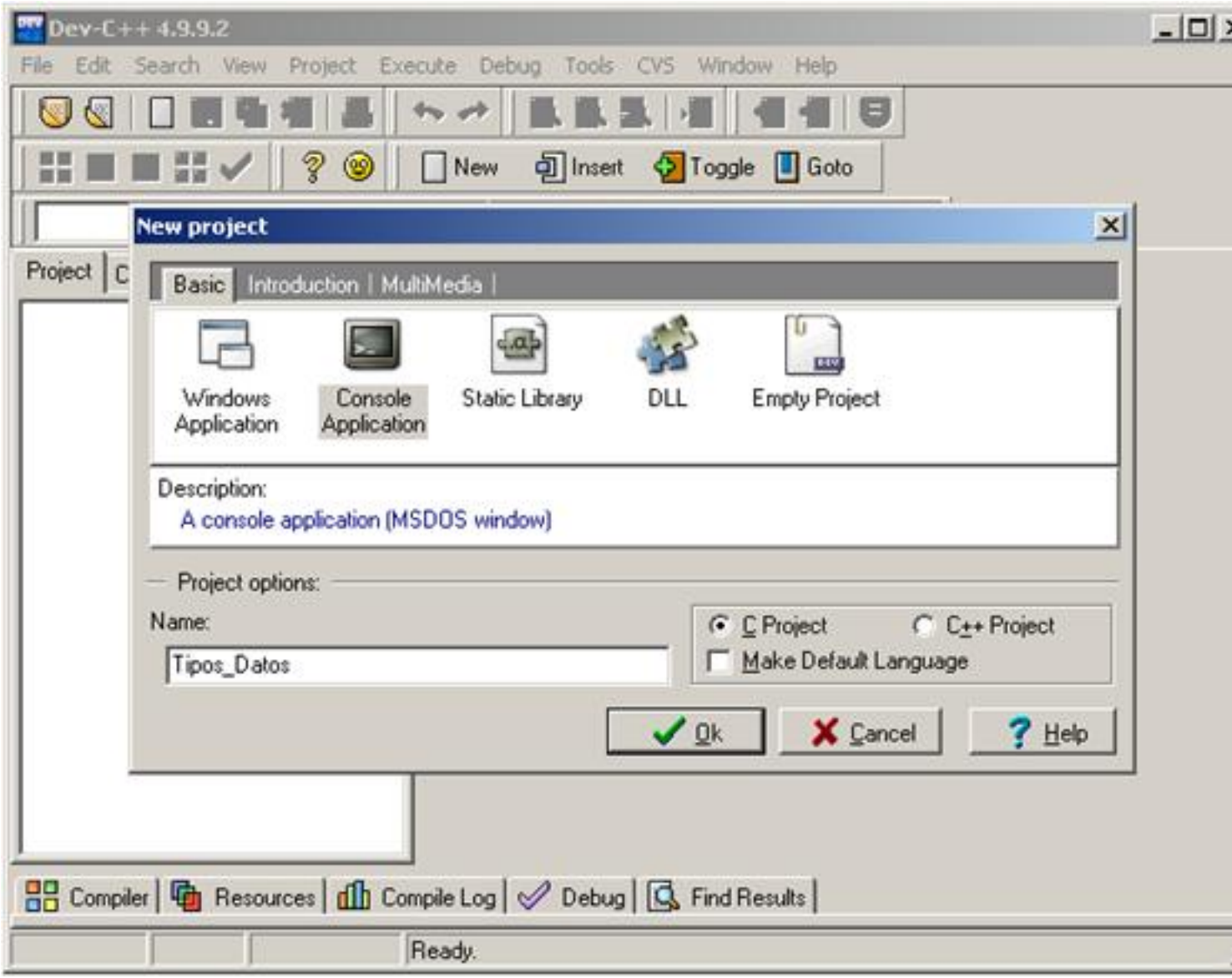
scanf(“%d”, numero);

scanf(“%d %s %f”, numero, &cadena, &real);

Programa Escritura por Pantalla

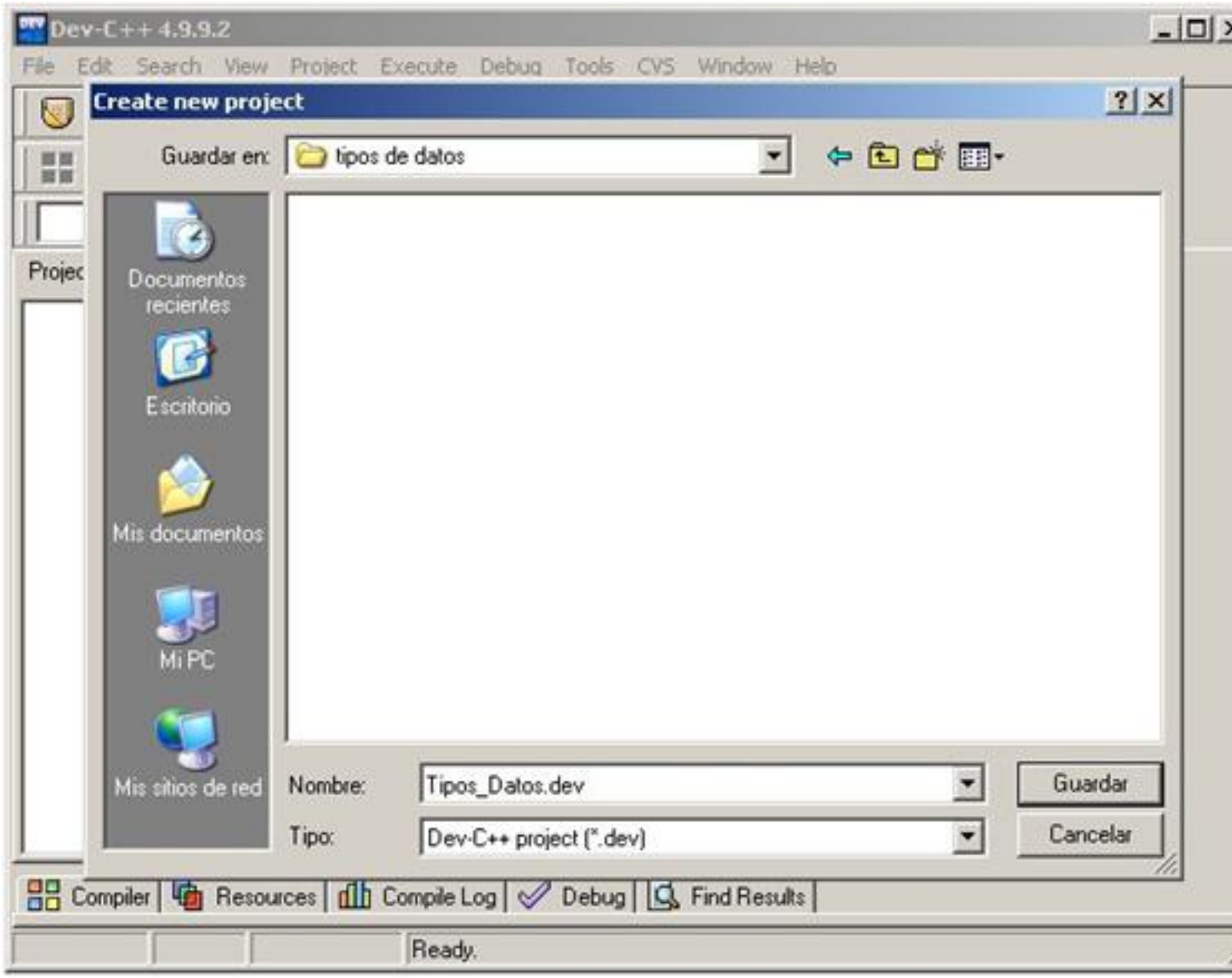
Vamos a hacer un programa que muestre el valor de los distintos tipos de datos que hemos visto anteriormente. El valor de los datos lo vamos a dar ya definido en el código.

Lo primero que tenemos que hacer es crear un nuevo proyecto:

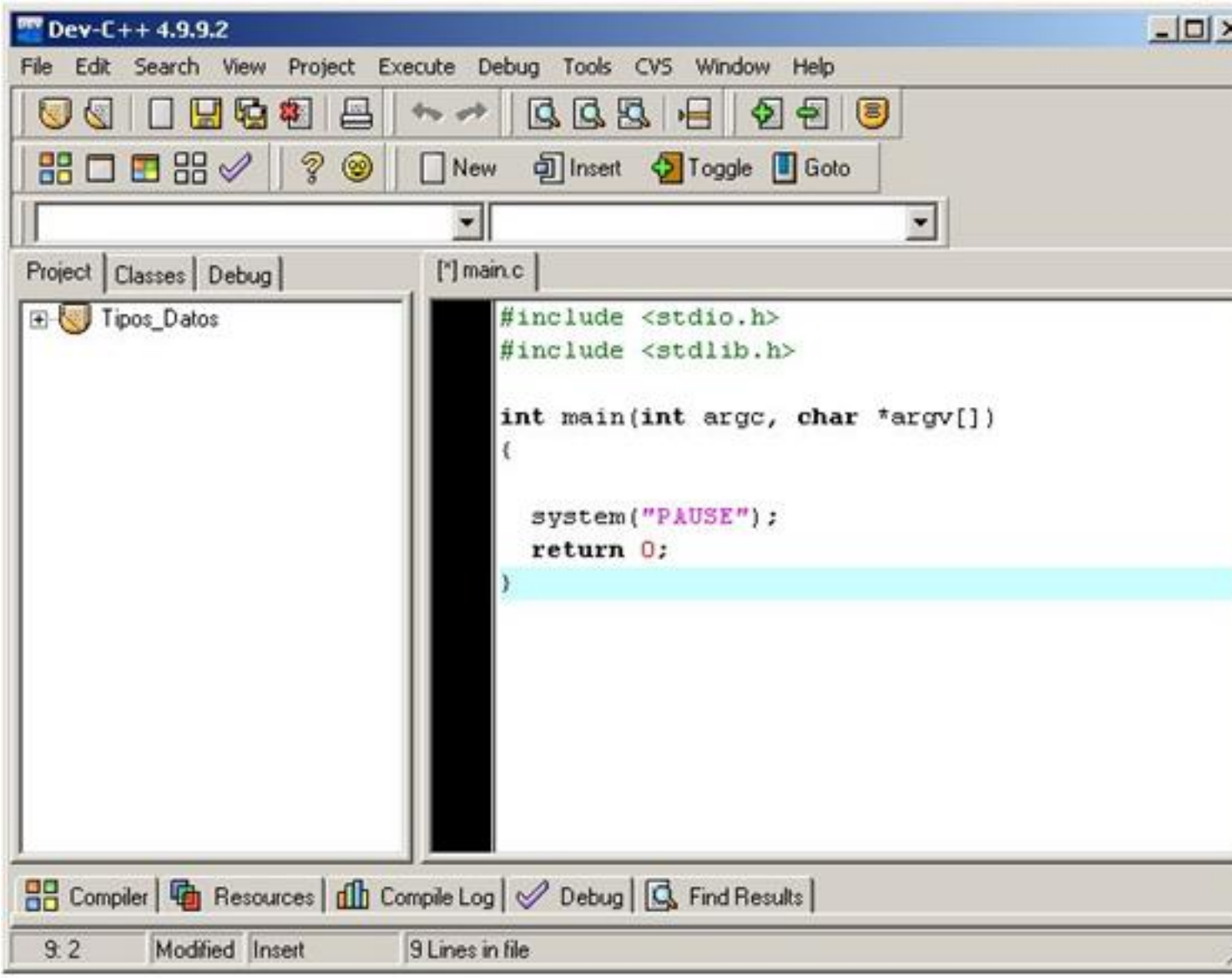


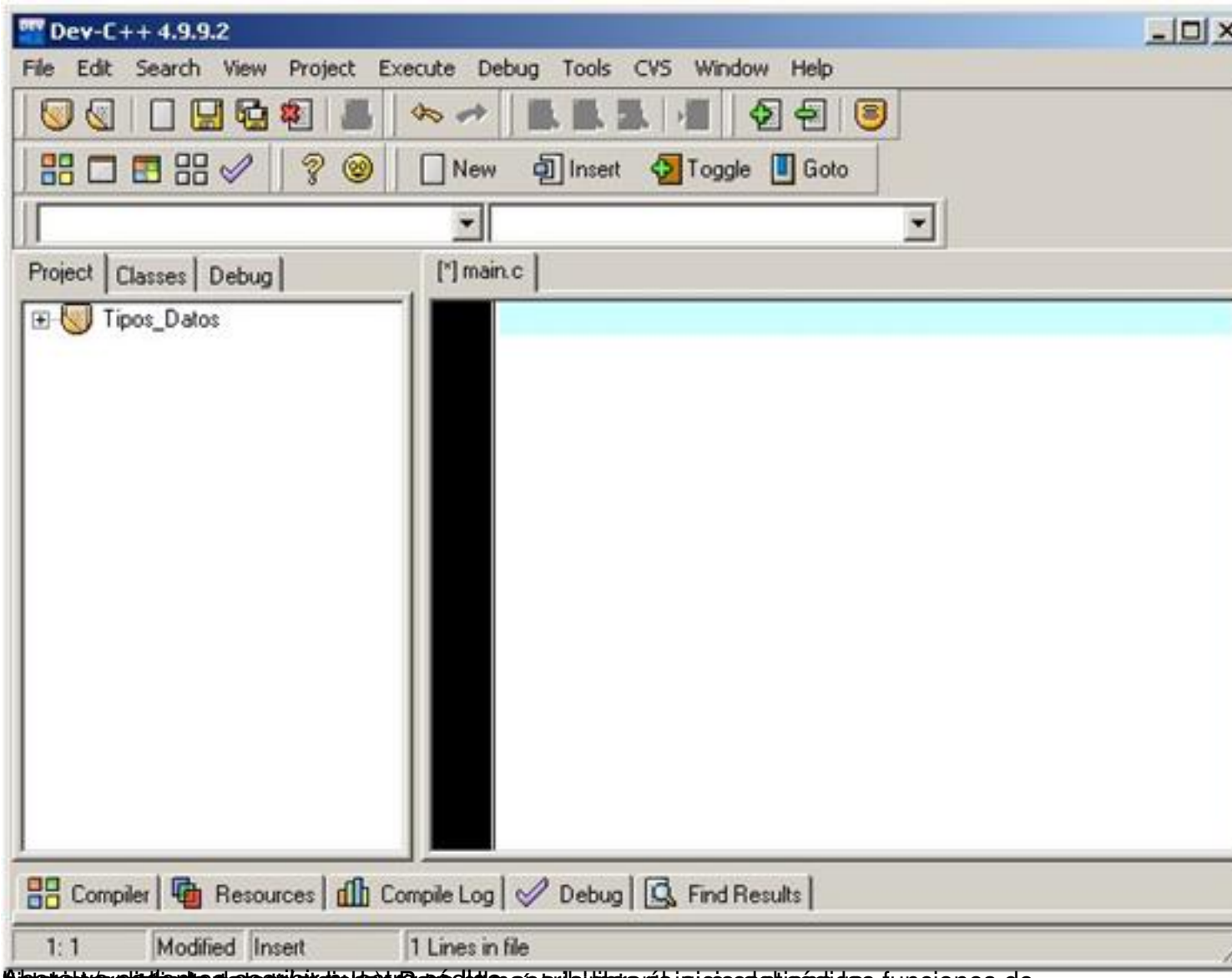
Creamos un nuevo proyecto que sea *Console Application*, seleccionamos C Project y le ponemos el nombre que queramos, que en este caso es:
Tipo_Datos
.

A continuación indicamos donde queremos guardar el proyecto que hemos creado.

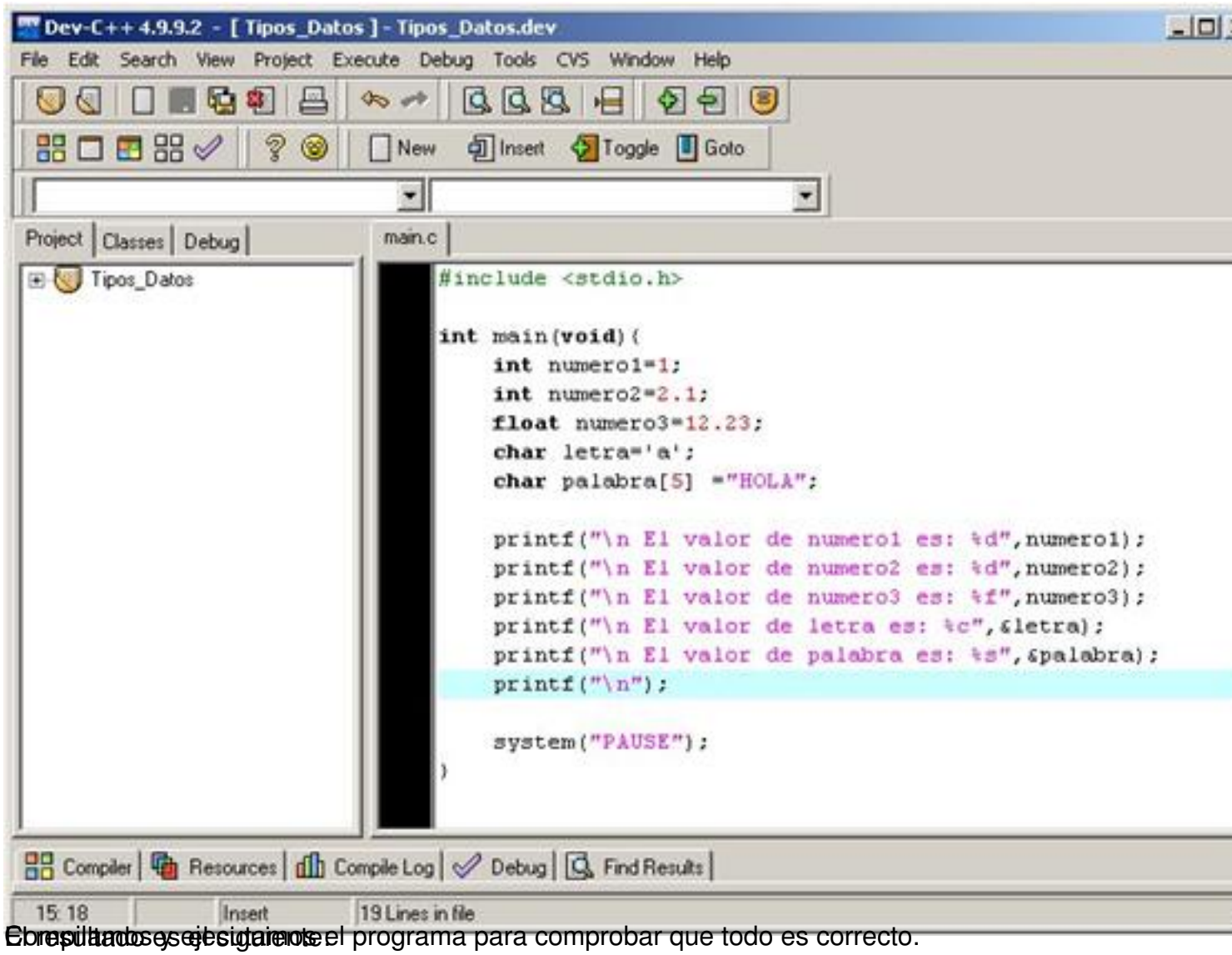


Por defecto, Dev-C++ crea el archivo *main.c*, pero lo borramos ya que queremos aprender a programar desde el principio.

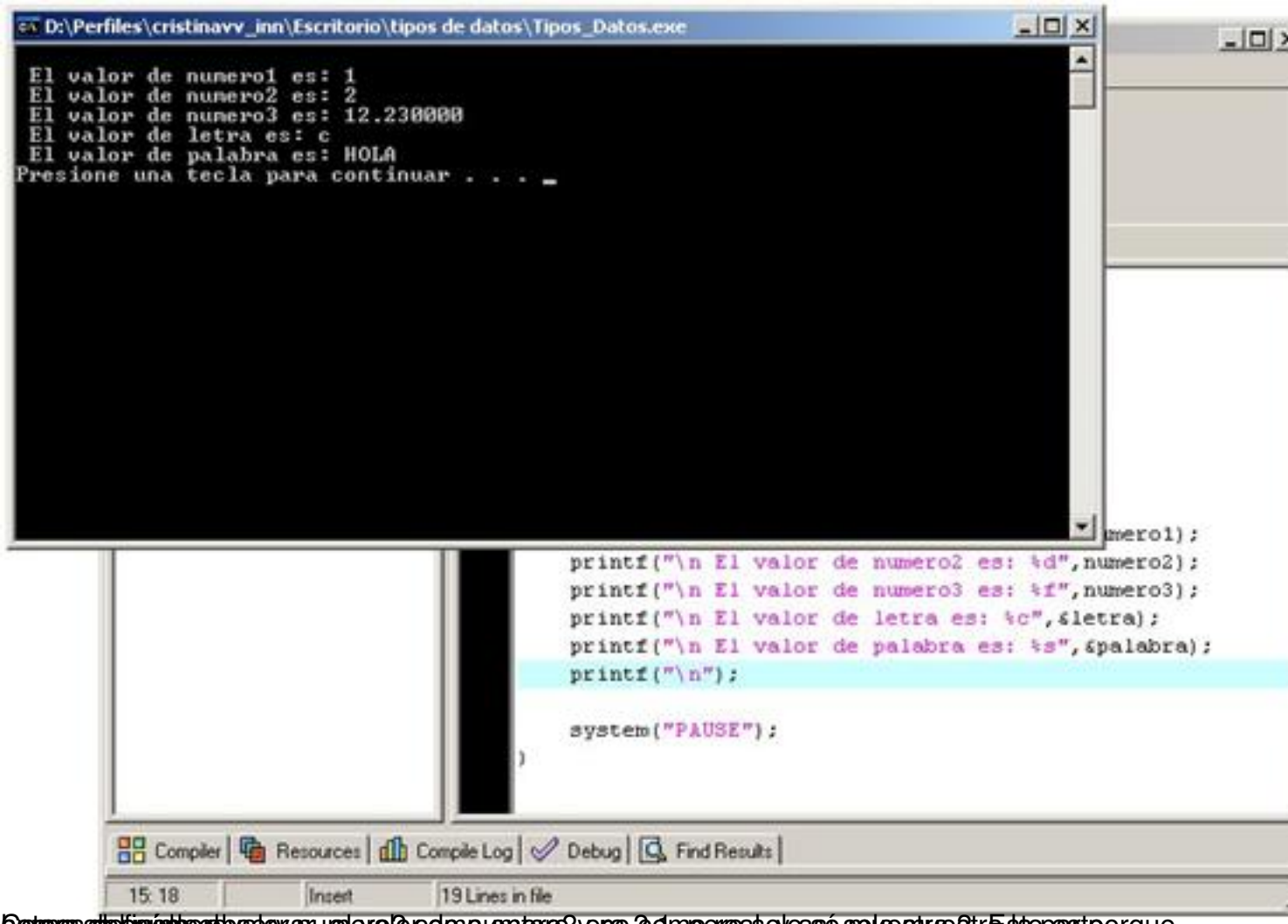

Dev-C++ 4.9.9.2
File Edit Search View Project Execute Debug Tools CVS Window Help
New Insert Toggle Goto
Project Classes Debug
Tipos_Datos
[*] main.c
#include <stdio.h>
#include <stdlib.h>
int main(int argc, char *argv[])
{
system("PAUSE");
return 0;
}
Compiler Resources Compile Log Debug Find Results
9: 2 Modified Insert 9 Lines in file



[illegible] hasta que pasemos a la siguiente lección, ponemos la



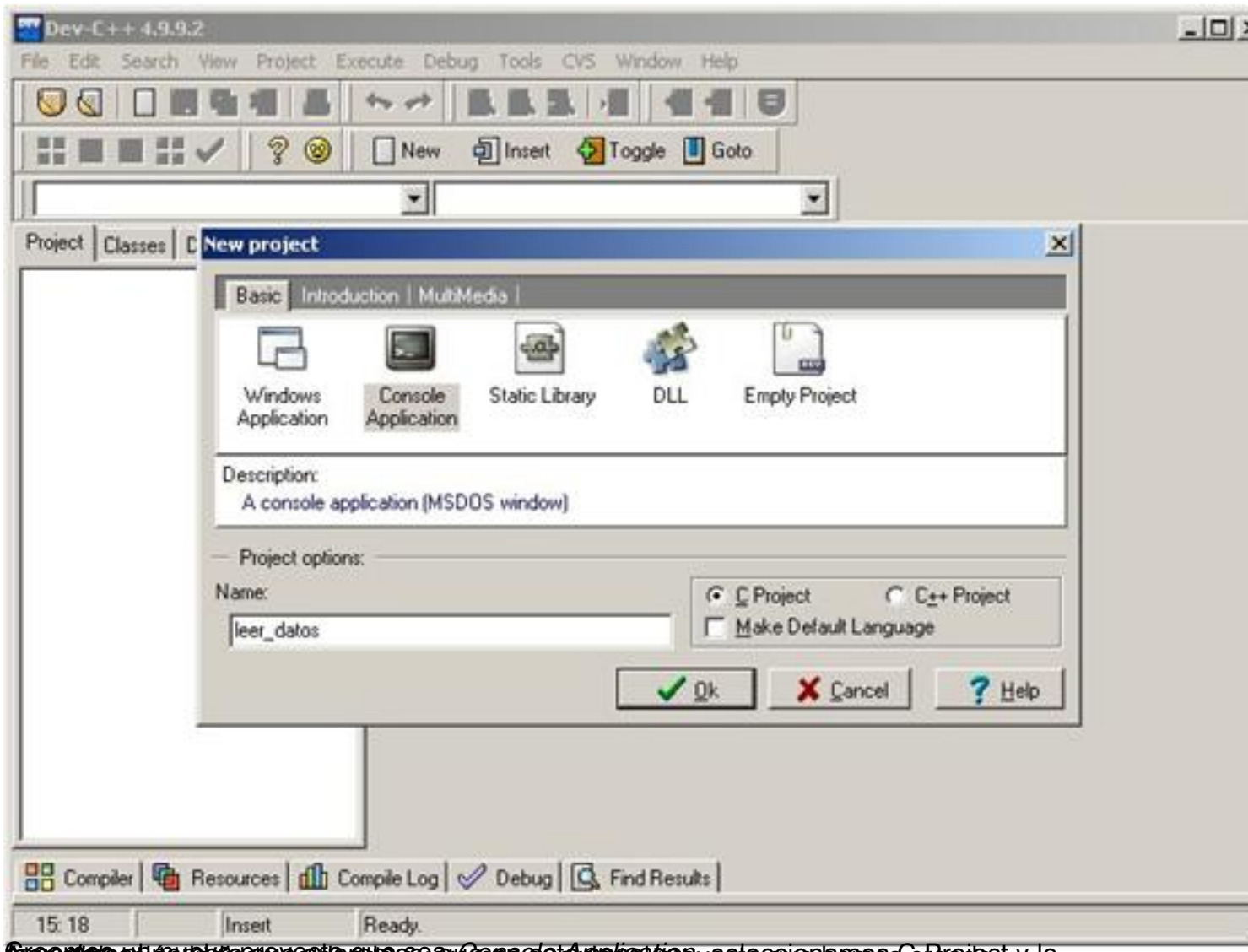
Compilamos y ejecutamos el programa para comprobar que todo es correcto.



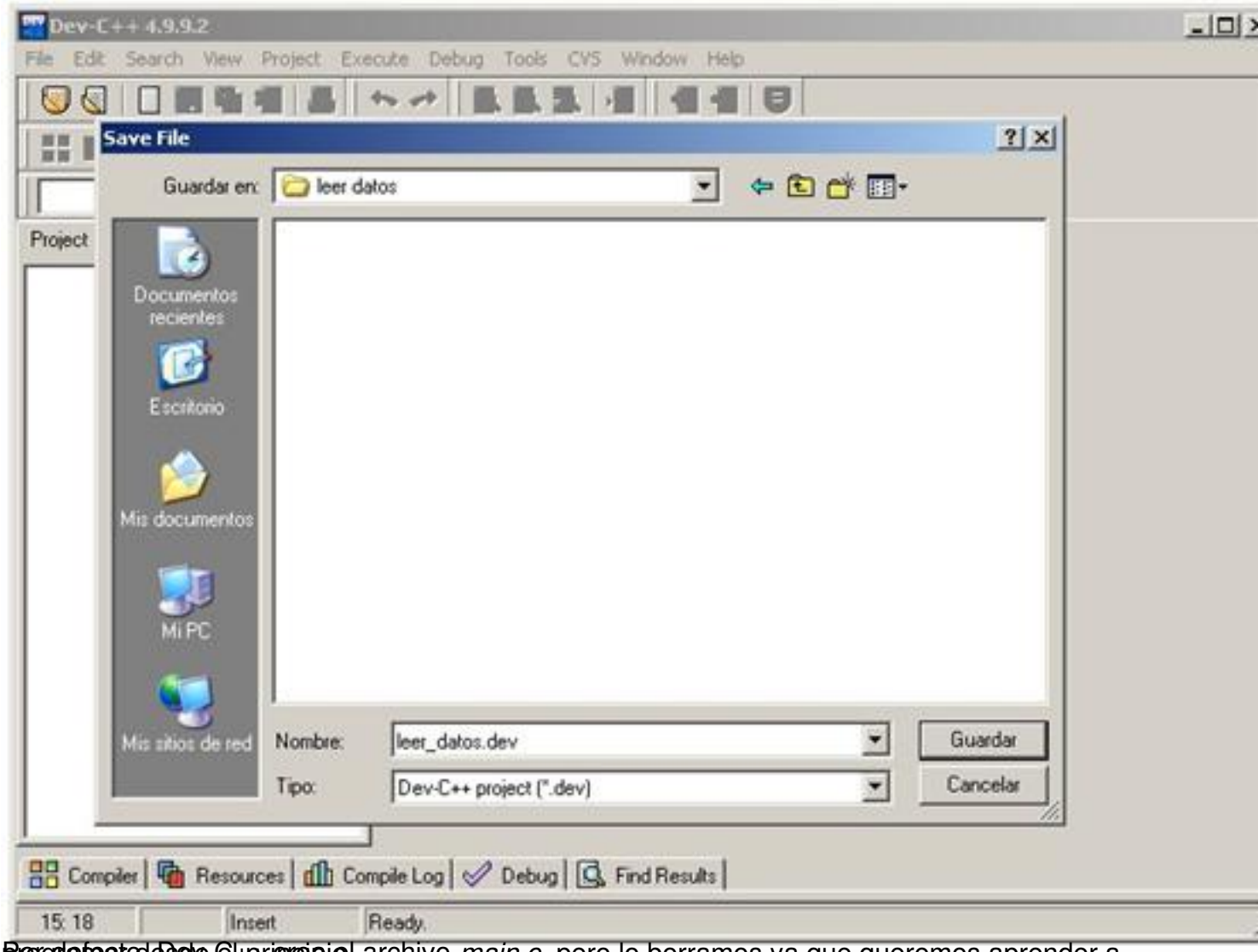
Programa de Lectura y Escritura de Datos

Vamos a hacer un programa que muestre el valor de los distintos tipos de datos que hemos visto anteriormente. El valor de los datos lo vamos a dar ya definido en el código.

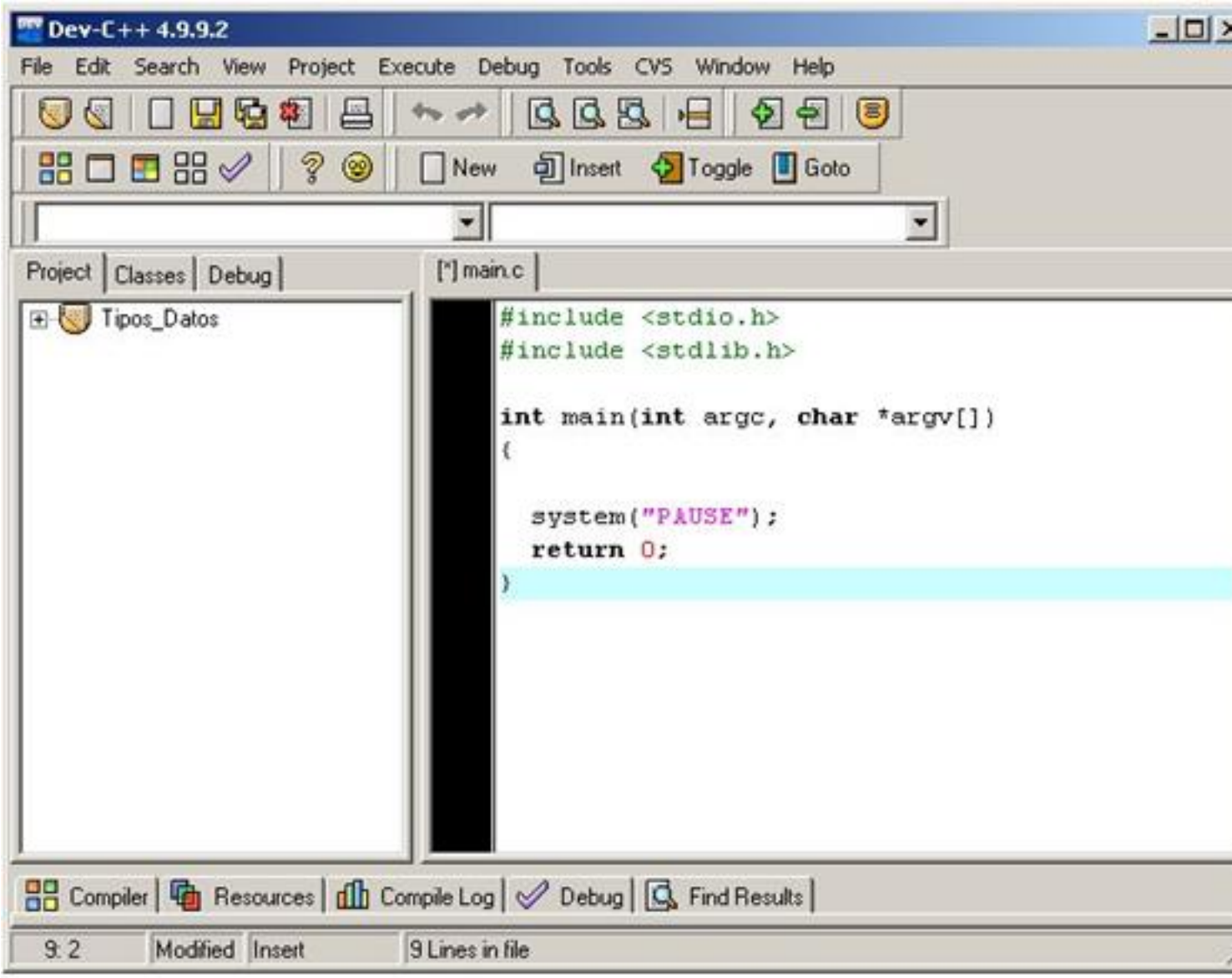
Lo primero que tenemos que hacer es crear un nuevo proyecto:

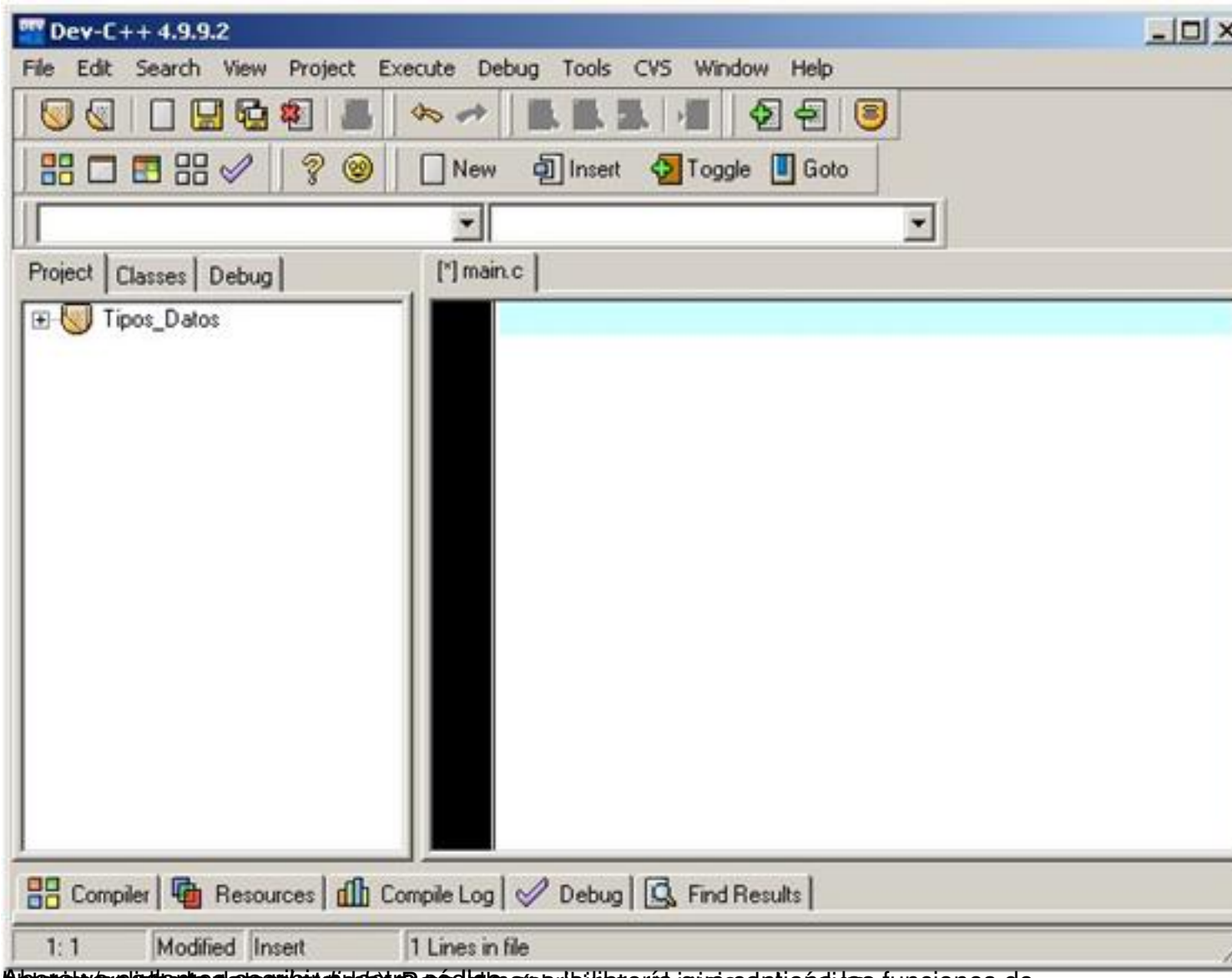


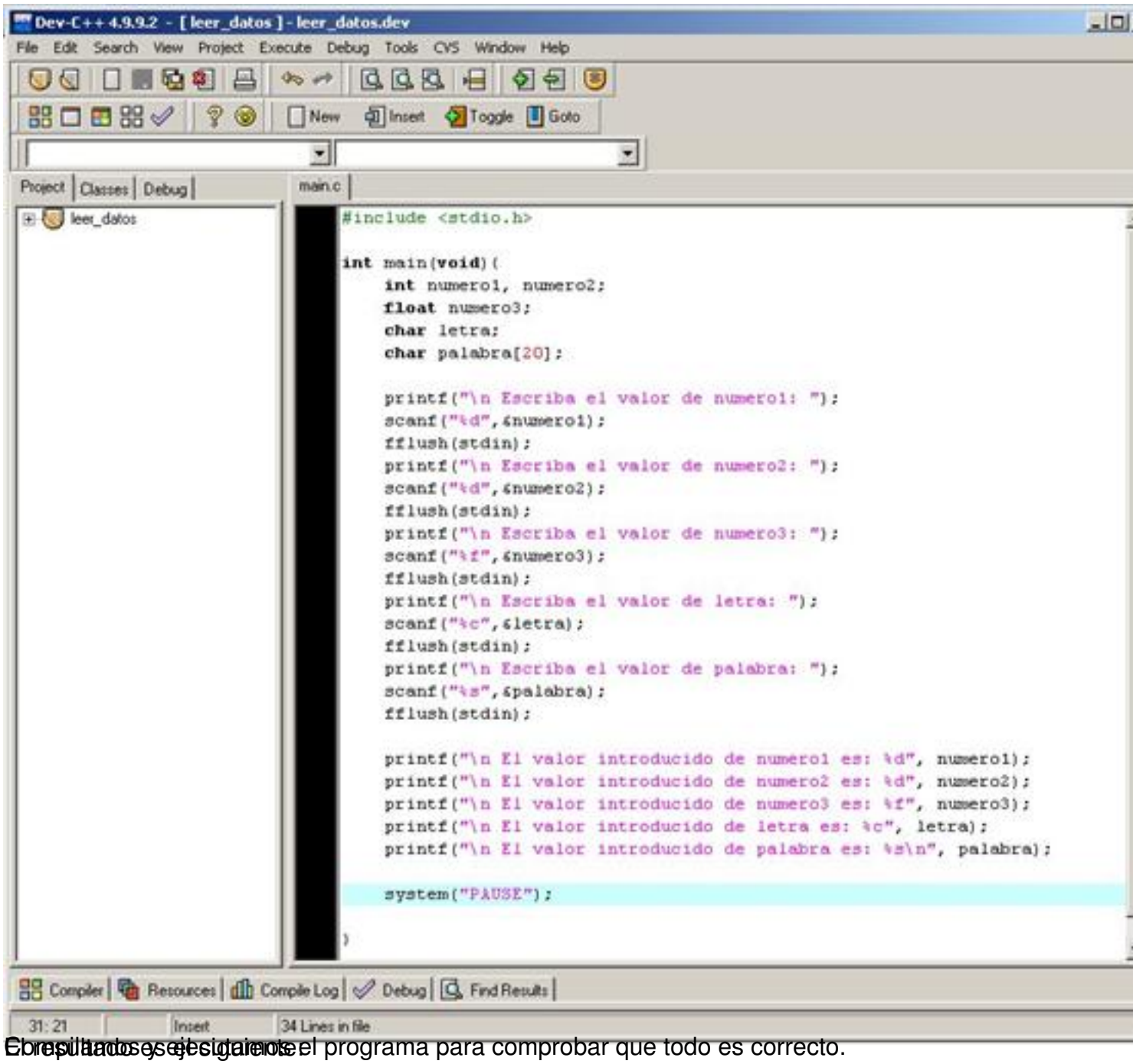
Creamos un nuevo proyecto que sea Console Application, seleccionamos C Project y le
A continuación indicamos donde queremos guardar el proyecto que hemos creado.



Por defecto Dev-C++ crea el archivo *main.c*, pero lo borramos ya que queremos aprender a programar desde el principio.

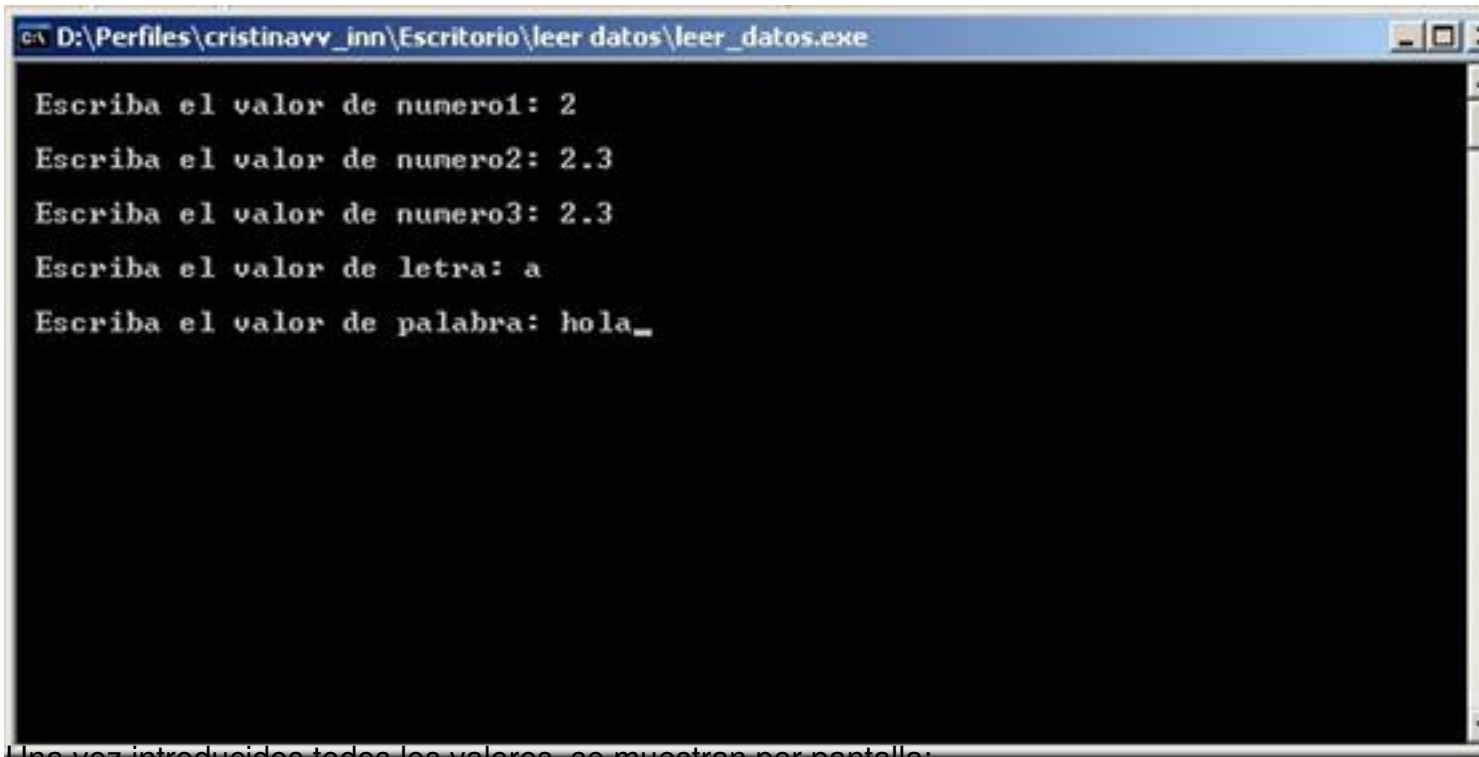

Dev-C++ 4.9.9.2
File Edit Search View Project Execute Debug Tools CVS Window Help
New Insert Toggle Goto
Project Classes Debug
Tipos_Datos
[*] main.c
#include <stdio.h>
#include <stdlib.h>
int main(int argc, char *argv[])
{
system("PAUSE");
return 0;
}
Compiler Resources Compile Log Debug Find Results
9: 2 Modified Insert 9 Lines in file


Dev-C++ 4.9.9.2
File Edit Search View Project Execute Debug Tools CVS Window Help
New Insert Toggle Goto
Project Classes Debug
[*] main.c
Tipos_Datos
Compiler Resources Compile Log Debug Find Results
1: 1 Modified Insert 1 Lines in file



El resultado es el siguiente

Compilamos y ejecutamos el programa para comprobar que todo es correcto.



Una vez introducidos todos los valores, se muestran por pantalla: