

LENGUAJE C

Tema 3 – Elementos de un programa II

ELEMENTOS DE UN PROGRAMA II

- ❖ Funciones.
- ❖ Procedimientos.
- ❖ Entrada / Salida de datos.
- ❖ Variables locales / globales.
- ❖ Programación con archivos múltiples.

FUNCIONES

Además de las funciones de biblioteca, el programador puede definir sus propias funciones que realicen determinadas tareas.

- ❖ **Función:** Secuencia de instrucciones agrupadas bajo un mismo nombre que realizan una tarea determinada.
- ❖ La función se ejecutará tantas veces como se la llame mediante su nombre.
- ❖ **Ventajas:**
 - ❖ Facilita la reutilización de código, aumentando la productividad del programador.
 - ❖ Descomposición de un problema en subproblemas más sencillos → se disminuye la complejidad del problema.
 - ❖ El uso de funciones mejora la estructura del programa, haciéndolo más legible y entendible.
- ❖ **Elementos de una función:**
 - ❖ **Nombre:** identificador de la función por el cual es invocada.
 - ❖ **Argumentos o parámetros:** datos que se le pasan a la función para que opere.
 - ❖ **Valor de retorno:** resultado de haber procesado la función.

DECLARACIÓN Y DEFINICIÓN

Para poder hacer uso de una función es necesario que ésta esté definida o declarada con antelación.

- ❖ Declaración de la función: Únicamente la cabecera o prototipo de la función (antes de main):

Tipo del resultado

```
int suma (int, int);
```

Tipos de los
parámetros

- ❖ Definición de la función: Todo el código de la función, su implementación (luego de main):

Necesario
para devolver
valor

```
int suma (int a, int b)
{
    int resultado;
    resultado = a + b;
    return resultado;
}
```

PASO DE PARÁMETROS A FUNCIONES

- ❖ En la definición de la función se especifica la lista de parámetros o argumentos que recibe y sus tipos.
- ❖ Puede no tener ningún argumento
- ❖ La sintaxis de la lista de argumentos es:
(tipo1 argumento1, tipo2 argumento2, ...)
- ❖ Al invocar (llamar) a la función se le deben pasar tantos argumentos como reciba y del tipo correcto.
- ❖ Existen dos formas de paso de parámetros:
 - **Por valor.** Se pasa como parámetro un valor (puede ser una constante, una variable, el resultado de una operación ...) este valor es utilizado en la función pero no es modificado su original.
 - **Por referencia.** Se pasa como parámetro la dirección en memoria de una variable. Esta variable podrá ser modificada desde dentro de la función. Los vectores y matrices también se pasan por referencia.

PASO DE PARÁMETROS A FUNCIONES 2

PASO POR VALOR

```
void intercambiar(int x, int y)
{
    int temp;
    temp = x;
    x = y;
    y = temp;
}

int main ()
{
    int a=5, b=6;
    intercambiar(a,b);
    printf("el valor de a: %d, el
valor de b: %d", a, b);
    return 0;
}
```

PASO POR REFERENCIA

```
void intercambiar(int *px, int *py)
{
    int temp;
    temp = *px;
    *px = *py;
    *py = temp;
}

int main ()
{
    int a=5, b=6;
    intercambiar(&a,&b);
    printf("el valor de a: %d, el
valor de b: %d", a, b);
    return 0;
}
```

PROCEDIMIENTOS

- ❖ Las funciones **void** permiten definir funciones que no devuelven nada
- ❖ El tipo **void** también permite escribir funciones que no reciben parámetros.

```
Void menu (void)
{
    printf ("Menu de opciones\n");
    printf (" (1) ingresar datos");
    printf (" (2) calcular valores");
    ...
}
```

LLAMADA A FUNCIONES

- ❖ Para que se ejecute una función, esta debe ser invocada mediante su nombre, seguido por los argumentos entre paréntesis.

nombre(arg1, arg2, ...);

- ❖ Los parámetros pueden ser variables, constantes, expresiones o el resultado de llamadas a otras funciones.
- ❖ Si la función invocada devuelve un valor, este podrá ser almacenado en una variable o utilizado como operando en alguna expresión.

variable = función2();

ENTRADA / SALIDA DE DATOS

- ❖ Las funcionalidades de entrada/salida en C no pertenecen a las palabras reservadas del lenguaje. Son funciones de librería (stdio.h), por ejemplo:
 - Entrada: `scanf()`.
 - Salida: `printf()`.

```
#include <stdio.h>
int main()
{
    float base, altura;
    printf("Teclee datos: ");
    scanf ("%f %f", &base, &altura);
    printf("La superficie es %f \n", base * altura);
    return 0;
}
```

FUNCIONES DE SALIDA DE DATOS

- ❖ **putchar(c);**
- ❖ Muestra el carácter c (tipos int o char) por pantalla en la posición actual del cursor. Su valor se interpreta como el código ASCII del carácter que se quiere mostrar.
- ❖ Ejemplos:
 - `putchar('a');` /* El carácter 'a' */
 - `putchar(97);` /* También el carácter 'a' */
 - `putchar('\n');` /* Salto de línea */
- ❖ **puts(cadena);**
- ❖ Muestra por pantalla una cadena de caracteres a partir de la posición actual del cursor. Tras mostrar la cadena salta a la línea siguiente
- ❖ Ejemplos:
 - `char cadena[]="Esto es una cadena";`
 - `puts(cadena);`
 - `puts("Esto es un ejemplo de puts");`

FUNCIONES DE SALIDA DE DATOS 2

- ❖ **printf(control, argumentos);**
- ❖ Muestra por pantalla datos de diferentes tipos según se especifique.
- ❖ control es una cadena de control que tiene dos clases de elementos, siempre entre comillas:
 - Caracteres normales que se imprimirán por pantalla directamente.
 - Caracteres especiales de formato que definen el modo en que se visualizarán los argumentos. Siempre empiezan con el carácter %.
- ❖ **Argumentos**
 - una lista de cero o más elementos separados por comas, generalmente variables o expresiones.
 - Para cada argumento deberá haber una secuencia especificando el formato deseado en control.

%-+0n.mX

(Lo mínimo a poner sería %X
Lo demás es opcional)

- : Alinear a la izquierda

+ : Incluir siempre el signo (+ ó -)

0 : Alinear relleno con ceros en lugar de espacios

n : Valor entero que indica el mínimo número de caracteres a mostrar

.m : m es un valor entero que indica el número de dígitos decimales a mostrar

X : Letra(s) que indica(n) el tipo de dato que se quiere mostrar

FORMATOS PARA PRINTF

Formato/control	Expresión	Resultado
%d %i	entero	entero decimal con signo
%u	entero	entero decimal sin signo
%o	entero	entero octal sin signo
%x %X	entero	entero hexadecimal sin signo
%f	real	real en notación punto
%e %E %g %G	real	real en notación científica
%c	carácter	carácter
%p	puntero	dirección de memoria
%s	string	cadena de caracteres
%ld %lu ...	entero largo	entero largo (distintos formatos)
%%		Imprime %

```
printf("%5.2f", 3.456);  
printf("%-10s", "hola");  
printf("%-5.2f", 3.452);
```

```
printf("%-0d", 336);  
printf("%f", 1234.56);  
printf("%e", 1234.56);
```

FUNCIONES DE ENTRADA DE DATOS

- ❖ `c = getchar();`
- ❖ Lee un carácter desde el teclado y devuelve su código ASCII, que puede almacenarse en una variable.
- ❖ Ejemplo:
 - `char c;`
 - `c = getchar();`
- ❖ `gets(cadena);`
- ❖ Lee una cadena de caracteres introducida desde el teclado por el usuario y la almacena en la variable `cadena`.
- ❖ Se almacenan caracteres hasta que se encuentra un salto de línea ('\n') (por ejemplo porque se ha pulsado la tecla intro.), que NO se almacena.
- ❖ Ejemplo:
 - `char cadena[20];`
 - `gets(cadena);`

Funciones de
entrada de
<conio.h>

La función `getch` lee un carácter del teclado sin visualizarlo en el monitor (sin eco); la función `getche` lee y visualiza en monitor.

FUNCIONES DE ENTRADA DE DATOS 2

- ❖ **scanf(control, variables);**
- ❖ Lee datos por teclado del tipo indicado.
- ❖ En control solamente deben aparecer caracteres de control.
- ❖ Las variables han de ir precedidas por el símbolo & (excepto las cadenas de caracteres), que es la forma de indicar en C que van a ser modificadas.
- ❖ El formato de la cadena de control es: %X
- ❖ Toma como delimitador los espacios en blanco y saltos de línea.
- ❖ No se puede usar para leer cadenas que contengan espacios.
- ❖ **fflush(stdin):** limpia el buffer cuando se leen caracteres y números de forma alterna.

FORMATOS PARA SCANF

Formato/Control	Tipo de dato
\n%c	char
%s	Cadena de caracteres
%d	int
%hu	unsigned short int
%ld	long int
%lu	unsigned long int
%f	float
%lf	double
%u	unsigned int
%hd	short int
%x	hexadecimal
%e	notación científica
%p	puntero

```
int a,*pa;  
float x;  
char c;
```

```
scanf("%d",&a); /* Lee  
un entero y lo  
almacena en a */
```

```
scanf("%f %c",&x,&c); /*  
Lee x y c */
```

```
scanf("%d",pa); /*  
PELIGROSO */
```

```
pa=&a; scanf("%d",pa);  
/* OK. Lee a */
```

VARIABLES LOCALES/GLOBALES

❖ **Variables locales**

- Argumentos y
- Variables definidas dentro del cuerpo de la función

❖ Las variables locales se crean en el Stack!

❖ La variable *i* de una función es **DISTINTA** de la *i* de otra (función)!.
❖

Variables globales

- Definidas fuera de las funciones (incluida main)

❖ Las variables globales se crean en el heap

❖ **“Alcance” (scope) de las variables:**

- Las locales son solo accesibles dentro de la función
- Las globales son comunes a todas las funciones (incluida main)

MODIFICADORES DE ACCESO

❖ La declaración de variables acepta los siguientes modificadores:

- `static` (locales o globales): El valor de la variable se conserva entre llamadas. Comportamiento similar a una variable global.
- `register` (locales): La variable es almacenada siempre (si es posible) en un registro de la CPU (no en memoria).
- `Auto` (locales): Las variables locales son auto por defecto. Dejan de existir cuando acaba la función donde están definidas.
- `Extern` : (globales): hace visible una variable local desde otro archivo.

Fichero1.c	Fichero2.c
<pre>int i; void f(void) { ... }</pre>	<pre>void f(void); extern int i; void g(void) { i = 26; f(); }</pre>

PROGRAMACIÓN EN MÚLTIPLES ARCHIVOS

Resulta conveniente, en general, escribir un programa en varios archivos. Un único archivo es considerado como principal y en él se implementa la función `main()`. Los demás archivos deben ir en parejas de archivo-cabecera/archivo-implementación. En el ejemplo, `principal.c` es el archivo principal, `biblioteca.h` es el archivo de cabecera y `biblioteca.c` es el archivo de implementación.

miBiblioteca.h

Prototipos de funciones
variables globales
definición de estructuras

miBiblioteca.c

```
#include "miBiblioteca.h"
```

implementación de las
funciones de miBiblioteca

Principal.c

```
#include "miBiblioteca.h"
```

otras funciones y declaraciones
implementación de la función `main()`
usando las funciones de miBiblioteca

WEBGRAFÍA & 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.
- ❖ Este documento se encuentra bajo Licencia Creative Commons 2.5 Argentina (BY-NC-SA), por la cual se permite su exhibición, distribución, copia y posibilita hacer obras derivadas a partir de la misma, siempre y cuando se cite la autoría del Prof. Matías E. García y sólo podrá distribuir la obra derivada resultante bajo una licencia idéntica a ésta.

❖ Autor:

Matías E. García

Prof. & Tec. en Informática Aplicada

www.profmatiasgarcia.com.ar

info@profmatiasgarcia.com.ar

 creative
commons



www.profmatiasgarcia.com.ar