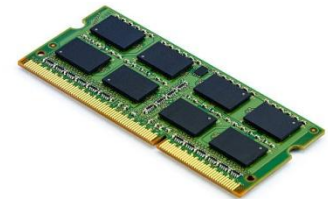


¿Qué es un tipo de dato?

- Un tipo de dato define:

- *el tamaño que dicho tipo va a ocupar en la memoria*
- *el rango de valores que puede almacenar dicho tipo*
- *la forma en que se almacenan en memoria los diferentes valores*
- *las operaciones que pueden realizarse con él*



¿Qué tipos de datos usa C?

- Numéricos:

- *Sin signo: para representar números sólo positivos.*
- *Con signo: para representar números que pueden ser tanto positivos como negativos.*
- *Decimales: para representar números con decimales.*

- Lógicos

- Caracter

Enteros Sin Signo

- *unsigned char*
- *unsigned short int o unsigned short*
- *unsigned int*
- *unsigned long int o unsigned long*
- *unsigned long long int o unsigned long long*

Enteros Con Signo

- *signed char*
- *short int o simplemente short*
- *int*
- *long int o simplemente long*
- *long long int o simplemente long long*

Decimales

- *float*
- *double*
- *long double*

Lógicos

- *bool*

En Ansi C se utilizan los enteros (int) para representar datos booleanos (*que sólo tienen dos valores: VERDADERO y FALSO*), utilizando el siguiente criterio:

- *Se considera VERDADERO para cualquier valor distinto de cero*
- *Se considera FALSO si es cero.*

Caracteres

- *char*
- *Almacenan un dígito correspondiente a:*
 - Caracteres alfabéticos: a b c ...z A B C ...Z
 - Caracteres numéricos: 0 1 2 ...9
 - Caracteres no imprimibles: espacio, tabulador, salto de línea ...
 - Caracteres especiales: + - * / ^ .,;< > \$?

String – cadena de caracteres

- Un string es un tipo de dato particular formado por una secuencia de caracteres agrupados que se utiliza para almacenar palabras. C no soporta string.
- Posee características particulares que lo diferencian:
 - Se crean indicando la cantidad de caracteres contenidos: `char nombre[20];` //se debe tener en cuenta el '\0'
 - Se le asigna un valor: `nombre="Juan"`
 - El nombre de la variable es la dirección de memoria donde fue alojado y por lo tanto se almacena desde el teclado como `scanf ("%s", nombre);`
 - Se imprime por teclado como `printf ("Su nombre es %s", nombre);`



60FEE8	60FEE9	60FEEA	60FEEB	60FEEC	60FEED	60FEEE	60FEEF	60FEF0	60FEF1	60FEF2	60FEF3	
60FEF4	60FEF5	60FEF6	60FEF7	60FEF8	60FEF9	60FEFA	60FEFB	60FEFC	60FEFD	60FEFE	60FEFF	
60FF00	60FF01	60FF02	60FF03	60FF04	60FF05	60FF06	60FF07	60FF08	60FF09	60FF0A	60FF0B	
60FF0C	60FF0D	60FF0E	60FF0F	60FF10	60FF11	60FF12	60FF13	60FF14	60FF15	60FF16	60FF17	
60FF18	60FF19	60FF1A	60FF1B	60FF1C	60FF1D	60FF1E	60FF1F	60FF20	60FF21	60FF22	60FF23	

Una celda de memoria corresponde a 1 byte
1 byte = 8 bits



	char=1 byte		short=2 bytes				int=4 bytes					
60FEE8	60FEE9	60FEEA	60FEEB	60FEEC	60FEED	60FEEE	60FEEF	60FEF0	60FEF1	60FEF2	60FEF3	
		float=4 bytes										
60FEF4	60FEF5	60FEF6	60FEF7	60FEF8	60FEF9	60FEFA	60FEFB	60FEFC	60FEFD	60FEFE	60FEFF	
				long=4 bytes								
60FF00	60FF01	60FF02	60FF03	60FF04	60FF05	60FF06	60FF07	60FF08	60FF09	60FF0A	60FF0B	
		double=8 bytes										
60FF0C	60FF0D	60FF0E	60FF0F	60FF10	60FF11	60FF12	60FF13	60FF14	60FF15	60FF16	60FF17	

long double=16 bytes															
60FF18	60FF19	60FF1A	60FF1B	60FF1C	60FF1D	60FF1E	60FF1F	60FF20	60FF21	60FF22	60FF23	60FF24	60FF25	60FF26	60FF27

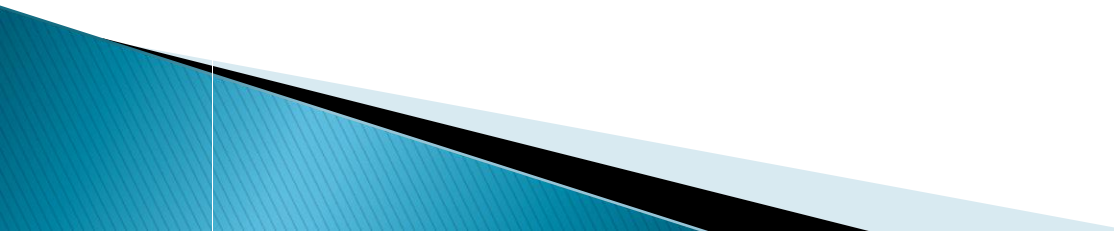
TIPO	Número de bits	Rango de valores
char	8	-128 a 127
unsigned char	8	0 a 255
short int	16	-32768 a 32767
unsigned short int	16	0 a 65535
int	32	-2147483648 a 2147483647
unsigned int	32	0 a 4294967295
long int	32	-2147483648 a 2147483647
unsigned long int	32	0 a 4294967295
long	32	-2147483648 a 2147483647
float	32	$1.18e-38 \leq X \leq 3.40e38$ Precisión científica (7-dígitos)
double	64	$2.23e-308 \leq X \leq 1.79e308$ Precisión científica (15-dígitos)
long double	80	$3.37e-4932 \leq X \leq 1.18e4932$ Precisión científica (18-dígitos)

Fuente: https://www.zator.com/Cpp/E2_2_4.htm

Variables

- Las variables se definen por los tipos de datos que indicamos al crearlas.

¿Qué es una variable?

- Es un espacio en memoria que el programador reserva para guardar un dato que recibe y que necesitará utilizar más tarde
 - La variable es una porción de memoria que ningún otro programa podrá utilizar
 - Para el compilador cada variable es una dirección de memoria específica en donde se aloja el dato guardado
- 


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

```
int main(){
    char un_char = 'a';
    short un_short = 31234;
    int un_int = 214755555;
    long un_long= -114755555;
    float un_float=12345.11234567890;
    double un_double=2.179000000045670000000000;
    long double un_long_double=4.0080000000000000000000337;

    printf("El valor de un_char es: %c, y en decimal es: %d\n", un_char,un_char);
    printf("El valor de un_short es: %d\n", un_short);
    printf("El valor de un_int es: %d\n", un_int);
    printf("El valor de un_long es: %d\n", un_long);
    printf("El valor de un_float es: %f, limitando los decimales a 2: %.2f, y en notaci%cn cient%cfica es: %e\n",un_float,un_float,162,161,un_float);
    printf("El valor de un_double es: %f, y en notaci%cn cient%cfica es: %e\n", un_double,162,161,un_double);
    printf("El valor de un_long_double es %lf y en notaci%cn cient%cfica es: %e.....\n", un_long_double,162,161, un_long_double);
    printf("Consultar el problema con long double en:\nhttps://stackoverflow.com/questions/26296058/cant-print-correctly-a-long-double-in-c\n");
    getchar();
    return 0;
}
```

	un_char		un_short			un_int									
60FEE8	60FEE9	60FEEA	60FEEB	60FEED	60FEED	60FEEF	60FEED	60FEF0	60FEF1	60FEF2	60FEF3				
		un_float													
60FEF4	60FEF5	60FEF6	60FEF7	60FEF8	60FEF9	60FEFA	60FEFB	60FEFC	60FEFD	60FEFE	60FEFF				
				un_long											
60FF00	60FF01	60FF02	60FF03	60FF04	60FF05	60FF06	60FF07	60FF08	60FF09	60FF0A	60FF0B				
		un_double													
60FF0C	60FF0D	60FF0E	60FF0F	60FF10	60FF11	60FF12	60FF13	60FF14	60FF15	60FF16	60FF17				
un_long_double															
60FF18	60FF19	60FF1A	60FF1B	60FF1C	60FF1D	60FF1E	60FF1F	60FF20	60FF21	60FF22	60FF23	60FF24	60FF25	60FF26	60FF27

¿Qué significa crear una variable?

- Las variables se crean al momento de ejecutar nuestra aplicación.
 - El compilador reserva un espacio libre en memoria y lo apunta a nuestra variable para que cada vez que la utilicemos vaya a dicha dirección
 - La porción (extensión) de memoria que reserva dependerá del dato que deseamos guardar
 - El valor almacenado puede cambiar en el transcurso de la ejecución del programa, pero siempre serán valores del tipo de dato al que pertenecen.
- 

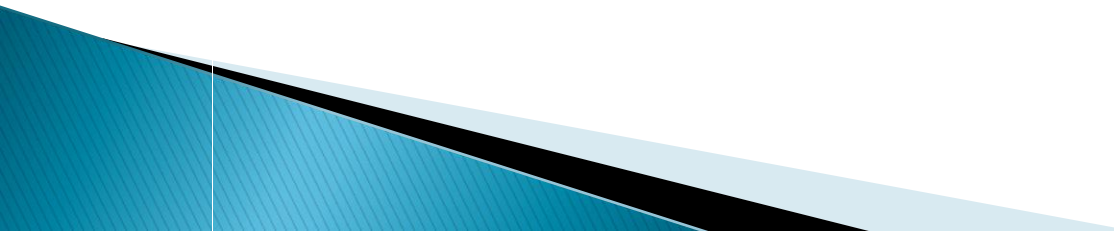
Ámbito de las variables

- ❖ El ámbito de una variable es la porción de código donde la variable está disponible (desde dónde se puede acceder a ella)
- ❖ Cuando una variable está disponible en una porción de código, diremos que es visible.
- ❖ Estudiaremos dos ámbitos:
 - Global
 - Local

Variables Globales

- ❖ Se declaran fuera del **main()**
- ❖ Son visibles en todo el código que sigue a su declaración
- ❖ Existen durante toda la ejecución del programa

Variables Locales

- ❖ Se declaran dentro del main o de una función (al principio del cuerpo)
 - ❖ Sólo son visibles dentro del cuerpo de código al que pertenecen
 - ❖ Se crean automáticamente cuando comienza la ejecución del bloque
 - ❖ Se destruyen automáticamente cuando se termina la ejecución del bloque
- 

Constantes

- ❖ El programador puede definir una constante la cual no podrá variar su valor una vez asignado

Para crear una constante float correspondiente a la cotización dólar (la cual no puede modificarse durante el uso de la aplicación) escribimos:

```
const float cotizacion = 90.71;
```

Constantes

- ❖ Se puede usar const antes o después del tipo de dato.
- ❖ Se inicializa la constante al crearla dado que no podrá cambiarse su contenido.
- ❖ La directiva del preprocesador **#define** es similar a la utilización de constantes sólo que NO se procesa como una constante.
- ❖ Al utilizar **#define** no se reserva espacio en memoria para un dato sino que al momento de compilar se reemplaza el nombre definido por el valor indicado.

Trabajando con variables

- Creación de variables
 - **int edad, mes, anio;**
 - **char sexo;**
 - **float sueldo;**
- Almacenamiento de valores
 - **edad = 22;**
 - **sexo = 'F';**
 - **sueldo = 4028.50;**

Trabajando con variables

- Almacenamiento de un valor introducido por teclado
 - `scanf ("%d", &edad);`
 - `scanf ("%c %f", &sexo, &sueldo);`
- Impresión del valor almacenado en una variable
 - `printf ("La edad introducida es %d", edad);`
 - `printf ("El sueldo promedio para un empleado de sexo %c es %.2f", sexo, sueldo);`