LENGUAJE C

Tema 2 – Elementos de un programa

ELEMENTOS DE UN PROGRAMA

- Comentarios.
- Identificadores.
- Constantes.
- Variables.
- Operadores.
- Sentencias o instrucciones.

COMENTARIOS

Los comentarios en C pueden ocupar varias líneas y se encuentran delimitados entre /* y */. Estos son ignorados por el compilador.

```
int main()
{
   /* Esto es un comentario de varias
      líneas.*/
   return 0;
}
```

IDENTIFICADORES

- Se utilizan para nombrar variables, funciones, etiquetas y elementos definidos por el usuario.
- Los primeros seis caracteres deben ser significativos (distinguirse de otro similar) y máximo puede tener hasta 31 caracteres.
- El primer carácter debe de ser una letra o subguión. Posteriormente pueden ser letras, números, signos de subrayado.
- Existe diferencia entre mayúsculas y minúsculas.
- No pueden emplearse palabras reservadas como identificadores.
- No pueden emplearse nombres de funciones que ya existan en el programa o en la librería de funciones de C.
- No puede llamarse main.

Convenciones:

- Empezar los nombres de funciones y de variables con una letra minúscula.
- Las constantes escritas con #define van con mayúsculas como #define PI 3.1416
- Las palabras intermedias comienzan con mayúsculas. sumaMatrices
- Utilizar el subguión para separar palabras intermedias. suma_Matrices
- Emplear nombres cortos para optimizar. (i, j, k, cont)

DELIMITADORES

Son símbolos que permiten al compilador separar y reconocer las diferentes unidades sintácticas del lenguaje.

Los principales delimitadores son:

- ; es necesario para finalizar sentencias o declaraciones.
- , separa dos elementos consecutivos de una lista.
- () enmarca una lista de parámetros.
- [] enmarca la dimensión o el subíndice de una tabla.
- {} enmarca un bloque de instrucciones o una lista de valores iniciales.

CONSTANTES

- Se utiliza para asignar un identificador a una constante, cuyo valor no se modificará durante toda la ejecución del programa.

```
#define  PI  3.1416
#define  NCOLS  20
```

- El pre-procesador de C, sustituye la ocurrencia de PI por el valor 3.1416 en todo el programa antes de efectuar la compilación, del mismo modo se sustituye NCOLS por 20, o sea, no se guarda memoria para la misma.

VARIABLES

Una variable es una posición de memoria cuyo valor puede ser cambiado durante la ejecución del programa.
Inicialmente el valor de una variable es indeterminado.
Todas las variables deben de ser declaradas para ser utilizadas.

<tipo de dato> <identificador>;
<tipo de dato> <identificador> = <valor>;

```
int nro;
float area, radio,
   volumen;
char letra = 'a';
double iva = 0,21;
```

Las variables pueden ser declaradas:

- Fuera de las funciones: Variables globales
- Dentro de una función: Variable local
- En la cabecera de una función: parámetros formales de la función (es una variable local)

MODIFICADORES DE ACCESO

- Const permite asignar a una variable un valor constante, es decir que una vez asignado a dicha variable su valor no podrá ser modificado durante el programa.

const <tipo dato> <identificador> = valor;

```
const int a=10;
const char pais[]="ARGENTINA";
```

- Volatile permite cambiar el valor de una variable por medios no explícitamente especificados por el programa. Por ejemplo la dirección de una variable global que apunta a un puerto externo.

volatile <tipo dato> <identificador> = valor;

```
volatile unsigned char *puerto = 0x30;
```

TIPOS DE DATOS BÁSICOS

Existen 5 tipos de datos básicos

- Entero
  - Int
- Real
  - Float (Simple precisión)
  - Double (Doble precisión)
- Carácter
  - Char (representacion interna en ASCII)
- Tipo vacío que se utiliza cuando una función no devuelve ningún valor
  - Void
- El tipo de datos lógico se implementa como entero
  - Cualquier valor distinto de 0 es verdad
  - Cualquier valor igual a 0 es falso

El tamaño y el rango de cada uno depende del compilador

La función sizeof() permite conocer el tamaño de cada tipo

Ciertos tipos básicos admiten diversos modificadores:

- unsigned :Sólo valores positivos (sin signo).
- signed : Valores positivos y negativos (por omisión).
- long : Formato largo (solo enteros).
- Short: Formato reducido (solo enteros).

Ejemplos:

```
unsigned int
signed char
long int (usualmente
representado como long)
unsigned long int
```

CARACTERES ESPECIALES

\0	Fin de cadena
\n	Nueva línea
\t	Tabulado
\b	Retroceso
\r	Retorno de carro
\f	Salto de página
\\	Contra barra
\'	Apóstrofe (comilla simple)
\"	Comillas
\a	Audio de alerta

OPERADORES

Son palabras o símbolos que implican una acción sobre ciertas variables. Pueden ser unarios (1 variable), binarios(2 variables) o ternarios (3 variables).

- Operadores Aritméticos
- Operadores Relacionales
- Operadores Lógicos
- Operadores de Asignación
- Operadores de Bits
- Operadores de Dirección

OPERADORES ARITMÉTICOS

Operador	Nombre	Descripción
+	Suma	5+2 → 7
-	Resta	5-2 → 3
*	Multiplicación	5*2 → 10
/	División	5/2 → 2
%	Módulo	5%2 → 1
(tipo de dato)	“Cast” forzado	(double)5 → 5.0

OPERADORES RELACIONALES

Operador	Nombre	Descripción
==	Igual a	if (a==‘s’)
!=	Diferente de	if (a!=null)
>	Mayor que	if (a>0.5)
<	Menor que	if (a<2l)
>=	Mayor o igual que	if (a>=2f)
<=	Menor o igual que	if (a<=3)

OPERADORES LÓGICOS

Operador	Nombre	Descripción
&&	Y (AND)	if ((a>3) && (a<9))
||	O (OR)	if ((a==2) || (a==3))
!	NEGADO (NOT)	if (!(a==3)) es igual a if (a!=3)

Importante:

FALSO es igual a cero.

VERDADERO es diferente de cero.

OPERADORES DE ASIGNACIÓN

Operador	Abreviado	No Abreviado
=	a=2;	a=2;
++	n++; o ++n;	n=n+1;
–	n–; o –n;	n=n-1;
+=	n+=2;	n=n+2;
-=	n-=2;	n=n-2;
*=	n*=2;	n=n*2;
/=	n/=2;	n=n/2;
%=	n%=2;	n=n%2;

OPERADORES DE BITS

Operador	Nombre	Descripción
<<	Corrimiento a la izquierda	b=a>>2;
>>	Corrimiento a la derecha	b=a<<3;
&	Y (AND) entre bits	c=a&128;
|	O (OR) entre bits	c=a|0x0a;
~	Complemento A1	c=~a;
^	O exclusivo (XOR)	c=^a;

OPERADORES DE DIRECCIÓN

Operador	Nombre	Descripción
*	Operador indirección	Me da el valor que está almacenado en una dirección de memoria. También sirve para declarar un puntero.
&	Operador dirección	Me da la dirección de memoria de una variable.

Otros operadores:

- Evaluación condicional

```
( a > b) ? a : b;
```

- Tamaño en bytes

```
sizeof ( variable ); sizeof ( tipo );
```

PRECEDENCIA DE OPERADORES

( ) [ ] ->	Alta prioridad
! ~ + - ++ – & * sizeof	Unarios
* / % + -	Aritméticos
<< >>	Corrimiento de bits
< <= > >= == !=	Relacionales
& ^ | && || ?:	Bits / Lógicos / Condicional
= *= /= %= += -= &= ^= |= <<= >>=	Asignación
,	Evaluación

SENTENCIAS (INSTRUCCIONES)

- Una sentencia es una instrucción o expresión en C que tiene una consecuencia. Pueden ser asignaciones, operaciones, llamadas a funciones.
- Todas las sentencias terminan con el signo de punto y coma ;
- Pueden ser simples o compuestas. Las compuestas van entre llaves:

```
{
    sentencia1;
    sentencia2;
    :
    sentencian;
}
```

- **Sentencias de Selección:**
  - **if – else, switch – case, ?:**
- **Sentencias de Repetición:**
  - **do – while, while, for**
- **Sentencias de Salto:**
  - **return, break, continue.**

SENTENCIAS DE SELECCIÓN IF - ELSE

```
if (expresión)
    sentencia;
 else
    sentencia;
```

Nota: una expresión en C es todo aquello que regresa un valor. Como por ejemplo una condición lógica, operaciones aritméticas, llamadas a funciones, una variable, una constante (numérica, carácter, etc.).

```
if (expresión)
  {
    sentencia1;
    sentencia2;
  }
else
  if (expresión2)
      sentencia;
  else
      sentencia;
```

SENTENCIAS DE SELECCIÓN SWITCH - CASE

```
switch(expresión)
{
     case 1:   sentencias;
                  break;
     case 2:   sentencias;
                  break;
      :
     case n:   sentencias;
                   break;
     default: sentencias_default;
                   break;
}
```

SENTENCIAS DE REPETICIÓN WHILE Y DO - WHILE

WHILE

```
while (expresión)
    sentencia;

while (expresión){
   sentencia1;
   …
   sentencian;
}
```

Do - WHILE

```
do sentencia;
while(expresión);

do{
   sentencia1;
   …
   sentencian;
}while(expresión);
```

SENTENCIAS DE REPETICIÓN FOR

```
for (inicialización; condición; incremento)
acción;

for (inicialización; condición; incremento){
acciones;
}

for(expr1;expr2;expr3)
Sentencia;
```

- expr1 y expr 3 son asignaciones o llamadas a funciones.
- expr2 es una expresión relacional.

- Si expr2 no esta presente se considera siempre verdadero →bucle infinito, unica salida break o return.

SENTENCIAS DE SALTO BREAK Y CONTINUE

- break:
  - Permite controlar las salidas de los bucles.
  - Provee una salida temprana para for, while, do, switch.

```
for(i=0;i<1000;i++){
   //hacer algo
   if(kbhit()) break;
}
```

- continue:
  - Es utilizada cuando la parte del bucle que sigue es complicada.
  - Provoca la próxima iteración del bucle cerrado a ser ejecutado inmediatamente.

BIBLIOGRAFÍA & LICENCIA

- Textos tomados, corregidos y modificados de diferentes páginas de Internet, tutoriales y documentos, entre los que destaco el libro: *C/C++ Curso de programación*, 2da Ed, Javier Ceballos, Alfaomega Ra-Ma.
-
- Autor:

Matías E. García

Prof. & Tec. en Informática Aplicada

www.profmatiasgarcia.com.ar

info@profmatiasgarcia.com.ar