Introducción al Lenguaje C

Lenguajes de Programación

Años y Evolución

1951-55 & Assembly, Lenguajes de expresiones (A0)

1956-60 & Fortran, Algol 58, COBOL, Lisp

1961-65 & Snobol, Algol 60, COBOL 61, Jovial, APL

1966-70 & Fortran 66, COBOL 65, Algol 68, Simula, BASIC, PL/I

1971-75 & Pascal, COBOL 74, C, Scheme, Prolog

1976-80 & Smaltalk, Ada, Fortran 77, ML

1981-85 & Turbo Pascal, Smaltalk 80, Ada 83, Postcript

1986-90 & Fortran 90, C++, SML, Haskell

1991-95 & Ada 95, TCL, Perl, Java, Ruby, Python

1996-00 & Ocaml, Delphy, Eiffel, JavaScript

2000- & D, C#, Fortran 2003, Starlog, TOM, ...

Clasificación por sus Estilos (Paradigmas)

- **Declarativo**:
  - Programación funcional (haskell, lisp)
  - Programación lógica (prolog)
- **Imperativo**:
  - Por procedimientos (pascal, C, C++)
  - Programación orientada a objetos (java, c++, eiffel)
- **Concurrencia**: transversal a los demás (java, C, erlang)

Historia del Lenguaje C

- Fue inventado por Dennis Ritchie en un DEC-PDP-11 en los Laboratorios BELL,
- En 1983 el Instituto de Estándares Americanos estableció un estándar que definiera al lenguaje C, conocido como ANSI C.
- Los principales compiladores de C llevan implementado el estándar ANSI C.
- Utilizado para la implementación del sistema UNIX, y muchas aplicaciones complejas.

Características del Lenguaje C

- Se cataloga como un lenguaje de nivel medio, puesto que combina elementos de lenguajes de alto nivel (Fortran, Pascal, Basic, etc.) con la funcionalidad del lenguaje ensamblador.
- Posee sólo 32 palabras reservadas (para escribir las sentencias), definidas por el comité ANSI. Deben escribirse en minúsculas (a diferencia de Pascal)

32 Palabras Reservadas

char	void	default	return
int	if	break	auto
float	else	continue	extern
double	do	goto	register
long	while	struct	const
short	for	union	static
signed	switch	enum	volatile
unsigned	case	typedef	sizeof

Elementos de un Programa

- Comentarios.
- Bloques
- Identificadores.
- Constantes.
- Variables.
- Declaración y definición de Funciones.
- Operadores.
- Sentencias o instrucciones.

Comentarios

- **Comentarios multi-línea**
  - **Comienzan con /*** y terminan con ***/**
  - No puede anidarse un comentario dentro de otro.
- Comentarios de **una sola línea** (C++, Java, Delphi).
  - Comienzan al principio de la línea con ***//***
- **Ejemplo:**

```
//Esto es un comentario
/* Esto también es
      un  comentario */
```

Bloques

- **Declaración de bloques:**
  - Comienzan con **{** y terminan con **}**.
  - En pascal: **begin** y **end**.
- **Ejemplo:**

```
{
    printf("ingrese un número  ");
    scanf("%f",&num);
 }
```

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.

Identificadores

- 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 palabras intermedias comienzan con mayúsculas.

```
sumaMatrices
```

- Utilizar el subguión para separar palabras intermedias.

```
suma_Matrices
```

- Emplear nombres cortos para optimizar. **(i, j, k, cont)**

Constantes

- Constantes de carácter.

  Ej. ‘a’, ‘0’, ‘\0x5’, ‘\0’, ‘\n’, ‘\t’, ‘$’, ’\\’, NULL

- Constantes enteras.

  Ej. 5, +5, -5, \05, \0x5, 5L, 5U, 5lu, etc.

- Constantes reales.

  Ej. 0.5f, 0.5, 5e-01f, 5.0e-01, (float)5, etc.

- Constantes de texto (Cadenas o “Strings”)

  “Esto es una cadena…”

#define

- Se utiliza para asignar un identificador a una constante.

  #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.

Variables y sus Declaraciones

- Una variable es una localidad de memoria cuyo valor puede ser cambiado durante la ejecución del programa.
- Todas las variables deben de ser declaradas para se utilizadas.

<tipo de dato> <identificador>;

- Ejemplo:

```
int a;
float area, radio, volumen;
```

Declaración y Definición de Funciones

- En C no existen los procedimientos como en Pascal, aunque pueden emularse. Son todas funciones.
- Declaración de una función:

  <tipo de dato> nombre (<tipo de dato>, <tipo de dato>, ...)

- Definición de una función:.

  <tipo de dato> nombre (<tipo de dato> <ident>, <tipo de dato> <ident>, ...){

  ...

  }

- Ejemplo de declaración de definición respectivamente:

```
double modulo3D( double, double, double );
double modulo3D( double x, double y, double z ){
  return(sqrt(x*x+y*y+z*z);
}
```

Pasaje de Parámetros de las Funciones

- C pasaje de parámetro por copia:

  <tipo de dato> nombre (<tipo de dato> <ident>, <tipo de dato> <ident>, ...)

- C no tiene pasaje de parámetros por referencia, pero se puede “imitar”.

const

- Es un modificador de acceso que me 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;

- Ejemplo:

```
const int a=10;
```

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 Dirección
- Operadores de Bits

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

Operadores Relacionales

Operador	Nombre	Descripción
==	Igual a	if (a==‘s’)
!=	Diferente de	if (a!=’s’)
>	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.

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 (Instrucciones)

- Sentencia de Asignación.
  - Operador de asignación = . Ej a=3;
- Sentencias de Selección.
  - if – else, switch – case, ?:
- Sentencias de Repetición.
  - do – while, while, for
- Sentencias de Salto.
  - return.
- Entrada/Salida .
  - scanf y printf.

Operadores de Asignación

Operador	Abreviado	No Abreviado
=	a=2;	a=2;
++	n++; (++n;)	n=n+1;
--	n--; (--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;

if-else

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

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

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 (númérica, carácter, etc.).

Operador Condicional ?:

(expresión)? sentencia1 : sentencia2;

expresión? sentencia1 : sentencia2;

Se ejecuta:

sentencia1 si **expresión** = **verdadero**

sentencia2 si **expresión** = **falso**.

Es un operador ternario y puede utilizarse para asignar variables:

a = (**expresión**)**? sentencia1:sentencia2**;

switch-case (case of de Pascal)

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

do while (repeat de Pascal)

```
do {
  ...
} while (expr);
```

while (while do de Pascal)

```
while (expr) {
  ...
  ...
}
```

Entrada/Salida estándar

- **scanf: leer desde el teclado.**
- **printf : mostrar por pantalla.**

```
{  ...
   printf("ingrese un número: ");
   scanf("%f",&num);
   res=cubo(num);
   printf("%f al cubo es: %f",num,res);
    ...
}
```

Algunas sentencias de salto

- **return: retorna el resultado de una función.**

```
int sucesor(int  n) {
   int aux;
   aux=n+1;
   return aux;
}
```

El compilador *gcc*

- **Ejemplos:**

  gcc hola.c

  compila el programa en C hola.c, genera un archivo ejecutable a.out. Ejecuta: ./a.out

  gcc -o hola hola.c

  compila el programa en C hola.c, genera un archivo ejecutable hola. Ejecuta: ./hola

  gcc -c hola.c

  no genera el ejecutable, sino el código objeto, en el archivo hola.o. Si no se indica un nombre para el archivo objeto, usa el nombre del archivo en C y le cambia la extensión por .o.

  gcc -c -o objeto.o hola.c

  genera el código objeto indicando el nombre de archivo.

- **Crear una biblioteca estática:**

  ar -r libejemplo.a hola.o hola2.o ...

Ejemplo

```
#include <stdio.h>
#include <math.h>
#define VALOR 5

double modulo3D( double, double, double);
double mod3; /* Variable global */

int main(int argc, char **argv){
  int x, y, z;
  x=y=z=VALOR;
  mod3=modulo3D(x,y,z);
  printf("\nEl módulo es: %lf",mod3);
  return 0;
}

double modulo3D( double x, double y, double z ){
  return(sqrt(x*x+y*y+z*z);
}
```

Dos programas equivalentes en C y Pascal

Programa en C:

```
/* Programa para multiplicar dos numeros */

#include <stdio.h>


/* Esta funcion devuelve la suma de los enteros
a y b */

int sumar(int a, int b)

{

   return a + b;

}

int a, b;

/* Linea principal del programa */

int main(void)

{

   printf("Introduce dos numeros enteros y pulsa
enter: ");

   scanf("%d %d", &a, &b);

   printf("La suma de %d y %d es %d\n", a, b,
sumar(a,b));

   return 0;

}
```

Programa en Pascal:

```
(* Programa para multiplicar dos numeros *)

program multiplicar_dos_numeros;


{ Esta funcion devuelve la suma de los enteros a
y b }

function sumar(a, b : integer) : integer;

begin

     sumar := a + b

end;

{ Linea principal del programa }

var

   a, b : integer;


begin

     write('Introduce dos numeros enteros y pulsa
enter: ');

     readln(a,b);

     writeln('La suma de ', a, ' y ', b, ' es ',
sumar(a,b))

end.
```

Programa en C que usa una biblioteca hecha en C

```c
#include <stdio.h>
extern int codificar(int x);
/* Archivo: prueba.c */
int main(void)
{
   int a, b;
   printf("Introduce un entero: \n");
   scanf("%d", &a);
   b=codificar(a);
   printf("La codificación de: %d es %d \n", a, b);
   return 0;
}
```

```c
/* Biblioteca. Archivo:
codificar.c */


int codificar(int x)
{
   return (x + 10)*3;
}
```

Pasos de compilación y ejecución:

1.gcc -c codificar.c

2. ar -r libmylib.a codificar.o

3. gcc -o prueba-biblio prueba.c -L. -lmylib

4. ./prueba-biblio

Referencias

http://www.tutorialspoint.com/cprogramming/c_quick_guide.htm

- **http://www.fismat.umich.mx/mn1/manual/node2.html #SECTION00260000000000000000**
- **Capítulo 7 del apunte de la asignatura “Análisis Comparativo de Lenguajes”. Archivo acl-book.pdf**