#1

UNDERCODE

TALLER DE PROGRAMACION C

TEMAS

- INTRODUCCIÓN
- IDES
- INSTALACIÓN
- PROCESO DE COMPILACIÓN
- EJEMPLOS
- VARIABLES
- OPERADORES

TUTOR

AURORA

Introducción

"C es un lenguaje de programación creado en 1972 por Dennis M. Ritchie en los Laboratorios Bell como evolución del anterior lenguaje B, a su vez basado en BCPL.

Al igual que B, es un lenguaje orientado a la implementación de Sistemas Operativos, concretamente Unix. C es apreciado por la eficiencia del código que produce y es el lenguaje de programación más popular para crear software de sistemas, aunque también se utiliza para crear aplicaciones.

En parte a causa de ser de relativamente bajo nivel y de tener un modesto conjunto de características, se pueden desarrollar compiladores de C fácilmente. En consecuencia, el lenguaje C está disponible en un amplio abanico de plataformas (más que cualquier otro lenguaje). Además, a pesar de su naturaleza de bajo nivel, el lenguaje se desarrolló para incentivar la programación independiente de la máquina. Un programa escrito cumpliendo los estándares e intentando que sea portátil puede compilarse en muchos computadores.

Se trata de un lenguaje fuertemente tipificado de medio nivel pero con muchas características de bajo nivel. Dispone de las estructuras típicas de los lenguajes de alto nivel pero, a su vez, dispone de construcciones del lenguaje que permiten un control a muy bajo nivel. Los compiladores suelen ofrecer extensiones al lenguaje que posibilitan mezclar código en ensamblador con código C o acceder directamente a memoria o dispositivos periféricos." Wikipedia 2013

Una vez finalizado por la excelente descripción de C pasemos a lo que sería en si el taller.

IDEs

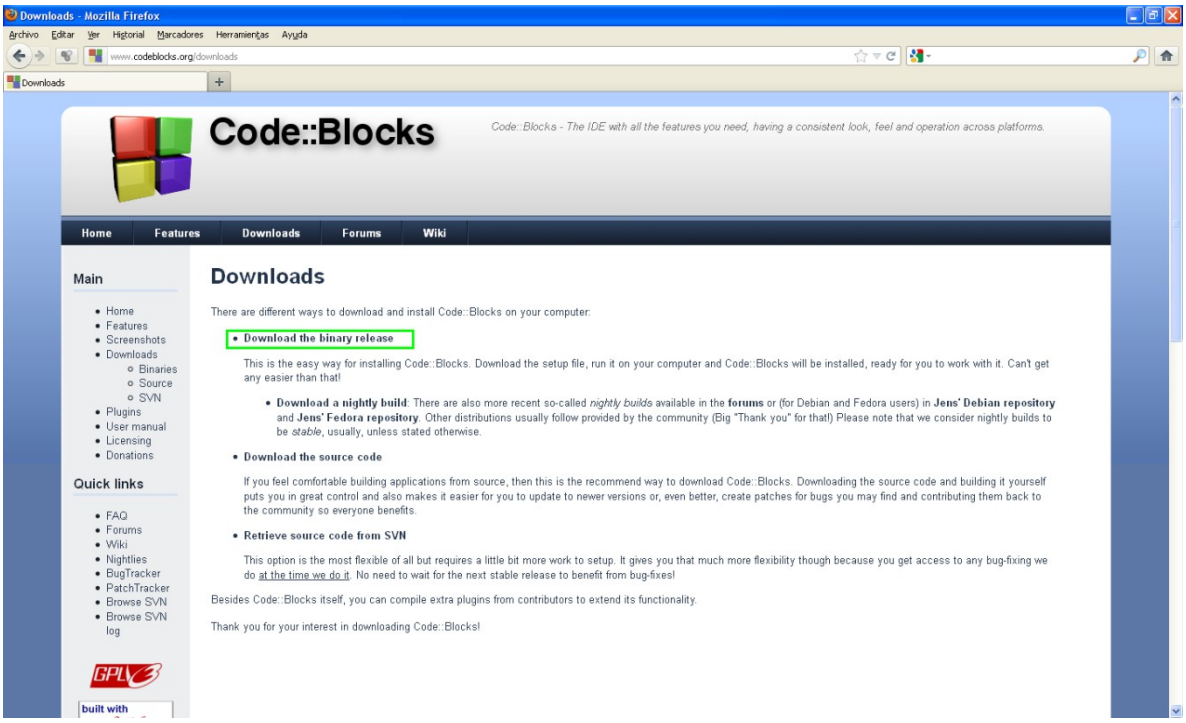
Empecemos con los IDEs, como saben un IDE es un software que posee varios componentes que nos permiten desarrollar aplicaciones de diferentes tipos, entre las herramientas que poseen un IDE están el editor, compilador, depurador y una interfaz gráfica. Entonces para poder desarrollar nuestras aplicaciones en C necesitamos seleccionar un IDE adecuado,

Existen diferentes IDEs para desarrollar en C, entre estos esta Eclipse, Visual Studio, Code::Blocks,Dev-C++ pero vamos a usar Code::Blocks, que tiene muy buenas críticas por los que programadores en C, considero que es mejor que Dev C++, también esta Eclipse Helios, personalmente he usado Eclipse pero no para programar en C, una de las

principales características que también es destacable de la mayoría de estas IDEs es: que son Software libre.

Empecemos obteniendo el IDE.

http://www.codeblocks.org/downloads



Descargamos la “relase binary”, descargan el instalador de acuerdo al SO que tengan instalado. Se necesitan 60 MB de espacio en disco para instalar.

Pasos de instalación

- Doble click en el instalador. Nos aparecerá la ventana de inicio de instalación.
- Click en el botón “Next”.
- Click en el botón “I Agree”, la próxima ventana será para seleccionar los componentes que deseamos, lo dejamos tal cual, damos click al botón “Next”.

- Nos pedirá el path de donde queremos instalar Code:Blocks que por defecto será “C:\Program Files (x86)\CodeBlocks”, si lo dejan así está bien, ustedes pueden elegir otro también si lo desean. Y después presionan en el botón “Install”.

La primera vez que deseen usar el programa les preguntara por el compilador que desean utilizar, seleccionan el que deseen si antes usaron alguno en particular caso contrario elegir GNU GCC Compiler.

Para desarrollar en C, necesitamos un compilador y es necesario entender el proceso para poder avanzar con el primer ejemplo, antes de seguir avanzando vamos a repasar el proceso de compilación.

Proceso de Compilación:

- Preprocesado consistente en modificar el código fuente en C según una serie de instrucciones (denominadas directivas de preprocesado) simplificando de esta forma el trabajo del compilador. Por ejemplo, una de las acciones más importantes es la modificación de las inclusiones (#include) por las declaraciones reales existentes en el archivo indicado.
- Compilación que genera el código objeto a partir del código ya preprocesado.
- Enlazado que une los códigos objeto de los distintos módulos y bibliotecas externas (como las bibliotecas del sistema) para generar el programa ejecutable final.

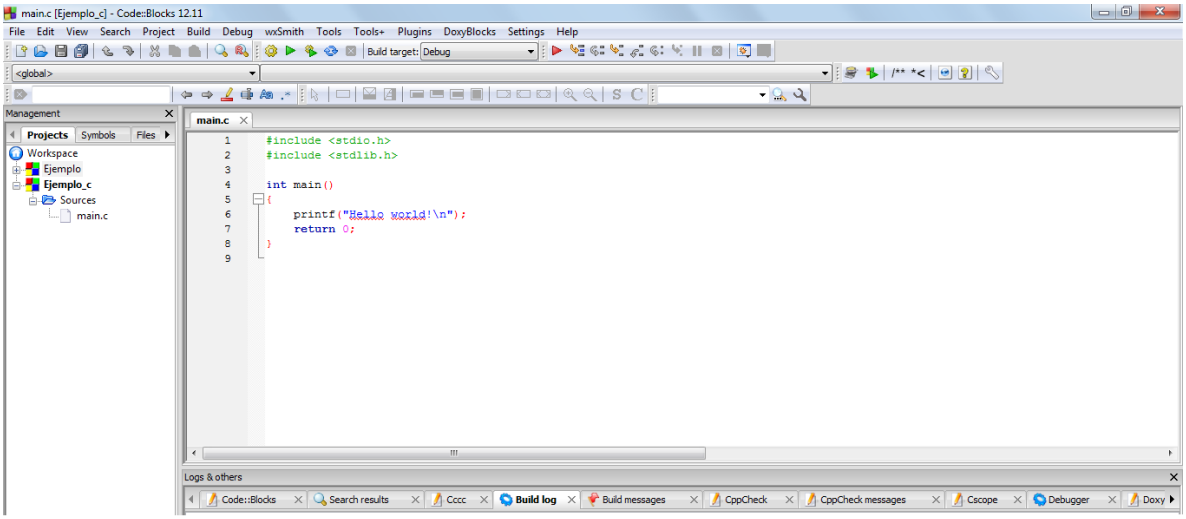
Primer Ejemplo en C.

Pasos

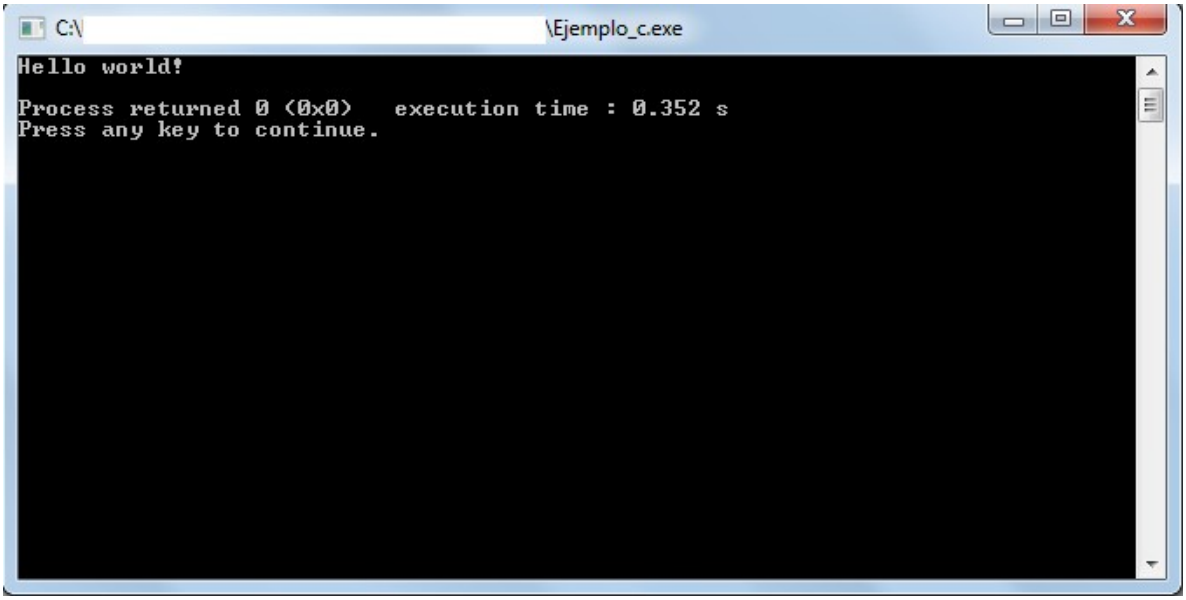
- Abrimos Code::Blocks
- Creamos un proyecto nuevo de consola “Console Application”, hacemos Click en Go->Next. No pedirá que seleccionemos C o C++, seleccionamos C.

- Después de lo anterior completamos con el nombre del proyecto. Next ->Finish

- Una vez creado el proyecto abrimos la carpeta src en el tab de Projects, podemos ver que ya tenemos el archivo main.c, con un típico “HelloWorld”



Si queremos ejecutarlo  usando este botón, nos mostrara un pop up con un mensaje de que necesita compilar primero para así obtener el ejecutable y luego poder ejecutar. Por eso primero  compilamos. Ahora si podemos ejecutarlo. Y nos mostrara una ventana con nuestro primer ejemplo.



Pueden ver mucha más información en las tabs que están en la parte inferior de la herramienta. Como por ej. Build log.

Veamos que tenemos en estas líneas de código:

```
1. #include <stdio.h>
2. #include <stdlib.h>
3.
4. int main()
5. {
6.     printf("Hello world!\n");
7.     return 0;
8. }
```

Nota: si llegan a tener un error relacionado con "make" cuando quieran compilar su programa, debe revisar si tienen instalado: MinGW - Minimalist GNU for Windows, caso contrario, pueden descargarlo de acá

"MinGW (Minimalist GNU for Windows), es una implementación de los compiladores GCC para la plataforma Win32, que permite migrar la capacidad de este compilador en entornos Windows."

Las primeras son las librerías stdio.h y stdlib.h que necesitamos para poder usar las funciones como **printf**, las librerías son un conjunto de funciones que contienen datos, código funcional que podemos usar en nuestro programa en este caso usamos la función **printf** para imprimir por pantalla. Notar que las librerías tienen *.h* al final, esto las destaca como librerías. Estas son librerías estándar, también se pueden crear librerías propias que veremos más adelante. Antes de usar cualquier función que se encuentra en alguna librería, se debe incluir la librería antes de hacer una llamada a la función, por eso se ponen al principio.

Después tenemos la función **main**; **main** es la primera función que se llama o más formalmente se invoca cuando se ejecuta un programa, nuestros programas en C empiezan y terminan en **main** incluso cuando tengamos un programa que posea varios módulos que estén fuera de esta función, o que se llamen fuera de esta función pero que de alguna forma u otra, alguna otra función que está en el **main** los invocara.

Como se puede ver **main** esta ante puesta por un tipo de devolución **int**; **int** es un tipo de dato, que por la ubicación indica que al final de la función debemos retornar un valor entero, esto es lo que hace la sentencia **return** que se utiliza para finalizar una función, esta sentencia se puede usar dentro de una función muchas veces por ejemplo cuando para diferentes valores ingresados por el usuario, necesitamos devolver diferente valores.

La función printf nos permite mostrar por pantalla, texto o valores de alguna operación, números. Además en la función **printf** también están los caracteres "**\n**" estos caracteres indica una nueva línea, también están otros como "**\t**". Esta función devuelve un numero entero positivo si se imprimieron los caracteres por pantalla, si llega a devolver un valor negativo es que hubo algún error.

Repasemos un poco las variables antes de seguir avanzando.

Variables

Tenemos diferentes tipos de variables con los que podemos trabajar en C, algunos son:

- Entero: tipo de dato entero pueden ser short, int, long y longlong; además de con signo o sin signo.
- Caracter: char;
- Flotantes: float y double;

Algunos ejemplos:

```
1. #include <stdio.h>
2. #include <stdlib.h>
3.
4. Int main()
5. {
6.    printf(" Empezando con ejemplos \n");
7.     int entero = 10;
8.     float flotante = 1.5;
9.     charcaracter = 'h';
10.     char buffer[20] = "en el taller";
11.     /*Para poder mostrar las variables usando la función
   printf, tenemos que especificar de que tipo es*/
12.     printf(" %s \n",buffer);
13.     printf(" Flotante y Entero: %d , %f\n",entero,flotante);
14.     printf(" Caracter: %c \n",caracter);
15.     printf(" Un caracter: %c \n",buffer[6]);
16.     return 0;
17. }
```

Para ejecutar este ejemplo compilamos y ejecutamos.

Veremos que en el ejemplo anterior hay algunas cosas extras de las que aun no hablamos, Primero los comentarios, en podemos dejar comentarios usando:

```
/*
* El texto que se encuentra aquí es un comentario y no será tomado como
tal por el
* compilador, este tipo de comentario puede contener varias líneas
*/

// Esta es otra forma de dejar comentario pero solo de una línea
```

Otra cuestión a notar es que en la función **printf** debemos que especificar de qué tipo es el valor que queremos mostrar por pantalla, indicando el formato esto se realiza usando el carácter de escape **%** más alguna letra que indica el tipo, a continuación veremos los más comunes:

Formato	Tipo
%d	Numero entero
%s	Cadena de caracteres
%c	Un carácter
%f	Nº flotante

Los operadores

Ahora vamos a ver algunos operadores que podemos usar:

- <
- >
- +
- -
- *
- /

Ejemplo:

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

int main()
{
  int numero = 10,numero_1; //Los enteros pueden ser sin o con signo, cuando no se especifica se considera con signo
  int sum;
  printf( "\n Ingrese un numero mayor a : %d \n\n", numero);
  scanf( "%d", &numero_1); // Esta función se utiliza para leer los datos ingresaros por el teclado, en este caso también se debe especificar de que tipo es el dato a ingresa, & indica el lugar de memoria donde se guardara el valor ingresado.
   sum = numero + numero_1;
  printf("\n La suma es %d\n", sum);

  return 0;
 }
```