Variables, constantes, identificadores y tipos de datos

Javier Fernández Rivera - www.aurea.es

Las Variables en C

A la hora de elaborar un programa es necesario usar unos datos, y para ello es imprescindible en muchos casos grabar esos datos en memoria para operar con ellos posteriormente. Recordemos que no es lo mismo grabar los datos en memoria que grabarlos en el disco duro. Cuando decimos grabar en memoria dentro de un lenguaje de programación nos estaremos refiriendo a grabar esos datos en la RAM.
Ahora bien, para grabar esos datos en la RAM podemos hacerlo racionalmente con dos elementos, llamados: variables y constantes.
Los dos elementos funcionan como fuentes de almacenamiento de datos, la gran diferencia entre los dos es que en el caso de las constantes su valor dado no varia en el transcurso de todo el programa o código.
Tanto las variables como las constantes podría decirse que son direcciones de memoria con un valor, ya sea un numero, una letra, o valor nulo.

Variables: Elementos de almacenamiento de datos o direcciones de memoria, que pueden variar en el desarrollo o proceso del programa.

Tanto las variables como las constantes las utilizamos a la hora de programar para almacenar en ellas unos datos determinados y poder nombrarlas en cualquier parte de nuestro código-programa para que nos devuelvan esos datos anteriormente introducidos.

Definición de una variable

Definir una variable es indicar el tipo de dato que va a contener y almacenar, y en función de esto reservar una determinada dirección de memoria o palabra de memoria (es la cantidad de bytes que ocupa un tipo de dato). Con lo que si queremos que una variable contenga un dato numérico deberemos de definirla de distinta forma que si queremos que contenga una cadena de caracteres. Cuando definimos una variable no le estamos introduciendo ningún valor, sino que únicamente estamos indicando que tipo de valor va almacenar posteriormente. Es obligatorio que siempre definamos una variable antes de introducir en ella un valor, y esto es así porque es necesario que se reserve en memoria un espacio limitado mas o menos grande dependiendo del tipo de dato que vaya a albergar dicha variable.

Para definir una variable

La sintaxis a seguir es: **[cualificador] <tipo> <ID>**

En manuales o tutoriales de informática lo que se encuentra entre corchetes es opcional y el resto es obligatorio.
Ante esa sintaxis, decimos que donde cualificador podemos poner opcionalmente uno de los disponibles que se encargan de variar o alterar el tipo de dato (será explicado a continuación), posteriormente ira el tipo de dato, ya sea entero, decimal, etc. Y por ultimo el identificador o nombre de la variable.

Tipos de datos

En cualquier lenguaje de programación es muy importante saber manejar los distintos tipos de datos que te permite almacenar y trabajar con ellos. Como ya vimos anteriormente las variables almacenan datos. Pero a la hora de definirlas hay que indicarles que tipo de dato van a almacenar. No es lo mismo que una variable almacene el numero “5000”, a que almacene la letra “C”. Existen 4 distintos tipos de datos en C, cada uno de los cuales reserva una zona mayor o menor en memoria (según sus necesidades), y esta

preparada para almacenar un tipo de dato en concreto. En otro lenguaje como el scriptting las variables se definen y una vez definidas puedes almacenar cualquier tipo de dato sin antes hacer alusión a el. Esto no pasa en C ni en la mayoría de lenguajes grandes. En C su potencialidad obliga a ello.

Existen 4 tipos de datos en C:

1) **int** Almacenan números enteros (sin decimales). Este tipo limita el rango numérico de 32767 a 32767. Este tipo reserva en memoria 16 bits o lo que es lo mismo 2 bytes.
2) **char** Almacena un carácter o caracteres, en realidad lo que hace es almacenar un numero del 0 al 255 que son los números que identifican a un carácter que corresponde a la tabla ascii. Este tipo ocupa en memoria 1 byte.
3) **float** Tipo para definir que una variable almacenara números decimales. Su rango numerico es de 3,4E-38 a 3,4E38. Su espacio en memoria es de: 4 bytes.
4) **double** Igual que la anterior pero es capaz de almacenar mayores cadenas numéricas. Rango de 1,7E-307 a 1,7E308. Su espacio en memoria es de 8 bytes.

Algunos tipos admiten unos prefijos a los cuales se les suelen llamar **cualificadores,** estos son: short (corto), long(largo), signed(con signo + o -), unsigned(sin signo).

Estos cualificadores se colocan al principio del tipo y lo cualifican, ejemplos:

Short int identificador: esto definirá a una variable de tipo entero y corto.
Unsigned float identificador: esto definirá una variable de coma flotante o sea decimal y sin signo.

Identificadores

Los identificadores son las palabras con las que identificamos o llamamos a una variable o constante. A la hora de dar nombre a una variable, o sea a la hora de identificar una variable debemos de tener en cuenta unas reglas de nomenclatura, marcadas por el lenguaje de programación en C.

Reglas de nomenclatura para los identificadores

Los nombres de los identificadores pueden llevar caracteres alfa-numéricos. Letras de la A a la Z excluyendo la ñ que no vale ☹. Y todos los números. Si es obligatorio que el nombre del identificador empiece por una letra y no un numero u otro signo.
Los identificadores solo aceptan el signo _ (guión bajo-subrayado), por tanto quedan excluidos otros como el signo de interrogación, los paréntesis, etc.
Tampoco podremos usar como identificadores las palabras reservadas de un lenguaje, palabras tales como: if, else, goto, etc.
El numero de caracteres de un identificador nunca debe sobrepasar los 31. Algunos compiladores solo reconocen los 8 primeros caracteres de los identificadores. Esto ya depende del potencial del compilador.
Recordemos que en C se distinguen entre mayúsculas y minúsculas con lo cual es distinto un identificador tal como, Suma que otro como suma. (Por convenio se ha decidido que los identificadores estén escritos en minúsculas.)

Lógicamente a la hora de dar nombre a una variable o constante, utilizaremos identificadores que nos hagan referencia o semejanza al dato que almacenan. Con lo que si necesitamos una variable para almacenar la cadena de caracteres: Hola, seria conveniente llamar o identificar esta variable como saludo y no como x. Los nombres de variables y constantes o sea los identificadores es adecuado que si su valor almacenado tiene relación unos con otros, los nombres de los identificadores también tengan esa relación (de forma léxica). Con todo esto lo que quiero decir es que nunca escatimemos en el tiempo de búsqueda para dar nombre a una variable o constante. Si le damos un nombre cualquiera luego al repasar el código todo nos resultara mas confuso.

Palabras reservadas

Hay ciertos indicadores, identificadores o palabras llamadas palabras reservadas (valga la redundancia) que no se pueden usar como identificadores, puesto que ya son usadas por el propio lenguaje para una finalidad determinada.
Estas palabras son por lo general en ingles ☹. Lo bueno es que no hace falta tener mucho nivel de ingles para comprenderlas ☺.

Algunas de estas palabras son: auto, break, case, char, const, continue, default, do, double, else, enum, extern, flota, for, goto, if, int, long, register, return, short, signed, sizeof, static, struct, switch, typedef, union, unsigned, void, volatile, while.

Definiendo variables

Una vez vista toda la teoría apliquémoslo a la practica

1) **int suma**
2) **long float resultado**
3) **char letra**

1: En el primer caso estamos definiendo una variable de tipo entero (int). O sea una variable que almacenara un dato numérico entero, sin decimales. Dicha variable es identificada por el nombre suma (su identificador).
2: En el segundo de los casos, definimos una variable de tipo de coma flotante o lo que es lo mismo decimal y a demás le atribuimos el cualificador long lo que hace que sea una variable de tipo decimal larga, esto hace que su rango numérico aumente. Y su nombre o identificador es: resultado.
3: En el tercer y ultimo caso definimos una variable de tipo carácter, y esta almacenara un carácter. Su ID es: letra.

Asignación de valores a variables

Es aquí cuando estamos verdaderamente introduciendo un dato dentro de una variable (dirección de memoria). Una vez que tenemos una variable definida, esta ya esta lista para albergar en su interior un dato pero esto solo podrá ser del tipo con el que se a definido la variable. Cuando metemos en una variable un dato se dice que la estamos declarando. Aunque el concepto y la diferencia entre definir y declarar no es muy claro.
Para mi, definir una variable es reservar el espacio en memoria y decir de que tipo va a ser, y cuando le metemos un dato ya estamos declarándola. Mas adelante esta definición se emplea a la inversa para las funciones.

Vamos a introducir valores a las variables definidas en el ejemplo anterior.

1) suma=2;
2) resultado=10.5;
3) letra='A';

1: Aquí estamos introduciendo el numero 2 dentro de la variable suma. Con lo que suma pasara a valir 2.
2: En el segundo caso, damos un numero decimal que introducimos dentro de la variable de tipo decimal. Así pues, la variable resultado ahora vale 10.5.
3: Por ultimo la variable de carácter "letra" le asignamos la letra a. Desde entonces siempre que sea nombrada o referida por su identificativo nos devolverá la letra a. Podríamos declararla de otra forma obteniendo el mismo resultado, tal y como es: letra = 65; De esta otra forma obtendríamos el mismo resultado pues que coge los números identiicativos a caracteres de la tabla ASCII.

Tamaño de una variable

Cuando definimos una variable, debemos saber con prioridad si va a almacenar una cantidad de bits pequeña o grande. Si la variable es de tipo entero (int) y almacena un numero mayor del rango que tiene este tipo de dato, se producirá un overflow. A la hora de hacer un programa debemos ser previsores e intentar poner el tipo de dato con un rango que soporte las posibles cifras a introducir en ese programa. Por ello es bueno conocer la forma de averiguar el rango de un tipo de dato, puesto que sabérselos de memoria es tontería e imposible.

Para conocer el tamaño o rango numérico que alcanza un determinado tipo de dato se usa la función **sizeof.**

```
#include <stdio.h>
main () {
 printf( "El tipo de dato entero (int) ocupa %i bytes\n", sizeof(int));
```

```
}
```

En este programa saldrá imprimido “4 bytes” . Sabiendo que un byte son 8 bits, (4*8) son el resultado de bits que ocupa el tipo de dato entero (int). Para calcular el máximo numero del rango que puede alcanzar este tipo de dato se hace elevando (2^32) a la 32 que era el numero de bits y obtenemos “4.294.967.296”. Pero esta cifra podría variar en función del uso de los ya anteriormente llamados cualificadores. La función “sizeof” se encuentra dentro de la librería “stdio.h”.

Uso de la función printf, pasando el argumento o parámetro de una variable

Nosotros cuando operamos con los distintos tipos de datos podemos o no saber la cifra que van a devolver. Lo mejor será verlo con un ejemplo.

```
#include <stdio.h>
main () {
 int num1,num2,resultado;
 num1=3;
 num2=2;
 resultado=num1+num2;
 printf( "\nEl resultado es: %d",resultado);
}
```

Lo que hacemos en este pequeño programa es:

1) Definimos 3 variables de tipo entero: num1,num2,resultado.
2) Le damos a num1 el valor “3”.
3) Le añadimos a num2 el valor “2”.
4) Le damos el valor a resultado de la suma de las dos anteriores variables (num1 + num2).
5) Imprimimos por pantalla el valor de resultado.

El meollo de la cuestión esta en el 5º paso. Con este tipo de paso de parámetros al printf estaremos continuamente trabajando así que no hay mas remedio que entenderlo ☺.
Tenemos la función: printf("\nEl resultado es: %d",resultado); Fijándonos en ella veriamos que el texto que se deberia de imprimir es el que esta entre comillas, o sea "\nEl resultado es: %d” Y así es lo único que no sale es el %d que es sustituido por el valor de la variable que hay al cerrar las comillas después de la coma.

Veamos otro ejemplo, que hace lo mismo pero con mejor presencia.

```
#include <stdio.h>
main () {
 int num1,num2,resultado;
 num1=3;
 num2=2;
 resultado=num1+num2;
 printf( "\nEl resultado de %d + %d es igual a %d",num1,num2,resultado);
}
```

Los primeros pasos hacen los mismo que el anterior ejemplo, debemos centrarnos en la ultima orden, la función printf. El primer %d es sustituido en orden por el valor de num1, el segundo por el valor de la variable num2, y el ultimo %d es sustituido por la ultima variable resultado.

Ahora bien, no siempre es %d lo que hay que poner para que lo sustituya por el valor de la variable pasada a la función. Según el tipo de dato que contenga la variable pasada debemos poner %d (enteros), %c (carácter), %f (decimales), etc.

Al %d se le llama “cadena de control”

Tipos de cadenas de control.

%c = carácter
%d = decimal
%e,f,g = real
%h = entero corto
%i = entero que puede ser octal decimal o hexadecimal.
%o = entero octal

%u = estero decimal sin signo
%x = entero hexadecimal
%s = cadena de caracteres acabada por un espacio.
%[...] = una cadena de caracteres que puede incluir espacios.

Ahora hagamos otro programa entero de ejemplo.

```
#include <stdio.h>
main () {
int num1,num2=0;
char txt;
clrscr();
printf("\nEste programilla mostrara el valor de 3 variables");
printf("\n");
num1=5;
txt='B';
printf("\nEl valor de la variable num1 es: %d",num1);
printf("\nEl valor de la variable num2 es: %d",num2);
printf("\nEl valor de la variable txt es: %c",txt);
printf("\n");
printf("\nFin del programa");
printf("\n");
}
```

Que pasaría si definimos una variable y le asignamos un valor mayor del rango numérico que nos permite el tipo de su asignación?
Lo que pasaría en tal cosa serías que si por ejemplo definimos una variable de tipo entero y supera el rango numérico de 32767 pasaría al siguiente numero y este seria el –32767. Puesto que no existe un numero mayor en el rango de ese tipo de definición de variables enteras. El compilador nos diría warnig y se produciría un overflow.

Las variables pueden ser según su funcionamiento o dato almacenado
Las variables por contador: Son un tipo de variables muy usadas en los bucles, rizos o procesos de repetición. Estas tipo de variables tienen un valor inicial y entran en un bucle y en función de una condición van incrementando o decrementando su valor inicial hasta la salida del bucle.
Las variables por acumulador: Este otro tipo de variables, lo que hacen es ir acumulando los valores, son las mas usuales.

Las variables pueden ser según donde estén definidas pueden ser
Variables globales: Este tipo de variables devuelven su valor en cualquier parte del procedimiento, afectan a todas las funciones que se encuentran en el programa.
Variables locales: Estas variables solo devuelven su valor en los procedimientos donde están definidas. Lo que quiere decir que están restringidas al procedimiento. Si se definen dentro de la función main solo tendrán vida dentro de esa función y no en funciones externas.

Constantes

Constantes : Elementos de almacenamiento de datos o direcciones de memoria, que no varían de ninguna forma durante el proceso del programa. Con lo que su valor será el mismo tanto al comienzo como al final de este.
Las constantes no solo funcionan en cualquier parte del procedimiento como las variables globales sino que también devuelven su valor almacenado en cualquier parte del código o programa.

Existen 4 tipos distintos de constantes
- C.enteras.
- C.de cóma flotante.
- C.de carácter.
- C.de cadena de caracteres.

Las constantes enteras: Este tipo de constantes pueden estar escritas en 3 tipos de sistemas numéricos, como son, el decimal, octal, y hexadecimal.
- Si la constante empieza por cero esta seria una constante entera escrita en octal (solo llevara del 0 al 7).

- Si la constante empieza por 0 o x será una constante entera escrita en hexadecimal (0 > 9 >F).
- En caso de no cumplirse estas condiciones anteriores estaríamos ante una constante entera decimal.

Las constantes de coma flotante o decimal
Distinguen la parte decimal de la entera por un punto, para el añadido de exponentes usa la letra “E”.

Las constantes de carácter
Se representa el valor de la constante encerrado entre apostrofos.

Las constantes de cadena
Almacena una cadena de caracteres que esta encerrada entre comillas.

También podemos especificar cualificadores en las constantes con lo cual tendríamos
- Si la constante acaba en UL estaremos ante una constante larga y sin signo.
- Si la constante acaba solo en U, estaremos ante una constante sin signo.

Si la constante acaba en L, estaremos ante una constante larga.

Como definir una constante?

Las constantes afectan a todas las funciones que se encuentran dentro del cualquier parte o función de nuestro programa.
Por ello se definen encima del main y debajo de los include. Las constantes se definen y se declaran al mismo tiempo. Se les da ya el valor. Esto se hace con una directriz “#define” a continuación va el identificador de esa constante o lo que es lo mismo su nombre, y por ultimo su valor.

Ejemplo:

```
#include <stdio.h>
#define PI 3.1416
main () {
printf( "\nEl numero PI vale: %f", PI);
}
```

Observemos en la segunda línea de nuestro código-programa. Ponemos la directriz #define que definirá constantes, a continuación su ID (identificador o nombre), un ultimo es pacio y al final el valor de esa constante.