Introducción a la Algorítmica y Programación (3300)

Prof. Ariel Ferreira Szpiniak - aferreira@exa.unrc.edu.ar
Departamento de Computación
Facultad de Cs. Exactas, Fco-Qcas y Naturales
Universidad Nacional de Río Cuarto

Teoría 2

Datos, tipos simples y compuestos (registro), variables, constantes.

Datos, tipos, objetos y valores
Introducción

Acciones formuladas

Romper seis huevos en un plato, Calentar el aceite en una sartén al fuego, teclear el primer número, teclear el segundo número, pulsar +, teclear 2.

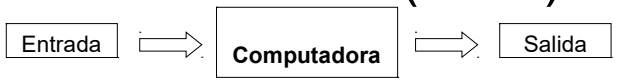
Objetos en un entorno

huevo nro.1, ..., huevo nro. 6, plato, sartén, primer número, segundo número, +, 2.

Características de un objeto

- **Nombre**: se utiliza para designarlo sin ambigüedad.
- **Utilización**: no intercambiable (se puede dividir con el objeto 2 pero no con el objeto plato)
- **Tipo**: característica común a todos los estados posibles del objeto. Los objetos huevo nro.1 y huevo nro. 2 son del tipo huevo. El primer número y el objeto 2 son de tipo numérico.
- **Valor**: en cada instante cada objeto tiene un valor. Este valor puede cambiar luego de la ejecución de una acción. Un objeto del tipo huevo representa en cada instante uno de los seis huevos. Cada viejo valor se destruye con el valor nuevo. Hay ciertos objetos que no cambian nunca de valor (2).

Datos, tipos, objetos y valores
Introducción (cont.)



Dato: representación de un objeto del mundo real mediante el cual podemos modelar aspectos del problema que se quiere resolver con un programa en una computadora.

Componentes de un programa
- *instrucciones*
- *estructuras para manipular datos*

♦ Las ***instrucciones*** representan las operaciones que ejecutará la computadora al ejecutar el programa.

♦ Los ***datos*** son los valores de información de los que se necesita disponer y en ocasiones transformar para ejecutar el programa.

▪ Conceptualmente, por una parte se tienen datos ***constantes***, datos que ***no cambian*** durante la ejecución del programa.

▪ Por otra parte, se pueden tener datos ***variables***, es decir que durante la ejecución del programa ***pueden cambiar.***

Datos, tipos, objetos y valores
Ejemplo

Determinar cantidad de objetos, su tipo, nombre y valor necesarios para resolver el siguiente problema:

"¿Cuánto ocupa una baldosa cuadrada de 20 centímetros de lado?"

Cantidad de objetos: dos

Tipo: número (ambos)

Nombre: lado, area

Valor:

lado=20

area= lado * lado

Objetos
Variables y Constantes

Variables

Definición: una **variable** es un objeto cuyo valor puede cambiar (no es invariable). Además, toda variable posee otros dos atributos:

- Un **nombre o identificador**, que sirve para designarla.
- Un **tipo**, que describe la utilización posible de la variable, es decir, el universo de valores posibles.

Cuando se declara una variable, se debe precisar su **nombre** y su **tipo**. Definir una variable es crear un objeto para el procesador. Si no se indica lo contrario, su valor es indeterminado.

Objetos
Variables y Constantes

Constantes

Definición: una **constante** es un objeto de valor invariable.

Es la realización de un valor de un tipo particular.

Cuando se declara una constante, se debe precisar su **nombre,** su **valor** y su **tipo**.

Definir una constante es crear un objeto para el procesador que no puede ser modificado mediante las instrucciones del programa.

Variables y Constantes
Declaración

Los nombres o identificadores están formados por letras, dígitos en cualquier orden y algunos símbolos especiales (excepto el primer carácter que debe ser una letra).

Convenciones que utilizaremos

Variables

Se escriben en letra minúscula. Si el nombre es largo se puede optar usar la notación camelCase

Ejemplo: velocMedia

Constantes

Se indican poniendo su nombre con letra inicial mayúscula e igualando a su valor. Notación PascalCase

Ejemplo: IvaMaximo

Tipos de datos

- Los algoritmos generalmente operan sobre datos de distinta naturaleza (números, letras, palabras, símbolos, etc.).
- Por lo tanto, los programas que implementan dichos algoritmos, necesitan alguna manera de representarlos.

Un *tipo de dato* es una clase de objetos de datos ligados a un conjunto de operaciones para crearlos y manipularlos.

Los tipos de datos se caracterizan por:

- Un rango de valores posibles,
- Un conjunto de operaciones realizables sobre ese tipo
- Una representación interna.

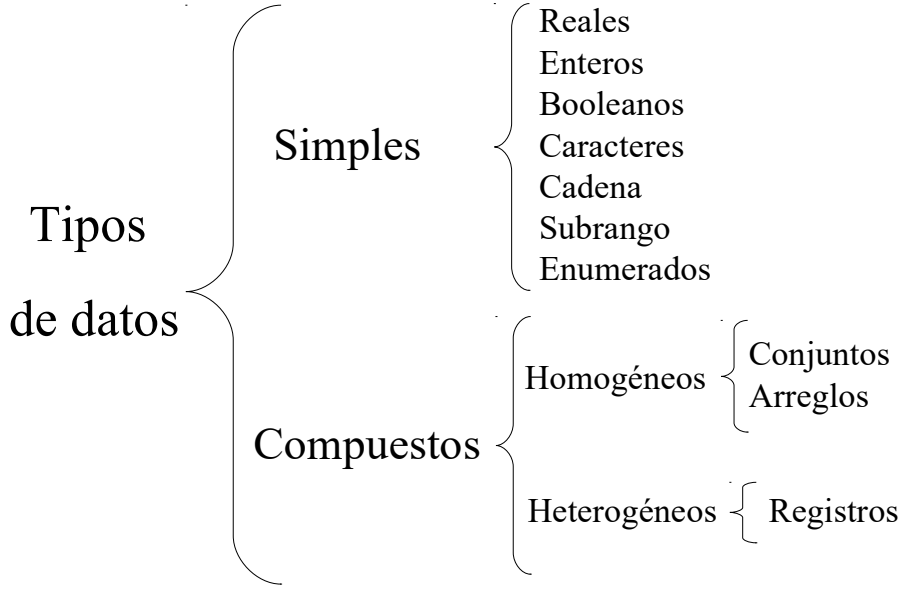
Por ello se dice que un tipo es el conjunto de valores posibles que puede tomar una variable.

Tipos de datos
Simples y Compuestos

Los objetos del mundo real generalmente son complejos. Por ello para poder representar esos objetos en programas que solucionen problemas se necesita de herramientas que faciliten dicha tarea.

Es por ello existen tipos de datos **Simples**, o Elementales, y **Compuestos**, o Estructurados. Estos tipos de datos permiten representar en una computadora las características principales de los objetos del mundo real.

Tipos de datos



Tipos de datos
Simples

Simples o Elementales

- **Numérico:** conjunto de valores numéricos que el procesador sabe tratar. Por ejemplo: 10 12,5 +22 -125,24. Según el procesador puede ser entero, real, etc.
- **Lógico:** conjunto de valores lógicos: **Verdadero** y **Falso.**
- **Carácter:** conjunto de caracteres que el procesador reconoce: letras minúsculas, mayúsculas, cifras y signos especiales. 'M' 'm' '5' '%'
- **Cadena:** conjunto de cadenas de caracteres que el procesador reconoce: "promNotas" "5mentarios".
- **Enumerados:** conjunto de valores definidos por el usuario
- **Subrango:** subconjunto de valores de un tipo determinado.

Tipos de datos
Compuestos

Compuestos

Agrupan varios valores para caracterizar una entidad particular. Por ejemplo un estudiante (nombre, apellido y documento). Un tipo compuesto es la unión de varios tipos elementales o compuestos.

Por ahora solo analizaremos el tipo compuesto denominado **Registro**. Más adelante veremos en detalle este tipo y el resto.

Tipos de datos simples
Tipo de dato numérico

El *tipo de dato numérico* es el conjunto de los valores numéricos que pueden representarse de dos formas:

- Enteros
- Reales

Tipos de datos simples
Tipo de dato numérico

Enteros

El *tipo de dato entero* es el tipo de dato numérico más simple de todos. Los elementos son del tipo: ..., -2, -1, 0, 1, 2, ...

• Dado que una computadora tiene memoria finita, la cantidad de valores que se pueden representar son finitos. Por esto existe un número entero máximo y otro mínimo.

• Hay sistemas de representación numérica que utilizan 16 dígitos binarios (*bits*) para almacenar en memoria cada número entero, permitiendo un rango de valores enteros entre -2^{15} y $+2^{15}$.

Otros sistemas utilizan 32 bits, por lo que el rango es entre -2^{31} y $+2^{31}$.

En general este máximo número entero se denomina MAXINT.

Tipos de datos simples
Tipo de dato numérico

Reales

Se debe tener en cuenta que el tipo de dato real tiene una representación finita de los números reales; dicha representación tiene una *precisión* fija, o sea, un número fijo de dígitos significativos.

La representación para números reales se denomina de *punto flotante*. Esta es una generalización de la conocida notación científica y consiste en definir cada número como una *mantisa* (parte decimal) y un *exponente* (posición de la coma).

Ejemplo: Sea el número 89412950000000000.

Su representación científica, en cualquier calculadora, es

8.941295×10^{16}

Tipos de datos simples
Tipo de dato numérico

Operaciones

Cada tipo de dato cuenta con un conjunto de operaciones posibles que se le pueden aplicar al mismo.

Las operaciones válidas para el tipo de dato numérico son:

suma (+), **resta** (-),
multiplicación (*),
división (/),
división entera (div),
módulo (mod).

Operadores aritméticos

igualdad (=), **desigualdad** (<>)
menor (<), **menor igual** (<=)
mayor (>), **mayor igual** (>=).

Operadores relacionales

Tipos de datos simples

Tipo de dato numérico

Operaciones

- No todas las operaciones definidas son aplicables a todos los tipos de datos numéricos.
- mod y div solo valen con operandos enteros y su resultado es entero.
- *Overflow:* cuando el resultado de la operación supera el máximo valor binario permitido por la representación.
- *Underflow:* cuando el resultado de la operación supera el mínimo valor binario permitido por la representación.
- Las expresiones que tienen dos o más operandos requieren reglas matemáticas que permitan determinar el orden de las operaciones.

Tipos de datos simples

Tipo de dato numérico

Operaciones

- El orden de precedencia para la resolución es:

1.operadores ***, /***

2.operadores **+, -**

3.operadores **div** (división entera) **y mod** (resto de la división entera)**.**

En caso que el orden de precedencia natural deba ser alterado, es posible la utilización de paréntesis dentro de la expresión.

Ejemplos

6 + 8 * 5	= 6 + 40	= 46
(6 + 8) * 5	= 14 * 5	= 70
5 * 2 + 7 + 4 * 3	= 10 + 7 + 12= 17 + 12	= 29

Tipos de datos simples

Tipo de dato lógico

El *tipo de dato lógico*, también llamado *booleano*, es un dato que puede tomar un valor entre un conjunto formado por dos posibles. Dichos valores son:

- **verdadero** (*true*)
- **falso** (*false*)

Se utiliza en situaciones donde se representan dos alternativas de una condición.

Por ejemplo, en el caso de tener que determinar si un valor es mayor que 10; la respuesta será *verdadera* en caso de serlo, de lo contrario será *falsa*.

Tipos de datos simples

Tipo de dato lógico

Operaciones

Los *operadores lógicos* o *booleanos* básicos son:

- negación (*no*),
- conjunción (y),
- disyunción (*o*).

El resultado de estas operaciones es el correspondiente a las conocidas *tablas de verdad*.

Tipos de datos simples

Tipo de dato lógico

Operaciones

Dados p y q datos lógicos, las tablas de verdad son:

p	no p
Verdadero	Falso
Falso	Verdadero

p	q	p y q
Verdadero	Verdadero	Verdadero
Verdadero	Falso	Falso
Falso	Verdadero	Falso
Falso	Falso	Falso

p	q	p o q
Verdadero	Verdadero	Verdadero
Verdadero	Falso	Verdadero
Falso	Verdadero	Verdadero
Falso	Falso	Falso

Tipos de datos simples

Tipo de dato lógico

Operaciones

Los operadores relacionales mencionados en el punto anterior, dan como resultado un valor lógico.

Ejemplos

(18 < 14)	da como resultado *falso*
(25 / 2.5 = 10)	da como resultado *verdadero*
no (8 > 3)	es *falso*.
(17 > 2) y (19 = 33)	es *falso*
(17 > 4) *o* (19 = 33)	es *verdadero*

Existe un orden de precedencia para los operadores lógicos:

1. operador ***no***
2. operador ***y***
3. operador ***o***

Tipos de datos simples

Tipo de dato caracter

Un tipo de dato caracter proporciona objetos de datos que contiene solo un carácter como su valor.

Este conjunto de valores se normalizó, entre otros, por un estándar llamado ASCII (American Standard Code for Information Interchange), el cual permite establecer un *orden de precedencia* entre los mismos.

- Letras mayúsculas: ‘A’, ‘B’, ‘C’, …, ‘Y’, ‘Z’
- Letras minúsculas: ‘a’, ‘b’, ‘c’, …, ‘y’, ‘z’
- Dígitos: ‘0’, ‘1’, ‘2’, …, ‘8’, ‘9’
- Caracteres especiales:‘!’, ‘@’, ‘#’, ‘$’, ‘%’, …

Se debe tener en cuenta que no es lo mismo el valor entero 0 que el símbolo carácter ‘0’.

Un valor del tipo de dato carácter es **solo uno** de los símbolos mencionados.

Tipos de datos simples

Tipo de dato caracter

Operaciones

Los operadores relacionales descriptos en el tipo de dato numérico, pueden utilizarse también sobre los valores del tipo de dato carácter.

Esto es, dos valores de tipo caracter se pueden comparar por =, <>, >, <, >=, <=. El resultado de cualquiera de ellos es un valor de tipo de dato lógico.

Ejemplos

(‘b’ = ‘B’) da como resultado *falso*
(‘c’ < ‘Z’) da como resultado *falso*
(‘c’ < ‘z’) da como resultado *verdadero*
(‘X’ > ‘5’) da como resultado *verdadero*
(‘ ‘ < ‘H’) da como resultado *verdadero*
(‘4’ = 4) no puede evaluarse

Funciones: Ord, Chr

Chr(64) devuelve ‘@’
Chr(65) devuelve ‘A’
Chr(97) devuelve ‘a’

Ord(‘@’) devuelve 64
Ord(‘A’) devuelve 65
Ord(‘a’) devuelve 97

Tipos de datos simples
Tipo de dato Cadena

Un tipo de dato cadena proporciona objetos de datos que contienen una serie de caracteres como su valor.

Las cadenas se representan como una secuencia de caracteres encerradas por comillas simples (apóstrofe). Los espacios en blanco también son considerados

Ejemplos:
'Eva'
'Achalay 550'
'estudiante universitario'

Tipos de datos simples
Tipo de dato Cadena

Operaciones

Las operaciones básicas entre cadenas son: *comparación, concatenación y asignación.*

• *Comparación:* Las comparaciones de las cadenas de caracteres se hacen según el orden de los caracteres en el código ASCII y con los operadores de relación. Las dos cadenas se comparan de izquierda a derecha hasta que se encuentran dos caracteres diferentes. El orden de las dos cadenas es el que corresponde al orden de los dos caracteres diferentes. Si las dos cadenas son iguales pero una de ella es más corta que la otra, entonces la más corta es menor que la más larga.

Ejemplos
'Alex' > 'Alas' {puesto que 'e' > 'a'}
'ADAN' < 'adan' {puesto que 'A' < 'a'}
'Damian' < 'Damiana' {'Damian' tiene menos caracteres}
'Lo gato' < 'Los gatos' {puesto que (blanco) < 's'}

Tipos de datos simples
Tipo de dato Cadena

Operaciones (cont.)

• *Concatenación:* La concatenación es un proceso de combinar dos o más cadenas en una sola cadena. El signo + se puede usar para concatenar cadenas.

***Ejemplos*:**

'UNIVERSIDAD '+'PUBLICA' = 'UNIVERSIDAD PUBLICA'

'patio'+'.'+'c'= 'patio.c'

• *Asignación*: Se puede asignar el valor de una expresión de cadena a una variable cadena.

***Ejemplos*:**

fecha ← 'lunes'

frase ← 'El próximo '+fecha+' inician las clases';

Tipos de datos simples definidos por el usuario
Enumerados

Son tipos de datos en los que el usuario establece los valores que componen el tipo. Esto se logra mediante la especificación de una lista de identificadores válidos.

Los identificadores deben comenzar con carácter, y no pueden repetirse.

El orden en que se definen es importante puesto que se pueden comparar. El menor de todos es el primero y el mayor es el último.

Marca = (Fiat, Renault, Peugeot, Ford) {tipo marcas de autos}

Dias = (Lun, Mar, Mie, Jue, Vie, Sab, Dom) {tipo días}

Genero = (Masculino, Femenino) {tipo género}

Tipos de datos simples definidos por el usuario
Subrango

Son tipos de datos definidos por el usuario. Se definen a partir de un ordinal estableciendo un límite inferior y uno superior.

Los tipos definidos mediante subrango admiten las mismas operaciones que el ordinal del cual proceden. Los subrangos se utilizan para dar una mayor legibilidad a los algoritmos.

1..12	{subrango del tipo entero}
A..Z	{subrango del tipo caracter}
Mie..Sab	{subrango del tipo dias definido por el usuario}

Tipos Compuestos

Los datos primitivos solo pueden almacenar un valor a la vez, por ejemplo, un entero o un real.

Para el caso en que sea necesario representar varios valores en una sola entidad es necesario utilizar datos no primitivos. Estos datos no primitivos se denominan *tipos compuestos, estructurados, estructuras de datos* o simplemente *estructuras*.

En las *estructuras* se almacenan colecciones de elementos del mismo tipo o de tipos diferentes.

Si todos los elementos son del mismo tipo se llaman *estructuras homogéneas*, en caso contrario se denominan *estructuras heterogéneas*.

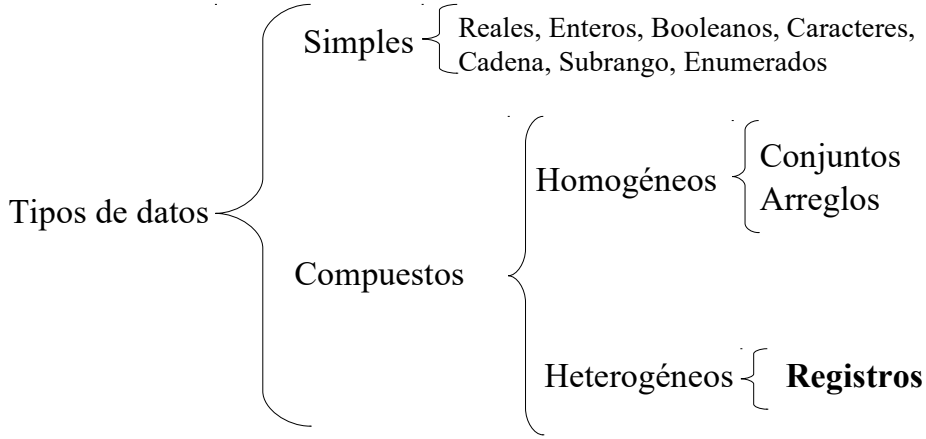
Definición: **es un conjunto de elementos reunidos bajo el mismo nombre colectivo.**

Tipos Compuestos

Un tipos compuesto (o estructurado) es un tipo de datos con las siguientes características:

- posee varias componentes, cada una de las cuales puede ser un tipo simple u otra estructura de datos;
- existe una relación entre los distintos componentes que la dota de un cierto significado y utilidad;
- se manipulan mediante una serie de operaciones.

Tipos de datos Registros

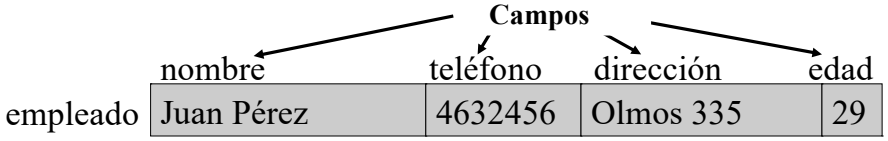


Registros

Un registro es un mecanismo mediante el cual se pueden agrupar elementos de varios tipos. De esta manera es posible representar o modelar entidades u objetos del mundo real, como por ejemplo un **empleado**.

Los registros están formados por **componentes**.

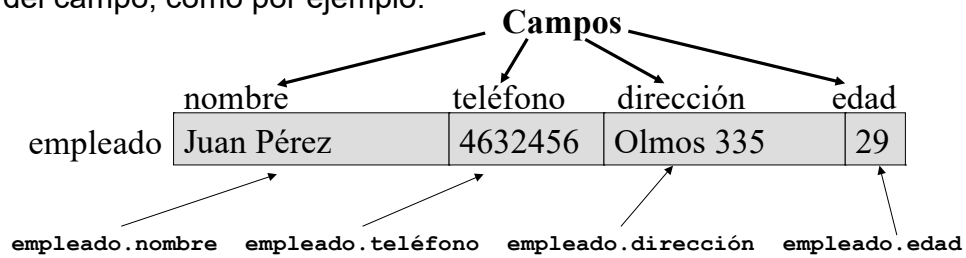
1. Los **componentes** de los registros pueden ser heterogéneos, por ejemplo, tipos de datos mixtos (strings, enteros, reales...).
2. Los **componentes** pueden ser simples o estructurados.
3. Los **componentes** de los registros se nombran con nombres simbólicos (identificadores) llamados ***campos***.



Registros
Selección de componentes

En Notación Algorítmica y en C la **selección de una componente** se realiza haciendo referencia al nombre del **campo.**

Esto se logra colocando el nombre del registro punto el nombre del campo, como por ejemplo:



De esta manera se puede acceder al contenido de cada campo, así como también se puede colocar información dentro de cada uno.

Lenguaje C - Tipos

Notación Algorítmica	**Lenguaje C**
Entero	-> int
Real	-> float
Caracter	-> char
Cadena	-> arreglo de char
Logico	-> int

\\cero es falso, cualquier otro valor es verdadero

< … >	-> struct

Notación algorítmica
Ejemplo de variables, constantes y asignación

```
Algoritmo CalcularIVA
Lexico
  Iva = 21 ∈ Z            //constante
  precioBruto ∈ R          //variable dato
  precioFinal ∈ R          //variable resultado
Inicio
  Entrada:precioBruto
  precioFinal ← precioBruto + (precioBruto*Iva)/100
  Salida:precioFinal
Fin
```

C

Ejemplo de variables, constantes y asignación

```
#include <stdio.h>
/* Lexico */

const float Iva=21; // Una forma de definir constantes
/* #define Iva 21 Otra forma de definir constantes*/

float precioBruto;        // variable dato
float precioFinal;       // variable resultado

void main(){
  //es separador de decimales es el punto, no la coma
  scanf("%f",&precioBruto);
  precioFinal = precioBruto + (precioBruto*Iva)/100;
  printf("El precio final es: %f \n", precioFinal);
}
```


Declaración y uso de registros

El registro anterior se puede declarar como:

TEmpleado = <

nombre ∈ Cadena,

telefono ∈ Cadena,

direccion ∈ Cadena,

edad ∈ Z^+

>

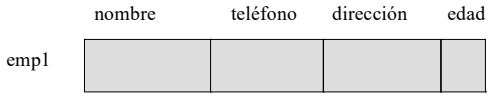
tipo

emp1 ∈ TEmpleado

variable

Declaración y uso de registros

Este es un ejemplo de carga del registro recientemente declarado y visualización de su contenido:




```
...
Entrada:emp1 // Obtener todos los datos personales del empleado
Salida:emp1 // Informar todos los datos personales del empleado
...
```


Declaración y uso de registros

Otra alternativa más detallada...


```
...
// Obtener los datos personales del empleado campo por campo
Entrada:emp1.nombre
Entrada:emp1.telefono
Entrada:emp1.direccion
Entrada:emp1.edad
// Informar los datos personales del empleado campo por campo
Salida:emp1.nombre emp1.telefono emp1.direccion emp1.edad
...
```

Declaración y uso de registros en C

```
typedef struct {
                char nombre[21];
                char telefono[21];
                char direccion[41];
                int edad;
                } TEmpleado;

TEmpleado emp1;
```


Declaración y uso de registros en C

Este es un ejemplo de carga del registro recientemente declarado y visualización de su contenido:


```
...
  printf("\n Ingrese Nombre: ");
  fgets(emp1.nombre,21,stdin);
  printf("\n Ingrese Telefono: ");
  fgets(emp1.telefono,21,stdin);
  printf("\n Ingrese Direccion: ");
  fgets(emp1.direccion,41,stdin);
  printf("\n Ingrese Edad: ");
  scanf("%d",&emp1.edad);
  printf("\n Los datos personales del empleado son: \n");
  printf("\n Nombre: %s \n", emp1.nombre);
  printf("\n Telefono: %s \n", emp1.telefono);
  printf("\n Direccion: %s \n", emp1.direccion);
  printf("\n Edad: %d \n", emp1.edad);
...
```


Ejemplo completo

```
#include <stdio.h>
typedef struct {
                 char nombre[21];
                 char telefono[21];
                 char direccion[41];
                 int edad;
               }TEmpleado;
TEmpleado emp1;
void main(){
  printf("\n Ingrese Nombre: ");
  fgets(emp1.nombre,21,stdin);
  printf("\n Ingrese Telefono: ");
  fgets(emp1.telefono,21,stdin);
  printf("\n Ingrese Direccion: ");
  fgets(emp1.direccion,41,stdin);
  printf("\n Ingrese Edad: ");
  scanf("%d",&emp1.edad);
  printf("\n Los datos personales del empleado son: \n");
  printf("\n Nombre: %s \n", emp1.nombre);
  printf("\n Telefono: %s \n", emp1.telefono);
  printf("\n Direccion: %s \n", emp1.direccion);
  printf("\n Edad: %d \n", emp1.edad);
}
```


Ejemplo

Ej. 1) Usando registros, definir el léxico para modelar la siguiente información de una persona.

- **Datos Personales**: nombre, apellido, dni (alfanumérico).
- **Dirección Completa**: calle, número, ciudad, provincia y código postal.

Ej. 2) Almacenar los datos de una persona en un registro.

Ej. 3) Mostrar los datos de una persona contenida en un registro.

Ej. 4) Almacenar y luego mostrar los datos de 3 personas.

Notación camelCase y PascalCase

camelCase es la práctica de escribir frases o palabras compuestas eliminando los espacios y poniendo en mayúscula la primera letra de cada palabra, excepto la primera. El nombre viene del parecido de éstas mayúsculas, entre las demás letras, con las jorobas de los camellos. Esta notación también se la conoce como lowerCamelCase.

ejemploDeCamelCase

PascalCase es la práctica de escribir frases o palabras compuestas eliminando los espacios y poniendo en mayúscula la primera letra de cada palabra. Esta notación también se la conoce como **UpperCamelCase**.

EjemploDePascalCase

Estos estándares son muy utilizados en varios lenguajes de programación.

Un ejemplo de su utilización es el lenguajes de programación Java donde los nombres de clase se escriben siguiendo PascalCase, mientras que para los nombres de métodos e instancias de clases se utiliza camelCase.

Notación Húngara

La notación húngara es un sistema usado normalmente para crear los nombres de variable. Es el sistema usado en la programación de algunos sistemas operativos, y programadores FORTRAN.

Consiste en prefijos en minúsculas que se añaden a los nombres de las variables, y que indican su tipo. El resto del nombre indica, lo más claramente posible, la función que realiza la variable.

Prefijo	Significado
b	Booleano
c	Carácter (un byte)
l	Entero largo de 32 bits
n	Entero de 16 bits

Ejemplos

nContador: la variable es un entero que se usará como contador.
bRespuesta: una variable booleana que almacena una respuesta.

Notación camelCase y PascalCase

Está demostrado que cuando se usa éste tipo de convenciones hay menos probabilidad de errores.

Se cometen menos errores cuando se escribe: CodigoPostal, que cuando se escribe codigopostal o CODIGOPOSTAL.

Algunos autores prefieren Codigo_Postal. Microsoft, en la plataforma .NET, abandonó los otros esquemas de nombres (particularmente la notacion húngara que usaba en MFC y que algunos desarrolladores utilizan en SQL: tbldirecciones) y se "paso" totalmente a PascalCase (CodigoPostal) o camelCase (codigoPostal).

Para la escritura de algoritmos y programas utilizaremos las dos notaciones, camelCase y PascalCase.

Ver la documentación sobre convenciones en Notación Algorítmica y C.

volver

Tabla de códigos ASCII

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

fuente: www.tablaascii.com.ar

Tabla de códigos ASCII Extendido

128	Ç	144	É	160	á	176	▒	193	┴	209	╤	225	ß	241	±
129	ü	145	æ	161	í	177	▒	194	┬	210	╥	226	Γ	242	≥
130	é	146	Æ	162	ó	178	▓	195	├	211	╙	227	π	243	≤
131	â	147	ô	163	ú	179	│	196	─	212	╘	228	Σ	244	⌠
132	ä	148	ö	164	ñ	180	┤	197	┼	213	╒	229	σ	245	⌡
133	à	149	ò	165	Ñ	181	╡	198	╞	214	╓	230	µ	246	÷
134	å	150	û	166	ª	182	╢	199	╟	215	╫	231	τ	247	≈
135	ç	151	ù	167	º	183	╖	200	╚	216	╪	232	Φ	248	°
136	ê	152	_	168	¿	184	╕	201	╔	217	┘	233	Θ	249	·
137	ë	153	Ö	169	_	185	╣	202	╩	218	┌	234	Ω	250	·
138	è	154	Ü	170	¬	186	║	203	╦	219	▉	235	δ	251	√
139	ï	156	£	171	½	187	╗	204	╠	220	▄	236	∞	252	_
140	î	157	¥	172	¼	188	╝	205	═	221	▌	237	ϕ	253	²
141	ì	158	_	173	¡	189	╜	206	╬	222	▐	238	ε	254	■
142	Ä	159	ƒ	174	«	190	╛	207	╧	223	▀	239	∩	255	
143	Å	192	└	175	»	191	┐	208	╨	224	α	240	≡		

fuente: www.tablaascii.com.ar

Bibliografía general

– **Tutoriales, manuales C:** en el sitio de la materia, sección Materiales.

– Biondi, J. y Clavel, G. "Introducción a la Programación. Tomo 1: Algorítmica y Lenguajes":
- Notación algorítimica (pags. 1 - 12)
- Entorno, Tipos, Variables, Constantes, Notación algorítmica (pags. 13 - 34)

– Scholl, P. y Peyrin, J.-P. "Esquemas Algorítmicos Fundamentales: Secuencias e iteración". (pags. 35 - 55)

– Wirth, N. "Algoritmos + Estruturas de Datos = Programas". (pags. 1-12).

– Quetglás, Toledo, Cerverón. "Fundamentos de Informática y Programación" http://robotica.uv.es/Libro/Indice.html
- Capítulo 3 pags. 81 a 90
- Capítulo 4

Bibliografía sobre registros

- Scholl, P. y J.-P. Peyrin, "Esquemas Algorítmicos Fundamentales: Secuencias e iteración", Barcelona, Ed. Masson, 1991.

- Lucas, M., J.-P. Peyrin y P. Scholl, "Algorítmica y Representación de Datos. Tomo 1: Secuencia, Autómata de estados finitos", Barcelona, Ed. Masson, 1985.

- Watt, David, "Programming Language Concepts and Paradigms", Prentice-Hall International Series in Computer Science (1990).

- Biondi, J. y G. Clavel, "Introducción a la Programación. Tomo 1: Algorítmica y Lenguajes", 2° ed., Barcelona: Masson, 1985.

- Clavel, G. y Biondi, J., "Introducción a la Programación. Tomo 2: Estructuras de Datos", 2° ed., Barcelona: Masson, 1985.

- De Guisti, A. "Algoritmos, datos y programas. Con aplicaciones en Pascal, Delphi y Visual Da Vinci. Prentice Hall.