



UNIVERSIDAD TECNOLÓGICA DEL CENTRO

GUÍA TEÓRICA

Materia: Herramientas de Informática I

"Si tú de mañana buscares a DIOS y rogaras al Todopoderoso; si fueres limpio y recto, Ciertamente luego se despertará por ti. Y hará próspera la morada de tu justicia. Y aunque tu principio haya sido pequeño, tu postrer estado será grande." Job 8:5-7.



Profesora : DELY M. GIL A.

VALENCIA, Enero 2008

CONTENIDO

1. Estructura de un programa en Lenguaje C
 - 1.1. Archivos de Inclusión o Cabecera
2. Variables
 - 2.1. Declaración de Variables
 - 2.2. Inicialización de variables.
3. Tipos de Datos
4. Constantes
 - 4.1. Constantes de Cadenas
 - 4.2. Caracteres no imprimibles
5. Operadores
 - 5.1. Operadores Aritméticos
 - 5.2. Operadores relacionales
 - 5.3. Operadores Lógicos
 - 5.4. Precedencia
6. Instrucciones de Entrada y salida formateada por pantalla
 - 6.1. scanf
 - 6.2. printf
7. Funciones de entrada y salida por consola.
8. Estructuras Selectivas
 - 8.1. Selectivas Simples
 - 8.2. Selectivas Dobles
 - 8.3. Selectivas Anidadas
 - 8.4. Selectivas Múltiples
9. Estructuras Repetitivas
 - 9.1. FOR
 - 9.2. WHILE
 - 9.3. DO



I. Estructura de un programa en lenguaje C

Todo programa en lenguaje C debe tener un programa o función principal MAIN(), en el que se desarrolla todo el código del programa las llamadas a funciones (es una función que se autoejecuta cuando se compila el programa).

Los comentarios comienzan por /* y terminan por */ cuando se trate de un bloque de sentencias. Si es una sola línea se comenta con //

Las instrucciones deben ir terminadas por punto y coma (;).

Las letras minúsculas son totalmente diferentes de las mayúsculas.

Toda palabra reservada se escribe en minúscula.

Los bloques de sentencias deben ir encerrados entre { al comienzo y } al final.
Inicio - Fin

1. Archivos de Inclusión o Cabeceras.

Los archivos de inclusión contienen definiciones y declaraciones que pueden ser incorporados en un programa específico que así lo requiera.

Para llamar un archivo de inclusión o cabecera es necesario hacer uso de la directiva **#include** la cuál tiene el siguiente formato:

```
#include nombre_de_archivo_cabecera
```

Donde nombre_de_archivo_cabecera es el nombre de un archivo que contiene las declaraciones y definiciones para un propósito específico. Este archivo se distingue por tener la extensión .h.

El nombre_de_archivo_cabecera se debe encerrar entre comillas (" ") o signos menor y mayor que (< >).

Ejemplo:

```
#include "stdio.h"  
#include <stdio.h>
```

Las comillas le dicen a C que busque primero en el directorio de trabajo actual el archivo de inclusión, si no lo encuentra, busca entonces en el directorio especificado en la línea de ordenes, y finalmente si aún no lo ha encontrado, entonces busca en el directorio standard que se halla definido durante la instalación.

Si el archivo de inclusión se encierra entre signos menor y mayor qué, el compilador busca primero en el directorio especificado en la línea de órdenes. si no lo encuentra busca entonces en el directorio standard. Jamás busca en el directorio de trabajo.

Sí en la inclusión se especifica nombre de ruta o camino, entonces busca en dicho directorio.

Estructura de un Programa en lenguaje C

```
// archivos cabecera
```



```

.....
//Constantes
.....
//TDA O Tipo Abstracto de Dato (Registro, Arreglos, Archivos)
.....
//Variables globales
.....
//Prototipos ( cabeceras de las funciones)
.....
void main(){
//Variables locales
.....
    clrscr();           // borra la pantalla
//sentencias ejecutables
.....
    getch();           // stop en la pantalla
}
//Implementación de los prototipos ( desarrollo del cuerpo de las funciones)
.....

```

II. Variables

Los identificadores de variables pueden contener 32 caracteres alfanuméricos, el subrayado(_) y el dólar (\$). El primer carácter debe ser una letra o el subrayado.

Las variables declaradas fuera de todas las funciones, incluyendo la principal main(), se consideran como **globales**. Las variables declaradas dentro de las funciones se consideran como **locales**.

1. Declaración de las variables

Formato: tipo lista_de_variables;
 tipo variable1 [,variable2, ... variable_n];

Ejemplo:
int a, b; float f, g; double d;

char ch; long l, k; signed char ca;

2. Inicialización de Variables

Formato: tipo variable = constante;

Ejemplo : char ch = 'a';
 float f = 123.675;



III. Tipos de Datos

Los tipos de datos en lenguaje C son los siguientes:

Nombre	Tipo	Bits	Tamaño
Unsigned char	Carácter sin signo	8	0 a 255
Char	Carácter	8	-128 a 128
Enum	Enumeración	16	-32,768 a 32,767
Unsigned int	Número sin signo	16	0 a 65,535
Short int	Número sin signo	16	-32,768 /32,767
Int	Número	16	-32,768 /32,767
Unsigned long	Número sin signo	32	0 to 4,294,967,295
Long	Número	32	-2,147,483,648
Float	Número real	32	3.4 * (10** -38)
Double	Número real de doble precisión	64	1.7 * (10** -308)
Long double	Número real	80	3.4 * (10** -4932)
void	Vacío	0	Sin valor

Cuando definimos cadenas de texto en C se debe definir como si fuera un arreglo de char o caracteres.

Ej.:

```
char saludo[5] = "hola";
printf("nombre %s", saludo);
```

Cuando definimos cadenas de texto en C se debe definir como si fuera un arreglo de char o caracteres.

Ej.:

```
char saludo[5] = "hola";
printf("nombre %s", saludo);
```

Dentro de la variable saludo se encuentra contenido el texto "hola".

La cantidad de caracteres son 5 comenzando desde la posición 0,1,2,3 y 4.

En la posición 0 de la cadena se almacena la letra "H".

En la posición 1 de la cadena se almacena la letra "o" y así sucesivamente

Cuando se define una cadena en la posición final de la cadena se pone \0 que significa el final mismo. Cuando nosotros trabajemos con cadenas podemos preguntar si es el final de la cadena (\0)

Cadena saludo

Posición 0	Posición 1	Posición 2	Posición 3	Posición 4
H	O	L	A	\0

La posición 4 de la cadena tiene el carácter de fin de línea que indica que la cadena llega hasta ahí.

De esta misma manera se puede imprimir el segundo carácter de la cadena

Ej.: usamos para mostrar en pantalla la función printf();

```
char saludo[5] = "hola";
printf("%c",saludo[2]); /* resultado : "l" */
```



IV. Constantes

Para la definición de constantes es necesario la directiva #define
#define identificador valor

Ejemplo:

```
#define PI      3.1416
#define ERR_1  printf("Error 26\n")
```

1. Constante de Cadenas

Las constantes de cadena se entrecomillan con comilla doble (").
Las constantes de carácter se entrecomillan con comilla simple (').

Ejemplo: "Esto es una cadena"

```
"a"
'a'
```

Nota: "a" es diferente de 'a'

La representación de 'a' como carácter y "a" como cadena en la memoria sería:

```
carácter  cadena
a         a\0
```

2. Constantes no imprimibles

Para los caracteres no imprimibles se utilizan códigos de barras invertidas:

\b	Retroceso
\a	Alarma
\f	Salto de página
\n	Salto de línea
\r	Retorno de carro o ENTER
\t	Tabulador horizontal
\"	Doble comilla
\'	Comilla simple
\\	Barra invertida
\v	Tabulador vertical (Impresora)
\0	Nulo

Ejemplo: ch = '\n';



V. Operadores

1. Operadores aritméticos

+	Suma
-	Resta
*	Multiplicación
/	División
%	Modulo
++	Incremento
--	decremento
()	Agrupar

Incrementar y Decrementar Variables

TIPO	RESULTADO
Variable++	Variable = variable +1
Variable- -	Variable = variable -1
Variable += incremento	Variable = variable + incremento
Variable -= incremento	Variable = variable - incremento

Precedencia

- 1) ++ -- ()
- 2) * / %
- 3) + -

2. Operadores relacionales

IGUAL (EQUAL)	= =
DISTINTO	!=
MAYOR	>
MENOR	<
Mayor =	>=
Menor =	<=

Precedencia

- 1) > >= < <=
- 2) == !=

3. Operadores lógicos

O (OR)	
NO (NOT)	!
Y (AND)	&&

Precedencia

- 1) !
- 2) &&
- 3) ||

4. Asignación

=

Porcentaje = 4;



VI. Instrucciones de entrada y salida formateada por pantalla

1. **printf()** : Escribe datos en diversos formatos, por lo tanto se denomina también salida formateada.

printf() escribe datos en la consola.

Formato:

```
printf("cadena_de_ctrl", lista_de_argumentos);
```

cadena_de_ctrl : Comienza con un signo de porcentaje (%) y luego le sigue el código de formato.

lista_de_argumentos : Son variables, constantes o expresiones a visualizar.

Tabla de ordenes de formato de printf()

Código	Significado
%c	visualiza un carácter
%d	visualiza decimal
%i	visualiza entero
%e	visualiza en notación científica
%f	visualiza datos en punto flotante
%g	usa %e ó %f la que sea más corta
%o	visualiza octal
%s	visualiza cadena de caracteres
%u	visualiza decimal sin signo
%x	visualiza hexadecimal
%%	imprime signo %
%p	visualiza un puntero

Ejemplo:

```
printf("Datos %c %d %s\n", 'a', 100, "caracteres");

/* visualiza : Datos a 100 caracteres */

/*****
 *
 * suma de valores
 *
 *****/

main()
{
    int a, b;

    a = 10; b = 40;

    printf("Suma de a + b = %d\n", a+b);
}
```

ESPECIFICADOR DE ANCHO DE CAMPO

Completa la salida con blancos o ceros para asegurar que es al menos la longitud mínima.



Si una cadena o número es mayor que el mínimo ancho especificado, esta se imprimirá completa.

El relleno por defecto es blanco. Si se desea un relleno con ceros se debe poner cero antes del especificador de ancho de campo.

Ejemplo:

```
printf("%05d", 48); /* visualiza :00048 */
printf("%5d", 48); /* visualiza : 48 */
```

Si se desea especificar posiciones decimales para un valor en punto flotante, se pone un punto decimal seguido por el número de decimales.

Ejemplo:

```
printf("%10.2f", 123.1283);
/* visualiza : 123.12 */
```

Nota : Tenga en cuenta que el valor que aparece antes del punto especifica el ancho total de campo incluyendo las posiciones decimales y el punto decimal. Lo anterior se podría leer como campo de tamaño diez de los cuales siete se tomarán para la parte entera, dos para decimales y uno para el punto.

Por defecto toda salida formateada se justifica a la derecha; si se desea un efecto contrario entonces se debe colocar un guión después del signo por ciento.

Ejemplo:

```
printf("%-5d", 48); /* visualiza :48 */
printf("%-5.2f", 48); /* visualiza :48.00 */
```

Cuando se usa el punto en una cadena; el valor que le sigue al punto indica el máximo ancho de la cadena.

Ejemplo:

```
printf("%5.7s", "José Perez");
```

El cinco indica que la cadena tendrá al menos cinco caracteres y un máximo de siete.

Si la cadena es más larga que el ancho máximo especificado, los caracteres adicionales se truncan por la derecha.

Ejercicio:

```
/*
*****
*
* formateo de la salida
*****/

main()
{
    int i = 100;
    float f = 3.5;

    printf("i = %4d, f = %2.2f\n", i, f);
}
```

MODIFICADORES DE FORMATO PARA printf() QUE PERMITEN VISUALIZAR ENTEROS CORTOS Y LARGOS

```
l ..... dato long
h ..... dato short
```

Los anteriores modificadores se pueden aplicar a los códigos de formato: d, i, o, u y x.



Ejemplo :

```
printf("%ld", p);      /* visualiza un long decimal */
```

2. **scanf()** : Función de entrada de propósito general que lee la corriente stdin.

Formato :

```
scanf("cadena_de_ctrl", lista_de_argumentos);
```

cadena_de_ctrl : Consta de tres clasificaciones de caracteres:

1. Especificadores de formato
2. Caracteres de espacio en blanco
3. Caracteres de no espacio en blanco

El código de formato le dice a **scanf()** que tipo de datos se lee a continuación.

Tabla de códigos de formato de scanf()

Código	Significado
%c	lee un carácter
%d	lee un decimal entero
%i	lee un decimal entero
%e	lee un número en punto flotante
%f	lee un número en punto flotante
%o	lee un número octal
%s	lee una cadena
%x	lee un número hexadecimal
%p	lee un puntero

Un carácter de espacio en blanco en la cadena de control provoca que **scanf()** salte sobre uno o más caracteres de espacio en blanco tabulado o return en la corriente de entrada.

Ejemplo:

```
scanf("%d %f %s", &i, &f, &str);
/* la entrada podría ser: 20 3.5      maria */
```

Un carácter de no espacio en blanco provoca que **scanf()** lea y descarte un carácter que no encaje.

Ejemplo:

```
scanf("%d,%d", &x, &y);
/* la entrada podría ser : 20,50 */
```

Si no se encuentra el carácter especificado terminará **scanf()**.

Supresión de la asignación con *.

Si en un **scanf()** se especifica el carácter %*, el campo de entrada no será tomado en cuenta.

Ejemplo:

```
scanf("%d%*c%d", &x, &y);
/* la entrada podría ser :20/40 */
/* x = 20      y = 40      */
```

En las órdenes de formato se puede especificar el modificador máximo ancho de campo.

Ejemplo :

```
scanf("20s", str); /* lee máximo 20 caracteres */
```



Si la entrada tiene más de 20 caracteres, el siguiente `scanf()` o cualquier llamada a una posterior entrada comenzará donde quedó la anterior.

Sea cuidadoso, la entrada para un campo puede terminar antes de completar la longitud máxima de la misma al encontrar un espacio en blanco. `scanf()` se moverá al próximo campo.

Ejemplo:

```
scanf("%2d %f %*d %2s", &i, &f, &nom);
/* al leer :5674 139 49abc74 quedará:
   i = 56    f = 74.00    saltará 139    nom = "49" */
```

La siguiente llamada a cualquier función de entrada comenzará a partir de a.

EJERCICIOS DE APLICACION

Desarrolle una serie de ejercicios para poner en práctica todo lo estudiado hasta el momento.

Programas ejemplo

```
/*
 * lee un valor y devuelve su cuadrado
 */
main()
{
    int a;
    printf("Teclea un valor :"); scanf("%d", &a);
    printf("Cuadrado de %d es igual a %d\n", a, a*a);
}
```

```
/*
 * Conversión de grados Celsius a fahrenheit
 * C = (5/9)(f-32)
 */
main()
{
    float fahr, celsius;

    printf("Grados fahrenheit :"); scanf("%f", &fahr);
    celsius = (5.0 / 9.0) * (fahr - 32);
    printf("%6.2f grados fahrenheit equivalen a %6.2f celsius\n",
           fahr, celsius);
}
```



VII. Funciones de Entrada y salida por consola.

Para el uso de estas funciones es necesario incluir el archivo "conio.h".

Función <code>getchar()</code>	:	Lee un carácter desde el teclado; espera por el retorno de carro.
Función <code>getche()</code>	:	Lee un carácter del teclado; no espera por retorno de carro.
Función <code>getch()</code>	:	Lee un carácter del teclado, no hace eco y no espera por retorno de carro.
Función <code>putchar()</code>	:	Escribe un carácter en la pantalla.
Función <code>gets()</code>	:	Lee una cadena desde el teclado.
Función <code>puts()</code>	:	Escribe una cadena en la pantalla.
Función <code>cprintf()</code>	:	Escribe carácter formateado.
Función <code>kbhit()</code>	:	Espera por la pulsación de una tecla.
Función <code>fgets()</code>	:	Lee una cadena de tamaño específico y la almacena en una variable.



VIII. Estructuras Selectivas

Las estructuras selectivas se utilizan para tomar decisiones lógicas, de ahí que suele denominárseles también Estructuras de Decisión o Alternativas

1. Estructuras Selectivas Simples

```
if(<condicion>) {
    sentencia_1;
    sentencia_2;
    .
    .
    sentencia_n;
}
```

Estructura Condicional Resumida:

?: Formato: (*<condicion>*) ? *<exp1>* : *<exp2>*

<expL> :Es la condición a ser evaluada.

<exp1> :Se ejecuta si *<expL>* es verdadera.

<exp2> :Se ejecuta si *<expL>* es falsa.

Ejemplo:

```
z = (a > b) ? a : b;
m = (mes > 6) ? 8 : 2;
```

```
#include "stdio.h"
main() /* convierte letra a tipo contrario */
{
    char ch;

    puts("Digita un carácter :");ch=getche();
    printf("%c\n", (islower(ch) ? toupper(ch) : tolower(ch)));
}
```

2. Estructuras Selectivas Dobles

```
if(<condicion>)
    sentencia;
else {
    sentencia_1;
    sentencia_2;
    .
    .
    sentencia_n;
}
```

3. Estructuras Selectivas Anidadas

```
if() else if()

if(<condicion>)
    sentencia;
else if(<condicion >)
    sentencia;
else if(<condicion >)
    sentencia;
.
.
.
else
    sentencia;
```



4. Estructuras Selectivas Múltiples

Formato:

```

switch(<var>) {
    case constante_1:
        secuencia_de_sentencias
        break;

    case constante_2:
        secuencia_de_sentencias
        break;

    .
    .
    .
    default :
        secuencia_de_sentencias
}

```

VII. Estructuras Repetitivas

Una estructura de control que permite la repetición de una serie determinada de sentencia se denomina BUCLE (LAZO o CICLO).

Clasificación:

Automáticas	{	Repita - para	(FOR)
Condicionales		Repita mientras	(WHILE-DO)
	{	Hacer _ Mientras	(DO-WHILE)

1. Bucle FOR:

Formato:

```

for(inicialización; <condición>; variación){
    sentencias;
}

```

inicialización: Es normalmente una asignación para dar un valor inicial a la variable controladora de ciclo.

<condición> : La expresión lógica determina hasta cuando ha de repetir el bucle.

variación : Forma o manera en que debe cambiar la variable del ciclo.

```

Ejemplo: #define MAXN 10
main()
{
    int i;

    for(i=0; i<MAXN; i++) printf("%d ",i);
}

```



```
main()
{
    int i;

    for (i = 10; i > 0; i--) printf("%d ",i);
}
```

```
main()
{
    int x;

    for(x=0; x<10; x +=2) printf("%d ", x);
}
```

```
main()
{
    int x, y;

    for(x=0, y=0; x<10; x++, y++)
        printf("x=%d, y=%d\n", x, y);
}
```

```
#include <stdio.h>
main()
{
    char ch;

    for(;;) {
        printf("Digita un carácter :"); ch=getche();
        if(ch=='$') break; /* rompe el ciclo */
    }
}
```

```
main()
{
    int x;

    for(x=0; x != 100;) {
        printf("Introduzca un número 100 para salir");
        scanf("%d",&x);
    }
}
```

```
#include <stdio.h>
#include <ctype.h>
main()
{
    int i, j, resp;
    char done = ' ';

    for(i=1; i<=100 && done != 'N'; i++) {
        for(j=1; j<=10; j++) {
            printf("Cuánto es %d + %d ?",i, j);
            scanf("%d",&resp);
            if(resp != i+j)
                printf("\nErrado\n");
            else
                printf("\nOk\n");
        }
    }
}
```



```
    }  
    printf("Sigue (S/N)? ");  
    done = toupper(getche());  
}  
}
```

2. Bucle WHILE-DO:

Formato: while (<expL>) sentencia;

Ejemplo:

```
#define FIN '*'  
#include <stdio.h>  
  
main()  
{  
    char ch = '\0';  
    while(ch != FIN) ch=getche();  
}  
  
#include "stdio.h"  
  
main()  
{  
    int c;  
  
    while((c=getchar()) != EOF)  
        putchar(c);  
}  
#include "stdio.h"  
  
main()  
{  
    int c, nc=1;  
  
    printf("%4d :", nc);  
    while((c=getchar()) != EOF) {  
        if(c == '\n') {  
            ++nc;  
            putchar(c);  
            printf("%4d :",nc);  
        } else  
            putchar(c);  
    }  
}
```

3. Bucle DO WHILE:

Formato: while (<expL>) sentencia;

Ejemplo:

```
#define FIN '*'  
#include <stdio.h>  
  
main()  
{  
    char ch = '\0';  
    while(ch != FIN) ch=getche();  
}
```



```
#include "stdio.h"

main()
{
    int c, nc=1;

    printf("%4d :", nc);
    while((c=getchar()) != EOF) {
        if(c == '\n') {
            ++nc;
            putchar(c);
            printf("%4d :",nc);
        } else
            putchar(c);
    }
}
```

LA SENTENCIA continue

continue : obliga a ejecutar la siguiente iteración y salta cualquier código entre medias.

Ejemplo:

```
main() /* imprime nros. pares */
{
    int x;

    for(x=1; x<100; x++)
        if(x%2)
            continue;
        printf("%d\n", x);
}
```

ETIQUETAS y goto

Una etiqueta es un nombre de identificador válido para C que termina en dos puntos (:).

Una etiqueta debe estar en la misma función donde está el goto que la utiliza.

Ejemplo:

```
loop:
    x++;
    if (x<100) goto loop;
```

goto puede ser apropiado cuando se desea salir de varios niveles de anidación.



GUÍA DE EJERCICIOS 1-2

<pre>EJERCICIO NRO. 1 // archivos cabecera #include<conio.h> #include<stdio.h> //Constantes //TDA //Variables globales //Prototipos void main(){ //Varibles locales clrscr(); // borra la pantalla printf("Bienvenidos al la UNITEC!!"); printf("\n\n.....fin del programa"); getche(); // Detiene la ejecucion del programa } //Implementación de los prototipos</pre>	<pre>EJERCICIO NRO. 2 ////////////////////// programa para lectura y escritura // archivos cabecera #include<conio.h> #include<stdio.h> //Constantes #define valor 10; #define precio 100.5 #define cadena "Bienvenidos al curso" //TDA //Variables globales //Prototipos void main(){ //Varibles locales int edad; clrscr(); // borra la pantalla printf("%s",cadena); printf("\n\nInserte su edad: "); scanf("%d",&edad); edad=edad+valor; printf("\n\nSu edad dentro de 10 años será de %d años",edad); getche(); // Detiene la ejecución del programa } //Implementación de los prototipos</pre>
--	--



GUÍA DE EJERCICIOS 3-4

<pre>EJERCICIO NRO. 3 // programa utilizando operaciones básicas // archivos cabecera #include<conio.h> #include<stdio.h> //Constantes #define desc 10; //TDA //Variables globales //Prototipos void main(){ //Varibles locales int codigo,i=1; float precio; clrscr(); // borra la pantalla printf("Insertar código de producto %d: ",i+1); scanf("%d",&codigo); printf("\nPrecio de producto %d ",codigo); scanf("%f",&precio); precio=precio-(precio/100)*desc; printf("\nEl precio con descuento es %.2f ",precio); ////////// printf("\nInsertar código del otro producto %d: ",i); scanf("%d",&codigo); printf("\nPrecio de producto %d ",codigo); scanf("%f",&precio); precio=precio-(precio*desc)/100; printf("\nEl precio con descuento es %.2f ",precio); getche(); // Detiene la ejecución del programa } //Implementación de los prototipos</pre>	<pre>EJERCICIO NRO. 4 // programa intercambio de digitos // archivos cabecera #include<conio.h> #include<stdio.h> //Constantes #define dig 10; //TDA //Variables globales //Prototipos void main(){ //Varibles locales int num,d1,d2,inv; clrscr(); // borra la pantalla printf("insertar un numero de 2 digitos: "); scanf("%d",&num); d1=num/dig; d2=num%dig; inv=(d2*10)+d1; printf("\nEl numero intercambiado %d ",inv); getche(); // Detiene la ejecución del programa } //Implementación de los prototipos</pre>
---	---



GUÍA DE EJERCICIOS 5-6

<pre>EJERCICIO NRO. 5 //// programa intercambio de dígitos con dos números de 3 Dígitos //// 234 y 567 llevarlo a 264 y 537 // archivos cabecera #include<conio.h> #include<stdio.h> //Constantes //TDA //Variables globales //Prototipos void main(){ //Variables locales int a,b,d1,d2,d3,d4,d5,d6,aux; clrscr(); // borra la pantalla printf("Insertar numero A: "); scanf("%i",&a); printf("\nInsertar numero B: "); scanf("%d",&b); d1=a/100; aux=a%100; d2=aux/10; d3=aux%10; d4=b/100; aux=b%100; d5=aux/10; d6=aux%10; a=(d1*100)+(d5*10)+d3; b=(d4*100)+(d2*10)+d6; printf("\nLos nuevos valores son %d y %d ",a,b); getche(); // Detiene la ejecución del programa } //Implementación de los prototipos</pre>	<pre>EJERCICIO NRO. 6 //// Ecuación de segundo grado // archivos cabecera #include<conio.h> #include<stdio.h> #include<math.h> //Constantes //TDA //Variables globales //Prototipos void main(){ //Variables locales int a,b,c; float x1,x2,raiz; clrscr(); // borra la pantalla printf("Insertar valor de a: "); scanf("%d",&a); printf("Insertar valor de b: "); scanf("%d",&b); printf("Insertar valor de c: "); scanf("%d",&c); raiz=sqrt(b*b-(4*a*c)); x1=(-b+raiz)/(2*a); x2=(-b-raiz)/(2*a); printf("\nlas raices son: %.2f y %.2f ", x1,x2); getche(); // Detiene la ejecución del programa } //Implementación de los prototipos</pre>
---	---



GUÍA DE EJERCICIOS 7-8

<pre>EJERCICIO NRO. 7 // archivos cabecera #include<conio.h> #include<stdio.h> //Constantes //TDA //Variables globales //Prototipos void main(){ //Varibles locales int num; clrscr(); // borra la pantalla printf("Insertar un número: "); scanf("%d",&num); if ((num%2)==0){ printf("El numero es par"); } else{ printf("El numero es impar"); } getche(); // Detiene la ejecucion del programa } //Implementación de los prototipos EJERCICIO NRO. 8 //// Operaciones con la estructura switch // archivos cabecera #include<conio.h> #include<stdio.h> //Constantes //TDA //Variables globales //Prototipos</pre>	<pre>void main(){ //Varibles locales clrscr(); // Borrar Pantalla float a,b; char op; gotoxy(20,2); printf("OPERACIONES ARITMETICAS "); gotoxy(25,4); printf("1. Realizar la operación Sumar"); gotoxy(25,6); printf("2. Realizar la operación Restar"); gotoxy(25,8); printf("3. Realizar la operación Multiplicar"); gotoxy(25,10); printf("4. Realizar la operación Dividir"); gotoxy(25,12); printf("Insertar operación a realizar: "); op=getche(); gotoxy(10,15); printf("insertar valor de a: "); scanf("%f",&a); gotoxy(40,15); printf("insertar valor de b: "); scanf("%f",&b); switch(op){ case 1: gotoxy(25,17); printf("La suma es: %.2f",a+b); break; case 2: gotoxy(25,17); printf("La suma es: %.2f",a-b); break; case 3: gotoxy(25,17); printf("La suma es: %.2f",a*b); break; case 4: gotoxy(25,17); printf("La suma es: %.2f",a/b); break; default: gotoxy(25,19); printf("La opción no es valida"); } getche(); // Detiene la ejecucion del programa } //Implementación de los prototipos</pre>
---	--



GUÍA DE EJERCICIOS 9-10

```
EJERCICIO NRO. 9
//// ejercicio de números promedio
// archivos cabecera
#include<conio.h>
#include<stdio.h>
//Constantes
//TDA
//Variables globales
//Prototipos

void main(){
//Varibles locales
char r='s';
int i=0,a=4;
int acum=0,valor;
clrscr(); // borra la pantalla
gotoxy(25,a),
printf("C A L C U L A R   P R O M E D I O");

while(r=='s'){
gotoxy(25,a+=2);
printf("Insertar el %d valor: ",i+1);
scanf("%d",&valor);
acum+=valor;
i++;
gotoxy(25,a+=2);
printf("Desea Continuar: ");
scanf("%c",&r);
}
printf("El promedio de los valores es:
%.2f",acum/i);
getche(); // Detiene la
ejecución del programa
}
//Implementación de los prototipos
```

```
EJERCICIO NRO. 10
//// Intercambiar todos los dígitos de un
numero
// archivos cabecera
#include<conio.h>
#include<stdio.h>
//Constantes
//TDA
//Variables globales
//Prototipos

void main(){
//Varibles locales
char r;
int num,aux,dig,inter;
do{
clrscr(); // Borrar Pantalla
gotoxy(20,3);
printf(" I N T E R C A M B I A R   D I G I T O
S ");
gotoxy(20,5);
printf("Insertar un valor: ");
scanf("%d",&num);
aux=num;
inter=0;
while(aux!=0){
dig=aux%10;
aux=aux/10;
inter=(inter*10)+dig;
}
gotoxy(20,7);
printf("El número intercambiado es:
%d",inter);
gotoxy(20,9);
printf("Deseas Continuar: ");
r=getch();
}while(r=='s');
getche(); // Detiene la
ejecución del programa
}
//Implementación de los prototipos
```