



UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE
INGENIERÍA

MI. Marco Antonio Martínez
Quintana

Estructura de Datos y
Algoritmos

Asignatura:

Estructura de Datos y Algoritmos 1



Actividad Lunes

#1 Repaso: Lenguaje C



Alumna

Citlali Cuahtepitzi Cuatlapantzi

Fecha

(09/Junio/2021)



Lenguaje C

El repaso fue hecho con el material antes realizado del mismo tema con algunas modificaciones*

ESTRUCTURA DE UN PROGRAMA

/*Programa que...*/	Comentarios
#include <stdio.h>	Declaración de archivos de cabecera (librerías)
#include <conio.h>	
#include <math.h>	
#include <string.h>	
#include <ctype.h>	
	Prototipo de las funciones
	Declaración de variables globales y constantes
main()	Programa principal
{	
	Declaración de constantes (locales)
	Declaración de variables (locales)
	Cuerpo del programa (estructuras de control)
}	
	Declaración de funciones
{	

}	

TIPOS DE DATOS

Tipo	Significado	Ancho en bit	Rango de valores (IBM PC)
char	carácter	8	-128 a 127
int	entero	16	-32,768 a 32,767
short	entero corto	16	-32,768 a 32,767
long	entero largo	32	-2,147,483,648 a 2,147,483,647
unsigned char	carácter sin signo	8	0 a 255
unsigned	entero sin signo	16	0 a 65,535
unsigned short	entero corto sin signo	16	0 a 65,535
unsigned long	entero largo sin signo	32	0 a 4,294,967,295
enum	enumerado	16	0 a 65,535
float	real	32	-3.4E-38 a 3.4E+38
double	real doble	64	1.7E-308 a 1.7E+308
Long double	real doble largo	64	1.7E-4932 a 1.7E+4932

OPERADORES

Aritméticos:			Lógicos:		Relacionales:		De asignación compuesta:		
							Ejemplo	Equivalencia	
+	Suma	a+b		o	<	Menor que	+=	s+ = 10	s=s+10
-	Resta	a-b	&&	y	>	Mayor que	-=	r- = 3	r=r-3
*	Producto	a*b	!	no	<=	Menor o igual que	*=	m* =4	m=m*4
/	División	a/b			>=	Mayor o igual que	/=	d/ = 2	d=d/2
%	Módulo (residuo de la división entera)	a% b (Residuo de a entre b)			!=	Distinto a	%=	x%= 33	x=x%33
pow	Potencia	Pow(a,b) (a elevado a la b)			==	Igual que			

INCREMENTO Y DECREMENTO

++	++i	Se incrementa i en 1 y a continuación se utiliza el nuevo valor de i en la expresión en la cual esté i
++	i++	Utiliza el valor actual de i en la expresión en la cual esté i y después se incrementa en 1
--	--i	Se decrementa i en 1 y a continuación se utiliza el nuevo valor de i en la expresión en la cual esté i
--	i--	Utiliza el valor actual de i en la expresión en la cual esté i y después se decrementa en 1

PRIORIDAD DE LOS OPERADORES ARIMÉTICOS		SECUENCIAS DE ESCAPE PARA PRINTF MÁS COMUNES	
Prioridad	Operación	Secuencia	Significado
1ª.	(), []	\n	Nueva línea
2ª.	!	\t	Tabulador
3ª.	*, /, %	\b	Espacio atrás
4ª.	+, -	\r	Retorno de carro
5ª.	<, >, <=, >=	\f	Comienzo de página
6ª.	=, !=	\'	Comilla simple
7ª.	&&,	\''	Comilla doble
8ª.	=, +, -, *, /, %=	\\	Barra invertida
		\xdd	Código ASCII en notación hexadecimal (cada d representa un dígito)
		\ddd	Código ASCII en notación octal (cada d representa un dígito)

PALABRAS RESERVADAS		FORMATOS PARA PRINTF Y SCANF		
abs()	Calcula el valor absoluto	Formato	printf()	scanf()
char	Tipo de dato carácter	Carácter simple	%c	%c
case	Si se cumple un caso	Cadena	%s	%s
default	Ninguna opción de la selectiva múltiple	Entero decimal con signo	%d	%d
typedef	Crea un nuevo nombre para un tipo de dato ya definido	Entero decimal con signo	%i	
for	Estructura repetitiva o de ciclo	Entero decimal, hexadecimal u octal		%i
int	Tipo de dato entero	Punto flotante (notación decimal)	%f	%f o %e
}	Fin del programa o de un bloque	Punto flotante (notación exponencial)	%e	%f o %e
do	Estructura repetitiva	Punto flotante (%f o %e, el más corto)	%g	
printf	Imprime en pantalla	Entero decimal sin signo	%u	%u
puts	Imprime una cadena	Entero hexadecimal sin signo	%x	%x
{	Inicio del programa o de un bloque	Entero octal sin signo	%o	%o
scanf	Lee una variable	Entero largo	%li	%li
gets	Lee una cadena de caracteres			
clrscr	Borra el contenido de la pantalla			
while	Estructura repetitiva			
void	Valor nulo			
main	Programa principal			
sqrt	Calcular raíz cuadrada			
float	Tipo de dato real			
struct	Registro o estructura			
return	Regresa valor a otra función			
break	Terminar el caso			
switch	Estructura selectiva múltiple			
if	Estructura selectiva			
else	La parte falsa de la selectiva			

INSTRUCCIONES DE SELECCIÓN

Sentencia “if”

```
if (condición)
{
    * * *
}
```

Sentencia “if- else”

```
if (condición)
{
    * * *
}
else
{
    * * *
}
```

Anidamiento o escalonamiento “if-else-if”

```
if (condición 1)
    ***
Else
    if(condición 2)
        ***
    else if (condición 3)
        ***
    else
        ***
```

De ser una única instrucción se pueden omitir corchetes

Sentencia “switch – case “

```
switch (expresión entera o carácter)
{
    case ‘a’: Instrucciones para a;
        ***
        break;
    case ‘b’: Instrucciones para b;
        ***
        break;
    case ‘c’: Instrucciones para c;
        ***
        break;
    ...
    default: Instrucciones para default;
        ( opcional de no ser ninguna de las anteriores)
}
```

INSTRUCCIONES DE REPETICIÓN

Bucle “for”

```
for( expr_ini(s) ; cond ; inc(s) )
{
    ***
}
```

expr_ini(s) : expresión de asignación para iniciar variable.
cond: expresión relacional o lógica (simple o compuesta).
inc(s); define como cambiarán las variables de control.

Bucle “while”

```
while (condición)
{
    ***
}
```

De ser una única instrucción Se pueden omitir corchetes.

Bucle “do while”

```
do
{
    ***
}
while (condición);
```

(La condición lleva ;)

ARREGLOS

Arreglos unidimensionales (vectores o listas)

```
tipo_dato identif_arreglo [tam_arreglo];  
tipo_dato identif_arreglo [tam_arreglo]={valores};
```

tipo_dato: tipo de dato de cada elemento del arreglo.
identif_arreglo: nombre que representa a todo el arreglo.
tam_arreglo: cantidad de elementos que contiene el arreglo.

```
int num[4]={ 9,8,2,4};          (Ubicación grafica)
```

9	8	2	4
0	1	2	3

Arreglos de caracteres

```
char cad[]={ Hola};          (Ubicación grafica)
```

No es necesario definir el tamaño, el compilador siempre añade un carácter nulo al final.

H	o	l	a	\0
0	1	2	3	4

- **gets:** Introduce una cadena de caracteres en el teclado hasta que se encuentra un carácter \n. Se agrega al final del arreglo un carácter de terminación NULL

```
gets(variable_cadena);
```
- **scanf:** Lee una cadena. El argumento correspondiente es un apuntador a un arreglo del tipo char, que es lo suficiente extenso para contener la cadena y un carácter de terminación NULL.

```
scanf("%s", &variable_cadena);
```

Arreglos bidimensionales (matrices o tablas)

```
tipo_dato identif_arreglo [tam_fila] [tam_col];  
tipo_dato identif_arreglo [tam_fila] [tam_col]={valores};
```

tam_fila= total de filas
tam_col= total de columnas

```
int a[3] [3]={ 1,2,3,4,5,6,7,8,9};          (Ubicación grafica)
```

a	0	1	2
0	1	2	3
1	4	5	6
2	7	8	9

Lectura

```
for ( i=0 ; i<filas ; i++ )  
for( j=0 ; i<columnas ; j++ )  
scanf( "%d", &mat [i][j] )
```

Impresión

```
for ( i=0 ; i<filas ; i++ )  
for( j=0 ; i<columnas ; j++ )  
printf( "%d", mat [i][j] )
```

FUNCIONES

- **Tipo_de_retorno:** es el tipo del valor devuelto por la función, o, en caso de que la función no devuelva valor alguno, la palabra reservada void.
- **Nombre_de_la_función:** es el nombre o identificador asignado a la función.
- **Lista_de_parámetros:** es la lista de declaración de los parámetros que son pasados a la función. Éstos se separan por comas. Debemos tener en cuenta que pueden existir funciones que no utilicen parámetros.
- **Cuerpo_de_la_función:** está compuesto por un conjunto de sentencias que llevan a cabo la tarea específica para la cual ha sido creada la función.
- **Return expresión:** mediante la palabra reservada return, se devuelve el valor de la función, en este caso representado por expresión.

Los **prototipos** de las funciones que se utilizan en un programa se incluyen generalmente en la cabecera del programa y presentan la siguiente sintaxis:

```
tipo_de_retorno nombre_de_la_función(lista_de_parámetros)
```

Definición

```
tipo_de_retorno nombre_de_la_función(lista_de_parámetros)  
{  
sentencias;  
}
```

Por ejemplo:

```
int cubo(int base)  
{  
int potencia;  
potencia = base * base * base;  
return potencia;  
}
```

ARCHIVOS

```
#include<stdio.h>
int main(void){
FILE *Apun_a_Archivo; /*Declara apuntador a Archivo */
Apun_a_Archivo=fopen("Archivo.txt","w"); /*Se crea un Archivo.txt y se le asigna su dirección Apun_a_Archivo */
putc('B',Apun_a_Archivo); /*Se escribe una letra al archivo abierto*/
fclose(Apun_a_Archivo); /*Se cierra el archivo creado */
}
```

Funciones		“MODOS” de abrir un archivo	
Nombre	Función	Modo	Significado
fopen()	Abre un archivo.	“r”	Abre un archivo existente sólo para lectura.
fclose()	Cierra un archivo.	“w”	Abre un nuevo archivo sólo para escritura. Si existe un archivo con el NombreDelArchivo especificado, será destruido y creado uno nuevo en su lugar.
fgets()	Lee una cadena de un archivo.	“a”	Abre un archivo existente para añadir, agrega información al final del archivo. Se creará un archivo nuevo si no existe un archivo con el NombreDelArchivo especificado.
fputs()	Escribe una cadena en un archivo	“r+”	Abrir un archivo existente tanto para lectura como para escritura.
fseek()	Busca un byte específico de un archivo.	“w+”	Abre un archivo nuevo para lectura y escritura. Si existe un archivo con NombreDelArchivo especificado, será destruido y creado uno nuevo en su lugar.
fprintf()	Escribe una salida con formato en el archivo.	“a+”	Abre un archivo existente para leer y añadir. Se creará un nuevo archivo si no existe un archivo con el NombreDelArchivo especificado.
fscanf()	Lee una entrada con formato desde el archivo		
feof()	Devuelve cierto si se llega al final del archivo.		
ferror()	Devuelve cierto si se produce un error.		
rewind()	Coloca el localizador de posición del archivo al principio del mismo.		
remove()	Borra un archivo.		
fflush()	Vacía un archivo.		