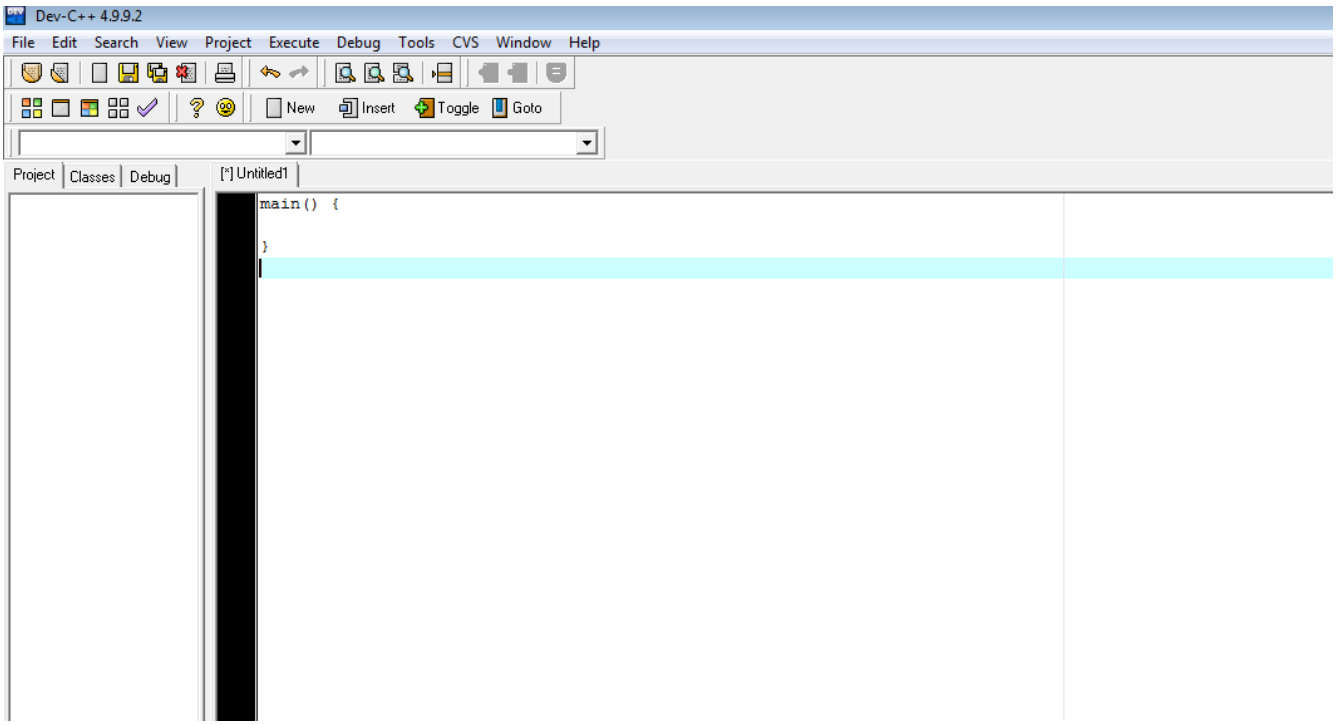
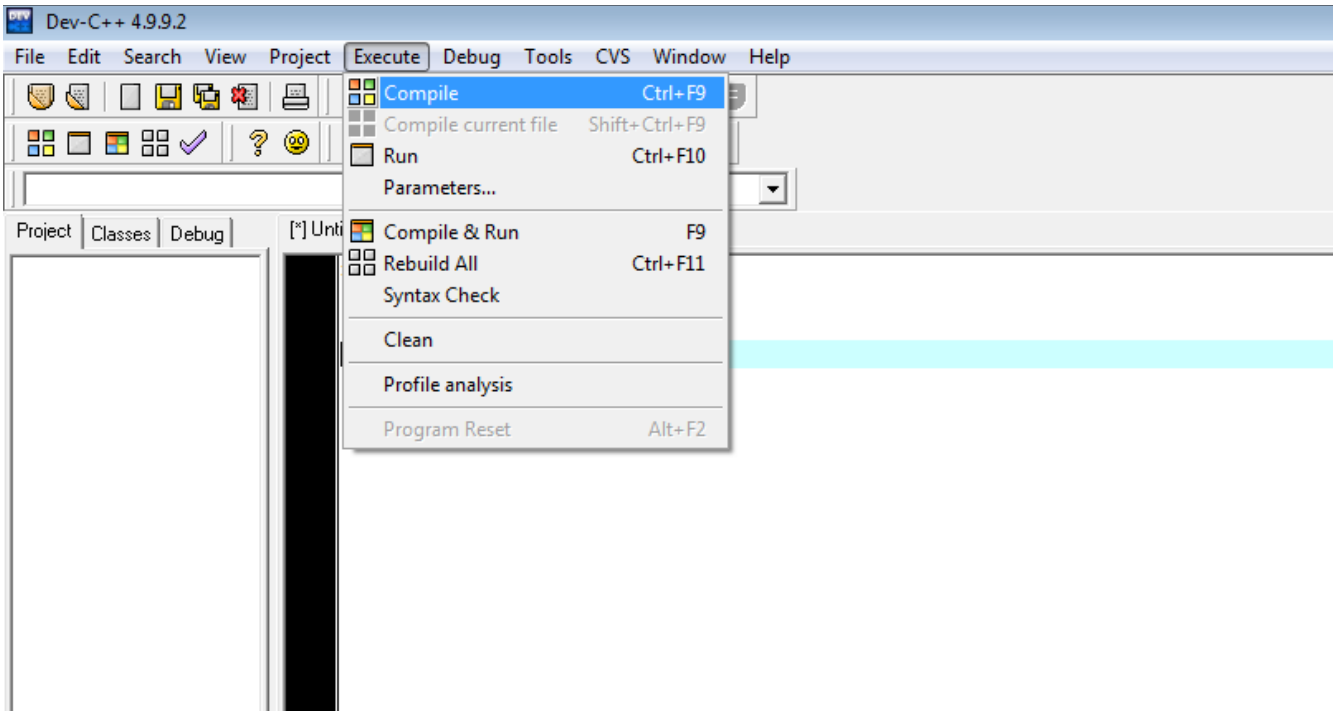
Código C y pequeño ejemplo

¿Qué tal, alumnos? ¿Cómo están? Voy a avisarles que las próximas clases seguramente vengan de manera tardada, porque ando un poco complicado con el tiempo.
A parte de ésto, quizás viendo para adelante vaya a abrir una página personal, dejando de lado a Udemy, para el curso. Así tenemos herramientas personalizadas, más maneras de ver los cursos, foros, etc, etc. :D

Bueno, veremos hoy código C y un pequeño ejemplo.
Para empezar veamos como tiene que ser nuestro programa.

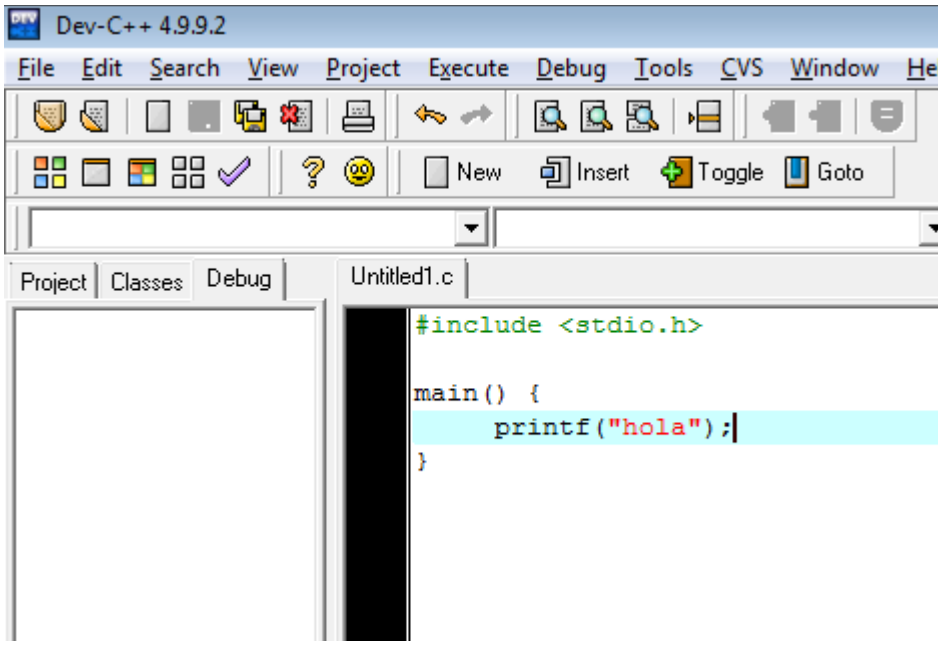


Eso sería lo mínimo e **indispensable** para que haya un programa. Se le llama la "**funcion main**", y es donde, entre sus **llaves**, irá el código que se ejecutará como programa principal. Primero le damos a "guardar como" y le damos al formato "**.c**" porque sino lo hará ".cpp" por defecto, y no es la idea. Podemos darle a **compilar** de esta manera o con (**ctrl+F9**):

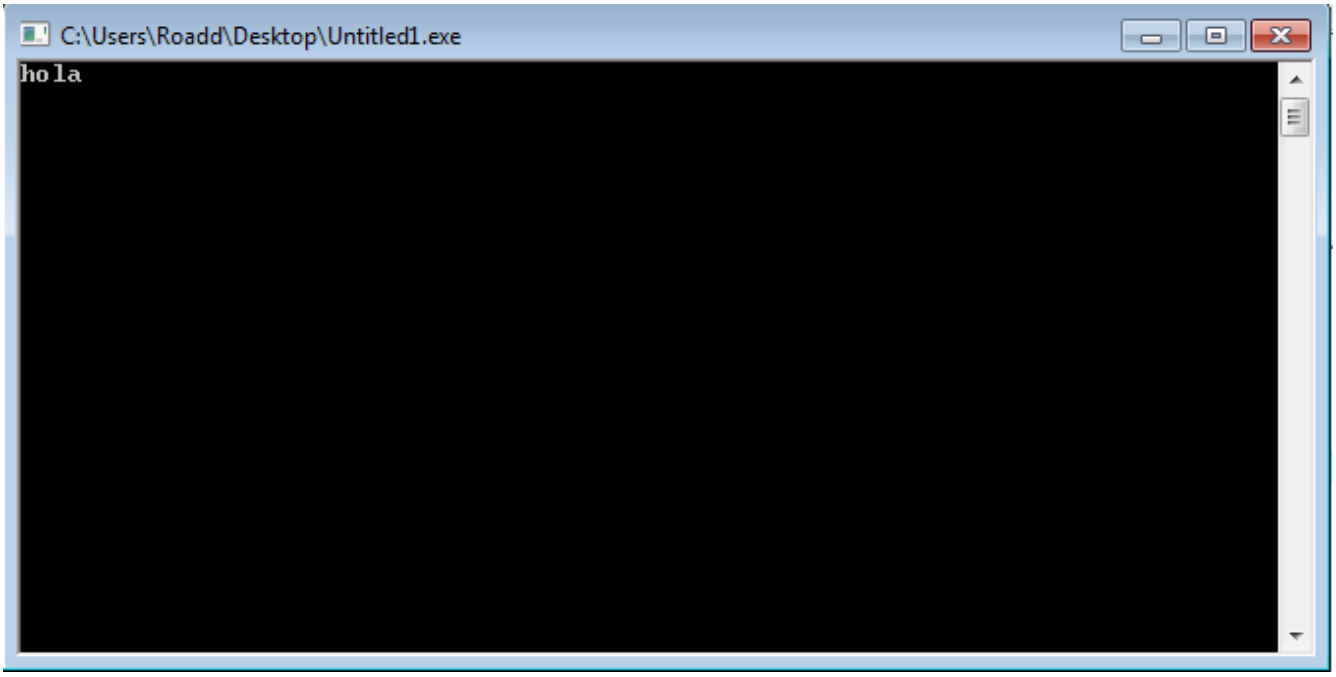


Compilar es **traducir** del lenguaje C al lenguaje de máquina para que una computadora pueda

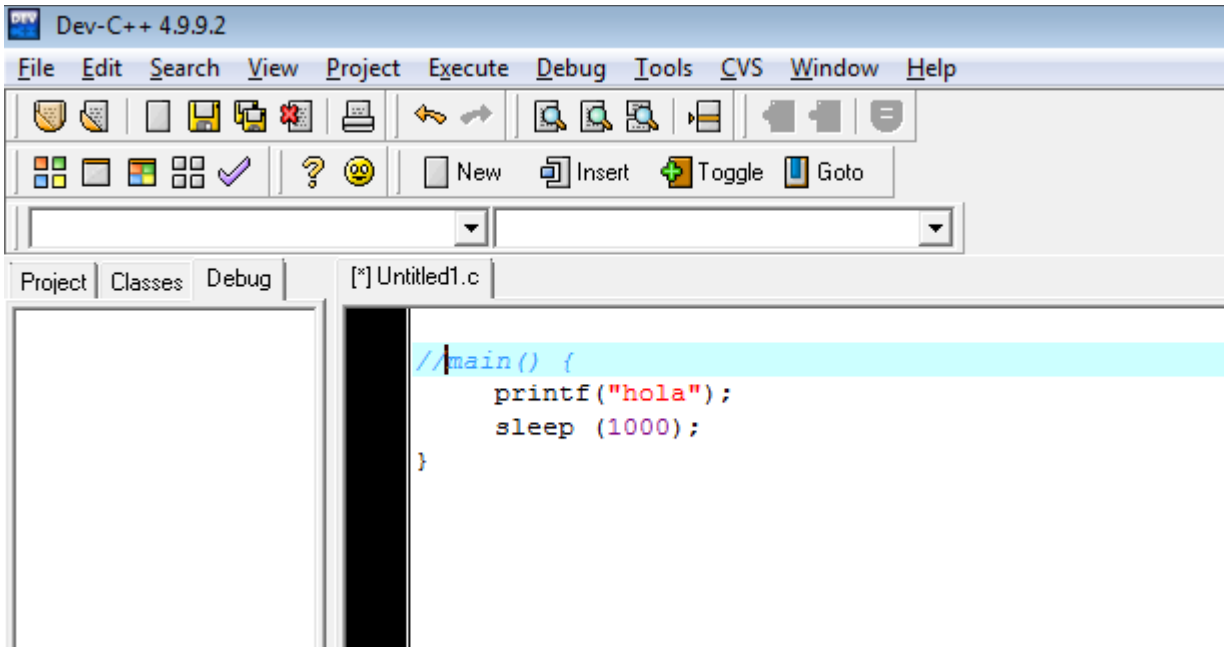
interpretar todo. Es decir que se generaría el deseado **.exe** y lo podríamos **ejecutar**. Si no tira ningún error, éso es exactamente lo que sucederá. Claro que **todavía** no hace nada, la funcion main sólo sirve para indicarle al compilador **dónde** debe empezar a ejecutar.
Si al compilar hubiesen errores, no compilaría.
Vamos a decirle que imprima algo en la pantalla:

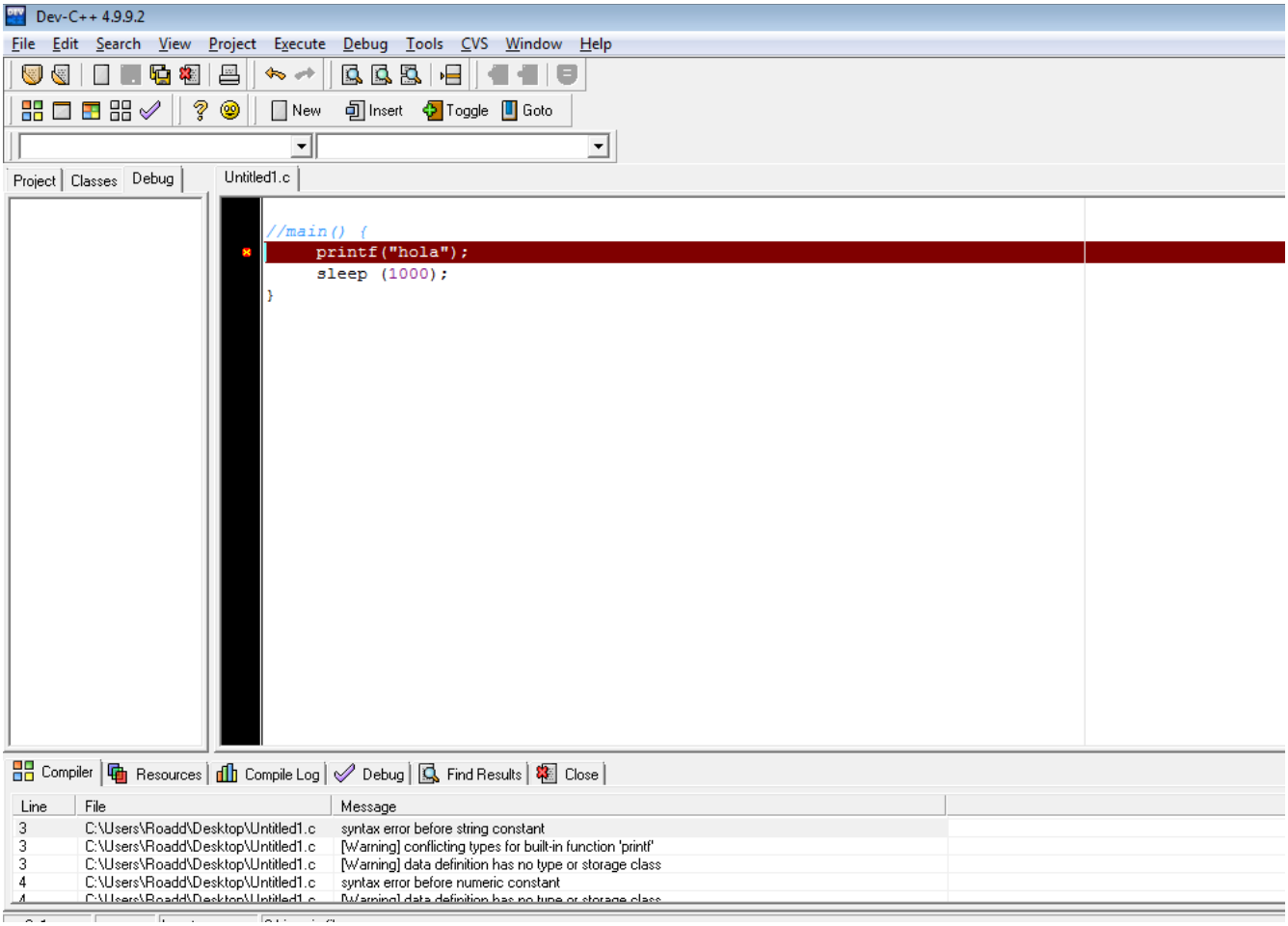


Bueno, acá tenemos 2 cosas nuevas. Tenemos la función "**printf("")** ". La función que acabo de mencionar sirve para **mostrar en pantalla**, lo que nosotros queramos. En nuestro caso queremos mostrar un **texto**, y lo haremos con **comillas**. Al **final** de los **comandos**, (como lo es el printf) debemos colocar un **";"** para darle a entender que es el **fin de línea**.
También está la función "**sleep()**", que hace esperar al programa, el tiempo en **milisegundos** que nosotros queramos. Yo puse **1000 milisegundos**, por lo que es igual a **un segundo**.
Compilemos y demos **run**, para mostrar el texto.



Ahora **¿Qué pasaría si me olvido de poner la librería pero llamo a la función?**
Borremos la línea del main, para ver el error. Pero de una manera distinta. Vamos a "**comentar**" la linea. Es decir, le vamos a decir al programa que esa línea no la ejecute y que es una **anotación** o comentario para el programador. Ésto lo hacemos con **dos barras** (//) al comienzo de la línea. Verán que hasta toma otro color. Ahora compilemos y demos **run**.





Vemos que nos da el **error**. Dev-C++ nos da algunas ayudas como por ejemplo, en dónde tuvo error al intentar compilar y por qué creé que no pudo compilar. Si siguieron el ejemplo, vean que el error da en la línea de abajo del main. ¿Por qué? Porque esa línea está **comentada** y no la toma. Quiere buscar la función main() y no la encuentra, y ahí es donde encuentra el error.

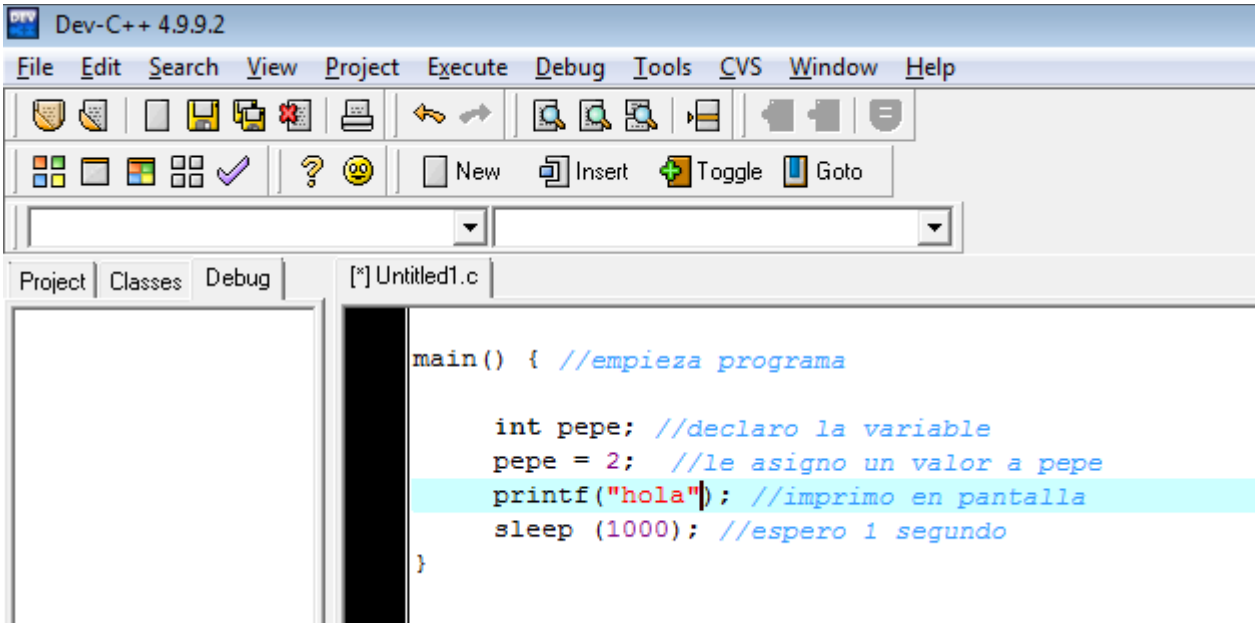
Avanzemos con la teoría. ¿Se acuérdan cuando, en el pseudocódigo, debimos **guardar** el **valor** que ingresaba el usuario o el número generado para usarlo después? Bueno, obviamente pasa lo mismo en todos los códigos. Los números que queremos guardar, debemos darle un nombre de guardado. Ésto es, en orden, primero decirle al programa, que reserve parte de la memoria para guardarlo con un nombre específico y luego darle un valor para poder interactuar con éstos. Los espacios de memoria reservados con un nombre específico, los llamamos "**variables**", porque su valor puede cambiar a través del tiempo. En programación se le llama al primer paso "**declarar variables**", porque le decimos al programa **cuánta memoria debe reservar**.

Claro que nosotros no siempre vamos a pensar en cuántos bytes debemos reservar, sino en lo que **necesitamos** para trabajar. Principalmente porque las variables tienen **tipos** y ya están **predefinidos** por el código (aunque más adelante veremos cómo podemos definir los nuestros, casi siempre es mejor usar las que nos da el programa). Para ésto no debemos incluir ninguna librería a parte. Los tipos son:

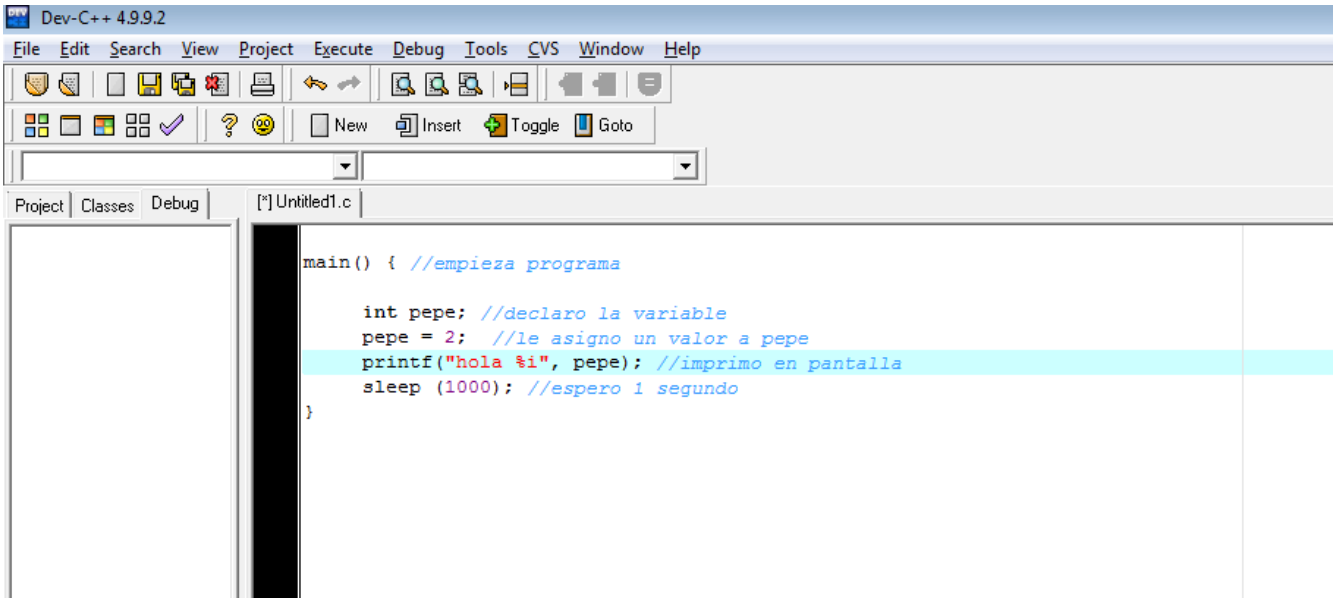
Tipo	Descripción	Bits	Rango de valores	Alias
SByte	*Bytes* con signo	8	[-128, 127]	sbyte
Byte	*Bytes* sin signo	8	[0, 255]	byte
Int16	Enteros cortos con signo	16	[-32.768, 32.767]	short
UInt16	Enteros cortos sin signo	16	[0, 65.535]	ushort
Int32	Enteros normales	32	[-2.147.483.648, 2.147.483.647]	int
UInt32	Enteros normales sin signo	32	[0, 4.294.967.295]	uint
Int64	Enteros largos	64	[-9.223.372.036.854.775.808, 9.223.372.036.854.775.807]	long
UInt64	Enteros largos sin signo	64	[0-18.446.744.073.709.551.615]	ulong
Single	Reales con 7 dígitos de precisión	32	$[1,5\times10^{-45} - 3,4\times10^{38}]$	float
Double	Reales de 15-16 dígitos de precisión	64	$[5,0\times10^{-324} - 1,7\times10^{308}]$	double
Decimal	Reales de 28-29 dígitos de precisión	128	$[1,0\times10^{-28} - 7,9\times10^{28}]$	decimal
Boolean	Valores lógicos	32	**true, false**	bool
Char	Caracteres Unicode	16	['\u0000', '\uFFFF']	char
String	Cadenas de caracteres	Variable	El permitido por la memoria	string

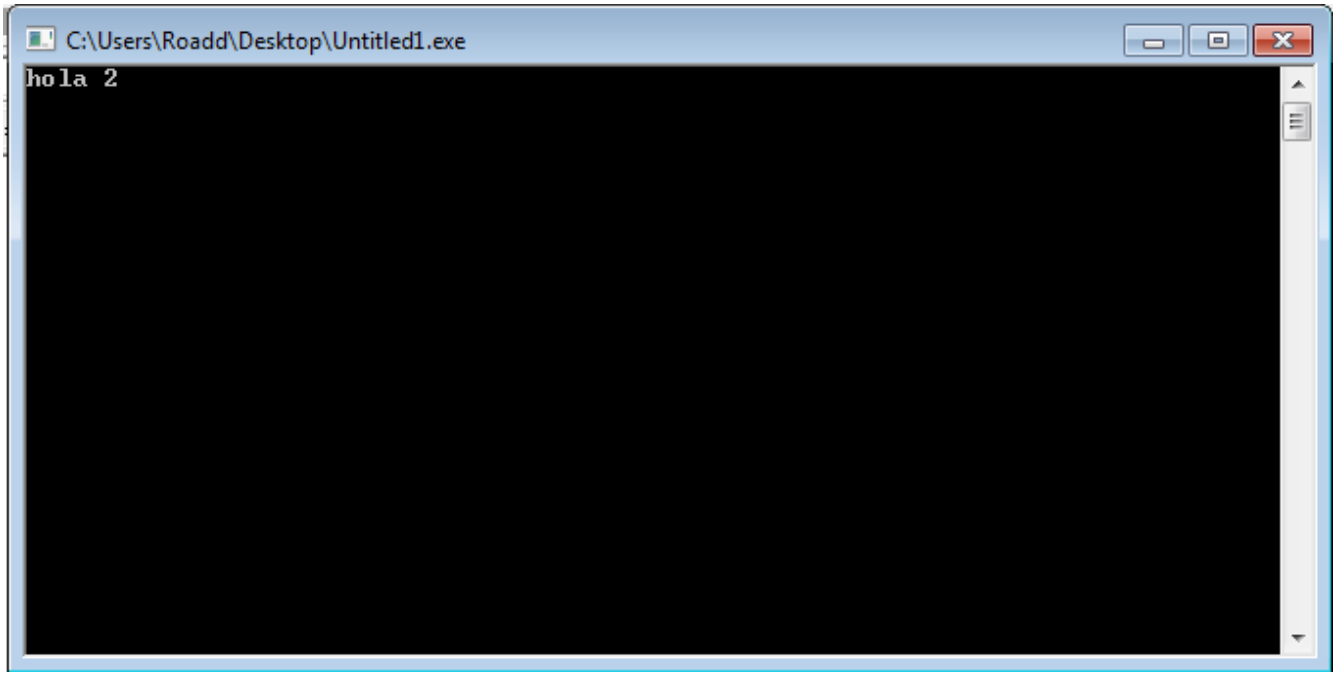
Veamos que en las columnas tenemos los **alias** que corresponden a los tipos de variables que podemos usar para declarar.

Para declarar una variable en el programa, debemos primero decir cuál es el tipo de variable y luego asignarle un nombre. Terminamos la línea con **";"**. Abajo le damos un valor diciendo **"alias variable = valor".**



Pero claro, en el printf queremos mostrar el valor de una variable sin importar cuál es el valor. Ésto lo hacemos poniendo directamente la variable. Debemos poner "**%i**" dentro del texto (esto quiere decir que va a haber un entero que se mostrará en pantalla), y luego de las comillas una coma para poner el nombre del "**int**" que se imprimirá en pantalla (así son las reglas del código:D). **Compilemos y hagamos run** :D.





Pondrá todo como si estuviera unido. Para el usuario que ve ésto, será **transparente.**

Bueno hasta aquí con C. Más adelante veremos más de ésto. Les aviso que no va a entrar en el examen la parte de práctica de código pero sí la teoría. :D Bueno, me despido por hoy.

Cualquier cosa pueden mandarme mail a: r0add@hotmail.com

Para donaciones, pueden hacerlo en bitcoin en la dirección siguiente:

1HqpPJbbWJ9H2hAZTmpXnVuoLKkP7RFSvw

Roadd.
