APRENDERAPROGRAMAR.COM

EJEMPLOS LENGUAJE C. DECLARAR VARIABLES Y ASIGNARLES CONTENIDO. PROGRAMA BÁSICO INT MAIN MOSTRAR MENSAJE. (CU00511F)

Sección: Cursos
Categoría: Curso básico de programación en lenguaje C desde cero

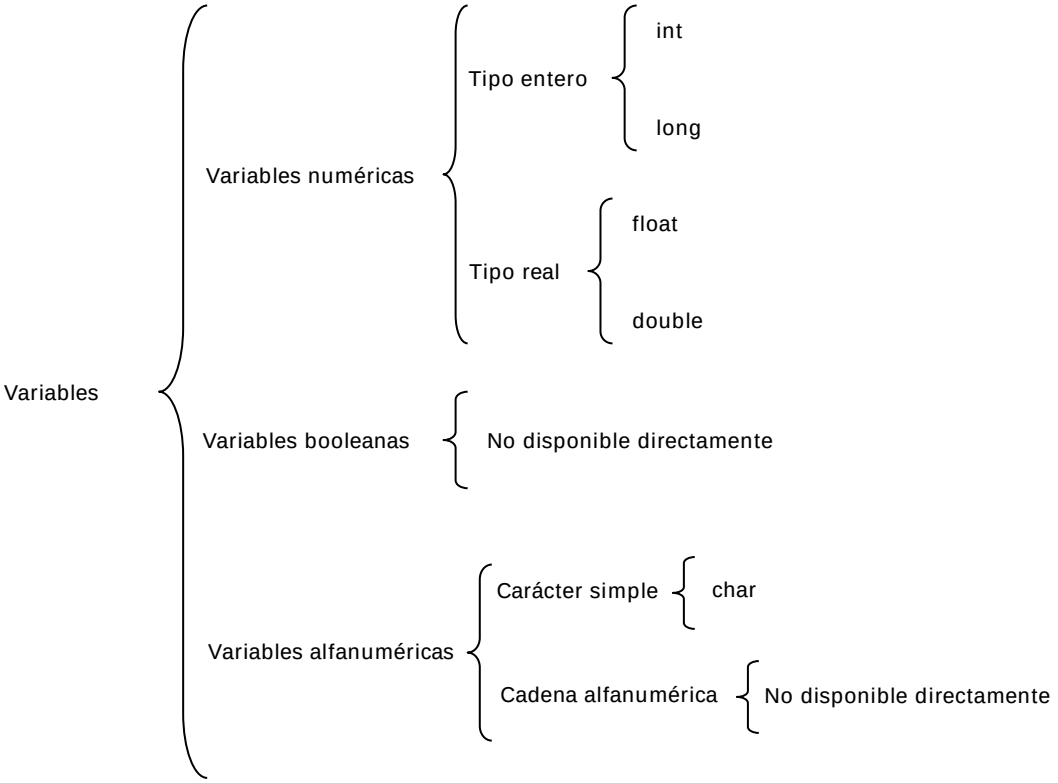
Fecha revisión: 2031

Resumen: Entrega nº11 del curso básico "Programación C desde cero".

Autor: Mario Rodríguez Rancel

ASIGNAR CONTENIDO A VARIABLES

En el curso Bases de la programación nivel I de aprenderaprogramar.com se indican unas normas y pautas básicas para asignar contenido a variables. Las normas para asignar contenido a variables en C son las mismas que se aplican a pseudocódigo, teniendo en cuenta estas equivalencias.



Ejemplos de asignación de contenidos son estos:

Declaración	Ejemplo asignación contenidos
int A;	A = 123;
float A;	A = - 3323.57;
char A;	A = 'p';
char A;	A = '1';
int salario;	salario = 30500;
float salario;	salario = 30500;

int A, B, suma;	A = 5 + 2; B = 32; suma = A + B; [suma valdrá 39]
int A, B, C, D, E;	A = 5; B = 32; C = A * B; [C toma el valor 160] D = A + C; [D toma el valor 165] E = D; [E toma el valor 165]
int agotamiento;	agotamiento = 0; [Usamos un entero para simular un booleano]

Un programa que conste de:

```
int A;
A = 7 * B;
```

```
int A, B;
A = 7 * B;
```

dará lugar a un error o resultado incierto debido a que *B* no está declarada en el primer caso, o a que la variable B no está inicializada en el segundo (no tiene un valor definido y el compilador no sabe qué valor asignarle).

En cambio:

```
int A, B=0;
A = 7 * B;
```

Supone que *A* valga cero, ya que B tiene asignado valor cero y *7 * B* equivale en este caso a *7 * 0*.

Vamos a crear un programa que declare una variable tipo *int* llamada *edad* y nos muestre un texto en pantalla. Para ello escribiremos el siguiente código:

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

int main() {
int edad;
edad=54;
printf ("La edad es %d años\n", edad);
printf ("Gracias por utilizar este programa del curso aprenderaprogramar.com");
return 0;
}
```

Explicaremos ahora las partes de las que consta el programa anterior y el resultado de su ejecución. En primer lugar, analicemos las líneas que componen el programa:

1. *#include <stdio.h>*
2. *#include <stdlib.h>*

Estas dos primeras líneas se ocupan de decirle al compilador que vamos a usar ciertas instrucciones cuyo funcionamiento está definido en archivos externos a nuestro programa. Estos archivos externos son studio.h y stdlib.h. La palabra clave #include se denomina **directiva del preprocesador** y los archivos stdio.h y stdlib.h se denominan **archivos de cabecera** del programa. En un programa podemos tener uno, dos ó más archivos de cabecera. En nuestro caso estamos incluyendo dos porque son los que incluye Code::Blocks cuando genera el programa básico de ejemplo.

Dichos archivos se encuentran en tu ordenador, por ejemplo el archivo stdio.h se encontrará en una ruta que será similar a C:\MinGW\lib\gcc\mingw32\4.7.2\include\ssp\stdio.h.

Al incluir estas líneas, estamos declarando que vamos a utilizar (o al menos que queremos tener a nuestra disposición) aquellas funciones del lenguaje que están disponibles en estos archivos externos. Prueba a eliminar estas línea y trata de ejecutar el programa. Obtendrás un mensaje de error. ¿Por qué? Porque estás tratando de usar alguna instrucción (como *printf*) que no está disponible porque no has incorporado los *include* que permiten hacer uso de ella.

¿Por qué se obliga en C a añadir archivos de cabecera incluso para funciones básicas del lenguaje? Posiblemente porque cuando se creó C se hizo teniendo muy en cuenta la eficiencia y el evitar sobrecargar el computador ocupando memoria inútilmente. Imagínate que eres el cliente de un restaurante ("el cliente del compilador") y que vas a pedir un menú. El dueño del restaurante no prepara todos los platos posibles y los pone a tu disposición, sino que espera a que tú le digas los platos que deseas y son únicamente esos los que prepara. De esta forma ahorra espacio y esfuerzo. En nuestro caso, en lugar de cargar en memoria y poner a tu disposición todas las posibles funciones del lenguaje, el compilador sólo carga y pone a disposición aquellas funciones que declaramos que vamos a usar. De este modo se gana en eficiencia.

3. **int main () { ... }** Las instrucciones que se encuentran dentro de las llaves del int main () { … } se suelen denominar coloquialmente "programa principal" o "instrucciones de la función principal". Podríamos decir que un programa en C consta de al menos una función (la función principal), pero puede tener muchas más funciones como veremos más adelante.
4. **int edad;** Corresponde a la declaración de una variable de tipo entero (*int*) y nombre edad.
5. **edad = 54;** Asigna a la variable edad el valor numérico indicado.
4. **printf ("La edad es %d años\n", edad);** Indica al ordenador que proceda a mostrar el contenido de la variable edad por pantalla. Aquí aparecen algunos elementos "extraños" como %d y \n cuyo significado explicaremos más adelante.
5. **printf ("Gracias por utilizar este programa de aprenderaprogramar.com")***;* Muestra un mensaje por pantalla. El contenido del mensaje lo hemos escrito dentro de paréntesis y dentro de comillas.
6. **return 0;** Esta instrucción la veremos de momento como una formalidad. Hemos dicho que el código dentro de los corchetes del *int main* se denomina función principal y formalmente una función debe devolver algo, en este caso un número entero. Para cumplir con este requisito escribiremos *return 0;* aunque también podríamos escribir *return 1;* ó *return 321;* si quisiéramos, sin que en este caso el resultado del programa se viera afectado. La sentencia *return* indica que el programa ha finalizado.

Ejecuta el programa pulsando el icono "*Build and Run*" o eligiendo la opción del menú *Build*. También puedes hacerlo pulsando la tecla *F9* (atajo para compilar y ejecutar el programa).

El resultado de ejecutar este programa será similar a:

```
La edad es 54 a±os
Gracias por utilizar este programa
```

Si te fijas en vez de escribirse “años” se ha escrito “a±os” o algo similar donde no se muestra la letra eñe. Esto tiene que ver con el origen anglosajón del lenguaje y explicaremos más adelante cómo se puede solucionar para visualizar algunos caracteres especiales para C como la ñ.

Si te fijas en vez de escribirse “años” se ha escrito “a±os” o algo similar donde no se muestra la letra eñe. Esto tiene que ver con el origen anglosajón del lenguaje y explicaremos más adelante cómo se puede solucionar para visualizar algunos caracteres especiales para C como la ñ.

Para cerrar la ventana de consola donde se muestra la ejecución del programa haz click sobre el aspa de cierre de la ventana o pulsa una tecla.

Prueba a introducir otras variables de tipo numérico y a dar lugar a que se muestren sus contenidos sobre la ventana de consola. Juega con los nombres de las variables, sus contenidos y la asignación de contenidos. De momento juega con variables tipo int y valores enteros, y procede a visualizar un mensaje sobre estas variables por pantalla. Más adelante veremos cómo usar otros tipos de variables.

También puede resultar de interés que compruebes qué ocurre si declaras una variable de un tipo y le asignas contenido de otro tipo.

Por ejemplo declara int salario; y asígnale el contenido salario = 655.34 y prueba a mostrarlo en pantalla. Ten en cuenta que en general en programación no se considera equivalente 655,34 (con coma como separador decimal) con 655.34 (con punto como separador decimal). Por tanto habremos de prestar atención al símbolo que usemos como separador decimal cuando debamos usarlo.

¿Qué ocurre cuando incumplimos las previsiones del compilador para asignar contenidos a variables (por ejemplo, para una variable A tipo int definir A = 53200000000, que está fuera del rango previsto)? No vamos a analizar los distintos casos que se pueden presentar, sino a tratar de dar una respuesta genérica. Cuando hacemos algo no esperado, como asignar un valor fuera de rango, asignar un valor que no concuerda con el tipo de la variable, sumar variables numéricas con alfanuméricas, asignar decimales a un número entero..., etc. pueden suceder varias cosas:

a) Salta un error. Un mensaje impide que el programa comience a ejecutarse, o bien el programa se detiene mientras se está ejecutando.

b) El programa se ejecuta pero los resultados no son los deseados.

c) El programa se ejecuta y los resultados son los deseados.

En resumen, es difícil prever lo que va a suceder, aunque se pueden estudiar y manejar las circunstancias. Por ejemplo una variable que se declare como tipo int pero a la que se asigna un contenido numérico real con dos decimales no da lugar a un error, pero sí a una pérdida de información derivada de que el número decimal se va a truncar a un entero. Las consecuencias de esta circunstancia habría que valorarlas para cada programa y circunstancias concretas.

Como programadores hemos de buscar programas “100% predecibles”, en los que no se pueda producir que "sea difícil prever lo que va a suceder". Por tanto intentaremos que la declaración y asignación de contenidos a variables se ajuste a las normas previstas. En última instancia, podremos incorporar instrucciones para tratar casos imprevistos.

EJERCICIO

Crea un programa en C que paso a paso contenga lo siguiente:

a) Los include que ya conocemos.

b) El int main de la misma forma que hemos visto.

c) Declarar una variable de tipo decimal doble denominada precio e inicializarla con un valor de 100.

d) Declarar una variable de tipo decimal doble e inicializarla con un valor del 4 por ciento (0.04).

e) Declarar una variable de nombre precioConImpuestos y asignarle como valor el resultado de multiplicar el contenido de la variable precio por la variable impuesto.

f) Mostrar un mensaje por pantalla que diga cuál es el precio con impuestos, utilizando la sintaxis que hemos visto.

¿Cuál es el resultado que se obtiene si en lugar de precio 100 usamos un valor de precio igual a 58.34?

Para comprobar si tus respuestas son correctas puedes consultar en los foros aprenderaprogramar.com.

Próxima entrega: CU00512F

Acceso al curso completo en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente:
http://www.aprenderaprogramar.com/index.php?option=com_content&view=category&id=82&Itemid=210