Tipos Básicos de Datos en C

Prof. Judith Barrios Albornoz

Departamento de Computación

Escuela de Ingeniería de Sistemas

Facultad de Ingeniería

Universidad de Los Andes

Semestre A_2013

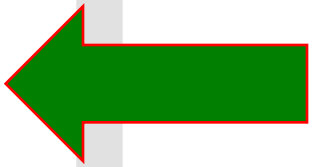
Este material toma como base el curso original
Prof. Narciso. Año 2000- 2006

¡Atención!

Elementos de un lenguaje de programación

Un sub-lenguaje para **definir los datos**:

¿Qué datos tenemos?
¿Cómo les llamamos?
¿Cómo son (tipo y/o estructura)?
¿Qué se puede hacer con ellos?



Un sub-lenguaje para **definir los algoritmos:**

¿Qué se hace con los datos?
¿En qué orden? ¿cuándo se hace?
¿Cuántas veces se hace?

Conceptos Básicos

¿**Dato?** *Expresión general que describe los objetos con los cuales opera un computador*

Existen varios tipos de datos:

- cifras de venta de un supermercado
- nota de un examen parcial
- cantidad de vuelos entre El Vigía y Caracas
- ….

Un *Lenguaje de Programación* incorpora un conjunto de **tipos de datos** que permite:

– conocer y utilizar las **propiedades del tipo de dato** y **los operadores que son aplicables** a dichos tipos de datos

– no importa el cómo se implementa este tipo de dato (su representación interna)

Conceptos Básicos

Cuatro clases básicas de datos:

1) **Número** (*punto fijo, punto flotante*)
2) **Lógico** (dos valores – verdadero/falso)
3) ***Caracter***
4) Dirección de memoria o **apuntador**

- ***Valor***
- ***Variable/constante***
- ***Memoria***
- ***Nombre***

Tipo de dato

Conceptos Básicos

Valor

Valor – ***elemento*** perteneciente a un ***conjunto*** representa ***un tipo de dato***

Todos los valores del ***conjunto*** deben cumplir la misma **propiedad**

el conjunto define el ***tipo de operaciones*** que se pueden aplicar sobre sus valores

Ejemplo*:*

El valor **2** es un elemento perteneciente al conjunto

$$A = \{-\infty \ldots, -1, 0, 1, 2, 3, 4, \ldots \infty\}$$

Conceptos Básicos

tipo de dato es equivalente a ***conjunto***

valor es equivalente a ***elemento***

Conceptos Básicos

Variable

Elemento de ***memoria*** que sirve para almacenar un ***valor***

referenciado por un ***nombre*** (*identificador*) perteneciente a un ***tipo de dato***

A diferencia del valor, la **variable** tiene **propiedades relacionadas con el espacio y el tiempo:**

*ocupa un **espacio** determinado **de memoria** que puede almacenar un **valor distinto en cada instante de tiempo***

Conceptos Básicos

Variable

Las variables contienen el estado de un programa

1. Se definen (se declaran)
2. Se crean
3. Se cargan con un valor inicial (datos de entrada)
4. Su valor se **modifica** (durante el programa)
5. Llegan a un valor final (resultados de salida del programa)

¡ Todo programa debe controlar sus variables!

Variable descontrolada ==== > > Posibilidad de ERROR !!!

Conceptos Básicos

Variable

Criterios a seguir con las variables:

- Cuantas menos es mejor
- Cada una con un **significado** muy claro e inmutable
- No hay que olvidarse de darles un **valor inicial**
- Controlar (y comprobar) que van **tomando valores** los valores previstos:
  - regularmente hay que aplicarles *predicados* que éstas deben satisfacer y notificar si alguna se sale de lo pautado
- Los lenguajes modernos obligan a **declarar** las variables antes de usarlas
  - permite chequear su existencia y la coherencia en su uso

Conceptos Básicos

Memoria

Puede ser interpretada como un conjunto de pares (***variable, valor***) que lleva asociada dos operaciones:

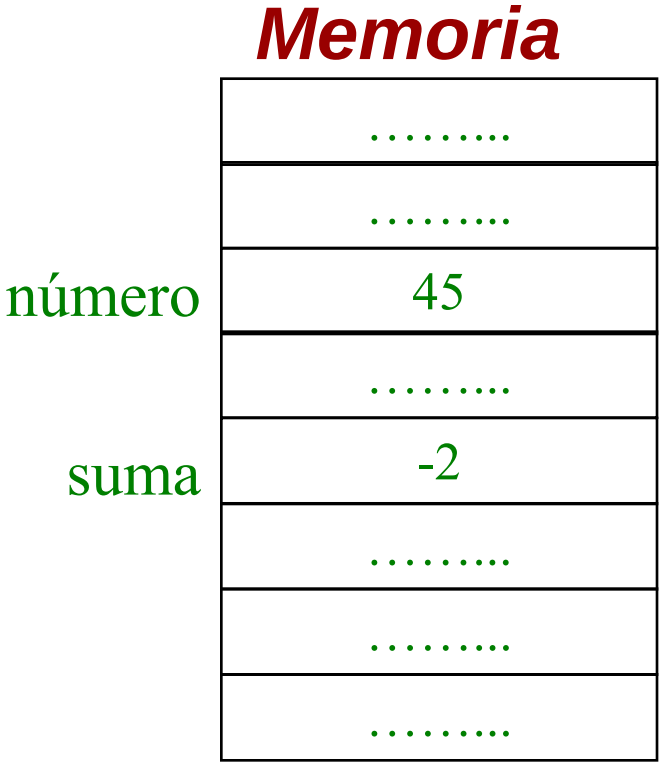
*Búsqueda (**lectura**)* y *Almacenamiento* (***escritura***)

Memoria = {(Variable$_1$, Valor$_1$), ...,(Variable$_n$, Valor$_n$)}

1) Búsqueda (Variable, Memoria) = Valor

2) Almacenamiento (Variable, Valor, Memoria)

Conceptos Básicos



Memoria = {…, (número, 45), …, (suma, -2), ...}

Conceptos Básicos

Memoria: **Asignación**

Expresión sintáctica de la operación de **Almacenamiento** **que modifica el contenido** de una variable

escribe el **valor** en la dirección de memoria asociada a la **variable**

Notación algorítmica

Nombre ← Valor o ***Nombre = Valor***

Notación en C ***Nombre = Valor***

Conceptos Básicos

Memoria: *Asignación*

numero = 45

Memoria = {…, (numero, 45), …, ...}

suma = -2

Memoria = {…, (numero, 45), …, (suma, -2) ...}

numero = 28

Memoria = {…, (numero, 28), …, (suma, -2) ...}

Conceptos Básicos

Memoria: *Asignación*

- Cada operación de asignación (*Almacenamiento*) transforma la memoria respecto al tiempo:

  pasa **de un estado a otro**

- La ejecución de un programa trae como efecto (*sin considerar los procesos de E/S*)

  la transformación de la memoria del contenido inicial en el contenido final

realiza un conjunto de **asignaciones** sobre las **n variables** que intervienen en el programa - ejecutadas según un flujo de control establecido

Conceptos Básicos

Memoria: *Asignación*

- La asignación **X = 7** equivale a decir que **X** es una variable a la que se le asigna el valor **7**
- La asignación **X = X + 3** evalúa la expresión derecha, en donde la variable con nombre **X** tiene el valor de **7**, sumándole la ***constante* 3** y **asignando** el resultado **(7 + 3) = 10** a la parte izquierda de la expresión, la **variable X**

De acuerdo a la programación estructurada, la asignación es una estructura de control **secuencial**

Conceptos Básicos

Nombre

Identificador que consta de varios caracteres ***alfanuméricos***, de los cuales el primero *normalmente* es una letra

Ejemplos

A221, notas, NOMBRES, NROCuenta, Pepe, temp, y, suma, Suma, NotaMaxima, sueldo_base, dias_en_anyo, ConjuntoDatos1, Ganancias95, Entero _temperatura

Conceptos Básicos

Memoria: *Asignación*

Cuádrupla **V = < N, T, R, K >** donde

N es el ***nombre*** de la variable

T su ***tipo de dato***

R una ***referencia*** de memoria asignada a la variable para su almacenamiento

K el ***valor*** almacenado

Ejemplo

V = < X, Entero, 10001, 7 > ≡	**X = 7**
nombre de la variable	X
tipo de dato	Entero
referencia de memoria	10001
valor	7

Conceptos Básicos

Tipo de dato

Es la explicación de un conjunto de ***valores***, denominado ***dominio***, sobre el cual se pueden realizar un conjunto de ***operaciones***

Toda variable debe estar asociada a un tipo de dato responsable de indicar el dominio de valores válidos para ella

Conceptos Básicos

Tipo de dato

Debido a que valores distintos pertenecientes a diferentes tipos de datos pueden tener la misma representación a nivel de máquina, la especificación del tipo de dato (**dominio, rango y operaciones aplicables**) permite controlar la interpretación para cada uno

Ejemplo

La secuencia de bits 01000001 (alfabeto binario de longitud 8) puede ser interpretada:

- Caracter ‘A’ en el tipo de dato *Caracter*
- Entero +65 en el tipo de dato Entero
- Real 4.5678 en el tipo de dato Real

Conceptos Básicos

Tipo de dato

Puede clasificarse como ***escalar*** o ***estructurado***

- ***Escalar*** o ***simple***: Aquel cuyo dominio presenta una propiedad de orden (*Entero, Real, Caracter, Lógico, Apuntador*)

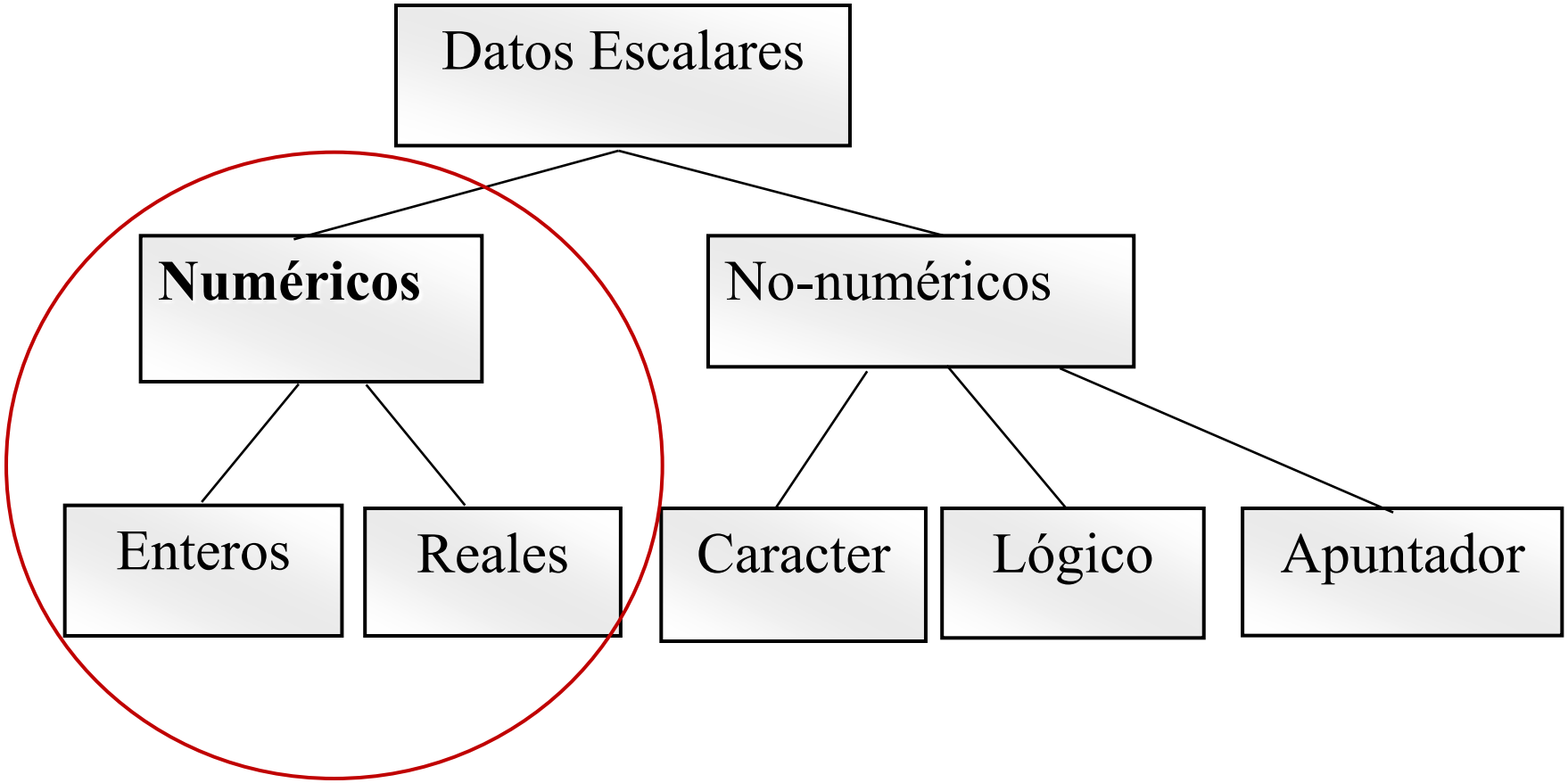
- ***Estructurado*** o ***compuesto***: Aquel que se define mediante composición de tipos de datos (*vector, cadena de caracteres, matriz, registro*)

Tipos de Datos en C /C++

Los *lenguajes de programación* ofrecen un **conjunto completo** de tipos de datos **escalares** y **estructurados** con las especificaciones del **dominio de sus valores** y sus **operaciones**

En este curso veremos algunos de los tipos de datos que ofrece el lenguaje de programación C/C++

Tipos de Datos Escalares en C/C++



Tipos de Datos Escalares (Numéricos) en C/C++

Tipo Entero

Subconjunto finito de los números enteros

-*rango o tamaño dependerá del lenguaje de programación en el que posteriormente se codificará el algoritmo y de la computadora utilizada*

Ejemplos

5 6 -15 4 0 20 17 -1340 26

Tipos de Datos Escalares (Numéricos) en **C**/C++

- ***Tipo Entero***

  Dependiendo del **número de bits** empleado en cada computadora (**n**), los **dominios del tipo de dato *Entero* varían** en

  a) $-2^{(n-1)}, ..., 0, ..., 2^{(n-1)} - 1$ **enteros positivos y negativos**

  Si **n = 16**

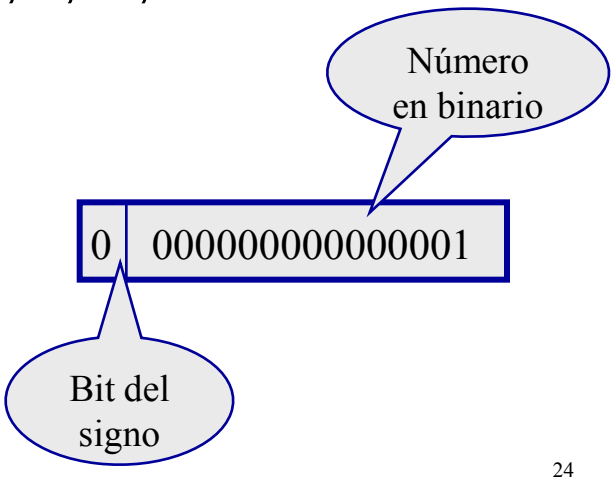
  $-2^{(16-1)}, ..., 0, ..., 2^{(16-1)} - 1 = -2^{(15)}, ..., 0, ..., 2^{(15)} - 1$

  = -32768, …, 0, …, 32767

  b) $0, ..., 2^n - 1$ enteros positivos

  Si **n = 16**

  $0, ..., 2^{16} - 1 =$ **0, ..., 65535**



Tipo de Dato **Entero** en **C**/C++

Tipo	Nro. bits	Rango
int	16	-32768 … +32767
long int	32	-2147483648 … +2147483647
unsigned int	16	0 … +65535
unsigned long int	32	0 … +4294967295

$-2^{31}, \ldots, 2^{31}-1$

$2^{32}-1$

signed int (16), **short int** (16), **unsigned short int** (16), **signed short int** (16), **signed long int** (32), **unsigned long int** (32), **long long int** (64), **insigned long long int** (64), **signed long long int** (64)

Tipo de Dato **Entero** en **C**/C++

- **Operadores**:

Operación	Operador	Operador en C/C++	Ejemplo
Suma	+	+	24 + 56
Resta	-	-	5 - 4
Multiplicación	x	*	4 * 5
División	/ o —	/	34 / 6
Resto o módulo		%	68 % 2
Incremento unitario		++	7++
Decremento unitario		--	6--

Tipos de **Datos Escalares** (Numéricos) en **C**/C++

- ***Tipo Real***

Subconjunto de los números reales limitado no sólo en el tamaño, sino también en cuanto a la **precisión** (*depende de la computadora*)

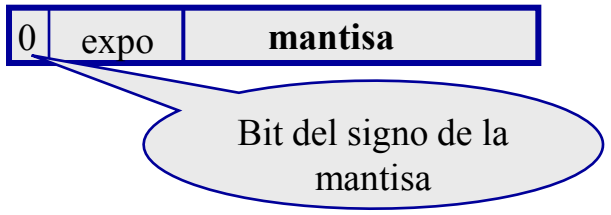
Se conocen como números de **punto flotante** cuya representación consta de una ***mantisa*** (parte fraccional), de una ***base*** y de un ***exponente*** (potencia a la cual se eleva la base)

- **Para el número $0.437875x10^3$ se tiene:**

  mantisa = 0.437875

  base = 10

  exponente = 3



Ejemplos

0.08 3739.41 -3.7452 52.3244 -8.12 3.0

Tipo de Dato **Real** en **C**/C++

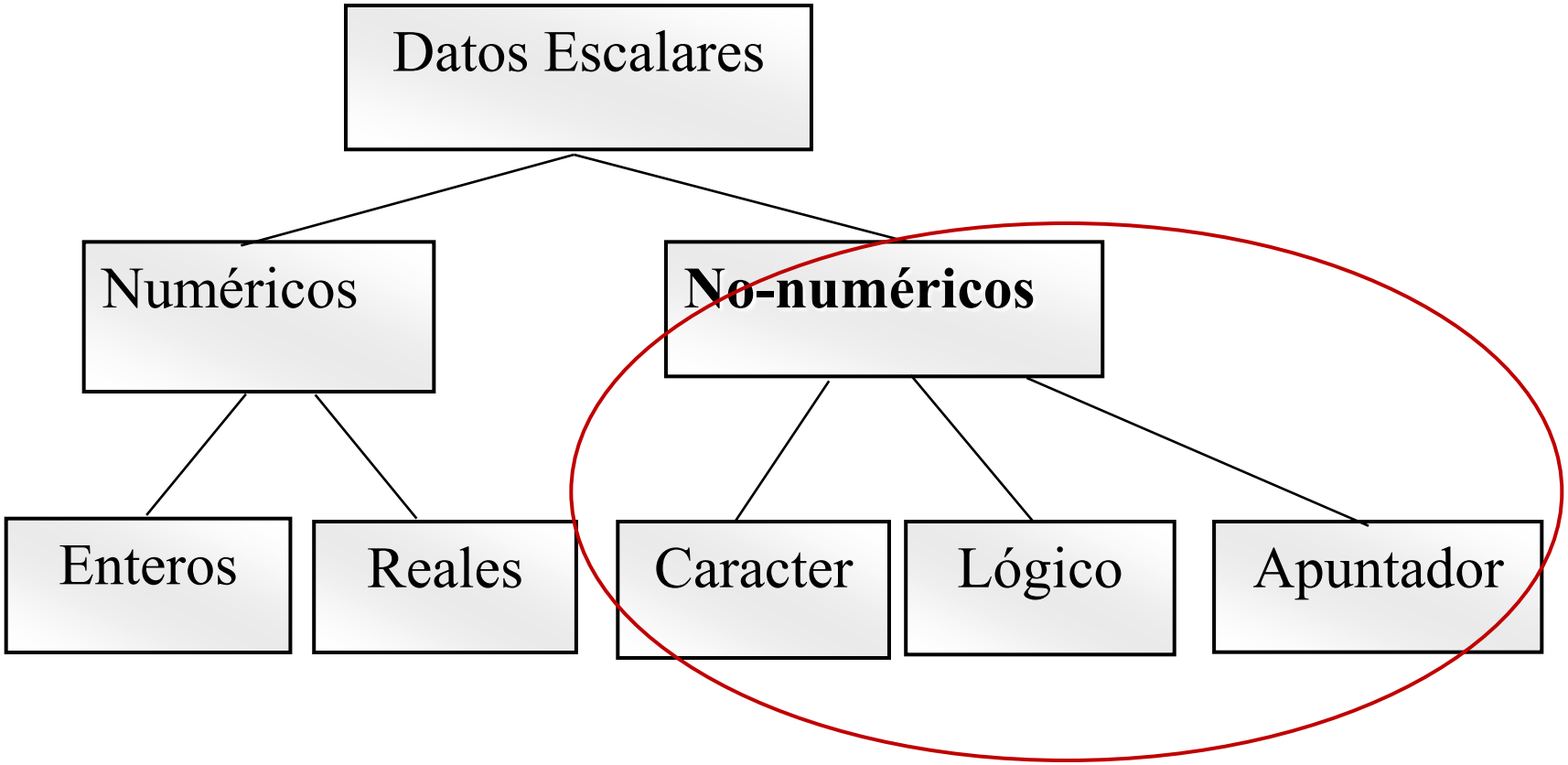
Tipo	Nro. bits	Rango	Precisión
float	32	1.17×10^{-38} a 3.4×10^{38}	6 dígitos decimales
double	64	2.22×10^{-308} a 1.79×10^{308}	10 dígitos decimales
Long double	128		10 dígitos decimales

Tipo de Dato **Real** en **C**/C++

Operación	Operador	Operador en C/C++	Ejemplo
Suma	+	+	7.89 + 56.4
Resta	-	-	6.23 – 2.45
Multiplicación	x	*	1.34 * 54.3
División	/ o —	/	34.6 / .96
Incremento unitario		++	7.55++
Decremento unitario		--	634.67--

Incremento o decremento de la parte **entera solamente**

Tipos de **Datos Escalares** en C/C++



Tipos de Datos Escalares (No-Numéricos) en C

Tipo Caracter

Conjunto finito y ordenado de los caracteres que la computadora reconoce (*letra, dígito, signo de puntuación, …)*

Un caracter es almacenado en un **byte** usando el **código** de **8 bits ASCII** (*American Standard Code for Information Interchange*) lo que da la posibilidad de representar

$2^8 = 256$ caracteres diferentes

Ejemplos

‘v’ ‘.’ ‘A’ ‘a’ ‘)’ ‘{‘ ‘+’ ‘9’ ‘*’

Tipo de Dato **Caracter** en **C**/C++

Tipo	Nro. bits	Precision
char	8	0 - 255
unsigned char	8	0- 255
signed char	8	-127 a 127

Operadores:
No tienen
Solo se leen y escriben

En este lenguaje los caracteres se pueden ***sumar, restar, multiplicar, dividir,*** calcular el resto, incrementar y decrementar, **sólo si usted los ve como enteros pequeños**, si los imprime como caracter obtendrá resultados en ASCII, donde no todos los caracteres son imprimibles

Tipos de Datos Escalares (No-Numéricos) en C/C++

Tipo Lógico (booleano)

Conjunto formado por los valores Verdadero y Falso

El tipo lógico en C++ existe si el compilador lo incluye, pero en el C/C++ ANSI no existe, por lo cual el usuario lo puede definir mediante la sentencia:

enum logico {falso, cierto};

Tipo Lógico (booleano)

Operación	Operador	Operador en C/C++
Y lógico	∧	&&
O lógico	∨	||
No (negación)	¬	!

Operador unario

!	Resultado
Falso	Cierto
Cierto	Falso

Operadores binarios

&&	Falso	Cierto
Falso	Falso	Falso
Cierto	Falso	Cierto

||	Falso	Cierto
Falso	Falso	Cierto
Cierto	Cierto	Cierto

Declaración de Variables

En **C**/C++ **todas las variables** que se usan en un programa deben ser **declaradas antes** de ser **usadas**

Declaración de Variables

Los objetivos de la declaración de variables son:

- **Asociar un tipo de dato y un identificador** (o nombre) a la variable para que el compilador pueda verificar la **correctitud** de las **operaciones** en donde interviene la variable

- Permitir que el compilador sepa cuánto **espacio** de **memoria** se necesita para almacenar el valor de la variable, y **asignar la dirección** de memoria donde este valor se va a almacenar

Declaración de Variables

Variables separadas por comas

tipo_de_dato ***lista_de_variables;***

Ejemplos

```
int día, mes, año;
int edad;
unsigned int A = 347;      /* Iniciación de
variable al momento de la declaración */
float pi = 3.14159;
double a, b, c;
unsigned long int B = 294967295;
long int C, distancia;
```

Declaración de Variables

Ejemplos

```
float precio, sub_total;
float costo_por_unidad;
char am_pm;
char letra = 'Z', suma = '+';
```

Tipos de Variables – *ámbito de validez*

- **Variable local**
  - Es aquella que está declarada **dentro** de un **bloque** delimitado por { }
  - Sólo se puede usar dentro del bloque en el que ha sido declarada

- **Variable global**
  - Es aquella que está declarada para **todo** el programa, es decir, **fuera de cualquier bloque o función**
  - Retiene su valor durante la ejecución de todo el programa

Declaración de Variables Locales:
Ejemplo 1

```
#include <stdio.h>”          /* Declaración de importación*/
void main ()
{
  /* Declaración de variables locales*/
      float base = 10.5, altura = 2.5, superficie;

  /* Cuerpo de la función*/
      Conjunto de sentencias
}
```

Declaración de Variables Locales:
Ejemplo 2

```
#include <stdio.h>                    /* Declaración de importación*/

enum logico = {falso, cierto};        /* Definición del tipo lógico */

void main ( ) {
    int i, j = 0, k;                  /* Declaración de variables locales */
    char car1, car2;
    float dividendo, divisor;
    int x, y, z;
    logico indicador = cierto;

    Conjunto de sentencias            /* Cuerpo de la función */
}
```

Declaración de Variables Locales:
Ejemplo 3

```
#include <stdio.h>
void main ()
{                                                          B1
        double x = 2.0;
        printf (%d,  x\n);                  /* se imprime 2.0 */
        {                                                  B2
                printf (%d\n,  x);          /* se imprime 2.0 */
                double x = 3.0;
                printf (%d\n, x);           /* se imprime 3.0 */
        }
    printf (%d\n, x);                       /* se imprime 2.0 */
}
```

Declaración de Variables Locales:
Ejemplo 4

```
void main( )
{
        int miEntero;
        miEntero = 503;          /* Valor inicial de miEntero es 503*/
        {
            int otroEntero;
            otroEntero = miEntero;
                                 /* Valor de otroEntero es 503*/
        }
}
```

B_1

B_2

Declaración de Variables Globales: *Ejemplo 5*

```
#include <stdio.h>
int valor;                    /* Declaración de variable global*/
void main()
{
    printf (“Introduzca un numero %i\n”);
    scanf (%d, &valor);
    printf (“ El numero leido es %i\n”, valor);
}
```

Declaración de Variables Globales y Locales: Ejemplo

```
#include <stdio.h>
int resultado;                    /* Declaración de variable global*/
void main( )
{
    int x, y, z;                  /* Declaración de variables locales*/
    printf ("Introduzca tres numeros enteros:");
    scanf ("%d%d%d", &x,&y,&z);
    resultado = x * y * z;        /* Calcular el producto de los tres numeros*/
    prinft ("El producto de los tres numeros es %d\n", resultado);
}
```

Constante

Valor que **no cambia durante la ejecución** de un programa

Puede ser ***numérica entera, numérica real, lógica, caracter***

Definición de Constantes (Globales)

#identificador valor

define

Asigna un valor a un identificador

Un proceso previo a la compilación sustituirá el identificador por el valor *en cualquier parte del programa donde aparezca el identificador*

Definición de Constantes (**Globales**)

Ejemplos

```
#define PI 3.14159
#define MAXIMO 256
#define PRIMERALETRA ‘A’
#define MENSAJE "Introduzca su nombre:"
```

Ejemplo

```
#include <stdio.h>                    /* Declaración de importación*/

#define LONGITUD 32                   /* Definición de constante*/

void main ( )
{
     int l1 = LONGITUD, x;            /*Declaración de variable local*/
     x = LONGITUD + 1;
   printf ("x = %d\n", x);
}
```

Valor inicial

Declaración de Constantes (Locales)

const ***tipo_de_dato*** *identificador* = valor;

```
void main ( )
{
    Declaración de constantes locales   /*Opcional*/
    Declaración de variables locales    /*Opcional*/

    Conjunto de sentencias              /* Cuerpo de la función*/
}
```

Una constante declarada de esta manera tan sólo puede usarse dentro del bloque en el cual ha sido declarada

Declaración de Constantes (Locales)

```
void main ( )
{
   const int LONGITUD = 32;
   int lon = LONGITUD;

    Conjunto de sentencias        /* Cuerpo de la función*/
}
```

Ejemplos

```
void main()
{
    const float pi;
    /* Error: pi debe ser iniciada*/
    /* en la declaración*/

    pi = 3.14159;
    /* Error: no se puede modificar*/
    /* el valor de una constante*/

}
```

```
#include <stdio.h>

void main()
{
    const float x = 7.0;

    prinft(%d, x\n);
}
```

Ejemplo:

Dado el radio de una esfera, calcular su área y su volumen

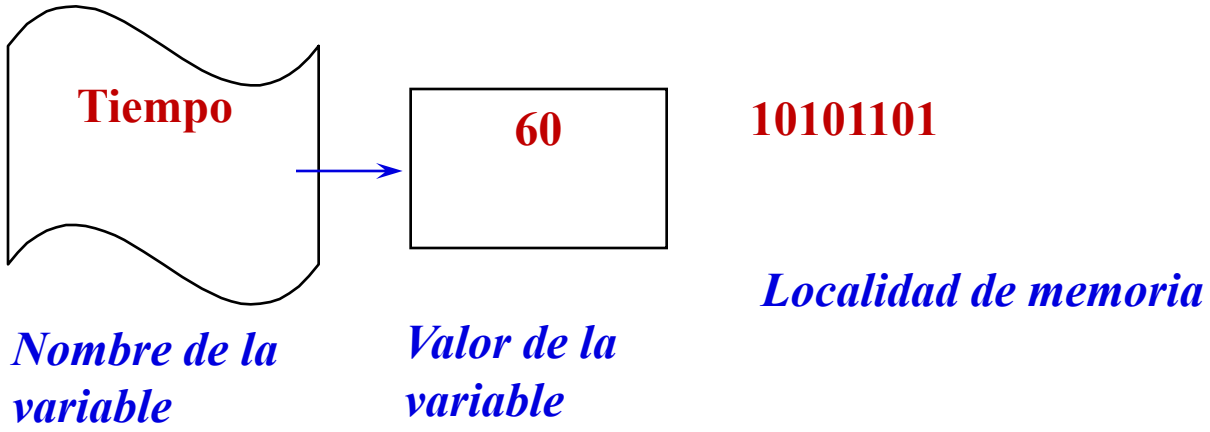
```
#include <stdio.h>
#define CUATRO 4.0              /* Definición de constante global */
float radio;                    /* Declaración de variable global */

void main ()
{
    const float PI = 3.14159; /* Declaración de constante local */
    float area, volumen;      /* Declaración de variables locales */
    printf ("Introduzca el radio de la esfera %f\n");
    scanf (%f, &radio);
    area = CUATRO * PI * radio * radio;
    printf ("Area = %f\n", area);
    volumen = area * (radio/3);
    printf ("Volumen = %f\n", volumen);
}
```

¡IMPORTANTE !

ES INDISPENSABLE ESPECIFICAR EL TIPO DE DATO DE *CADA* CONSTANTE Y DE *CADA* VARIABLE

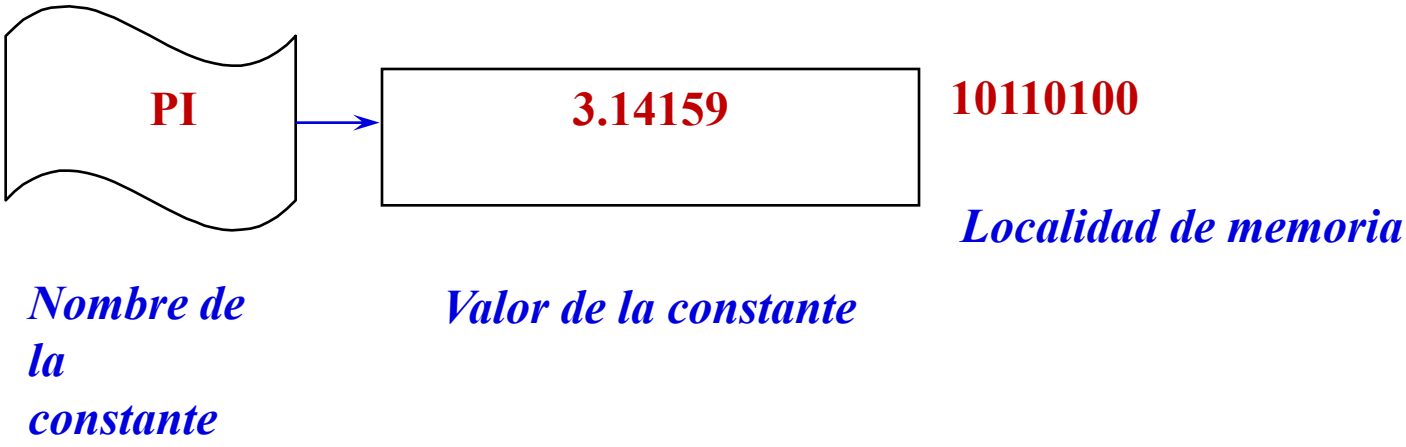
Hay que recordar que:

Al declarar una variable se reserva espacio de memoria principal y se etiqueta con el identificador correspondiente



Hay que recordar que:

Al definir o declarar una constante se reserva espacio de memoria principal y se etiqueta con el identificador correspondiente.



Ejercicios

- Definir cada una de las siguientes constantes
  - Como globales
  - Como locales

  - **613**
  - **613.0**
  - **-613**
  - **‘6’**
  - **‘G’**
  - **-3.012×10^{15}**
  - **17×10^{12}**
  - **π**
  - ***e***
  - **- Masa del electrón (en kg)**
  - **- Diámetro atómico (en cm)**