Introducción al Lenguaje de Programación 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

Lenguaje de Programación C

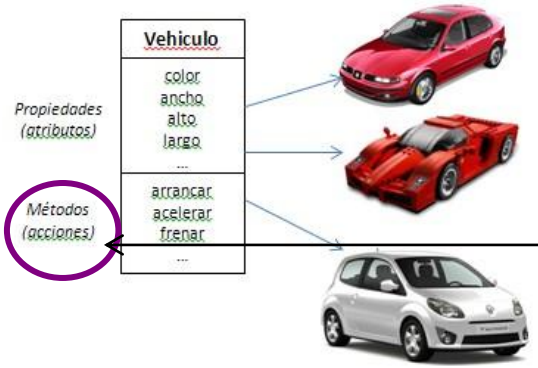
- Fue diseñado e implementado por *Brian Kernighan y Dennis Ritchie* en 1972, a partir de los lenguajes BCPL (1967) y B (1970)
  - Su desarrollo está estrechamente vinculado al del sistema operativo UNIX
- Combina características de los lenguajes de alto nivel (*sentencias de control y manipulación de d*atos) y de los lenguajes de bajo nivel (*manejo de bits, bytes, palabras y apuntadores*)
- Es independiente del *hardware* (es portable)

Enfoques de Programación

- ✓ Programación **estructurada**
  - ✓ *evolucionó en los 70s*
  - ✓ *Dahl, Dijkstra y Hoare*

1

- ✓ Programación **orientada a objetos**
  - ✓ *OOP: Object- Oriented Programming*
    - ✓ *evolucionó en los 80s - C++*



Los **objetos** se describen a través de una parte estructural y de una parte de comportamiento (**métodos**)

Los métodos se **implementan** usando programación estructurada (1)

Programación Estructurada

- Enfoque disciplinado que permite escribir programas estructurados, utilizando las siguientes tres **estructuras de control** bien definidas:
  - **Secuencial** (*asignación, lectura, escritura*)
  - **Decisión** o **selección** (*simple, doble, múltiple*)
  - **Repetición** (*repita-mientras, hacer-mientras, repita-para*)

- Los programas estructurados son fáciles de probar, depurar y modificar

- **Programación orientada a acciones** donde la unidad básica es la función - ***imperativa***

Lenguaje de Programación C

- ✓ *Características generales*
  - ✓ Simple
  - ✓ Sintaxis Flexible
  - ✓ Flujo de control estructurado
  - ✓ Tipos de datos variados
  - ✓ Amplia variedad de operadores
- ✓ *Elementos sintácticos*
  - ✓ Palabras clave
  - ✓ Identificadores
  - ✓ Delimitadores
  - ✓ Comentarios

Lenguaje de Programación C

Palabras clave o palabras reservadas

Aquellas que tienen un significado especial para el compilador y que están reservadas para uso especial del lenguaje de programación

Palabras reservadas en C

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

Lenguaje de Programación C

✓ Identificadores

- ✓ Nombres que permiten señalar, mencionar o hacer referencia a los diferentes objetos manipulados por un programa
- ✓ Deben resultar significativos, sugiriendo lo que representan
- ✓ No pueden ser palabras reservadas
- ✓ Los compiladores de **C**/C++ reconocen hasta un máximo de treinta caracteres, aunque el identificador puede tener cualquier longitud

Lenguaje de Programación C

✓ Identificadores

- ✓ Se componen de una serie de caracteres que pueden ser letras, dígitos o el carácter de subrayado (_)
- ✓ El primer carácter NO PUEDE SER un dígito
- ✓ Las letras mayúsculas y las minúsculas son diferentes

Lenguaje de Programación C

Ejemplos:

Identificadores válidos

A1, a1, sueldo_base, entero1, entero2, NOMBRES, nombres, notas, Contador, P, Q, bandera_Venezuela, sueldoNeto, cedulaIdentidad

Identificadores inválidos

1A, 1a, #sueldo_base, 1_entero, +NOMBRES, ;Contador, =bandera

Lenguaje de Programación C

✓ Delimitadores

- ✓ Signos especiales que permiten al compilador separar y reconocer las diferentes *unidades sintácticas* del lenguaje:
  - ✓ **;** (*punto y coma*). Se considera el de *terminación*, que es necesario cuando finaliza cada una de las sentencias o declaraciones
  - ✓ **,** (*coma*). Separa dos *elementos consecutivos* de una lista
  - ✓ **( )** (*paréntesis*). Enmarca una lista de *parámetros*, *expresiones* o *condiciones*
  - ✓ **[ ]** (*corchetes*). Enmarca la dimensión o el subíndice de un *arreglo*
  - ✓ **{ }** (*llaves*). *Enmarca un bloque* de sentencias o una lista de valores iniciales

Lenguaje de Programación C

Comentarios

- Permiten que el programador **documente** sus programas
- Sirven para facilitar la legibilidad de un programa.
- Son ignorados por el compilador

Tipos de **comentarios**

- *Comentarios iniciales*: objetivo general del programa
- *Comentarios en cada línea*: pasos cruciales del programa

Lenguaje de Programación C

Comentarios en C

En C los comentarios **empiezan** por los caracteres ***/**** y **terminan** con los caracteres ****/***

Pueden comprender varias líneas y estar distribuidos de cualquier forma,

todo aquello que está entre el

- ***/**** (*inicio del comentario*) y el
- ****/*** (*fin del comentario*)

es **ignorado por el compilador**

Comentarios en C

```
/* Esto es un comentario simple. */

/* Esto es un comentario más largo,
    distribuido en varias líneas. El
    texto se suele alinear por la izquierda. */

/*****************************************
* Esto es un comentario de varias        *
* líneas, encerrado en una caja para     *
* llamar la atención                     *
*****************************************/
```

Características de los Comentarios

- Deben ser coherentes con el programa
- Deben ser relevantes
- Deben mantenerse al día
- Siempre deben enriquecer el programa: conceptos, gráficos, relaciones entre partes, etc.

Programas en C

- ✓ Las *primeras tres líneas son comentarios*: nombre del programa, fecha de creación y objetivo del programa
- ✓ El programa consiste de la función *main*

  void main()
- ✓ Todo programa debe tener una función *main*
- ✓ La llave que abre **{** indica el **comienzo** del cuerpo de la función, mientras que la llave que cierra **}** indica el **final** del cuerpo de la función

```
{
    int a, b, c;

    c = a + b;
}
```

Programas en C

int a, b, c;

- ✓ Línea que le dice al compilador *que se van a usar en el programa* ***tres variables enteras a, b y c***
- ✓ El compilador reserva espacio en memoria para estas variables
- ✓ *Toda variable* usada en un programa *debe ser declarada* y se le debe asignar un *tipo de dato*
- ✓ **Sentencias ejecutables** del programa :

```
a = 3;
b = 5;
c = a + b;
```

Programa en C

Nuestra primera función en C

```
void main( )
{
   … las instrucciones en lenguaje C se escribe aquí!
 }
```

Programas en C

Sintaxis formal 1

```
/* Comentarios*/
Declaración de importaciones
Definición de constantes                    /* Opcional*/
Definición de tipos                         /* Opcional*/
Declaración de prototipos                   /* Opcional*/
Declaración de variables globales           /* Opcional*/

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

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

Definición de funciones                     /* Opcional*/
```

Programas en C

Sintaxis formal 2

```
/* Comentarios*/
Declaración de importaciones
Definición de constantes                    /* Opcional*/
Definición de tipos                         /* Opcional*/
Declaración de variables globales           /* Opcional*/

Definición de funciones                     /* Opcional*/

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

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

Atención - importante!!!!

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 hacemos con los datos?
¿En qué orden (cuándo se lo hacemos)?
¿Cuántas veces?

Conjunto de sentencias

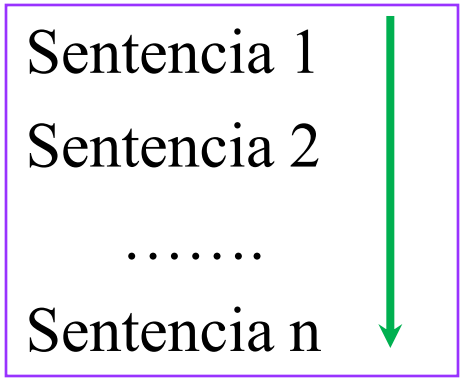
Estructuras de control

- El proceso de diseño del algoritmo, y posterior codificación del programa, consiste en definir las acciones o sentencias que resolverán el problema

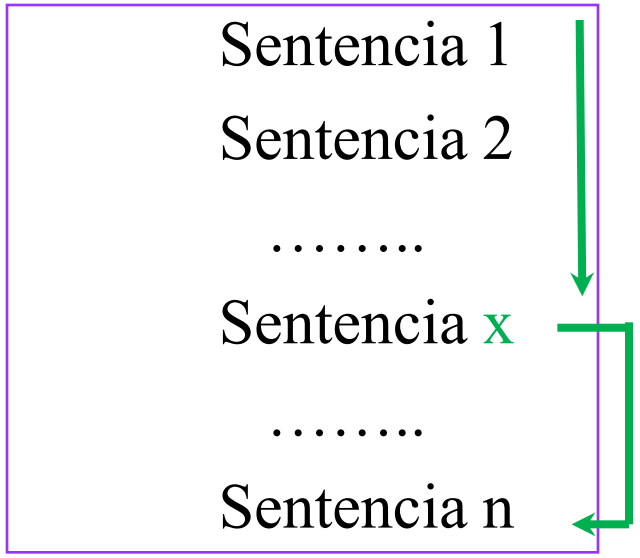
- En general, en un programa, las sentencias son ejecutadas una después de la otra, en el orden en que aparecen escritas

Estructuras de Control

Programa lineal: Las sentencias se ejecutan secuencialmente en el orden en que aparecen escritas



Programa no lineal: Se interrumpe la secuencia mediante sentencias de bifurcación



Tipos de Estructuras de Control

Atención:

La *programación estructurada* utiliza ***tres*** **estructuras de control**

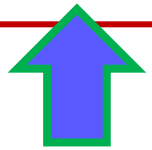
Todos los programas pueden ser escritos en términos de estas estructuras de control (*Bohm y Jacopini*)

- **Estructuras secuenciales**
- **Estructuras de decisión o selección**
- **Estructuras de repetición**

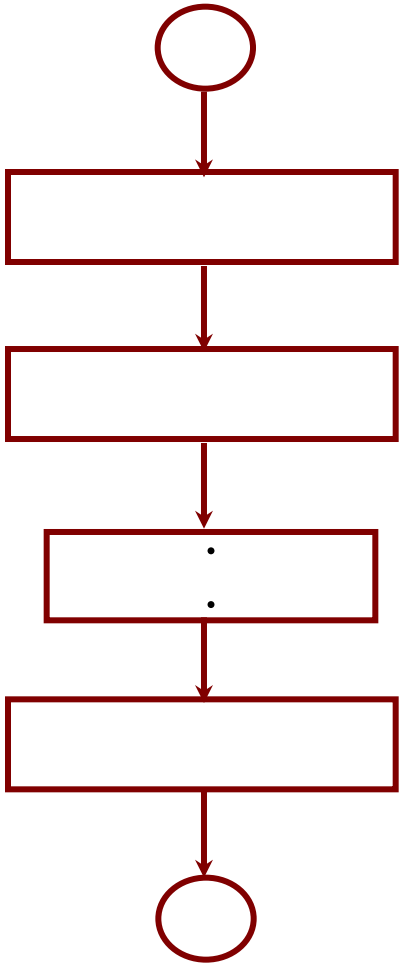
Estructuras Secuenciales

Se ejecutan en secuencia sin posibilidad que la sentencia siguiente a ejecutar pueda ser otra diferente de la que sigue en la secuencia

Tipo de sentencia	**Pseudocódigo** inglés	**Pseudo-código en** *español*	**Código en en C** / C++
Comienzo de proceso	**begin**	**Inicio**	{
Fin de proceso	**end**	**Fin**	}
Entrada (lectura)	**read**	**Leer**	**scanf** /cin
Salida (escritura)	**write**	**Escribir**	**printf** / cout
Asignación	A ← 5 ó A = 5	A ← 5 ó A = 5	A = 5



Estructuras Secuenciales



Ejecución secuencial

Punto único de entrada o salida

Estructura secuencial

Sentencia de Entrada (*Lectura*)

Permite leer determinados valores (datos de entrada) y asignarlos a determinadas variables
Los datos de entrada se introducen a la computadora mediante los dispositivos de entrada (teclados, unidades de disco, etc.)

Notación algorítmica

Leer (lista de variables de entrada)
Leer (a, b, c)

Notación en C

scanf ("%cadena de control", &Arg1, &arg2, &…argn);

scanf *("%d %i %d", &a, &b, &c);*

Sentencia de **Entrada/Salida** (*Lectura*)

Especificadores de formato

%c	carácter
%d	entero decimal con signo
%i	entero decimal con signo
%e	notación científica con e minúscula
%E	notación científica con E mayúscula
%f	coma flotante
%g	usar &e o %f el que sea más corto
%G	usar &E o %F el que sea más corto
%o	octal sin signo
%s	cadena de caracteres
%u	enteros decimales sin signo
%x	hexadecimales sin signo (letras minúsculas)
%X	hexadecimales sin signo (letras mayúsculas)
%p	mostrar un puntero
%n	el argumento es un puntero entero al que se le asigna el numero de caracteres escritos – un puntero a una variable
%%	imprimir %

Sentencia de Salida (*Escritura*)

Permite escribir los resultados de un programa

La salida puede aparecer en un dispositivo de salida (*pantalla, impresora, etc.*)

Notación algorítmica

Escribir (mensajes y/o variables de salida)

Escribir ("El resultado es", resultado)

Notación en C

printf ("%variables de control formato", lista de argumentos) ;

printf ("El resultado es %tipo" , resultado);

Ejemplo 1

```
/*  Primer programa en C  */

#include <stdio.h>

void main()

{  /* Comienzo del programa principal */

   printf (“Programando en C\n”);

 } /* Fin del programa principal */
```

Ejemplo 2

```
/*otro programa en C*/

#include <stdio.h>

void main()
{   /* Comienzo del programa principal*/
    printf (“Mi nombre es Judith\n”);
    printf (“¿Cual es el tuyo ?\n”);
    scanf (“%s”, &nombre);
    printf (“mi nombre es %s”, nombre);
}   /* Fin del programa principal*/
```

Ejercicios

1

```
/* Programa que dibuja los contornos de un
     cuadrado con + */

#include <stdio.h>

void main()
{    /* Comienzo del programa principal*/
     ..............
     ..............

}    /* Fin del programa principal*/
```

2 *Hacer un programa en C* para dibujar la letra **E** utilizando ***