Tipos Básicos de Datos en C

Andrés Arcia
Departamento de Computación
Escuela de Ingeniería de Sistemas
Facultad de Ingeniería
Universidad de Los Andes

Recordar.

Programa = Datos + Algoritmos

Entonces se necesitan:

Un sublenguaje para definir los datos:

- Representación del problema en datos
- Nombres de las variables
- Tipos (enteros, reales, caracteres, ¿nuevos tipos?)
- Operaciones sobre los tipos (sumas, restas, juntura, ¿nuevas operaciones?, etc.)

Un sublenguaje para definir los algoritmos:

- Qué le hacemos a los datos.
- En qué orden (cuándo se lo hacemos).
- Cuántas veces.

Conceptos Básicos

***Dato*: En el computador un dato** siempre se resume a un conjunto de bits. Un conjunto de reglas sobre esos bits forman un tipo de dato. Por ejemplo: un dato de tipo entero, un dato de tipo caracter.

Dos tipos de datos básicos

1) **Números: Entero, punto flotante de precisión variable, dirección de memoria.**
2) **Carácter: Básicamente su utilizan para formar palabras.**

- **A partir de estas dos clases de datos, los lenguajes de programación incorporan diversos tipos de datos que son reutilizables.**
- **Las propiedades y operadores aplicables a dichos tipos de datos nos permiten modelar problemas del quehacer en la computación (muy amplio hoy en día, practicamente ubicuo).**

Conceptos Básicos

- **Valor**
- **Variable**
- **Memoria**
- **Nombre**
- **Tipo de dato**

Conceptos Básicos

Valor

La definición más intuitiva de *valor* es similar a la de *elemento* perteneciente a un *conjunto*, el cual representa *un tipo de dato*. Todos los valores del *conjunto* deben cumplir la misma propiedad, y 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 = {0, 1, 2, 3, 4, ...}

Conceptos Básicos

Valor

tipo de dato es equivalente a *conjunto*

valor es equivalente a *elemento*

Conceptos Básicos

Variable

Espacio **de *memoria* que sirve para almacenar un *valor*, referenciada por un *nombre*, y perteneciente a un *tipo de dato*.**

A diferencia del valor, la variable tiene propiedades espacio-temporales, es decir, ocupa un espacio determinado de memoria que puede almacenar un valor distinto en cada instante de tiempo.

¿Cómo se define un instante de tiempo?

Conceptos Básicos

Variable

Las variables que habitan en un programa definen el estado en cualquier instante de tiempo.

1. **Se definen (declaración)**
2. **Se crean**
3. **Se modifican**
4. **Se destruyen**

Dentro de las modificaciones estan:

Se cargan con un valor inicial (datos de entrada)
Su valor se modifica (programa)
Llegan a un valor final (resultados de salida)

Conceptos Básicos

Variable

Criterios a seguir con las variables:

- **Cada variable debe tener un objetivo claro e inmutable.**
- **No olvidarse de darles un valor inicial.**
- **Controlar (y comprobar) que van tomando valores esperados: regularmente aplicar predicados que deben satisfacer y notificar si alguna se sale de lo previsto (programación defensiva).**
- **Los lenguajes fuertemente tipeados obligan a declarar las variables antes de usarlas, lo que 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* y *Almacenamiento*.

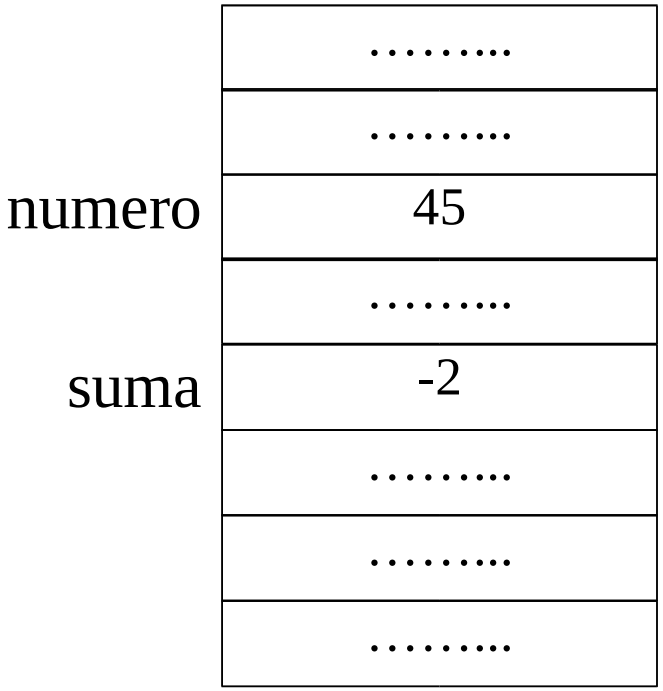
1) Memoria = {(Variable$_1$, Valor$_1$), ...,(Variable$_n$, Valor$_n$)}

2) Búsqueda(Variable, Memoria) = Valor

3) Almacenamiento(Variable, Valor, Memoria)

Conceptos Básicos

Memoria



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

Conceptos Básicos

Memoria: Asignación

Expresión sintáctica de la operación de *Almacenamiento*. Modifica el contenido de una variable.

Notación algorítmica

Nombre ← Valor ó Nombre = valor

Notación en C

***nombre_variable* = ϖαλο ρ;**

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 de un estado a otro respecto al tiempo.

La ejecución de un programa tendrá como efecto (sin considerar los procesos de E/S) la transformación de la memoria inicial en otra final, realizando 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

FechaInicio, NumeroDePersonas, Tmp10, direccion_1, direccion_2, _EstaActivado_, Tiempo_De_Ejec_00

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, y 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 agrupación de un conjunto de *valore*s, sobre el cual se pueden realizar un conjunto de *operacione*s.

Toda variable debe estar asociada a un *tipo de dat*o con el que se puede determinar univocamente el dominio de valores.

Conceptos Básicos

Tipo de dato

Ejemplo

El tipo de dato *fecha* podría estar representado por los *atributos* DIA, MES y AÑO, siendo los tres de tipo *entero*.

Las *operaciones* aplicables sobre el tipo de dato *fecha* podrían ser:

- **Mostrar fecha**
- **Incrementar día**
- **Incrementar mes**
- **Incrementar año**
- **Intervalo entre dos fechas**

Conceptos Básicos

Tipo de dato

Ejemplo

El tipo de dato *coordenada* podría estar representado por los *atributos* X, Y y Z, siendo los tres de tipo *entero*.

Las *operaciones* aplicables sobre coordenada podrían ser:

- **Distancia entre dos coordenadas (puntos)**
- **Desplazar un punto**
- **Mostrar el valor de X**
- **Mostrar el valor de Y**
- **Mostrar el valor de Z**

Conceptos Básicos

Tipo de dato

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

Ejemplo

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

- **Carácter 'A' en el tipo de dato *carácte*r, o**
- **Entero +65 en el tipo de dato entero.**

Conceptos Básicos

Tipo de dato

Puede clasificarse como *escalar* o *estructurado*.

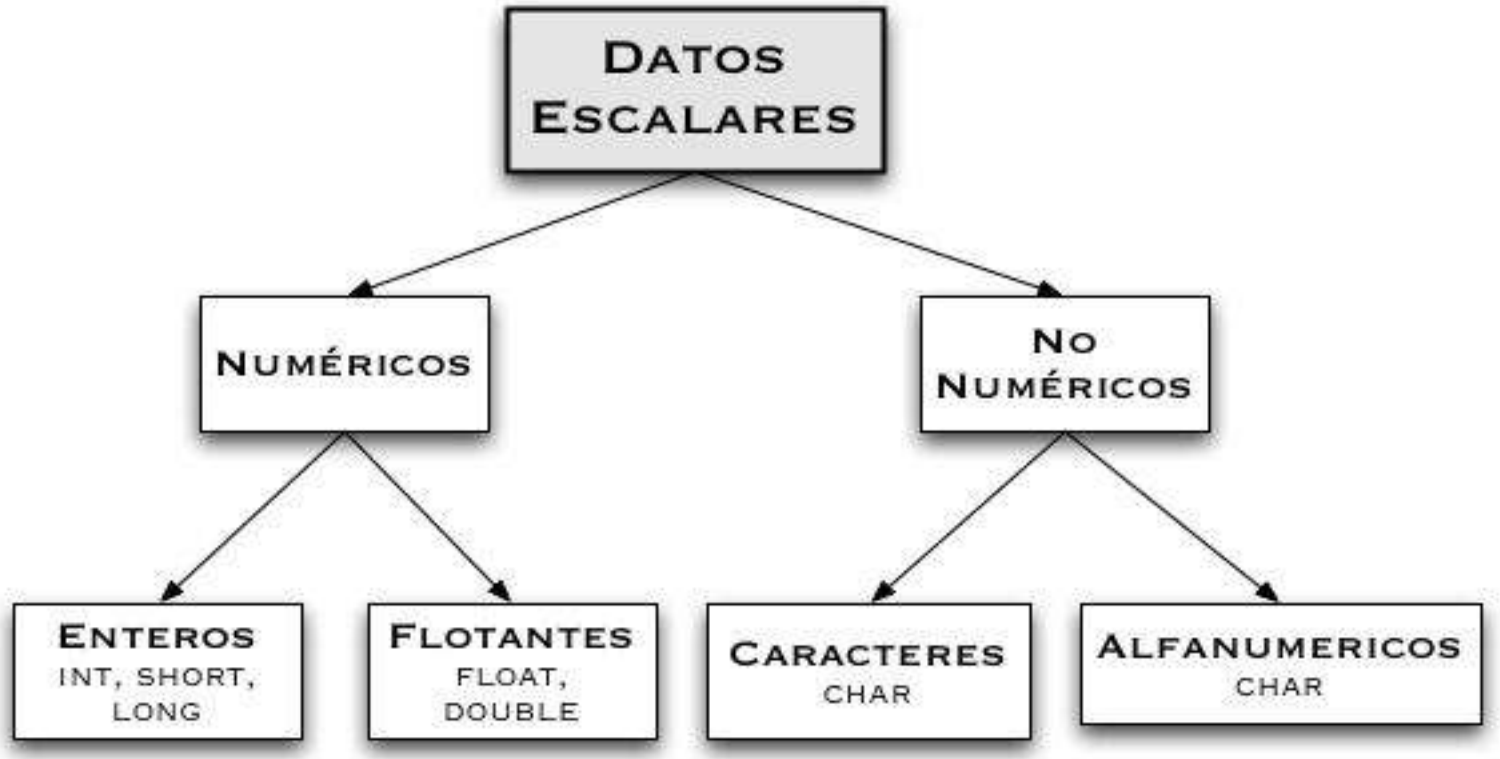
- ***Escalar o simple*: Aquel cuyo dominio presenta una propiedad de orden (entero, real, carácter, lógico).**
- ***Estructurado o compuesto*: Aquel que se define mediante composición de tipos de datos (vector, cadena de caracteres, matriz, registro).**

Tipos de Datos en C

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

En este curso vamos a estudiar algunos de los tipos de datos que ofrece el lenguaje de programación C.

Tipos de Datos Escalares en C



Tipos de Datos y Tamaños

char: un byte. Este tipo almacena un caracter según la tabla local.

int: un entero. Refleja el tamaño de los enteros en la máquina local.

float: punto flotante de precisión simple.

double: punto flotante de precisión doble.

Informacion sobre tipos

Si desea saber con exactitud sobre los tipos de datos que soporta su computador, busque en los archivos:

- limits.h
- float.h

Deberian estar en **/usr/include**

Constantes (literales)

Un entero es escrito como un número directo. Ej: 67890 es un entero.

Un entero largo se escribe con una l (ele) al final. Ej: 345l.

Un entero sin signo se escribe con una u al final. Ej: 454u.

Los modificadores de tipos se pueden mezclar. Ej: 123UL.

Un numero exponencial o con una f al final indica punto flotante.

Tipos de Datos Escalares (Numéricos) en C

Tipo Entero

Subconjunto finito de los números enteros. El tamaño del subconjunto dependerá del número de bits que correspondan al tipo de datos. Mientras más bits mayor será el tamaño del subconjunto.

Ej:

Entero corto (8 bits) llega hasta 255

Entero (en un Pentium IV, 32 btis) llega hasta el orden de los 4000 Megaunidades.

Tipos de Datos Escalares (Numéricos) en C

- *Tipo Entero*

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

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

Si el n = 16

$-2^{(16-1)}, \ldots, 0, \ldots, 2^{(16-1)} - 1$

$= -2^{(15)}, \ldots, 0, \ldots, 2^{(15)} - 1$

$= -32768, \ldots, 0, \ldots, 32767$

b) $0, \ldots, 2^n - 1$ ***enteros positivos***

Si n = 16

$0, \ldots, 2^{16} - 1 = 0, \ldots, 65535$

Tipo de Dato Entero en C

Tipo	# bits	Rango
Int	32	-2147483648 ... 2147483647
Long int	32	Idem
Unsigned int	32	0 ... 4294967295
Long long	64	0 ... 18446744073709551615

Tipos de Datos Escalares (Numéricos) en C

Tipo Real

- **Subconjunto de los números reales limitado no sólo en el tamaño, sino también en cuanto a la precisión. 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 0.437875×10^3 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

Tipo	Nro. bits	Rango	Precisión
float	32	$1.17x10^{-38}$ a $3.4x10^{38}$	6 dígitos decimales
double	64	$2.22x10^{-308}$ a $1.79x10^{308}$	15 dígitos decimales

Tipos de Datos Escalares (No-Numéricos)

Tipo Carácter

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

Un carácter 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' '*'

Cada caracter tiene un equivalente en el tipo entero.

Hay otros sistemas de codificación aparte del ASCII, ej UTF-8.

Tipo de Dato Carácter

Tipo	Nro. bits
char	8

Declaración de Variables

En C todas las variables que van a ser usadas en un programa deben ser declaradas al principio de la función o al principio del programa.

Los objetivos de la declaración de variables son:

- Asociar un tipo de dato y un identificador único a la variable. Esto también permite 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

tipo_de_dato lista_de_variables;

Ejemplos

```
int dia, mes, anio;
int edad;
unsigned int A = 347;
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 = ‘+’;
bool error = false;
```

Tipos de Variables

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> ⇐ importacion de encabezados
int main ()
 {
    double base = 10.5,  ⇐ declaración de vars.
           altura = 2.5,
           superficie;

    CUERPO DE LA FUNCION
}
```

Bloque Función

Declaración de Variables Locales: Ejemplo 2

```
#include <stdlib.h>   ⇐ biblioteca

char logico;   ⇐ variable global

   void main () {
        int i, j = 0, k;     ⇐ variable local
            char car1, car2;
        float dividendo, divisor;
        int x, y, z;
        char indicador = 1;

        CUERPO DEL PROGRAMA
   }
```

Declaración de Variables Locales: Ejemplo 3

```
#include <stdio.h>
void main ()
{
        double x = 2.0;
        printf("%f",x);        ⇐ imprime 2.0
        {
                printf("%f",x);        ⇐ imprime 2.0
                double x = 3.0;
                printf("%f",x);        ⇐ imprime 3.0
        }

        printf("%f",x);        ⇐ imprime 2.0
}
```

B_1

B_2

Declaración de Variables Locales: Ejemplo 4

```
void main( )
{
        int miEntero;
        miEntero = 503;
        {
                int otroEntero;
                otroEntero = miEntero;
        }
}
```

B_1

B_2

Declaración de Variables Globales y Locales: Ejemplo

```
#include <stdio.h>
int resultado;

int main()
{
     int x, y, z;

     printf("Introduzca tres numeros enteros: ");
     scanf("%i%i%i",&x,&y,&z);
     /* Calcular el producto de los tres numeros */
     resultado = x * y * z;
     printf("El producto de es %i", resultado);
     return 0;
}
```

Ejemplo:

```
#include <stdio.h>

int main()
{
      int grad_c, grad_f;
      int intervalo;

      intervalo = 20;
      grad_c = 0;
inicio:
      printf("Cantidad de grados centígrados: %i \n", grad_c);
      /* NO FUNCIONA ASI: grad_f = 5/9*(grad_c – 32) */
      grad_f = 5*(grad_c - 32)/9;
      printf("Conversión a grados farenheit: %i \n", grad_f);
      grad_c = grad_c + 20;
    if (grad_c <= 300)
                goto inicio;
      printf("Fin de la ejecución\n");
}
```

Constantes

Constante

Valor que no cambia durante la ejecución de un programa.

Puede ser numérica entera, numérica real, lógica, carácter.

Definición de Constantes (Globales)

#define identificador valor

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>

#define LONGITUD 32

int main ()
{
    int l1 = LONGITUD, x;
    x = LONGITUD + 1;
    printf("x = %i",x);
}
```

Declaración de Constantes (Locales)

const tipo_de_dato identificador = valor;

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

    Conjunto de sentencias             // Cuerpo de la funcion
}
```

Declaración de Constantes (Locales)

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

    CUERPO DEL PROGRAMA
}
```

Ejemplos

```
void main()
{
     const float pi;
     // Error: pi debe ser inicializada
     // en la declaracion

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

}
```

```
#include <stdio.h>

void main()
{
     const float x = 7.0;

     printf("%f",x);
}
```

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

```
#include <stdio.h>
#define CUATRO 4.0
 float radio;

int main ()
   {
      const float PI = 3.14159;
      float area, volumen;

      printf("De el radio de la esfera: ");
      scanf("%f",&radio);
      area = CUATRO * PI * radio * radio;
      printf("Area = %f\n",area);
      volumen = area * (radio/3);
      printf("Volumen = %f\n",volumen);
     return 0;
    }
```

Recuerde.

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

Ejercicios

Definir cada una de las siguientes constantes (consulte las fuentes apropiadas para los valores que se necesiten).

A) Como globales

B) Como locales

- **613** **- Número de Avogrado**
- **613.0** **- Masa del electrón (en Kg)**
- **-613** **- Diámetro atómico (en cm)**
- **'6'**
- **"PD-10"**
- **$-3.012x10^{15}$**
- **$17x10^{12}$**
- ***e***