APUNTES DE LENGUAJE DE PROGRAMACIÓN

ELABORADOS POR EL PROFESOR DE LA ACADEMIA DE SISTEMAS DIGITALES, TURNO VESPERTINO.
ING. R. DIMITRI CAB CORDERO

Índice.

Introducción.

La realización de estos apuntes esta fundamentada en la necesidad de una bibliografía apegada al temario de la asignatura de lenguaje de programación C, impartida en el CECYT No. 1 "Gonzalo Vázquez Vela" en el turno Vespertino.

Los apuntes están dirigidos a todos los estudiantes de la carrera de Técnico en Sistemas Digitales, a los cuales se les da una pequeña introducción a una de las áreas mas explotadas en la actualidad como es la programación de computadoras. En estos apuntes, se dan los principios básicos de programación en el lenguaje "C" específicamente, siendo este el lenguaje de mayor uso y versatilidad para las aplicaciones de control y automatización por computadora. Aunque la tecnología ha tendido hacia los lenguajes visuales basados en el sistema Windows de Microsoft, el lenguaje C sigue siendo una herramienta útil para dicha tarea junto con JAVA.

Se incluyen apéndices que están orientados a una complementación con los distintos conocimientos que se deben de adquirir paralelamente con la asignatura destacando secciones que no se encuentran en otros libros, como es el caso de errores de compilación y ejecución.

Los apuntes fueron creados en base a la planificación temática de la asignatura durante semestres anteriores, por lo que, aunque no abarcan todos los temas del temario original, se cubren los temas principales y necesarios para el correcto aprendizaje de la asignatura. Aprendizaje corroborado y comprobado con la satisfacción de los alumnos egresados que estudian una carrera en el nivel superior donde han demostrado sus firmes conocimientos en lenguaje C.

Con lo antes mencionado, estamos seguros de que estos apuntes serán de mucha utilidad para los alumnos que deseen iniciarse y aprender en una de las áreas mas reconocidas de la carrera de Técnico en Sistemas Digitales: "La programación de computadoras".

COMPETENCIA PARTICULAR 1
Realiza programas con funciones básicas de entrada – salida de datos, declarando variables de diversos tipos y realizando operaciones matemáticas simples basándose en algoritmos y utilizando el entorno integrado

INTRODUCCIÓN A LA PROGRAMACIÓN

Breve semblanza histórica del lenguaje C.

El lenguaje de programación C se llama de esa manera debido a que su predecesor fue un Lenguaje de programación llamado B desarrollado por Ken Thompson en los laboratorios Bell Telephone.

El creador del lenguaje C fue creado por Dennis Ritchie en los laboratorios Bell Telephone en 1972- El lenguaje no fue creado para su lenguaje de programación "popular" sino para desarrollar el sistema operativo LTNI)(, el cual es usado en muchas de las minicomputadoras. Desde un principio el lenguaje C tuvo como propósito ser útil: permitir a los programadores atareados que las cosas se pudieran hacer.

Como el C es un lenguaje "proceso y flexible" por lo que no tardo tiempo en que saliera de los laboratorios Bell Telephone y se difundiera rápidamente. De repente los programadores de todo el mundo se encontraban haciendo programas de todo tipo con el lenguaje C. como consecuencia varias organizaciones comenzaron a utilizar muy pronto sus propias versión del lenguaje C, y como eran diferentes empresas las que implementaban el lenguaje C empezaron a existir pequeñas diferencias entre cada paquete, por lo que cada vez el lenguaje C se hacia menos compatible con otras versiones del mismo lenguaje.

Por lo que para resolver el problema, el American Nacional Standards Institute "ANSI" (Instituto Nacional Americano de Estándares). Formó un comité en 1983 para elaborar una versión estándar del C que llegó a ser como el C estándar ANSI.

Ya una vez que sabemos algunas características históricas del lenguaje C es necesario que sepamos que los lenguajes mediante se programa una computadora se pueden clasificar en los siguientes tipos:

***Lenguaje de máquina**: es el lenguaje natural de una computadora el cual está relacionado con el diseño de hardware de la computadora, estos lenguajes consisten en cadenas de números y están orientados a los elementos de la computadora.*

Lenguaje ensamblador*- son aquellos que convierten los programas de lenguaje ensamblador en lenguaje de máquina. Por ejemplo las instrucciones de un microprocesador.*

Lenguaje de alto nivel*: Son aquellos con los que se pueden escribir enunciados simples para poder llevar a cabo tareas complicadas. Estos lenguajes permiten a los programadores escribir instrucciones lo mas parecidas al lenguaje común.*

Por todo lo anterior podemos concluir que el lenguaje C en un lenguaje de alto nivel, ya que las instrucciones son simples y son parecidas al lenguaje común (en inglés) y es l por medio de este lenguaje que en la actualidad se hacen gran cantidad de software, al grado que el mismo WINDOWS esta realizado por medio de lenguaje C.

RAP 1:
Conoce el concepto de programa y cuáles son sus elementos, así como su estructura general.

Edición de un programa.

Para desarrollar un programa en lenguaje C, se puede editar en cualquier editor de texto, ya que el lenguaje C es un lenguaje de alto nivel, lo que permite escribir las instrucciones en el lenguaje común (Ingles) que se utiliza. Una sola instrucción efectúa tareas complejas.

Al salvar el programa en un editor de texto cualquiera, debe salvarse con la extensión .c ó .cpp. entre los editores de texto comunes y útiles tenemos los siguientes:

- *block de notas*
- *wordpad*
- *word*
- *works*

Para salvar un archivo con una extensión especifica, debe ponerse entre comillas, ya que de no realizarse esto , el archivo guardara el archivo con la extensión por default.

Ejemplo:

Forma correcta:

"programa1.cpp" **** el archivo se guardara con la extensión especificada ****

Forma incorrecta:

programa1.cpp **** el archivo se guardara con la extensión por default***

programa1.cpp.txt *

Debemos recordar que para ejecutar un programa, no solo basta con editarlo, si no hace falta una aplicación para su compilación y ejecución.

Existen varias aplicaciones disponibles para el desarrollo de programas en lenguaje C como pueden ser:

- *Turbo C*
- *Quick C*
- *Borland C*
- *C para UNIX*

Borland C.

Borland C es una aplicación popular y muy eficiente, diseñada para la edición, compilación y ejecución de los programas en lenguaje C. Cuenta con un editor para la elaboración de los programas; un compilador, un depurador y un ligador (linker) para la creación de los programas ejecutables.

Para ingresar a Borland C desde DOS se debe acceder al subdirectorio ***c:\borlandc\bin>*** *y teclear el comando* ***bc****. Para acceder desde Windows utiliza el explorador de Windows y selecciona la carpeta de BorlandC, luego seleccionar la carpeta BIN y por ultimo hacer clic dos veces sobre el icono del archivo bc.exe.*

Cuenta con un ambiente de desarrollo similar al utilizado en Pascal y Qbasic. Posee una barra de menús con los siguientes menús:

- *FILE * menú con opciones de archivo **
- *EDIT * menú con opciones de edición **
- *SEARCH * menú con opciones de búsqueda **
- *RUN * menú con opciones de ejecución de programa **
- *COMPILE * menú con opciones de compilación de programa **
- *DEBUG* menú con opciones de depuración de errores **
- *PROYECT * menú con opciones de creación de proyectos **
- *OPTIONS * menú con opciones de configuración de la aplicación **
- *WINDOW * menú con opciones de visualización **
- *HELP * menú con opciones de ayuda **

Borland C es una aplicación muy completa, con muchas características y utilidades para el desarrollo de aplicaciones computacionales. Por lo que en este capitulo se dará una breve descripción de las utilidades que se necesitan conocer, para el desarrollo de programas en un nivel básico.

Menú FILE.

NEW: *Opción para la creación de un nuevo archivo.*

OPEN: *Opción para abrir un archivo existente con extensión .cpp*

SAVE: *Opción para salvar un archivo.*

SAVE AS: *Opción para salvar un archivo con un nombre propuesto por el usuario.*

SAVE ALL: *Opción para salvar todos los programas que estén en la pantalla.*

CHANGE DIR: *Opción para cambiar de directorio raíz.*

PRINT: *Opción para imprimir un archivo.*

DOS SHELL: *Opción para salir un momento a DOS (teclear exit para regresar).*

QUIT: *Opción para salir de Borland C.*

Menú EDIT.

UNDO: *Opción para deshacer la ultima acción realizada.*

REDO: *Opción para rehacer la acción que se había deshecho.*

CUT: *Opción para cortar.*

COPY: *Opción para copiar.*

PASTE: *Opción para pegar.*

CLEAR: *Opción para borrar.*

COPY EXAMPLE: *Opción para copiar un texto ejemplo preseleccionado de la ventana actual a la ventana de ayuda clipboard.*

CLIPBOARD: *Opción para abrir la ventana de clipboard, la cual guarda el texto cortado y copiado de otras ventanas.*

Menú SEARCH.

FIND: *Opción para buscar una palabra en el archivo.*

REPLACE: *Opción para reemplazar una palabra.*

SEARCH AGAIN: *Opción para repetir la búsqueda.*

GO TO LINE NUMBER: *Opción para ir a una línea de código.*

PREVIOUS ERROR: *Opción para mover el cursor a la línea de código del mensaje de error previo.*

NEXT ERROR: *Opción para mover el cursor a la línea de código del mensaje de error siguiente.*

LOCATE FUNCTION: *Opción para buscar una función.*

Menú RUN.

RUN: *Opción para ejecutar un programa.*

PROGRAM RESET: *Opción que realiza las siguientes funciones:*

- *Detiene la sesión actual de depuración.*
- *Libera la memoria que el programa ocupa.*
- *Cierra cualquier archivo abierto que el programa este usando.*

GO TO CURSOR: *Opción para ejecutar el programa hasta la línea de código donde este el cursor.*

TRACE INTO: *Opción para seguir después de la ejecución del programa renglón por renglón.*

STEP OVER: *Opción para seguir la ejecución paso a paso.*

Menú COMPILE.

COMPILE: *Opción para compilar un archivo.*

MAKE: *Opción para crear un archivo ejecutable.*

LINK: *Opción para ejecutar el ligador.*

REMOVE MESSAGE: *Opción para borrar los mensajes de error.*

Menú DEBUG.

Este Menú sirve para evaluar varias variables y parámetros de los programas.

Menú PROJECT.

Este Menú sirve para la integración de varios programas en un solo proyecto.

Menú OPTIONS.

Este Menú sirve para la configuración de la aplicación.

Menú WINDOW.

Este Menú sirve para la configuración de las ventanas de edición.

Menú HELP.

Este Menú proporciona la ayuda para la utilización de Borland C.

Conceptos básicos.

Computadora: *Dispositivo capaz de efectuar cálculos y tomar decisiones lógicas a velocidades millones y hasta miles de millones de veces masa rápidas que un ser humano.*

Programa: *Conjunto de instrucciones para procesar datos.*

Lenguaje de alto nivel: *Lenguaje en el que una sola instrucción efectúa tareas complejas. Permite que los programadores escriban instrucciones parecidas al lenguaje común que utiliza que utiliza la gente y contiene las notaciones matemáticas comunes.*

Compilar: *Traducir el programa de C en código de lenguaje de maquina.*

Enlace o Ligador: *Vincula el código objeto con el código de las funciones faltantes (como son librerías), generando un archivo ejecutable.*

Cargador de programa: *Carga en la memoria el archivo ejecutable.*

Variable: *Localidad de memoria de la computadora donde puede almacenarse un valor que será empleado por el programa.*

Directivas.

#include <archivo de encabezado o libreria>

Directiva que incluye un programa o librería al programa en C.

#define *nombre* texto de reemplazo.

Directiva que define un nombre simbólico o constante simbólica como una cadena de caracteres especial..

Comandos de Borland C.

Teclas	Función

F1.....Ayuda

F2.....Salvar archivo

F3.....Abrir archivo

F4.....Correr el programa hasta donde esta el cursor

F5.....Maximizar ventana de edición

F6.....Cambiar de ventana de edición

F7.....Correr programa paso a paso

F8.....Correr programa paso a paso sin entrar a funciones

F9.....Crear programa ejecutable

F10.....Menú

ALT+F9.....Compilar el programa

CTRL+ F9.....Correr el programa.

ALT+ X.....Salir de Borland C

SHIFT+ SUPR.....Cortar

CTRL+ INSERT.....Copiar

SHIFT+INSERT.....Pegar

CTRL+F1.....Ayuda de instrucción especifica.

Cuerpo general de un programa en lenguaje C.

Comentarios

Inclusión de archivos

Definición de constantes y macros

Definición de prototipos y funciones

Variables globales

main ( )

{

variables locales

flujo de sentencias

}

Variables locales flujo de sentencias Definición de funciones

Los componentes que deben contener nuestros programas en C son:

La función main ( ): *el único componente que es obligatorio en cada programa en C es la función maín ( ). En su forma más simple main ( ) consiste en el nombre main, seguido por un par de paréntesis vacíos () y un par de llaves { }. Dentro de las llaves se encuentran enunciados que forman el cuerpo principal del programa. Bajo circunstancias normales la ejecución del programa comienza con el primer enunciado de main ( ) y termina con el último enunciado de maín ( ), que es donde se cierran la llave "}".*

La directiva #include*: la directiva #include da instrucciones al compilador C para que añada al contenido de un archivo de inclusión al programa durante la compilación. Un archivo de inclusión es un archivo de disco separado que contiene información necesaria para el compílador. Varios de estos archivos algunas veces llamados archivos de encabezado se proporcionan con el compilador. Nunca se necesita modificar la información de estos archivos y ésta es la razón por la cual se mantiene separados del código fuente. Todoslos archivos de inclusión deben tener la extensión* ***.h*** *(por ejemplo* ***stdio.h, math.h,etc.****).*

En la sección de apéndices se muestran algunas funciones incluidas dentro de las blibliotecas o librerías.

La definición de variables*: una variables es un nombre asignado a una posición de almacenamiento de datos durante la ejecución del programa. En lenguaje C una variable debe ser definida antes de que pueda ser utilizada. Una definición de variable le informa a compilador el nombre de la variable y el tipo de datos que van a guardar. Por ejemplo si el dato es un valor entero, un número muy grande, o bien un carácter.*

EL prototipo de función*: proporciona al compilador C el nombre y los argumentos de una función en el. Programa y debe aparecer antes de que la función sea usada. Un prototipo de función es diferente de una definición de función que contiene las instrucciones actuales que hacen a la función.*

Los enunciados del programa*. el trabajo real de un programa C es hecho por sus enunciados. Los enunciados de C despliegan información en la Pantalla leen entrada del teclado ejecutan operaciones matemáticas llamen funciones leen archivos de discos y hacen todas las otras operaciones que un programa necesitan evaluar. La mayor parte de este libro está dedicado a enseñarte los diversos enunciados de C. por el momento, recuerde que en el código fuente los enunciados de C son escritos*

La definición de función*: una función es una sección de código independiente que es escrita para ejecutar determinada tarea. Cada función tiene un nombre, y el código de cada función es ejecutado incluyendo el nombre de la función en una instrucción de programa. A esta se le llama llamado de función.*

Los comentarios en el programa*: el compilador ignora todos los comentarios y por lo tanto no tiene ningún efecto sobre la manera en que funciona el programa. Se puede poner lo que se quiera en un comentario y esto no modifica la manera en que trabaje el programa. Un comentario puede ser definido por un "/" y un*

** " (/* esto es un comentario */) o por dos " / "(\\ esto también es un comentario \\).*

A continuación se muestra un ejemplo de un programa en C. Intente identificar los elementos descritos anteriormente.

```
#include<stdio.h>
#include<conio.h>
#define pi 3.1416
int r,d,p,a;
main()
{
clrscr();
printf("Dame el valor del diametro:");
scanf("%d", & d);
/*Calculo del perimetro*/
p=pi*d;
printf("el valor del perimetro es:%d\n", p);
/*Calculo del area*/
r=d/2;
a=pi*(r*r);
printf("El valor del area es:%d\n", a);
getch();
return 0;
}
```

El programa más básico de C

```
#include<stdio.h>
#include<conio.h>
main()    /funcion principal del programa/
{
clrscr();/limpia pantalla/
printf("==============MI PRIMER PROGRAMA====================================");
getch(); / detiene la pantalla para ver a ejecución del programa/
return 0;/retorna en contro al programa principal/
}
```

En este programa la instrucción ***printf*** *tiene como función mas elemental la de imprimir en pantalla un mensaje y su formato es el siguiente:*

printf ("Mensaje");

Modifique el programa anterior para ponerle todos los datos de su equipo al mensaje pero use las siguientes secuencias de escape para darle mayor presentación a la salida de su programa.

Secuencias de escape

Carácter	Secuencia de escape	VALOR ASCII
campana (alerta)	*\a*	*007*
espacio atrás (backspace)	*\b*	*008*
tabulador horizontal	*\t*	*009*
nueva línea (cambio de línea)	*\n*	*010*
tabulador vertical	*\v*	*011*
alimentación de página	*\f*	*012*
retorno de carro	*\r*	*013*
comillas (")	*\ ”*	*034*
apóstrofo (')	*\ ’*	*039*
interrogación	*\?*	*063*
barra invertida	**	*092*
nulo	*\o*	*00C*

RAP 2:
Aplica las funciones básicas de entrada y salida de datos.

Instrucciones printf y scanf.

C*omo se vio en el programa anterior la instrucción* ***printf*** *sirve para salida de datos a través de la pantalla incluyendo los datos que tenga guardados la computadora mientras que la instrucción* ***scanf*** *sirve como una instrucción de entrada de datos a través del teclado hacia el programa, dicho en otras palabras nos va a servir para recibir y guardar los datos o valores que el usuario teclee en la ejecución de un programa.*

A continuación presentamos la sintaxis de dichas instrucciones.

printf ("mensaje");

printf("%*",argumento);

scanf ("%*",&dirección)

el ***"*"*** *indica el carácter de conversión que dependerá del tipo de dato que se este trabajando el cual se muestra a continuación en la siguiente tabla:*

El ***argumento*** *se refiere al nombre de la variable que contenga el valor el cual se pretende que aparezca en el letrero del programa del programa.*

La ***dirección*** *se refiere al nombre de la variable la cual "guardara" el dato valor accesado por el usuario a través del teclado.*

Caracteres de conversión más usados de scanf y printf

Carácter de conversión	Significado
%c	el dato es un carácter
%d	el dato es un entero decimal
%e	el dato es un valor en coma flotante
%f	el dato es un valor en coma flotante
%g	el dato es un valor en coma flotante
%h	el dato es un entero corto
%i	el dato es un entero decimal, hexadecimal o entero
%o	el dato es un entero octal
%s	el dato es una cadena seguida por un espacio en blanco (el carácter nulo \0 se añade automáticamente al Final)
%u	el dato es un entero sin signo
%x	el dato es un entero hexadecimal
[...]	el dato es una cadena que puede contener espacios en blanco

Un* prefijo *puede preceder ciertas conversiones de caracteres.

Prefijo	*Significado*
H	*dato corto (entero corto o entero sin signo corto)*
l	*dato largo (entero largo, entero largo sin signo o real de doble precisión)*
L	*dato largo (real en doble precisión largo)*

Ejemplo

```
int a;
short b;
long c;
unsigned d;
double x;
char str[ 80 ];
scanf ("%5d %3hd %12 ld %12lu %15lf" ,&a, &b, &c, &d, &x)
scanf ( “%d \n " , str ) ;
```

```
#include<stdio.h>
#include<conio.h>
#define pi 3.1416
main()
{
int edad,anio;
char nombre[24],sexo;

clrscr();
printf("Teclea Tu nombre\n");
scanf("%s", & nombre);
printf("teclea su sexo\n M masculino \n  F femenino \n\a");
scanf("%d",&sexo);

printf("teclea tu año de nacimiento");
scanf("%d",&anio);
edad=2020-anio;
clrscr();
printf(" la persona de sexo %c de nombre %s en el 2001 tendra %d
años",sexo,nombre,edad);
getch();
return 0;
}
```

RAP 3:
Emplea los distintos operadores disponibles en el lenguaje aplicando sus reglas de prioridad.

TIPOS DE OPERADORES Y EXPRESIONES.

Tipos de datos.

Tipo de dato	Descripción	Requerimientos típicos de memoria
int	cantidad entera	2 bytes o 1 palabra (varía de una computadora a otra)
short	cantidad entera corta (puede contener menos dígitos que int)	2 bytes o 1 palabra (varía de una computadora a otra)
long	cantidad entera larga (puede contener más dígitos que int)	1 o 2 palabras (varía de una computadora a otra)
unsigned	cantidad entera sin signo (no negativa) (la cantidad máxima permisible es aproximadamente el doble que int)	2 bytes o 1 palabra (varía de una computadora a otra)
char	carácter	1 byte
signed char	carácter con valores en el rango de -128 a 127	1 byte
unsigned char	carácter con valores en el rango de 0 a 255	1 byte
float	número de coma flotante (un número que contiene un punto decimal y/o un exponente)	1 palabra
double	número de coma flotante en doble precisión (más cifras significativas y un exponente que puede ser mayor en magnitud)	2 palabras

long double	*número de coma flotante en doble precisión (puede tener mayor precisión que un double)*	*2 o más palabras (varía de una computadora a otra)*
void	*tipo especial de dato para funciones que no retornan ningún valor*	*no aplicable*
enum	*constante de enumeración (tipo especial de int)*	*2 bytes o 1 palabra (varía de una computadora a otra)*

Nota: *El calificador unsigned puede aparecer con short int 0 long int, por ejemplo unsigned short int* (o *unsigned short), o unsigned long int* (o *unsigned long).*

Reglas de conversión

Estas reglas se aplican a operaciones aritméticas entre dos operadores con distintos tipos de datos. Puede existir alguna variación de una versión de C a otra.

1. *Si uno de los operandos es long double, el otro será convertido a double y el resultado será un long double*
2. *En otro caso, si uno de los operandos es double, el otro se convertirá a double y el resultado será double,*
3. *En otro caso, si uno de los operandos es float, el otro será convertido a float y el resultado será float.*
4. *En otro caso, si uno de los operandos es unsigned long int, el otro será convertido a unsigned long int y el resultado será unsigned long int.*
5. *En otro caso, si uno de los operandos es long int y el otro es unsigned int, entonces:*

   a) Si unsigned int puede convertirse a long int, el operando unsigned int será convertido y el resultado será long int.

   b) En otro caso, ambos operandos serán convertidos a unsigned long int y el resultado será unsigned long int.
6. *En otro caso, si uno de los operandos es long int, el otro será convertido a long int y el resultado será long int.*
7. *En otro caso, si uno de los operandos es unsigned int, el otro será convertido a unsigned int y el re. sultado será unsigned int.*

8. Si no se puede aplicar ninguna de las condiciones anteriores, entonces ambos operandos serán convertidos a int (si es necesario) y el resultado será int.

Notar que algunas versiones de C convierten automáticamente todos los operandos en coma flotante a doble precisión.

Reglas de asignación

Si los dos operandos en una expresión de asignación son de tipos distintos, entonces el valor M operando dela derecha será automáticamente convertido al tipo M operando de la izquierda. La expresión completa de asignación será de este mismo tipo. Además:

1. Un valor en coma flotante puede truncarse si se asigna a un identificador entero.

2. Un valor en doble precisión puede ser redondeado si se asigna a un identificador de coma flotante (simple precisión).

3. Una cantidad entera puede ser alterada si se asigna a un identificador de entero más corto o a un identificador de carácter (algunos de los bits más significativos pueden perderse).

Operadores

El lenguaje C soporta varios tipos de operadores que podemos clasificar bajo los siguientes rubros:

- ***Operadores aritméticos***
- ***Operadores relacionales***
- ***Operadores lógicos***
- ***Operadores lógicos al nivel de bit***
- ***Operadores de asignación***
- ***Operadores de manipulación de datos en bajo nivel Operadores de expresión primaria***
- ***y Operadores Especiales***

A continuación se listan los distintos tipos de operadores y su significado semántico en C. La descripción de las correspondientes operaciones aritméticas y lógicas se presenta, conjuntamente con la descripción de los tipos de datos a los cuales son aplicables los diferentes operadores, en el Capítulo 5.

Operadores aritméticos

Negativo	*-*
Suma	*+*
Resta	*-*
Multiplicación	***
División	*/*
Residuo	*%*

Operadores relacionales

Menor que	<
Mayor que	>
Menor que o igual a	<=
Mayor que o igual a	>=
Igual que	==
Diferente de	=!

Operadores lógicos

Negación	¡=
Función 0	||
Función Y	&&

Operadores lógicos al nivel de bit (compuertas lógicas)

Complemento a 1	~
Función 0	|
Función Y	&
Función 0 Exclusiva	A

Operadores de asignación

Asignación simple	=
Incremento aritmético	+=
Decremento aritmético	-=
Incremento geométrico	*=
Decremento geométrico	/=
Modulo	%=
Corrimiento a la derecha.	»=
Corrimiento a la izquierda	«=
Ventana 0 lógica	|=
Ventana Y lógica	&=
Ventana 0 Exclusiva	^=

Operadores tipo lenguaje ensamblador

Corrimiento a la derecha	»
Corrimiento a la izquierda	«
Indirección	*
Apuntador	&
Incremento geométrico	*=
Decremento geométrico	/=

Operadores de expresión primaria

Apuntador *

Función ( )

Arreglo [ ]

Campo (modo indirecto) ->

Campo (modo directo)

Operadores de funciones especiales

Conversión de tipos (casta) (tipo)

Expresión condicional ?:

Expresión intermedia ,

Precedencia de los operadores matemáticos y paréntesis.

Es importante tomar en cuenta cuando se introduce una formula a un programa que los operadores matemáticos se van a ejecutar mediante un orden y direcciones establecidas. A la cual se muestra a continuación:

Orden	*Operador*	
Primero	***++ --***	***(incremento y decremento)***
Después	**** / %***	***(producto, división y módulo)***
Y al final	***+ -***	***(suma y resta)***

T*ome en cuenta que cuando se encuentran en una expresión mas de un operador con el mismo orden de precedencia se ejecutaran de izquierda a derecha.*

Uso de paréntesis

Y *por ejemplo cuando requerimos que se realice primero una suma antes de la división como en el caso de un promedio de números es necesario el uso de paréntesis de agrupación los cuales se eliminarán como en álgebra de adentro hacia fuera y antes de ejecutar cualquier otro operador.*

Observe que no es lo mismo : p= 10+5+9/3 = 18

Que: p=(10+5+9)/3= 8

Observe que todas estas operaciones nos darán diferente resultado:

1) *10 +5*4%3-4+1= 10+20%3-4+1 = 10+2-4+1 = 9*
2) *10+4%3*5-4+1 = 10+1*5-4+1 = 10+5-4+1= 12*
3) *10+4%3*(5-4+1)= 10+4%3*2 = 10+1*2 = 10+2 =12*
4) *(10+5)*4%3-4+1 =15*4%3-4+1 = 60%3-4+1= 0-4+1 = -3*

Actividades

Competencia particular 1

1.- Que diferencia existe entre un lenguaje de alto y de bajo nivel

2.- ¿cuándo se dice que se esta programando en lenguaje de máquina?

3.- escriba el nombre de un lenguaje de programación predecesor del lenguaje c.

4.- mencione el nombre de 5 lenguajes de alto nivel,

5.-¿ qué significa compilar un programa?

6.- ¿cuando se de dice que un archivo es ejecutable?

7.- ¿escribe el cuerpo general de un programa en c.

8.- el nombre que recibe la función principal de un programa en c. es:

9.- menciona en qué librería se encuentra cada una de las siguientes instrucciones e indica su funcion.

A) printf

B) getch()

C) clrscr()

D) pow(a,b)

E) sin(a)

F) delay(n)

G) a%b

H) scanf

I) return

J) getch()

10.- cual de los siguientes tipos de variables reserva mas espacio de memoria. carácter, doble precisión, flotante,

11.- escriba los caracteres de conversión para los diferentes tipos de datos en c.

12.- enumere los operadores aritméticos de c.

13.- ¿ cuáles son los operadores lógicos?

14.- si las variables son declaradas como de tipo entero encuentre el valor que daria c para las siguientes expresiones:

si: a=8, b=2, c=3.

*A) (a+b-c)*2%3+5*

*B) a%3+b*c*(a-c)*

C) 5(a+b-3%2+a/b*(b+c)*

15.- convierta las siguientes formulas para que puedan ser utilizadas en un programa de c.

A) tres resistores en paralelo.

B) formula general.

C) ley de senos.

***II.- IMPLEMENTA LA CODIFICACIÓN PARA LOS PROGRAMAS QUE SE PIDEN**:*

1.- implemente un programa en c que presente en pantalla el calendario del mes de septiembre utilizando secuencias de escape.

2.- realice un programa que guarda en diferentes variables en nombre, primer apellido, sexo y edad.

3.- realice un programa que calcule el valor de la potencia base "2" solicitada por el usuario.

4.- escriba un programa que determine el angulo de un triangulo rectángulo, el usuario debera dar el valor de los catetos.

5.- haga un programa que calcule la aceleración de un cuerpo(en el sistema mks), el usuario debera dar como dato: la velocidad final (en m/s), velicidad inicial(en m/s), y tiempo (en horas).

6.- diseñe un programa que determine la resistencia total de un de un circuito de 2 resistencias en paralelo:

el usuario accesara los datos de los resistores en kilo ohms, pero el resultado debera ser presentado tanto en ohms como en kilohms.

7.- mediante un programa haga usted que la computadora sea capaz de promediar 5 numeros.

8.- realice un programa que determine los valores de x1 y x2 para una ecuacion cuadratica. el usuario debera ingresar los coheficientes.

9.- implemente un programa que calcule la corriente en miliamperes de un circuito a partir de valores de de voltaje y resistencias dadas en volts y ohms respectivamente.

10.-Encuentre los valores que daría "C" para las siguientes expresiones:

*A) (4+3-5)%2+6*5/2-7*

*B) 20/10+5*6%3*

*C) (4*6-3+1)-(4*2-4)+6+2*3*

D) 6+1-7+8%3%2+1

E) 100(20+5) + 3%10*

11.-Convierta las formulas que se indican para que puedan ser realizadas correctamente en un programa:

a) Promedio de 4 números cualesquiera..

b) 5 resistencias en serie.

c) La ley de Coulmb.

d) Teorema de Pitágoras.

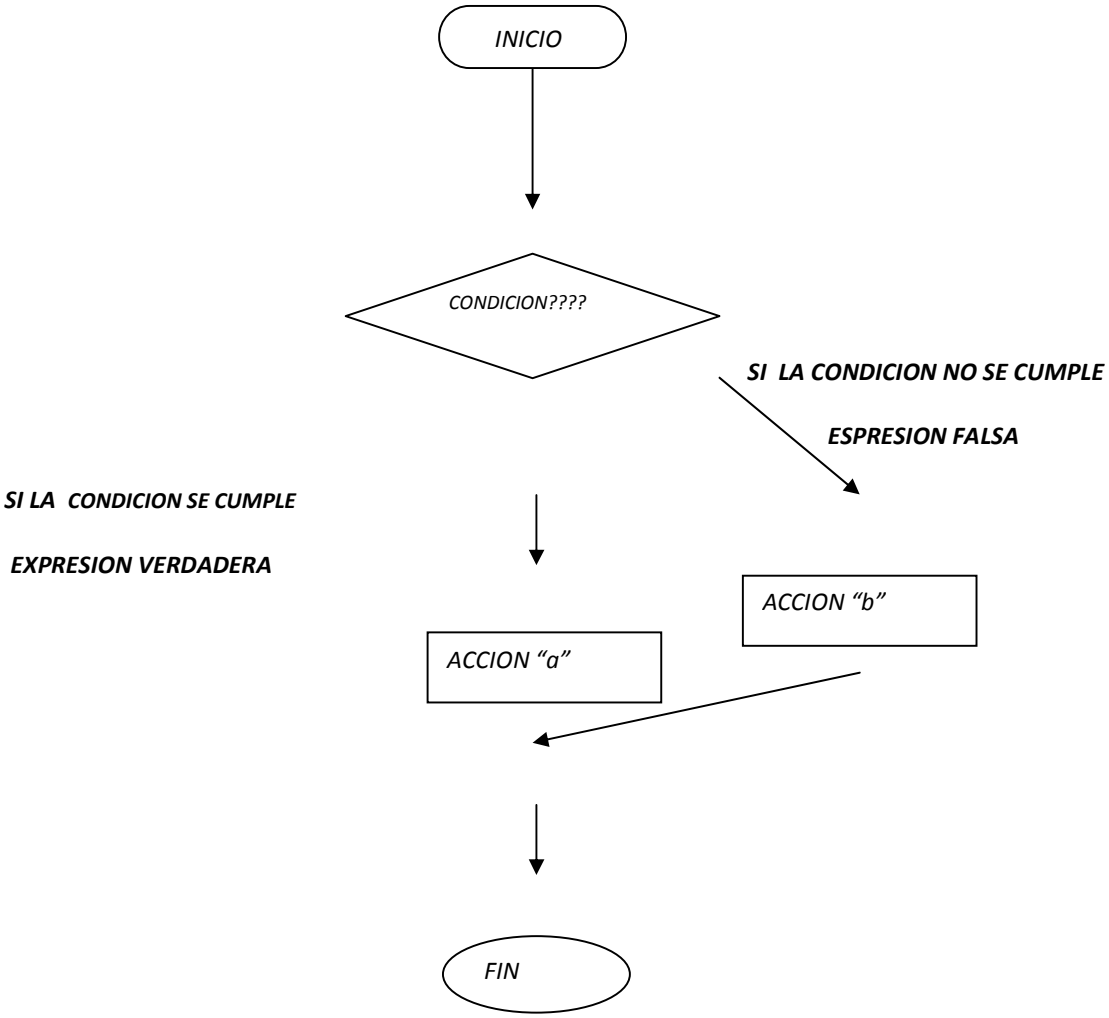
e) 3 resistores en paraelo.

COMPETENCIA PARTICULAR 2
Diseña programas empleando las estructuras de control condicionales y repetitivas basadas en el modelo de la programación estructurada.

Sentencias de control

La sentencia de control IF –ELSE se utiliza cuando en un programa requerimos que se realice una acción dependiendo de que una condición sea verdadera y en el caso de que dicha condición no se cumpla realizar otra acción.

Dicho en otras palabras Cuando nosotros hacemos una pregunta no realizaremos lo mismo si la respuesta en afirmativa o negativa pero sin embargo es obvio que tendremos que realizar una de loas dos acciones dependiendo de la respuesta.



Como se observo en el diagrama la condición debe ser una condición que se responda con falso o verdadero por ejemplo: a<6, c<=t, k==0, etc. Y si la condición es verdadera se realizara la acción "A" y en el caso de que no se cumpla se realizara la acción "b". (Note que solo se realizara una de las acciones.)

RAP 1:
Conoce los principios de la programación estructurada, así como las sentencias de control secuenciales.

Sintaxis de la sentencia if- else

```
if (CONDICIÓN)
{
acción o sentencia  "A" (cuando sea verdadera la condición)
}
else
{
acción o sentencia "b" (cuando no se cumpla la condición)
}
```

ALGUNOS PUNTOS IMPORTANTES:

Tome en cuenta que las llaves delimitan las sentencias o acciones a realizar para cada caso.

Cuando es una sola instrucción la contenida puede ser omitida la llave.

En los casos cuando no se va a realizar una acción cuando no se cumpla la condición se puede omitir el ELSE.

Un ejemplo sencillo de dicha instrucción es un programa que a partir de la edad dada por el usuario de cómo salida si la edad es mayor o menor de edad.

```
#include <stdio.h>

#include <conio.h>

main()

{

int edad;

clrscr();

printf("teclea el valor de tu edad");

scanf("%d",&edad);

if(edad<18)

{

clrscr();

printf( "ERES MENOR DE EDAD");

}

else

{

clrscr();

printf("ERES MAYOR DE EDAD");

}

 getch();

return 0;

}
```

A continuación se muestra un ejemplo de un programa que determina la hipotenusa y el ángulo de un triangulo rectángulo así como el valor del ángulo pero en el caso de que alguno de los catetos dados valga cero el programa manda un mensaje de error mediante una sentencia de control IF ELSE

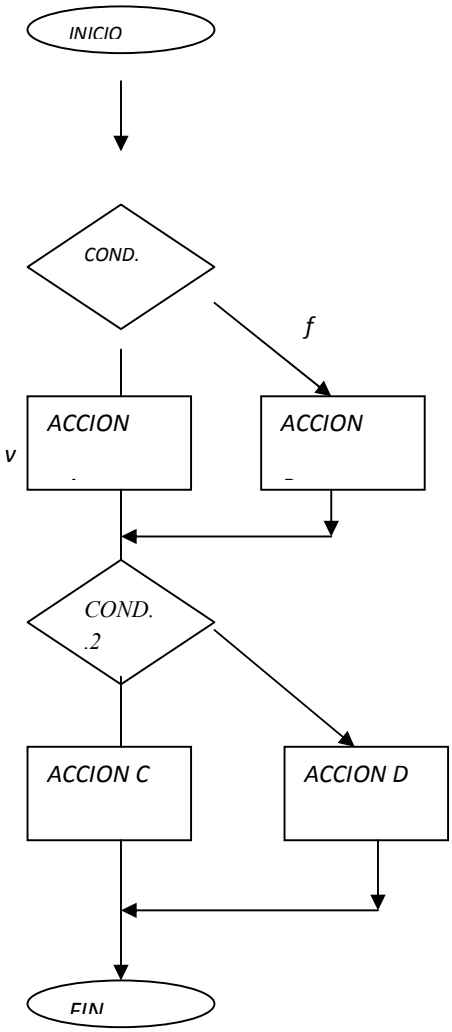
```
#include <stdio.h>

#include <conio.h>

#include <math.h>

#define PIRAD 57.295779

main()

{

float a,b,c;

double angulo,t;

clrscr();

printf("\n\n\n\nEscribe el valor del cateto opuesto: ");

scanf("%f",&b);

printf("\n\nEscribe el valor del cateto adyacente: ");

scanf("%f",&a);

c=sqrt((a*a)+(b*b));

t=b/a;

angulo=(atan(t))*(PIRAD);

if(a<=0 || b<=0)

{

clrscr();

printf("\n\n\n\n\n\n\t\t-Error al teclear valores de los catetos!");

}

else

{
```

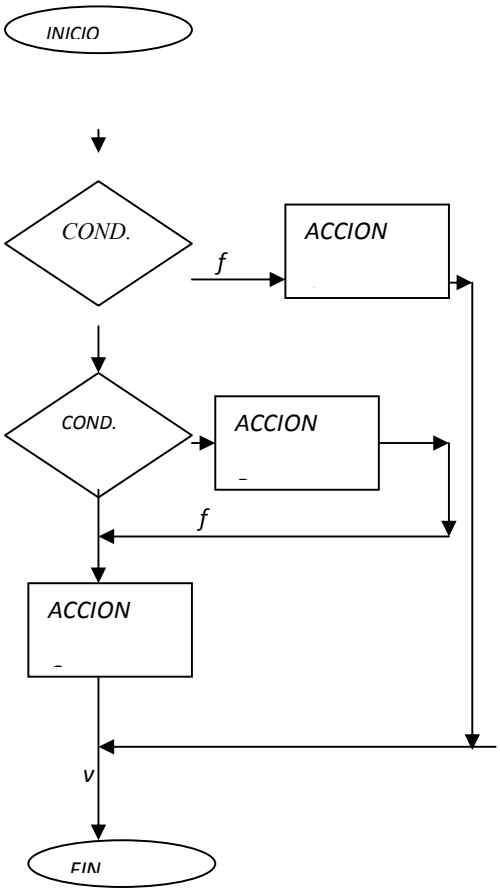
```
clrscr();

printf("\n\n\n\n\n\n\t\tLa hipotenusa vale: %f",c);

printf("\n\n\n\t\tEl ángulo vale %lf ø ",ángulo);

}

getch();

return 0;

}
```

RAP 2:
Aplica las distintas estructuras de control condicional simples y anidadas.

SENTENCIA IF ANIDADAS.

Cuando nos referimos a el término ***anidado*** *debemos entender que se refiere cuando dos o más sentencias se encuentran una dentro de otra, osea, no de forma lineal por Ejemplo observe los diagramas de flujo y advierta que diferencia existe.*





Si observa en el caso 1 con la primera condición da la opción para que se realice la acción A ó la Acción B para después por medio de la condición 2 se realice la acción C o bien la D, lo cual lo podríamos interpretar como dos sentecias IF- ELSE continuas una después de la otra .

Pero para el caso 2 vemos que la condición 2 solo se puede ejecutar cuando la condición 1 sea verdadera, de lo cual podemos observar que la condición 2 está dentro de la condición 1 por lo que este ejemplo se trata de un IF ELSE dentro de otro IF ELSE a lo que daremos el nombre de ***IF's anidados*** *.la sintaxis para los dos casos de presenta a continuación:*

Caso 1

```
If(condicion 1)
{
accion a
}
else
{
accion b
}
if(condicion 2)
{
accion c
}
else
{ accion d
}
```

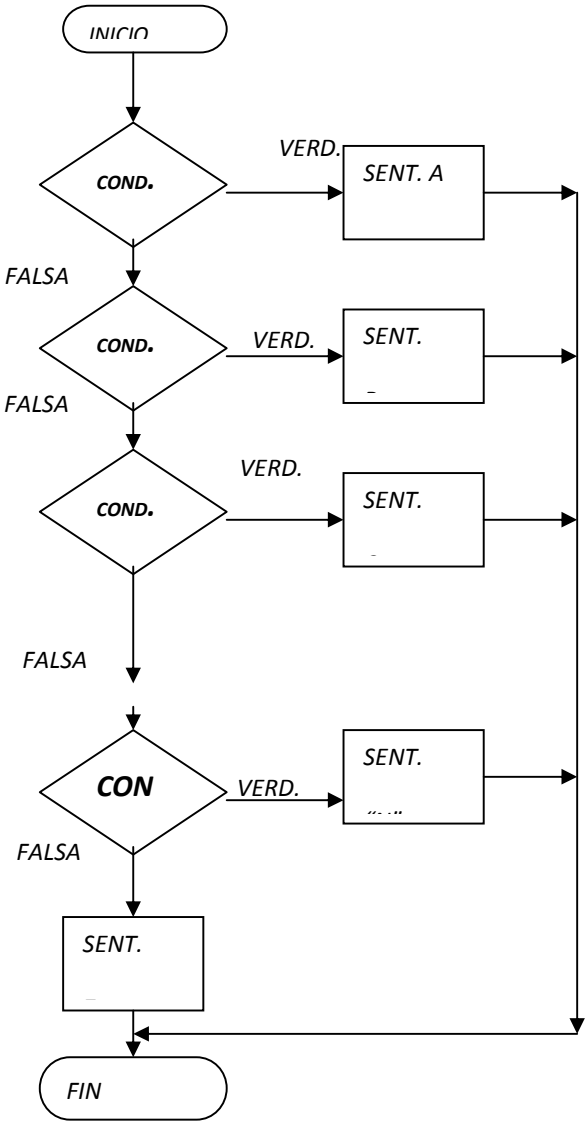
Caso 2

```
If(condicion 1)
{
 if  (condicion 2)
  {  accion b
   }
  else
   {  accion c
    }
 }
else
{accion a
}
```

Observe cuidadosamente el uso de las llaves y vea que para el caso 2 estas son las que van a servir para indicar el anidamiento de unas sentencias con otras.

SENTENCIA ELSE-IF

Esta sentencia se utiliza cuando requerimos ser más específicos en la condición del programa con respecto al if-else el cual en su forma simple solo tiene una condición y dos posibles sentencias a realizar dependiendo que esta se cumpla o no. De tal forma que para ser más específicos podemos condicionar en varias ocasiones , cada una de las condiciones tendrá una acción asociada la cual solo se realizara cuando dicha condición sea verdadera y de no cumplirse la primera seguirá "preguntando" en casa una de las condiciones y si ninguna de ellas se cumpliera se realizara una sentencia asociada un else en un diagrama de flujo se vería de la siguiente forma:


INICIO
COND.
VERD.
SENT. A
FALSA
COND.
VERD.
SENT.
FALSA
COND.
VERD.
SENT.
FALSA
CON
VERD.
SENT.
FALSA
SENT.
FIN

SINTAXIS DE LA SENTENCIA ELSE- IF

If (condicion)

{ *Sentencias A}*

else if(condición 2)

{*Sentencias B****}***

else if(condicion 3)

{*Sentencias C****}***

……..

else if(condicion "n")

{*Sentencias "N"****}***

else

{*Sentencias Z}*

Observe este sencillo ejemplo del uso de la sentencia else-if

```
# include <stdio.h>
# include <conio.h>
#include <dos.h>
main ()
{
int opcion;
```

```
clrscr();


printf("teclea un numero del 1 al 5");
scanf("%d",&opcion);
if(x==1)
{
printf("TU COLOR FAVORITO ES EL VERDE\n\t\a");
delay(300);
printf("suerte")
}
else if(opcion== 2)
{
clrscr();
printf("TU COLOR FAVORITO ES EL AZUL\n\t\a");
delay(300);
printf("suerte");
}
else if(opcion== 3)
{
clrscr();
printf("TU COLOR FAVORITO ES EL VIOLETA \n\t\a");
delay(300);
printf("suerte");
}
else if(opcion== 4)
{
```

```
clrscr();
printf("TU COLOR FAVORITO ES EL AZUL\n\t\a");
delay(300);
printf("suerte");
}
else if(opcion== 5)
{
clrscr();
printf("TU COLOR FAVORITO ES EL AMARILLO\n\t\a");
delay(300);
printf("suerte");
}
else
{
printf("tecleaste un numero no validpo rtecuerda que era del 1 al 5");
printf("mas cuidado para la proxima");
getch();
return 0;
}
```

Del programa anterior observe lo siguiente:

Que es importante el uso de las llaves.

Que todas las condiciones estaban relacionadas con una variable en este caso "opcion"

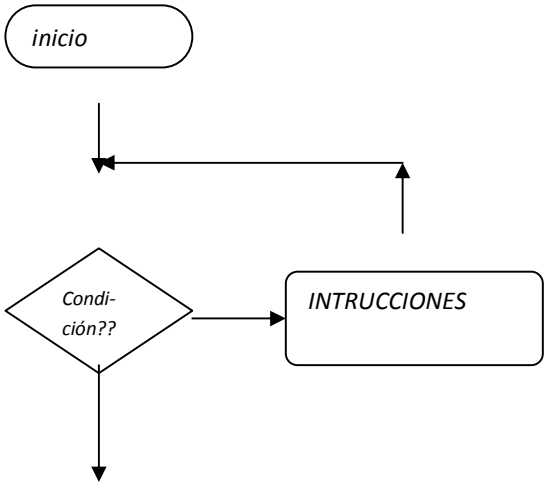
Solo una de las acciones se va a realizar en función de que se cumpla la condición

Que en el caso que no se cumpla ninguna de las condiciones se va a realiza la acción que este dentro del único else (el de el último).

RAP 3:
Emplea las diversas estructuras de control repetitivas simples y anidadas.

Sentencia de Control "while"

***A**l enunciado while es también se le llama "ciclo while" o bien "bucle do whie" el nombre de "ciclo " o "bucle" lo recibe ya que esta instrucción se utiliza cuando queremos que un segmento de programa (un bloque de instrucciones) se realice un cierto numero de veces, tales como salidas de datos o entradas. A continuación se ilustra por medio de un diagrama de flujo.*



CUANDO **SE EJECUTA UNA SENTENCIA WHILE PASA LO SIGUIENTE:**

1.- Se evalúa la condición.

2.- Si la condición se evalúa falsa (no se cumple la condición) la ejecutará la instrucción siguiente fuera del ciclo o bucle

3- Si la condición se evalúa verdadera (la condición se cumple) se ejecutan las instrucción contenidas dentro del ciclo while.

4- Una vez ejecutadas vuelve a condicionar (Paso 1).

SINTAXIS DE LA SENTENCIA WHILE

while(condición)

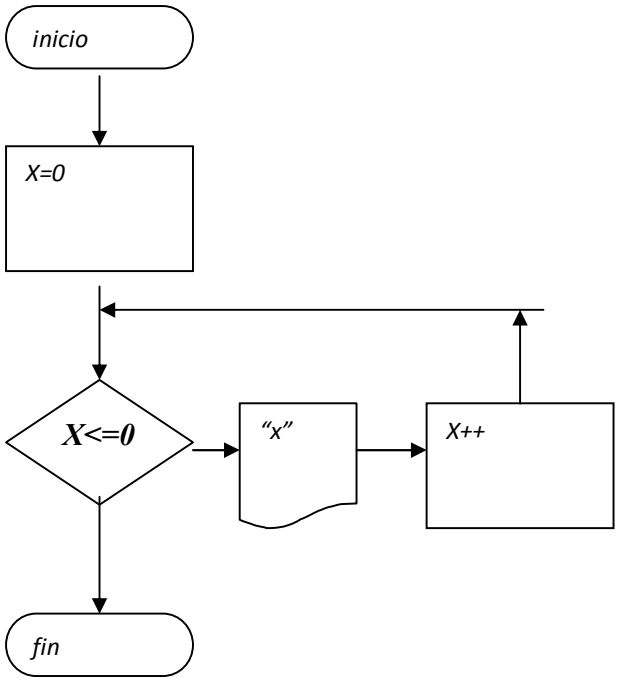
{

instrucciones;

}

Un ejemplo claro de aplicación de una sentencia while es un contador, pues es más fácil repetir con algunas modificaciones un "printf" por medio de un bucle, que poner una instrucción "printf" por cada numero que se desee poner en la cuenta.

Ejemplo :




```
main()
{
int x=0;
while(x<=0)
{

printf("%d",x);

x++;

}
```

Observe lo siguiente:

Las llaves están delimitando las instrucciones que contiene el ciclo.

El programa imprimirá una cuenta de 1 en 1 del 0 al 10.

Observe la condición.

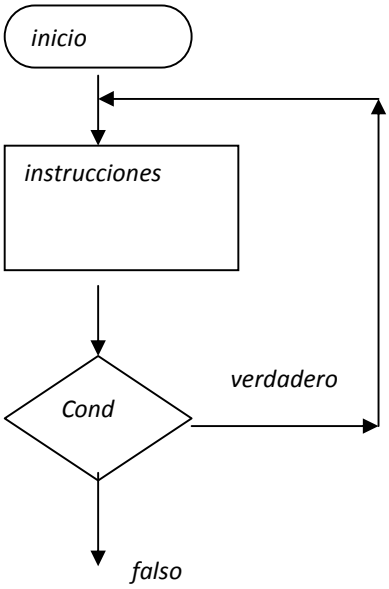
Aquí se presenta otro ejemplo de uso de la sentencia "while":

¿ Qué salida da el siguiente programa?

```
# include <stdio.h>
# include <conio.h>
# include <math.h>
main ()
{
clrscr ();
int i=0,n,j,potencia;
printf ("Programa que cuente desde 2 exp 0 hasta 2 exp n\n");
printf ("Dame el valor del exponente");
scanf ("%d", &n);
while (i<=n)
{
potencia=pow(2,i);
i++;
}
printf ("%d", potencia);
getch ();
return 0;
```

Sentencia de control do-while

Esta sentencia de control al igual que la "while" es una sentencia que nos va servir para hacer ciclos mo bucles dentro de nuestro programa básicamente con una diferencia; en lo que en la sentencia "while " si la condición se cumple se jecuta o no se ejecuta una serie de instrucciones , en la "do-while " se ejecutaran al menos una vez las instrucciones para que después se condiciones, observe el diagrama de flujo.



continúa

CUANDO SE EJECUTA UNA SENTENCIA do-while PASA LO SIGUIENTE:

1.- Se ejecutan las instrucciones correspondientes (las que se pretendesnponer dentro del ciclo).

2.- la condición es evaluada , si es verdadera la ejecucuón se va hacia el paso 1 pero si es falsa el ciclo termina y se ejecutan las instrucciones imediatas.

Sintaxis de una sentencia do-while

do {

Instrucciones

} while (cond);

Observe lo siguiente:

1.- al igual que en el "while" las llaves delimitarán las instrucciones que se desean poner dentro del ciclo.

*2.- do while es una de las pocas instrucciones que si llevan **;** en la condicion.*

3.- Observe que las instrucciones aun cuando la condición sea verdadera se ejecutaran almenos una vez.

Ejemplos de aplicación de la sentenciado-while.

Ejemplo 1.- Contador descendente.

```
main()
{
int a=100;
do{
printf(""%d",a);
a-=5;
    } while(a>=0);
 }
```

Ejemplo 2.- programa que calcula el promedio de"n" números usando la sentencia do-while.

```
main()
{
float  n,c,s,x,p;
clrscr();
printf (" PROGRAMA QUE CALCULA EL PROMEDIO  DE LA CANTIDAD DE NUMEROS QUE TU  QUIERAS \n");
delay(550);
printf("TECLEA LA CANTIDAD DE NUMEROS");
scanf("%f", &n);
c=1;
s=0;
do{
printf("INGRESA  tu %3.0f  er   VALOR A PROMEDIAR  \n",c);
scanf("%f",&x);
c++;
s+=x;
  }while(c<=n);
 p=s/n;
printf("TU  PROMEDIO TOTAL DE TUS %2.0f NUMEROS ES %f",n,p);
}
```

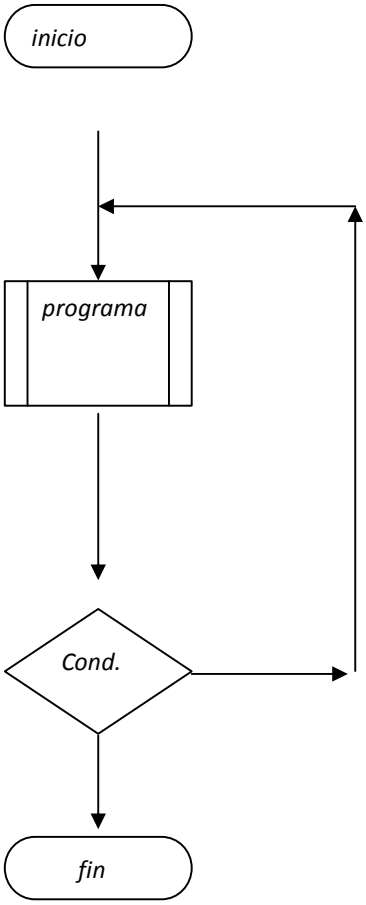
ACTIVIDADES:

1.- Modifique el programa anterior para que determine:

El valor de "n" resistores en serie

El valorde "n" resisores en paralelo.

Otra aplicación muy útil para una sentencia do while es usarla para la ejecución repetida de programas cuando sea puesta una condición:



.

Ejemplo 3.- aqui se muestra un ejemplo de un programa que se ejecuta de Nuevo cuando se tecle al final un “5”.

```
# include <stdio.h>
# include <conio.h>
# define pi 3.1416
main ()
{
int r;
float d,a,p;
do{
clrscr ();
printf("Programa para calcular el area y diametro de una circunferencia\n");
printf("Dame el diametro\n");
scanf("%f",& d);
a=4*pi*d/2;
p=pi*d;
printf("El area es: %f\n",a);
printf("El perimetro es: %f",p);
printf(“PARA VOLVER A EJECUTAR EL PROGRAMA APRIME UN 5”);
scanf(“%d”,&r);
}while(r==5);
getch ();
return 0;}
```

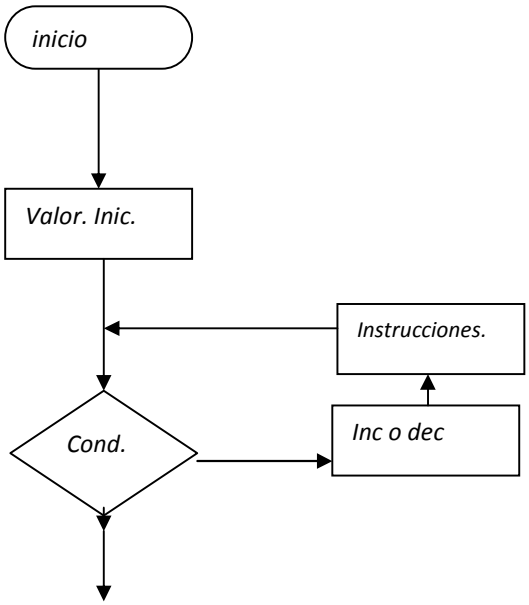
IMPORTANTE

Recuerde que es posible anidar diversos tipos de sentencias de control solo usando correctamente las llaves.

Sentencia de Control “ for ”

La sentencia de control “ ***ciclo for*** *” al igual de la sentencias while y do-whille son sentencias que dan lugar a bucles o ciclos dentro del programa, destacando cada uno de ellas con sus características especificas, por ejemplo el ciclo for básicamente es una sentencia mediante la cual vamos a poder diseñar contadores de una forma mucha mas fácil que con ciclos vistos anteriormente y cuyas aplicaciones veremos mas adelante.*

Visto desde un diagrama de flujo el ciclo for funciona de la siguiente forma:



¿Qué sucede cuando se ejecuta una instrucción “for”?

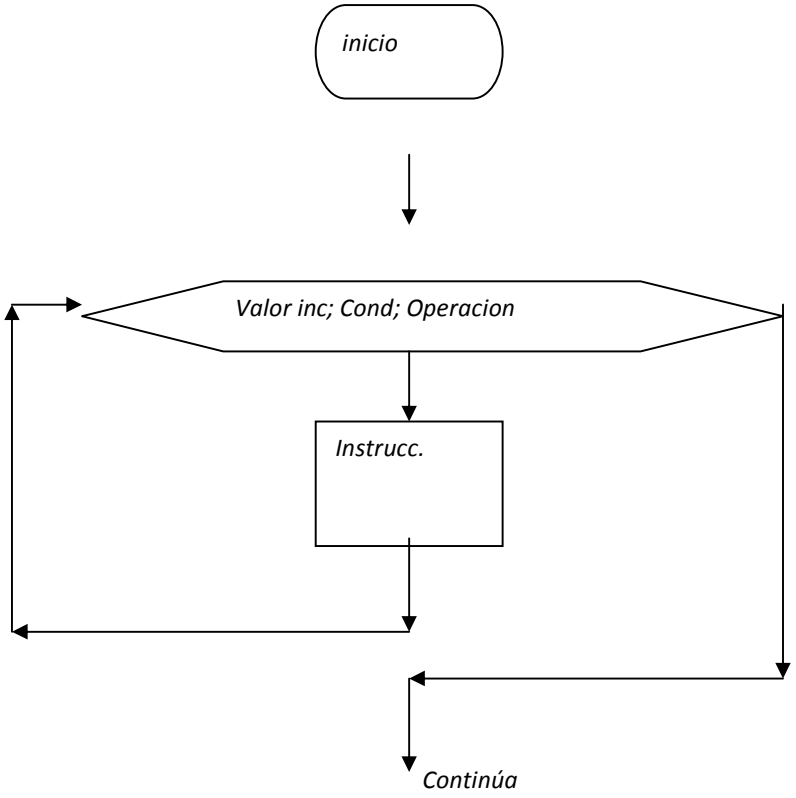
1.- Se asigna un valor inicial a una variable que será evaluada.

2.- Se condiciona la variable , si la expresión es falsa sale del ciclo y continua con la instrucción siguiente , pero si es verdadera continua.

3.- Se realiza sobre la variable una operación (típicamente un incremento o bien decremento).

4.- Se realizan las instrucciones contenidas dentro del ciclo. Se va al paso 2.

Otra simbología del for en diagramas de flujo.



Observe la otra forma que se tiene de expresar el ciclo for observe que se colocaron tres datos en una solo lugar el valor inicial, la condición que se debe de cumplir para que se ejecuta el bucle así como la operación que ya se había mencionado normalmente son incrementos o decrementos, se colocaron precisamente de esa forma por que por sintaxis todos esos datos lleva la instrucción, observe a continuación.

Sintaxis de la sentencia for.

for(valor inicial; condición; operación)

{

instrucciones;

}

IMPORTANTE:

- *Recuerde que el carácter ";" solo va entre las condiciones y no al final.*
- *Un error en la condición puede provocar ciclos infinitos, asea que nunca terminen provocando asó problemas con la maquina.*

Ejemplo 1

Como ya se dijo el ejemplo más típico de un ciclo for es un contador observe la sencillez del programa:

```
main()
{
 int a;
clrscr();
printf(" PROGRAMA CONTADOR DE 2 EN 2 HASTA EL 100");
for(a=0, a<=100;a+=2)
{
printf("%d\t",a);
}
}
```

Ejemplo 2

El factorial de un numero se define matemáticamente de varias formas y este valor tiene gran utilidad en probabilidad y estadística pero en forma general se darán algunos ejemplos con el fin de que usted solo observe que se utiliza una forma de conteo su símbolo es ***"!".***

a) *El factorial de 3 se escribe 3!= 3(2)(1)=6*

b) *El factorial de 5 se escribe 5¡ = 5(4)(3)(2)(1) = 120*

A continuación se muestra un programa que determina el valor del factorial de un número por que como se observa el factorial es la suma de los productos de los elementos de un contador.

```
# include <stdio.h>
# include <conio.h>
main()
{
clrscr ();
int i,f=1,a;
printf("Programa para calcular el factorial de un numero\n");
printf("Dame el numero\n");
scanf("%d",&a);
for(i=1;i<=a;i++)
{
f=f*i;
}
printf("\t%d\n",f);
getch();
return 0;}
```

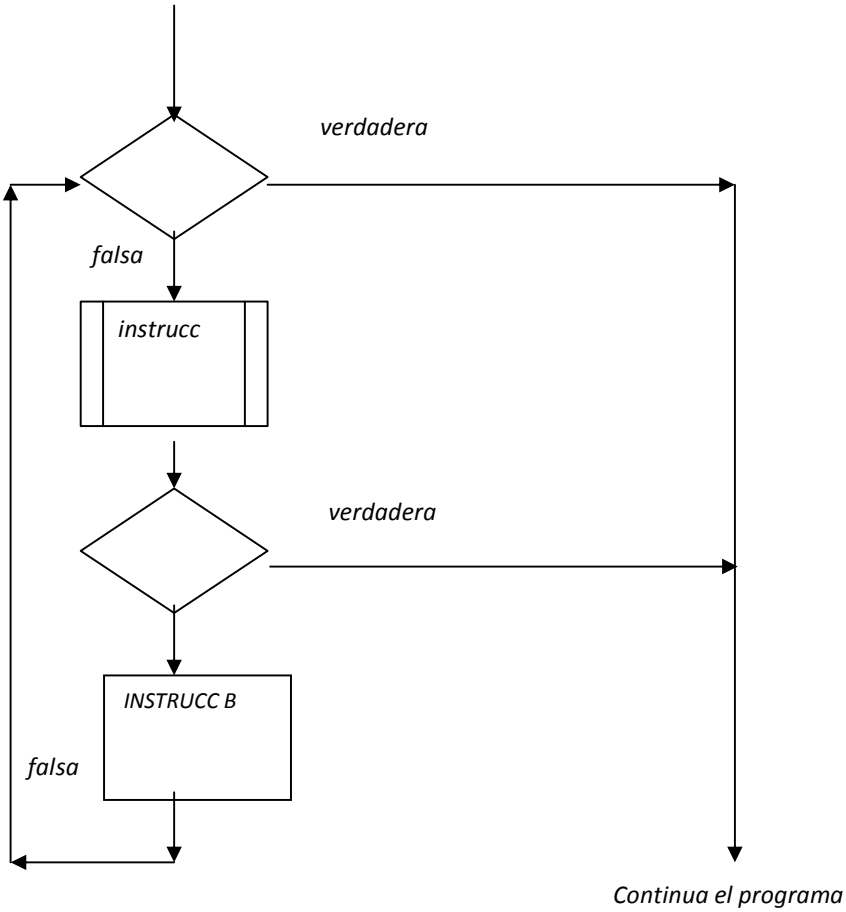
Suponga que se ingresa un 6 ¿cuál sería la salida del programa?

Instrucción break.

La sentencia break es una instrucción que sirve para salir de un ciclo (sentencia iterativa) o bien de un "case" en el caso de la sentencia switch en un cualquier punto del programa.

La función básica es que cuando se cumple una condición asociada con la instrucción break el programa se sale del bucle en donde se este trabajando.

A continuación se muestra por medio de un diagrama de flujo.



Observe lo siguiente.

En lo general la estructura muestra un ciclo como los que ya se vieron (while o for, etc) que depende de la condición 1, si esta es cierta se seguirá realizando dicho bucle con la característica que dentro de

dicho bucle habrá una condición por medio de un if la cual cuando sea verdadera (se cumpla) y tener enseguida una instrucción break la ejecución del bucle se interrumpirá y se saldrá incluso omitiendo la ejecución de la instrucción(es) señaladas como B.

SINTAXIS DE LA SENTENCIA BREAK

If(condición)

break;

***A** continuación presenta un ejemplo de un programa que determinara el promedio de hasta cien números el usuario deberá pulsar el carácter "f"para indicar que es el último número.*

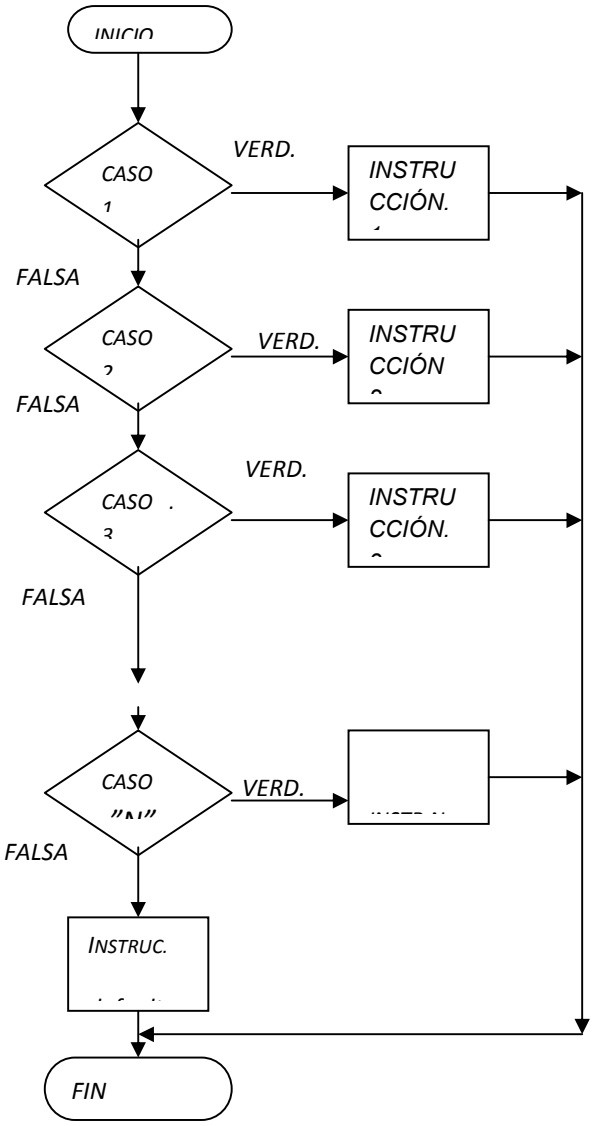
Si observamos al oprimir el carácter "f" el break provocara que el programa se salga del bucle y continuara ejecutando las instrucciones fuera del ciclo for.

```
main()
{
int contador;
flota suma=0, promedio, numero;
char salida;
clrscr();
printf(" ESTE PRPGRAMA DE CALCULA  EL PROMEDIO DE HASTA CIEN NUMEROS\n CUANDO SE LO INDIQUE INGRESE LOS NUMEROS Y AL TERMINAR DE INGRESAR EL ULTIMO NUMERO OPROIMA LA TECLA f \n");
printf(" OBSERBAREMOS LA FUNCION DE LA SENTENCIA BREAK");
for( contador=0;contador<=100;contador ++)
    {
     printf("INGRESA EL VALOR DE LA CANTIDAD %d",contador);
     scanf( " %f",&numero);
```

```
      suma+=numero;
      printf("\n\nSI ES EL ULTIMO NUMERO TECLEA LA f  DE LO CONTRARIO CUALQUIER OTRA\n");
       scanf("%c",&salida);
       if(salida =='f')
       break;
     }
promedio= suma/numero;
printf("\n EL PROMEDIO DE LOS %d NUMEROS ACCESADOS ES; %f\n");
}
```

Instrucción switch.

La instrucción switch en una instrucción que sirve para seleccionar uno de varios programas contenidos en una misma estructura, dicho en otras palabras nos da la posibilidad de hacer un menú en donde se selccionara uno de varios conjustos de instrucciones visto en un diagrama de flujo se expresaria como:



Si observamos el diagrama de flujo observamos que una sola variable se podrá condicionar un gran numero de veces (al igual que el sentencia else-if) y de varios grupos de instrucciones solo se ejecutarán los que se encuentren dentro de cada condición si dicha condición se cumple y (que en el diagrama son llamadas casos) pero si ninguna de las condiciones o casos se cumplen entonces se realizará un bloque instrucciones contenidas dentro del default.

Sintaxis de la sentencia switch

switch(variable)

{

case ´caracter ':

*instrucciones del primer caso**;***

break;

case ´caracter´:

instrucciones del Segundo caso;

break;

case ´caracter´:

intrucciones del tercer caso;

break;

*****

case ´caracter´:

instrucciones del caso"n"

break;

dafault:

instruciones cuando no se cumpla ningún caso;

break:

}

Donde:

variable *corresponde a una variable de tipo caracter donde se guardara la selección ingresada por el usuario.*

caracter *corresponde al caracter conel cual se ejecutaran el bloque de instrucciones respectivo.*

break *como ya se vió tiene como función la de sacar la elecución del programa de una estructura en este caso la del swith una vez que se ha ejecutado uno de los bloques de instrucciones.*

EJEMPLO

A continuación se presenta un programa que ilustra la sentencia swtch, cuya salida será un horóscopo seleccionado por el usuario, observe cada una de sus líneas.

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

main ()
{
char escoge;
clrscr ();
printf("Programa que de el horoscopo para cada signo zodiacal escogido\n");
printf("Escoge un signo zodiacal\n");
printf("Aries            a\n");
```

```
printf("Cancer          b\n");
printf("Tauro           c\n");
printf("Geminis         d\n");
printf("Leo             e\n");
printf("Libra           f\n");
printf("Virgo           g\n");
printf("Sagitario       h\n");
printf("Escorpion       i\n");
printf("Acuario         j\n");
printf("Piscis          k\n");
printf("Capricornio     l\n");
scanf("%c",&escoge);
switch(escoge)
{
case 'a':
printf("Jupiter te dara unos dias con muchas sorpresas. Tal vez puedas\n");
printf("sentirte un poco aturdido por que a veces no sabes controlar tus\n");
printf("emociones. A veces te gusta llevarle la contraria a los demas y\n");
printf("esto puede traerte problemas.");
break;
case 'b':
printf("Eres sentimental la mayoria de las ocasiones y necesitas mucho mas\n");
printf("apoyo del que supones. Tu familia es uno de tus mayores refugios.\n");
printf("Trata de que los demas se involucren en tu vida y veras que es\n");
printf("mas facil superar tus problemas.");
break;
case 'c':
printf("En estos dias te va visitar un amigo al que no ves desde hace\n");
```

```
printf("tiempo con el que te vas a divertir mucho. Tal vez empieces a extra¤ar\n");
printf("a tu pareja, pronto van a volver a estar juntos\n");
break;
case 'd':
printf("Alguien que te agrada muy pronto te va invitar a salir, lo mejor es\n");
printf("que tu tambien le agradas y lo mas probable es que inicien una\n");
printf("relacion. Tus amigos van a organizar un viaje de fin de semana,\n");
printf("date un tiempo y ve con ellos.");
break;
case 'e':
printf("En ocasiones puedes ser muy rebelde y por esta razon te metes en\n");
printf("problemas de los cuales te cuesta trabajo salir. Uno de tus mejores\n");
printf("amigos puede decirte algo sobre tu pareja que te intrigara. Platica\n");
printf("con ambos para escucharlos y no acabar con ninguna relaci¢n");
break;
case 'f':
printf("Para alcanzar tus objetivos tienes que poner mucho de tu parte\n");
printf("y no dejarle todo a la suerte. Conoceras a una persona de Geminis\n");
printf("con la que te vas a identificar muy bien. Aprovecha tu tiempo libre\n");
printf("para descansar y reponer tus energias.");
break;
case 'g':
printf("Despues de todos los problemas que has tenido en la relacion con\n");
printf("tu pareja, las cosas empezaran a mejorar, asi que vas a entrar en \n");
printf("un periodo tranquilo y positivo. Un -amigo- que te quiere hacer da¤o\n");
printf("puede inventar un algo, procura no hacer caso de ello.\n");
break;
case 'h':
```

```
printf("No te gusta que te mencionen tus errores, pero acuerdate que quienes\n");
printf("te lo dicen son las personas que mas te quieren.\n");
printf("Los planes que tienes con respectoa tu trabajo se van a cumplir\n");
printf("favorablemente y estaras mas tranquilo\n");
break;
case 'i':
printf("En menos tiempo del que crees se va presentar la oportunidad de\n");
printf("mejorar tus ingresos asi que aprovecha al maximo. Por falta de \n");
printf("dialogo tienes problemas con personas que conoces hace poco tiempo\n");
printf("y quiza sea el momento perfecto para darte cuenta de como son realmente\n");
case 'j':
printf("Mucha gente esta en espera de que hagas algo mal para poder hacerte\n");
printf("da¤o, por eso tienes que concentrarte para evitar que los demas\n");
printf("hablen mal de ti. A veces tienes temor al rechazo en tu grupo de\n");
printf("amigos porque eres mas serio que ellos.\n");
break;
case 'k':
printf("Tus problemas familiares estan a punto de solucionarse definitivamente\n");
printf("lo cual te permite dedicarle mas tiempo a tus amigos y asuntos\n");
printf("personales. Es un buen momento para iniciar un romance. Alguien\n");
printf("que tu conoces desconfia un poco de tus capacidades.\n");
break;
case 'l':
printf("Estas en un periodo excelente para cambiar los aspectos de tu vida\n");
printf("que no te agradan mucho. Te pierdes mucho en conversaciones absurdas\n");
printf("con personas demasiado necias que tienes cerca, mejor dejalas a un\n");
printf("lado y busca a alguien mas positivo\n");
break;
```

```
default:
printf("OPCION INCORRECTA ");
break;
}
getch();
return 0;

}
```

IMPORTANTE

- En mayoría de los programas que engloban diversos tipos de instrucciones para que la programación sea óptima deberán ser implementados con funciones tema que se abordara en el próximo capitulo.

Actividades

Competencia particular 2

1. *Implemente un programa libre sobre la aplicación de la sentencia else-if.*

2. *Indique en qué se diferencia una sentencia if-else de una else if.*

3. *Realice un programa usando la sentencia else if que de la opción de calcular de la ley de ohm:*
    a. *I a partir de V y R*
    b. *V a partir de I y R*
    c. *R a partir de I y V*

4. *Implemente un programa que usando la sentencia while de cómo salida un contador del 0 al 100. (los números deberán aparecer de uno en uno en pantalla con retardos de tiempo).*

5. *realice un programa que de cómo salida una tabla de multiplicar dada por el usuario.*

6. *Haga un programa que de cómo salida los valores de y para la ecuación y = 4x+6 en un intervalo de x de –5 a 5.*

7. *realice un programa que de cómo salida un contador en donde los intervalos , inicio y final de conteo serán dados por el usuario.*
8. *Implemente un programa que imprima todos los caracteres ASCII usando sentencia do- while.*
9. *(es un contador definido pero para que imprima los valores hay que poner en el printf de salida hay que sustituir %c)*

10. *2.- mediante una sentencia do while haga que el programa de un contador de 1 en hasta 100 se repita cuando sea oprimido un numero diferente de 2 .(¡= diferente).*

11. *Realice un programa que Imprima en pantalla todos valores del seno de 0 a 360 grados mediante una sentecia "do-while" .*

12. *Realice un programa que imprima los valores de 2 a la "0" a 2 a la "n" .*

13. *Implemente un programa que solo imprima los números divisibles entre 3 y entre 2 de una cuenta de 0 a 100.*
14. *(anidar un do-while con un if.)*
15. *¿Cómo podría hacer un programa que realice la misma función pero utilizando en su estructura en for que de lugar a un contador descendente?*

16. *Realice un programa que determine el promedio de "n" números elegidos por el usuario,*

17. *Implemente un programa que de cómo salida los valores de la función coseno de grado engrado(sexagecimal)) en un intervalo ascendente dado por el usuario.*

18. *Por medio de una sentencia for haga un programa que imprima en pantalla lo siguiente: (la impresión deberá ser carácter por carácter).*

    1. *********
    2. *********
    3. *********
    4. *********
    5. *********

19. *Realice un programa que tabule la distancia recorrida en un intervalo de tiempo dado por usuario de un móvil de aceleración constante la cual también será proporcionada por el usuario.*

20. *Implemente un contador que cuente primero de uno en uno del 0 al 99 para después ir descendiendo de uno en uno hasta llegar de nuevo a cero.*

21. *Realice un programa que sea un contador descendente con intervalos, valor inicial y final dados por el usuario pero cuando los valores ingresados no correspondan a un contador descendente no mande un mensaje de error y vuelva a preguntarlos.*

22. *Haga un contador de 0 a 100 de 2 en 2 el cual se deberá ejecutar de forma ininterrumpida cuando el usuario oprima un tecla.*

23. *Realice un programa que imprima en pantalla las 10 tablas de multiplicar.*

24. *Determine la función y salida del siguiente programa:*

```
# include <stdio.h>
# include <conio.h>
main()
   {
   clrscr ();
   int p,i,v,s=0;
   float p;
   printf("Dame la cantidad \n");
   scanf("%d",&p);
```

```
for(i=0;i<=p-1;i++)
{
printf("Dame el numero\n");
scanf("%d",&v);
s=s+v;
}
p=s/p;
printf("El resultado  es: %f",p);
getch();
return 0;
}
```

25. *Implemente el programa que determine el suma de "n" resistencia en paralelo pero que se utilice la tecla para indicar que se a accesado el último valor del resistor.*

26. *realice un programa que pregunte una clave secreta hasta 1000 veces, en el caso de ingresar la clave correcta se debe de ejecutar un programa cualesquiera.*

27. *Modifique el programa anterior añadiendo un bucle infinito.*

28. *Mediante la función de números aleatorios realice una ruleta electrónica.*

29. *implemente un programa que por medio de un menú de la opción de investigar las principales características de las sentencias de control vistas hasta el momento.*

30. *Al terminar dicho programa deberá preguntar si se vuelve a ejecutar o no.*

31. *realice un programa que de la opción de calcular cualquiera de las tres variables de la formula de v=d/t;*

32. *implemente un programa libre que ilustre un uso de sentencia switch.*

33. *Realice un programa que de la opción de imprimir en pantalla un contador ascendente o bien , descendente. Para cada caso los valores iniciales , finales e intervalos de conteo deberán ser dados por el usuario.*

34. *Modifique la codificación del siguiente programa para que realice la misma función pero con una sentencia swItCh.*

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

main ()
{
clrscr ();
char opcion;
int Voltaje,i,r,v;
float intensidad, Resistencia;
printf("Programa que da la opcion de calcular alguna de las tres variables de la ley de Ohm\n");
printf("\t Menu\n");
printf("\t Voltaje              V\n");
printf("\t Intensidad           I\n");
printf("\t Resistecia           R\n");
```

```
printf("\t Selecciona una opcion\n");
opcion=getche();
if(opcion=='V')
{
printf("Dame la intensidad\n");
scanf("%d",&i);
printf("Dame la resistencia\n");
scanf("%d",&r);
Voltaje=i*r;
printf("Voltaje = %d v\n",Voltaje);
}
else
{
if(opcion =='I')
{
printf("Dame el voltaje \n");
scanf("%d",&v);
printf("Dame la resistencia \n");
scanf("%d",&r);
intensidad= v/r;
printf("intensidad=%fA", intensidad);
}
```

```
else
{
printf("Dame el voltaje\n");
scanf("%d",&v);
printf("Dame la intensidad\n");
scanf("%d",&i);
Resistencia=v/i;
printf("Resistencia=%f Ohm",Resistencia);
}
}
getch ();
return 0;
}
```

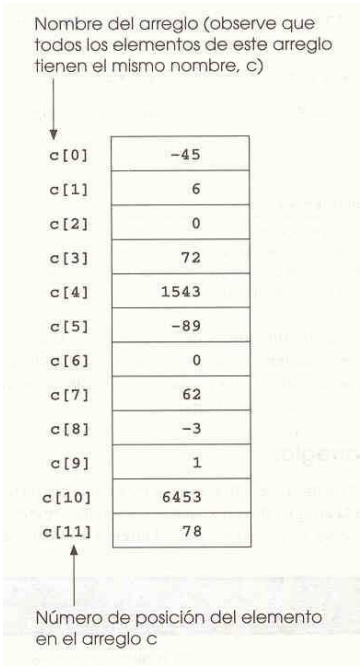
COMPETENCIA PARTICULAR 3
Realiza programas que implementan arreglos como herramienta de solución para resolver las problemáticas planteadas.

Arreglos.

Los arreglos son estructuras de datos que consisten de elementos de información del mismo tipo que están relacionados.

Definición.

Un arreglo es un grupo consecutivo de localidades de memoria que tienen el mismo nombre y el mismo tipo de datos. Para referirnos a una localidad o elemento particular de un arreglo, especificamos el nombre del arreglo y el numero de posición del elemento particular dentro de este.



RAP 1:
Utiliza los arreglos como herramienta auxiliar en la simplificación de variables del mismo tipo.

Sintaxis (Forma de declararlo).

Tipo_de_datos nombre_del_arreglo*[*numero_de_elementos*];*

Donde:

Tipo_de_datos *: Se refiere al tipo de datos que vas a guardar en el arreglo, recuerda todos los datos tienen que ser del mismo tipo (int, float, char, double, etc.).*

nombre_del_arreglo *: Es el nombre con el que te vas a referir al arreglo durante todo el programa, elige un nombre no mayor a 8 caracteres y que sea relacionado con el uso que le vas a dar al arreglo.*

numero_de_elementos *: Es la cantidad de elementos que vas a guardar en el arreglo.*

Ejemplos de declaración de arreglos:

```
int c[12];          /* Arreglo de 12 datos enteros */

char cadena[30];    /* Arreglo de 30 caracteres */

float numeros[100]; /* Arreglo de 100 numeros flotantes*/
```

El primer elemento de cualquier arreglo es el elemento cero. Por lo tanto, nos referimos al primer elemento del arreglo nombre*[0];*

Al numero de posición entre los corchetes se le conoce como subíndice. Los subíndices deben ser enteros o expresiones enteras.

Ejemplo

El siguiente programa es un ejemplo de limpiar o poner a ceros todas las localidades de un arreglo, además de imprimirlo en la pantalla.

```
/* Programa que inicializa en cero e imprime un arreglo */

#include <stdio.h>
#include <conio.h>

void main(void)
{
 int i, n[10];

 clrscr();
 for(i = 0; i < 10; i++)           /* Inicializa el arreglo en ceros */
      {
       n[i] = 0;
      }
 printf("Elemento \t Valor \n");
 for(i = 0; i < 10; i++)           /* imprime el arreglo */
      {
```

```
       printf("\t %d \t\t %d \n",i,n[i]);
      }
 getch();
}
```

El siguiente programa es un ejemplo de como introducir valores al arreglo desde el inicio de la ejecución del programa.

```
/* Programa que inicializa con valores predeterminados e imprime un arreglo */

#include <stdio.h>
#include <conio.h>

void main(void)
{
 int i, a[6]={32,26,64,18,95,14};  /* Inicializacion de un arreglo */
                                    /* con valores predeterminados */
 clrscr();
 printf("Elemento \t Valor \n");
 for(i = 0; i < 6; i++)            /* imprime el arreglo */
      {
       printf("\t %d \t\t %d \n",i,a[i]);
      }
 getch();
}
```

Los siguientes programas son aplicaciones del uso de los arreglos.

Ejemplo

```
/* Programa que calcula la suma de los elementos de un arreglo */
#include <stdio.h>
#include <conio.h>
void main(void)
{
 const int datos = 6;
 int i, total = 0, n[datos]={1,6,4,8,5,4};/*Inicializacion del arreglo*/
                                     /* con valores predeterminados */
 clrscr();
 printf("Elemento \t Valor \n");
 for(i = 0; i < datos; i++)            /* imprime el arreglo */
      {
       printf("\t %d \t\t %d \n",i,n[i]);
      }
 for(i = 0; i < datos; i++)            /* suma de los elementos */
      {
       total += n[i];
      }
 printf("La suma de todos los elementos del arreglo es %d \n", total);
 getch();
}

/* Programa que imprime cadenas de caracteres guardados en arreglos */
```

```
#include <stdio.h>
#include <conio.h>
void main(void)
{
 int i;
 char cadena1[4] = {"HOLA"}, cadena2[6]; /*Inicializacion de la cadena1*/
                                  /* con valores predeterminados */
 clrscr();
 printf("introduce una cadena de caracteres de 6 letras maximo > ");
 scanf("%s", &cadena2);
 printf("\nLa cadena 1 tiene la palabra > %s", cadena1);
 printf("\nLa cadena 2 tiene la palabra > %s", cadena2);
 printf("\nLos elementos de los arreglos son:");
 printf("\n   cadena1 \t    Caracter \n");
 for(i = 0; i < 4; i++)            /* imprime cadena 1 */
      {
       printf("\t %d \t\t %c \n",i,cadena1[i]);
      }
 printf("\n   cadena2 \t    Caracter \n");
 for(i = 0; i < 6; i++)            /* imprime cadena 2 */
      {
       printf("\t %d \t\t %c \n",i,cadena2[i]);
      }
 getch();}
```

IMPORTANTE¡

Es importante que observes el uso de la estructura for() para el acceso a los arreglos, esta permitirá manejar arreglos mas complejos que a continuación se verán.

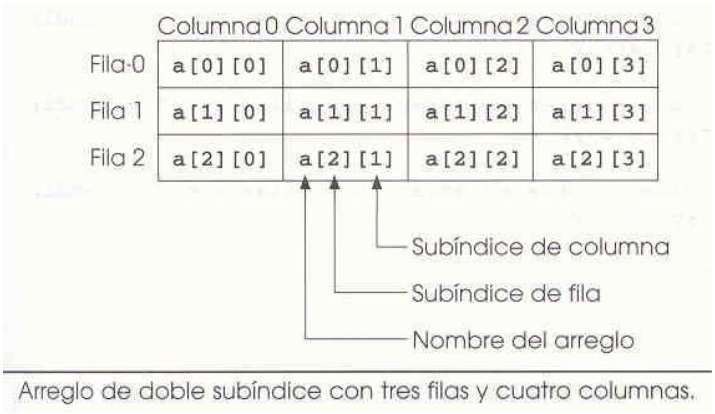
RAP 2:
Emplea los arreglos como herramienta para implementar cadenas de caracteres.

Arreglos de múltiples subíndices.

Un uso común de los arreglos de múltiples subíndices es la representación de tablas de valores que consisten en información ordenada en filas y columnas. Para identificar un elemento en particular de una tabla, debemos especificar dos subíndices: el primero identifica la fila del elemento y el segundo la columna.

Ejemplo:

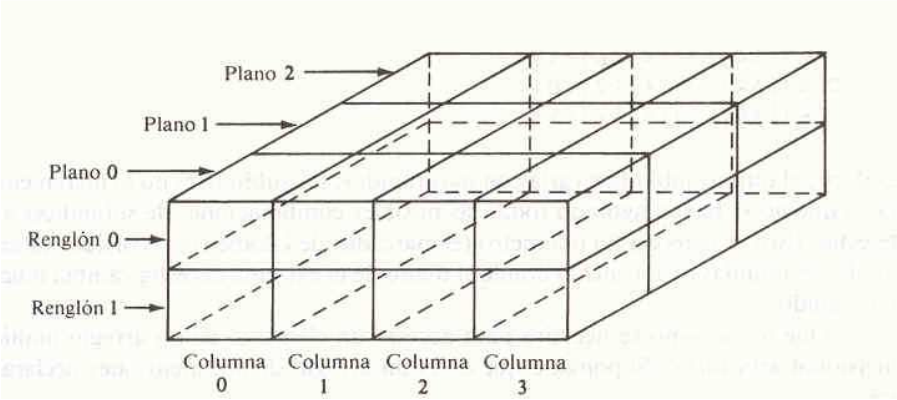
int a[3][4]; */* arreglo de tres filas por cuatro columnas */*



Arreglo de doble subíndice con tres filas y cuatro columnas.

Los arreglos de dos subíndices se conocen como arreglos bidimensionales. Los arreglos pueden tener mas de dos subíndices, llamándose multidimensionales.

Ejemplo : int múltiple [2][4][3]; */* Arreglo tridimensional */*



EJEMPLO 3

Este programa es un ejemplo de cómo inicializar un arreglo de dos subíndices e imprimirlo en pantalla.

```
/* Programa que inicializa e imprime un arreglo bidimensional*/

#include <stdio.h>
#include <conio.h>

void main(void)
{
 int i, j, arreglo[2][3] = {{1,2,3},{4,5,6}};  /* Inicializacion */
                                /* con valores predeterminados */
```

```
 clrscr();
 printf("Los valores del arreglo son: \n");
 for(i = 0; i < 2; i++)            /* imprime el arreglo bidimensional */
      {
       for(j = 0; j < 3; j++)
       {
        printf("%d  ",arreglo[i][j]);
       }
       printf("\n");
      }
 getch();}
```

Observa que ahora ocupas dos for() para poder acceder al arreglo bidimensional. Por lo tanto, de ahora en adelante, toma en cuenta que el numero de subíndices te va indicar el numero de for a utilizar para poder manejar arreglos multidimensionales.

EJEMPLO 4

El siguiente programa es una aplicación muy util de un arreglo bidimensional.

```
/* Programa que calcula la suma de dos matrices usando arreglos*/


#include <stdio.h>
#include <conio.h>


void main(void)
{
 int i, j, a[2][3], b[2][3], c[2][3];
```

```
clrscr();
for(i = 0; i < 2; i++)            /* datos para la matriz A */
     {
      for(j = 0; j < 3; j++)
      {
       printf("Elemento a[%d][%d] ",i,j);
       scanf("%d", &a[i][j]);
      }
     }
for(i = 0; i < 2; i++)            /* datos para la matriz B */
     {
      for(j = 0; j < 3; j++)
      {
       printf("Elemento b[%d][%d] ",i,j);
       scanf("%d", &b[i][j]);
      }
     }
for(i = 0; i < 2; i++)            /* Imprime la matriz C */
     {
      for(j = 0; j < 3; j++)
      {
       c[i][j] = a[i][j] + b[i][j];
       printf("\nElemento c[%d][%d] = %d", i, j, c[i][j]);
      }
     }
getch();}
```

Actividades

Competencia particular 3

1.- Desarrolla un programa que dada una secuencia de 10 datos enteros diferentes, los ordene en forma ascendente. Utiliza arreglos.

2.- Desarrolla un programa que acepte un password de 8 caracteres, si el password es el correcto que imprima "ACCESO AL SISTEMA", de no ser así, imprimirá "ACCESO DENEGADO" y utiliza la instrucción exit 1; *para salir del programa.*

3.- Desarrolla un programa que determine si una palabra es palíndroma. Un palíndromo es una palabra que se lee igual al derecho y al revés. Ejemplos: ojo, radar, anilina, etc.

4.- Desarrolla un programa que lleve el control de tus calificaciones de las materias de 5º. Semestre, colocando en las filas el nombre de la materia y 3 columnas para las calificaciones de los departamentales, y por ultimo una cuarta columna que despliegue tu calificación promedio por materia. Al final deberás desplegar tu promedio semestral.

5.- Desarrolla un programa que realice la multiplicación de una matriz de 2x3 por una matriz de 2x2.

6.- El metodo de ordenamiento por selección consiste en elegir, dentro de una serie de numeros, el numero menor de los elementos restantes, se escoge el siguiente numero menor y asi sucesivamente. Realice un programa que ordene ascendentemente una matriz de n x m elementos enteros positivos. Utilice dos matrices: una matriz para leer los elementos y otra para colocar los elementos ordenados. Una funcion de usuario leera los elementos de la matriz y otra funcion la ordenara.

7.- Dado un conjunto de datos propuesto, hacer un programa que realice lo siguiente:

- *Almacenar los datos en un arreglo unidimensional.*
- *Contar el numero de datos negativos y el numeros de datos positivos e indicar cuales son los mayores.*

8.- Hacer un programa para calcular el valor de la expresión:

$$p = \left(\sum_{i=1}^{n} x_i\right)/n$$

9.- Hacer un programa para calcular el valor de la expresión:

$$P = \prod_{i=1}^{m} \sum_{j=1}^{n} a_{ij} = (a_{11}+a_{12}+\ldots+a_{1n})(a_{21}+a_{22}+\ldots+a_{2n})(\ldots)$$

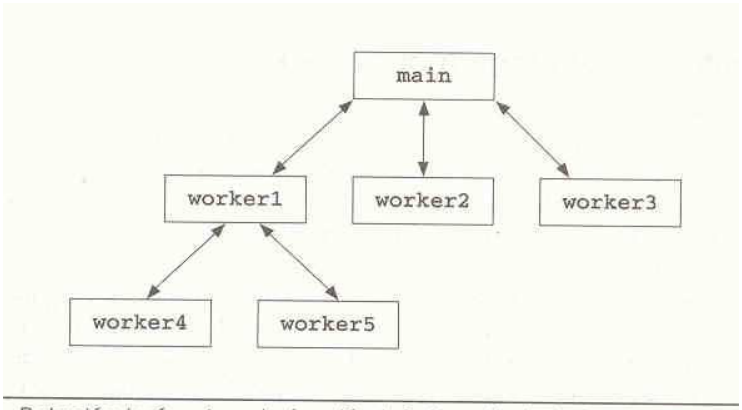
10.- Hacer un programa para calcular el valor de la expresión:

$$p = \sum_{I=1}^{n} e^{i}$$

COMPETENCIA PARTICULAR 4
Desarrolla programas utilizando funciones matemáticas, gráficas y de entrada – salida de datos mediante puerto, así como las diseñadas para uso específico de la problemática planteada.

FUNCIONES.

La mejor forma de desarrollar y mantener un programa grande es construirlo a partir de módulos o componentes mas pequeños, los cuales son mas fáciles de manejar que el programa en forma completa. Esta técnica se llama divide y venceras.



Relación jerárquica de función jefe/función trabajador.

Definición.

Los módulos o componentes en lenguaje C se llaman funciones. ***Las funciones son bloques de código que realizan una tarea especifica.*** *Un ejemplo, son las funciones printf() y scanf(), que llevan a cabo una tarea elemental, como son las de imprimir en el monitor y leer del teclado respectivamente y que conforman parte de una biblioteca o librería.*

Las ***bibliotecas o librerías****, se componen de funciones que permiten al programador efectuar ciertos cálculos o acciones comunes. El objetivo de estas funciones es el no rescribir código ya existente, sino solamente hacer uso de él en el momento que se necesita. Sin embargo, no todas las funciones realizan las tareas deseadas, así que el mismo programador, puede realizar sus propias funciones, para llevar a cabo la realización de un programa.*

RAP 1:
Utiliza las funciones predefinidas en el lenguaje.

INTRODUCCIÓN A LOS GRÁFICOS

Las computadoras están dotadas de un conjunto de caracteres s gráficos en modo de texto, a los que podemos tener acceso mediante el código ASCII, mediante los cuales podemos generar marcos para la representación de resultados e incluso agruparlos para generar algunas figuras en la pantalla, cuadros de desplegado de menús y otros usos muy básicos, a esta forma de graficación se le denomina graficación en línea o en modo de texto.

Otra forma de graficación que se emplea para colocar figuras en pantalla se realiza a través de lenguaje de programación, en donde por medio de funciones especiales se pueden hacer diversas figuras geométricas en pantalla mediante las cuales podemos realizar figuras mas complejas. Forma en donde se requiere el uso de la computadora en modo gráfico.

Los monitores tienen dos formas de operación una denominada modo de texto y otra modo gráfico. El modo de texto es el que los monitores tienen por omisión y es mediante el cual trabajamos para elaborar un programa y ejecutarlo y en donde la pantalla se podría medir en la cantidad de caracteres que caben en pantalla y lo menor que podría mostrar en pantalla seria un carácter. Para el caso de modo gráfico del monitor este debe de ser habilitado a través de la interfase gráfica y en este modo a la mínima información que podemos colocar en el monitor se le denomina píxel, y es este el que denomina la definición de la pantalla de forma directamente proporcional a mayores píxeles en pantalla será mayor la definición de dicho monitor, debido a que existen en el mercado muchos adaptadores y modos gráficos estándares Borland provee una interfase denominada " Borland Graphics Interfase" **BGI**, al cual contiene un juego de funciones gráficas escritas en código de bajo nivela través del cual se establece la conexión con los diversas tarjetas gráficas.

INICIALIZACIÓN DE LA INTERFASE BGI

El primer paso a seguir en la mayoría de los programas de graficación es cambiar el desplegado del monitor de modo de texto a modo gráfico y habilitar la interfase BGI para cargar la memoria del manejador adecuando su hardware al modo gráfico y una vez realizado esto, poder utilizar las funciones gráficas en pantalla.

Todos los archivos del modo gráfico deberán iniciar con la librería ***graphics.h*** *y posteriormente inicializar el modo gráfico con la función* ***initgraph()*** *yabuna vez hechas el programa se deberá incluir al final la función* ***closegraph()*** *para así regresar la computadora a modo de texto al terminar de ejecutarlo.A continuación se muestra la sintaxis para la función initgraph.*

initgraph ([int far]*gdriver,[int far]*gmode, [char for]* "Ruta de directorio";

Los dos primeros parámetros son apuntadores a valores que especifican el manejador y el modo gráfico utilizar, el tercer argumento es terminador nulo que especifica la trayectoria en donde se localizan los archivos manejadores de gráficos.

A continuación se muestran los manejadores y modos gráficos en BGI.

MANEJADORES GRÁFICOS EN BGI

CONSTANTE	*VALOR*
DETECT	*O (SOLICITUD DE AUTODETECCION*
CGA	*1*
MCGA	*2*
EGA	*3*
EGA64	*4*
EGAMONO	*5*
IBM8514	*6*
HERCMONO	*7*
ATT400	*8*
VGA	*9*

La función initgraph puede configurar automáticamente el adaptador de video correspondiente al monitor y se inicializa ***gdriver*** *con la constante numérica del sistema* ***DETECT siendo esta la mas usada*** *en virtud de que el uso de constantes como lo indica la tabla anterior requiere que el programador tenga un conocimiento claro del monitor con el cual trabajará, e incluso el programa solo seria valido para ciertos equipos especificados.*

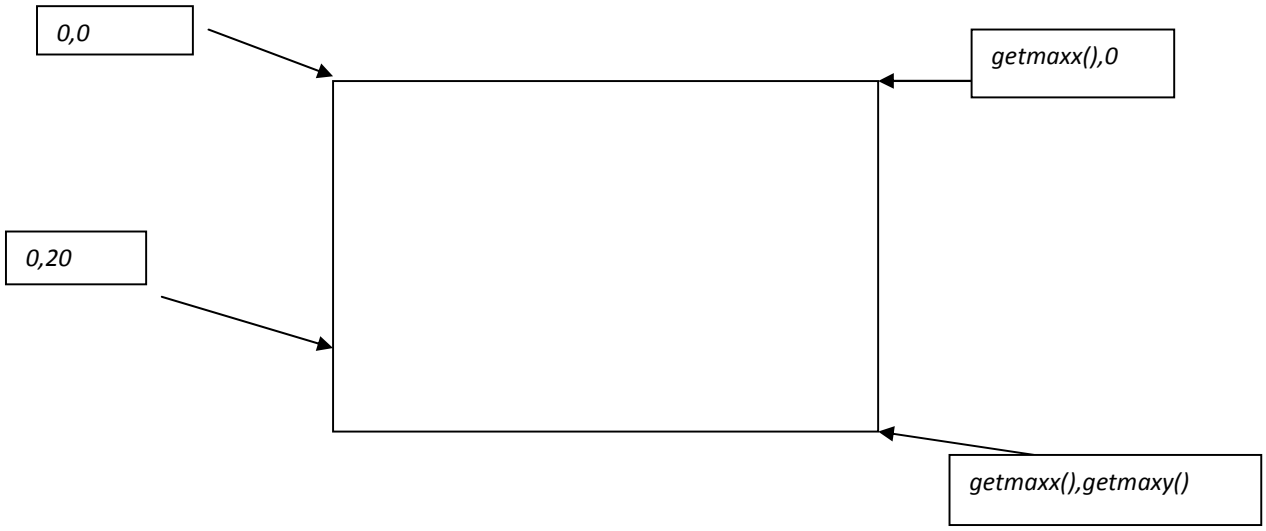
SINTAXIS GENERAL PARA INICIALIZAR EL MODO GRÁFICO .

```
# include <graphics.h>
#include <stdio.h>
main (void)
{
int gdriver = DETECT, gmode, erorcode;

initgraph( &gdriver,gmode, "C:\\borlandc\\bgi");

errorcode= graphresult();
if (error !=grOk)
   {
      printf("Error al inicializar modo gráfico del tipo %s \n", grapherromsg(errocode));
      printf("Oprime una tecla para continuar");
      getch();
      exit(1);
   }
closegraph();
}
```

El cuerpo del programa que se muestra corresponde a la estructura mínima para poner al monitor en modo gráfico y ahí mismo se muestra como se cierra para regresar al modo de texto, carece de instrucciones por lo que si se ejecutara solo se vería un ligero parpadeo en pantalla.

TRABAJO EN MODO GRÁFICO CON COORDENADAS DE PANTALLA

Como ya se dijo el trabajo en modo gráfico implica el uso delas unidades fundamentales de pantalla, llamados píxeles que bien los podríamos explicar como un punto en pantalla , donde cada uno de los cuales tendrá una coordenada dentro de la pantalla y como ya se comento la cantidad que tendrá cada monitor dependerá del tipo de cada uno de ellos, la localización de ellos será por medio de coordenadas en el eje x y y, los valores de x se contaran en la parte horizontal de la pantalla de izquierda a derecha donde la coordenada "0" será el extremo izquierdo de la misma y para el caso de la "y" el valor "0" será la parte superior , mientras mayor sea este valor estaremos hablando de una coordenada mayor. Es importante destacar que dichas coordenadas no corresponde a ningún cuadrante de el plano cartesiano, tal y como estamos acostumbrados a usarlos en matemáticas. Por lo que hay que poner especial cuidado.



Las valores getmaxx() es el valor máximo en el eje de las "x"

Y getmaxy() valor máximo en el eje de las "y"

FUNCIONES DE LA LIBRERIA ESTANDAR DE "C" graphics.h

Funciones de control del modo de video.

CONTROLADORES DE GRÁFICOS Y MACROS DE MODOS:

- DETECT,
- CGA,
- MCGA,
- EGA,
- EGA64,
- EGAMONO,
- IBM8514, /'*1 - 6 */
- HERCMONO,
- ATT400,
- VGA,
- PC3270, /*7 - 10 */
- CURRENT_DRIVER = -1

COLORES ESTÁNDAR DE "C" EN MODO GRÁFICO:

- BLACK
- BLUE
- GREEN
- CYAN
- RED
- MAGENTA
- BROWN
- LIGHTGRAY
- DARKGRAY
- LIGHTBLUE
- LIGHTGREEN
- LIGHTCYAN
- LIGHTRED
- LIGHTMAGENTA
- YELLOW
- WHITE

GRÁFICOS BÁSICOS

A continuación se dan algunas de las instrucciones básicas para construir figuras que son en su mayoría geométricas para así conformar programas básicos en ser referenciados por si nombre en mayúsculas o por su numero que le corresponda.

circle(int x, int y, int radius);

D*ibuja un circulo de radio radius con el centro en la posición especificada por x, y; usando el color de dibujo actual.*

line(int x1, int y1, int x2, int y2);

Traza una linea desde la posición especificada por x inicial, y final, con el color del dibujo, estilo y grosor de la línea actual.

putpixel(int x, int y, int color);

Pone la posición determinada por x e y del color especificado por color.

setcolor(int color);

Establece el color actual del dibujo.

Color	Numero
Negro (BLACK)	0
Azul (BLUE)	1
Verde (GREEN)	2

Cyan (CYAN)	*3*
Rojo (RED)	*4*
Magenta (MAGENTA)	*5*
Café (BROWN)	*6*
Gris Claro (LIGHTGREY)	*7*
Gris Oscuro (DARKGREY)	*8*
Azul Claro (LIGHTBLUE)	*9*
Verde Claro (LIGHTGREEN)	*10*
Cian Claro (LIGHTCYAN)	*11*
Rojo Claro (LIGHTRED)	*12*
Magenta Claro (LIGHTMAGENTA)	*13*
Amarillo (YELLOW)	*14*
Blanco (WHITE)	*15*

floodfill(int x, int y, int border);

Rellena cualquier figura cerrada, tiene como parámetro las coordenadas de un punto inferior de la figura y el color de las líneas que formaran la figura.

setfillstyle(int pattern, int color);

Rellena, con un determinado color determinados objetos. Los valores admisibles de pattem, son:

- *EMPTY FILL*
- *SOLID_FILL*
- *LINE_ FILL*
- *LTSLASH_FILL*
- *SLASH FILL*
- *BKSLASH_FILL*
- *LTBKSLASH_FILL*
- *HATCH_FILL*
- *XHATCH_FILL*
- *INTERLEAVE_FILL*
- *WIDE_DOT FILL*
- *CLOSE_DOT FILL*
- *USER_FILL*

setfillpattern(char far *upattern, int color);

Permite definir un patrón de relleno.

ESCRITURA DE TEXTO EN MODO GRÁFICO.

outtext(char far *textstring);

Escribe la cadena apuntada por textstring en la posición actual.

settextstyle(int font, int direction, int charsize);

Se utiliza para cambiar estilo, tamaño o dirección del texto. El tipo font puede tomar uno de los siguientes valores:

- *DEFAULT FONT = 0*
- *TRIPLEX FONT = 1*
- *SMALL_FONT = 2*
- *SANS SERIF FONT= 3*
- *GOTHIC_FONT = 4*

Los valores de dirección pueden ser:

- *HORIZ_DIR 0*
- *VERT_DIR 1*

(Los valores que puede tomar charsize son de 0-10.)

outtextxy(int x, int y, char far *textstring);

Permite escribir el texto en una posición determinada de la ventana gráfica. La cadena se escribe en las coordenadas de la ventana gráfica.

- *BKSLASH_FILL*
- *LTBKSLASH_FILL*
- *HATCH_FILL*
- *XHATCH_FILL*
- *INTERLEAVE FILL*
- *WIDE_DOT FILL*
- *CLOSE_DOT FILL*
- *USER FILL*

setfillpattem(char far *upattem, int color);

Permite definir un patrón de relleno.

ESTADO DE LA PANTALLA EN MODO GRÁFICO.

gettextsettings(struct textsettingstype far *texttypeinfo);

Se utiliza para obtener información sobre la pantalla de gráficos.

getviewsettings(struct viewporttype far *viewport);

Carga en una estructura información diversa sobre la ventana gráfica actual.

```
struct viewporttype {
        int left, top, right, bottom;
        int clip;
```

Los campos left, top, right, y bottom (izquierda, arriba, derecha y abajo) contienen las coordenadas superior izquierda e inferior derecha de la ventana gráfica.

```
struct textsettingstype {
        int font;
        int direction;
        int charsize;
        int horiz;
        int vert;
};
```

La variable font contiene uno de los siguientes valores:

- *DEFAULT FONT = 0,*
- *TRIPLEX FONT =1,*
- *SMALL_FONT = 2,*

- *SANS_SERIF_FONT = 3,*
- *GOTHIC FONT = 4*

El elemento dirección debe valer HORIZ_DIR(valor por defecto) para texto horizontal o VERT_DIR para vertical. El elemento charsize es un multiplicador que se usa para modificar el tamaño del texto. Los valores horiz y vert son la manera en que se justifica el texto, los cuales pueden ser los siguientes:

- *LEFT TEXT = 0*
- *CENTER_TEXT = 1*
- *RIGHT TEXT = 2*

- *BOTTOM_TEXT = 0*
- *CENTER_TEXT = 1*
- *TOP TEXT = 2*

FUNCIONES DE MANIPULACIÓN DE LA PANTALLA EN MODO GRÁFICO.

clearviewport(void);

Limpia la ventana gráfica activa.

getimage(int left, int top, int right, int bottom, void far *bitmap);

copia parte de una ventana gráfica en un buffer.

imagesize(int left, int top, int right, int bottom);

Devuelve el numero de byte que se necesitan para salvar una imagen.

putimage(int left, int top, void far *bitmap, int op);

Copia el contenido de un buffer a la ventana gráfica.

setactivepage(int page);

Determina que pagina se vera afectada por las rutinas gráficas.

Setviewport (int left, int top, int right, int bottom, int clip);

Crea una ventana gráfica.

setvisualpage(int page);

Determina la página que se muestra.

El valor de op determina la manera exacta en que se escribe la imagen en la pantalla, los valores validos son:

- *COPY PUT, /*MOV*/*
- *XOR_PUT, /*XOR*/*
- *LEFT_TEXT = 0,*
- *CENTER_TEXT =1,*
- *RIGHT TEXT = 2,*
- *BOTTOM TEXT = 0,*
- *CENTER TEXT =1,*
- *TOP TEXT =2.*
- *OR_PUT, /*OR*/*
- *AND_PUT, /*AND*/*
- *NOT_PUT /*NOT*/*

FUNCIONES DE GRÁFICOS Y DE LA PANTALLA DE TEXTO.

arc(int x, int y, int inicio, int fin, int radius);

Dibuja un arco desde inicio hasta fin (dado en grados hexadecimales) a lo largo de una circunferencia invisible centrada en x, y y de radio radius.

bar(int left, int top, int right, int bottom);

Dibuja una barra rectangular cuya esquina superior izquierda queda definida por lef, top y su esquina inferior derecha por right, bottom. La barra se rellena con el color y relleno actuales.

bar3d(int left, int top, int right, int bottom, int depth, int topflag);

es igual a la anterior, con la diferencia que produce una barra tridimensional de depth pixels de profundidad. Se traza el contorno de la barra con el color actual del dibujo.

cleardevice(void);

Limpia la pantalla y establece la posición actual 0,0.

cprintf(const char *str, ...);

Esta función se encuentra en conio.h, esta funci6n es como el printf(), excepto en que escribe en la ventana de texto actual en lugar de stdout (salida estandar).

cscanf(const char *str, ...);

Esta función se encuentra en conlo.h, esta función es como scanf(), excepto en que lee la informaci6n desde la consola en lugar de desde stdin,(entrada estandar).

closegraph(void);

Termina el use del estado gráfico.

delline(void);

Borra la línea de la ventana activa que contenga el cursor.

detectgraph(int far *graphdriver,int far *graphmode);

Determina el tipo de adaptador gráfico, si existe alguno, de la computadora

drawpoly(int numpoints, int far *poiypoints);

Dibuja un polígono utilizando el color actual del dibujo.

ellipse(int x, int y, int stangle, int endangle, int xradius, int yradius);

Dibuja una elipse con el color actual del dibujo. El Centro de la elipse esta en x, y. La longitud de los ejes x e y viene dada por xradius y yradius.

fillellipse(int x, int y, int xradius, int yradius);

Dibuja y rellena una elipse usando el color y relleno actual.

fillpoly(int numpoints, int far *polypoints);

Primero dibuja el objeto, en el color actual de dibujo consistente en numpoints puntos definidos por las coordenas x,y en el arreglo al que apunta polypints.

floodfill(int x, int y, borde);

Rellena un objeto con el color y relleno actuales, dadas las coordenadas de cualquier punto interior al objeto, y el color de su borde.

getarccoords(struct arccoordstype far *arccoords);

Da valores a los campos de la estructura a la que apunta arccoords con las coordenadas relativas a la ultima llamada a arc(). La estructura arccoordtype se define como:

```
struct arccoordstype {
int x, y;
int xstart, ystart, xend, yend;
 };
```

getaspectratio(int far *xasp, int far *yasp);

Copia la razon de aspecto de la x en la variable a la que apunta xasp y la razbn de aspecto de la y en la apuntada por yasp.

getbkcolor(void);

Devuelve el color de fondo actual. Con cualquiera de los colores estándar de `C".

getcolor(void);

Devuelve el color actual del dibujo. Con cualquiera de los colores estándar de "C".

struct palettetype *far getdefaultpalette(void);

Devuelve la paleta definida por defecto por el controlador de gráficos con el que se llamo intgraph(). La estructura palettetype esta definida como:

struct palettetype {

unsigned char size;

signed char colors[MAXCOLORS+1];

};

char *far getdrivername(void);

Devuelve el nombre del controlador de gráficos actual.

getfillpattem(char far *pattern);

Rellena el arreglo al que apunta pattem. con los ocho bytes que forma el patrón de relleno actual

getfillsettings(struct fillsettingstype far *fillinfo);

Da valores a los campos de la estructura a la que apunta fillinfo con el numero de patrón de relleno y el color actual. La estructura flllsettingtype se define como:

struct fillsettingstype {

int pattern;

int color;

};

getgraphmode(void);

Devuelve el modo de registro actual. El valor que devuelve no corresponde al que asocia el BIOS con ese modo de video usando un controlador de video actual, el valor del controlador lo podemos encontrar en la definición de controladores y macros de gráficos para "C".

getiinesettings(struct linesettingstype far *lineinfo);

Da valores a los campos de la estructura a la que apunta lineinfo con el tipo de linea actual. La estructura linesettingstype se define como:

struct linesettingstype {

int linestyle;

unsigned upattem;

int thickness;

};

getmaxcolor(void);

Devuelve el mayor valor valido para el color en el modo de video actual.

getmodename(int mode_number);

Devuelve el nombre del modo especificado. El valor del modo se obtiene en la llamada a initgraph().

getpalette(struct palettetype far *palette);

Carga la estructura a la que apunta palette con el numero de la paleta actual. La estructura palettetype se define como:

struct palettetype {

unsigned char size;

signed char colors[MAXCOLORS+1]; };

};

getpalettesize(void);

Devuelve el número de colores de la paleta actual.

gettext(in izq, int arr, int der, int aba, volt * buf);

Copia el texto de un rectángulo con esquina superior izquierda e inferior derecha en el buffer al que apunta buf.

graphdefaults(void);

Restaura el sistema gráfico a sus parámetros por defecto.

graphresult(void);

Devuelve el valor que representa el resultado de la ultima operación gráfica.

initgraph(int far *graphdriver, int far *graphmode, char far *pathtodriver);

Se usa para inicializar el sistema de gráficos y carga el controlador de gráficos apropiado.

insline(void);

Inserta a una línea en blanco en la posición actual del cursor.

installuserdriver(char far *name, int huge (*detect)(void));

Permite instalar controladores BGI creados por otros.

pieslice(int x, int y, int stangle, int endangle, int radius);

Dibuja un sector de circulo (trozo de una tarta) que cubre un ángulo igual a endangle, stangle, usando el color actual del dibujo.

rectangle(int left, int top, int right, int bottom);

Dibuja, en el color actual de dibujo un rectángulo definido por las coordenadas left, top y right, bottom.

restorecrtmode(void);

Restaura la pantalla al modo en que estaba antes de la llamada de initgraph().

sector(int X, int Y, int StAngle, int EndAngle, int XRadius, int YRadius);

Dibuja un sector de elipse con el color actual de dibujo y lo rellena usando el color y el patrón de relleno actuales.

setallpalette(struct palettetype far *palette);

Cambia todos los modos de los colores de la paleta de EGAIVGA.

La estructura patttetype se define como:

```
struct palettetype {
        unsigned char size;
        signed char colors[MAXCOLORS+1];
};
```

setaspectratio(int xasp, int yasp);

Da a la razón de aspecto x el valor al que apunta xasp y a la razón de aspecto y el valor al que apunta yasp.

setbkcolor(int color);

Cambia el color del fondo al color que se especifica en color.

setgraphmode(int mode);

Establece el modo gráfico actual al especificado por mode, el cual debe ser un modo valido para el controlador de gráficos.

setiinestyle(int linestyle, unsigned upattern, int thickness);

Determina el aspecto de las líneas que dibuja cualquier función que trace limas. El parámetro linestyle indica el estilo de línea el cual puede ser:

- *SOLID_LINE = 0*
- *DOTTED_LINE = 1*
- *CENTER_LINE = 2*
- *DASHED_LINE = 3*
- *USERBIT_LINE = 4*

El parámetro thickness tendrá uno de los siguientes valores:

- *NORM_WIDTH =1*
- *THICK_WIDTH = 3*

setpalette(int colomum, int color);

Cambia los colores que muestra el sistema de video.

setrgbpalette(int colornum,int red, int green, int blue);

Cambia los colores que muestra el sistema de video. Solo se puede usar en aquellos sistemas de gráficos que soporten pantallas RGB (rojo-verde-azul), como el IBM 8514 y la VGA.

settextjustify(int horiz, int vert);

Establece la manera en que se alinea el texto con la CP. Los valores de horiz y vert determinan el efecto de esta función tal como se muestra a continuación.

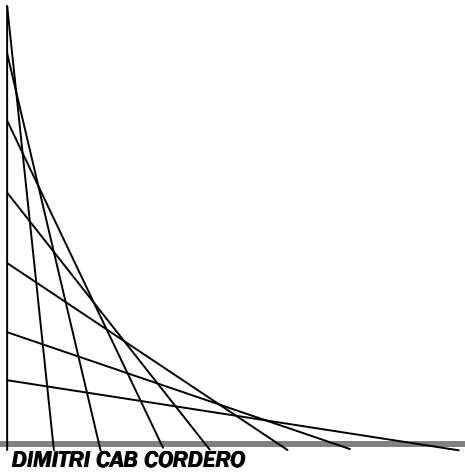
- *LEFT TEXT = 0, CP a la izquierda*
- *CENTER TEXT =1, CP en el centro*
- *RIGHT TEXT = 2, CP a la derecha*
- *BOTTOM TEXT = 3, CP abajo*
- *TOP TEXT = 4. CP arriba*

A continuación se dan algunos ejemplos de programas en modo gráfico

1.- programa que dibuja un circulo centrado en pantalla:

```
#include<conio.h>
#include<stdio.h>
#include<math.h>
#include<graphics.h>
#include<dos.h>
main()
{
clrscr();
int a=DETECT,b;
initgraph(&a,&b,"C:\\BORLANDC\\BGI");
cleardevice();
setcolor(BLUE);
setbkcolor(WHITE);
circle( getmaxx()/2,getmaxy()/2,50);
getch();
closegraph().
}
return 0;
```

2.- El siguiente programa da una salida similar a esta figura pero con mas líneas y mas juntas.



```
#include<conio.h>
#include<stdio.h>
#include<math.h>
#include<graphics.h>
#include<dos.h>
main()
{
clrscr();
int x0,x1,y0,y1,x2,y2;
int a=DETECT,b;
initgraph(&a,&b,"C:\\BORLANDC\\BGI");
cleardevice();
x0=0,y0=0;
x1=0,y1=getmaxy();
x2=getmaxx(),y2=getmaxy()/2;
x3=getmaxx(),y3=getmaxy();
setbkcolor(14);
line(x0.y0,x1,y1);
line(x1,y1,x2,y2);
for(y0=0;y0<=y2;yo+=5)
{
setcolor(4);
line(x0,y0,x1,y1);
delay(30);
x1+=5
}
getch();
closegraph();
return(0);
}
```

3.- Ejecute el siguiente programa y encuentre su función.

```
#include<conio.h>
#include<stdio.h>
#include<math.h>
#include<graphics.h>
#include<dos.h>
main()
{
clrscr();
int x1,y1,x2,y2,x3,y3,w,i;
int a=DETECT,b;
initgraph(&a,&b,"C:\\BORLANDC\\BGI");
x1=0,y1=0;
```

```
x2=getmaxx()/2,y2=getmaxy()/2;
x3=getmaxx(),y3=getmaxy();
setbkcolor(14);
for(w=15;w>=1;w-=3)
{
setcolor(4);
line(x1,y2,x3,y2);
delay(30);
setcolor(4);
line(x2,y1,x2,y3);
delay(25);
}
for(i=x1;i<=235;i=i+=3)
{
setcolor(BLUE);
line(x2,y1+i,x2+i,y2);
delay(30);
setcolor(YELLOW);
line(x2,y1+i,x2-i,y2);
delay(30);
setcolor(GREEN);
line(x2,y3-i,x2+i,y2);
delay(30);
setcolor(WHITE);
line(x2,y3-i,x2-i,y2);
delay(25);
}
getch();
closegraph()
return(0);
}
```

A continuación se muestran diversos ejemplos en los que se utilizan diversas instrucciones en modo grafico

1.- Diversas figuras con relleno por medio de llamado de funciones

```
#include <stdlib.h>
#include <conio.h>
#include <stdio.h>
#include <graphics.h>
#include <dos.h>
#include "a:ayuda.h"
#define T 1000
#define A 20
#define B 20
#define C 20

struct REGPACK reg;

struct circulo{
int x1,y1,r;
int relleno;
}gcir[A][B];

struct linea{
int x1,y1,y2,x2;
int relleno;
}glin[A][B];

struct rectang{
int x1,y1,x2,y2;
int relleno;
}rectangu[A][B];

struct ellipse{
int x1,y1,x2,y2;
int relleno;
}gellipse[A][B];

struct texto{
```

```
int x1,y1;
char texto;
int relleno;
}gtex[A][B];


void formato(void);
void presenta(void);
void rata(void);




main ()

{
// declaracion de variables//
int gdriver=DETECT,gmode,color_relleno;
int x1,x2,y1,y2,r, relleno=0, proc_term=0, temp=0;
int i,f,a,b,c,d,e;
char opcion, *color,texto[C],salva[C];

d=0;
do
{
if(gellipse[i][d].x1!=0){
gellipse[i][d].x1=0;
gellipse[i][d].y1=0;
gellipse[i][d].x2=0;
gellipse[i][d].y2=0;
}
d=d++;
}while(gellipse[i][d].x1!=0);

c=0;
do
{
if(rectangu[i][c].x1!=0){
```

```
rectangu[i][c].x1=0;
rectangu[i][c].y1=0;
rectangu[i][c].x2=0;
rectangu[i][c].y2=0;

}
c=c++;
}while(rectangu[i][c].x1!=0);

b=0;
do
{
if(glin[i][b].x1!=0){
glin[i][b].x1=0;
glin[i][b].y1=0;
glin[i][b].x2=0;
glin[i][b].y2=0;

}
b=b++;
}while(glin[i][b].x1!=0);

f=0;
do{
if(gcir[i][f].x1!=0){
gcir[i][f].x1=0;
gcir[i][f].y1=0;
gcir[i][f].r=0;
}
f=f++;
}while(gcir[i][f].x1!=0);




presenta();
initgraph(&gdriver,&gmode,"");
formato();
```

```
rata();
color_relleno=15;

b=0; c=0;d=0;
e=0; f=0;


do
{
// prepara pantalla//

reg.r_ax=1;
proc_term=0;

// toma la opcion deseada //

opcion=getch();
setfillstyle (1,15);
bar (20,470,640,480);
setfillstyle (1,color_relleno);
setcolor(color_relleno);
do
{
// espera que se presione un boton del raton//
reg.r_ax=3;
intr(0x33, &reg);


/*************************Proceso de buscar*********************************/


if (opcion=='b' || opcion=='B'){



f=0;
do{
if(gcir[i][f].x1!=0){
x1=gcir[i][f].x1;
y1=gcir[i][f].y1;
```

```
r=gcir[i][f].r;
color_relleno=gcir[i][f].relleno;
if (relleno==1) pieslice (x1,y1,0,360,r);
else circle (x1,y1,r);
f=f++;
}
}while(gcir[i][f].x1!=0);


b=0;
do
{
if(glin[i][b].x1!=0){
x1=glin[i][b].x1;
y1=glin[i][b].y1;
x2=glin[i][b].x2;
y2=glin[i][b].y2;
color_relleno=glin[i][b].relleno;
line (x1,y1,x2,y2);
b=b++;
}

}while(glin[i][b].x1!=0);

c=0;
do
{
if(rectangu[i][c].x1!=0){
x1=rectangu[i][c].x1;
y1=rectangu[i][c].y1;
x2=rectangu[i][c].x2;
y2=rectangu[i][c].y2;
color_relleno=rectangu[i][c].relleno;
if (relleno==0)rectangle (x1,y1,x2,y2);
else bar(x1,y1,x2,y2);
c=c++;
}

}while(rectangu[i][c].x1!=0);
```

```
d=0;
do
{
if(gellipse[i][d].x1!=0){
x1=gellipse[i][d].x1;
y1=gellipse[i][d].y1;
x2=gellipse[i][d].x2;
y2=gellipse[i][d].y2;
color_relleno=gellipse[i][d].relleno;
if (relleno==0) ellipse (x1,y1,0,360,abs(x1-x2),abs(y1-y2));
else fillellipse (x1,y1,abs(x1-x2),abs(y1-y2));
d=d++;
}
}while(gellipse[i][d].x1!=0);


e=0;
do
{
if(gtex[i][e].x1!=0){
x1=gtex[i][e].x1;
y1=gtex[i][e].y1;
settextjustify (LEFT_TEXT,CENTER_TEXT);
settextstyle(SANS_SERIF_FONT,HORIZ_DIR,2);
texto[temp]=gtex[i][e].texto;
color_relleno=gtex[i][e].relleno;
outtextxy (x1,y1,texto);
e=e++;
}
}while(gellipse[i][e].x1!=0);

proc_term=1;
reg.r_bx=0;
}

/*****************************************************************************/


/*****************************Proceso de salida*******************************/
if (opcion=='q' || opcion=='Q'){
```

```
e=0;

d=0;
do
{
if(gellipse[i][d].x1!=0){
gellipse[i][d].x1=0;
gellipse[i][d].y1=0;
gellipse[i][d].x2=0;
gellipse[i][d].y2=0;
}
d=d++;
}while(gellipse[i][d].x1!=0);

c=0;
do
{
if(rectangu[i][c].x1!=0){
rectangu[i][c].x1=0;
rectangu[i][c].y1=0;
rectangu[i][c].x2=0;
rectangu[i][c].y2=0;

}
c=c++;
}while(rectangu[i][c].x1!=0);

b=0;
do
{
if(glin[i][b].x1!=0){
glin[i][b].x1=0;
glin[i][b].y1=0;
glin[i][b].x2=0;
glin[i][b].y2=0;

}
b=b++;
}while(glin[i][b].x1!=0);
```

```
f=0;
do{
if(gcir[i][f].x1!=0){
gcir[i][f].x1=0;
gcir[i][f].y1=0;
gcir[i][f].r=0;
}
f=f++;
}while(gcir[i][f].x1!=0);
closegraph();
exit(1); }

/*************************************************************************/

/*************************Proceso de nuevo*********************************/
if(opcion=='n' || opcion=='N')
{
formato();
proc_term=1;
reg.r_bx=0;
}

/*************************************************************************/

/*************************Proceso de relleno*******************************/
if (opcion=='F' || opcion =='f')
{
//tiene relleno ya?//
if (relleno==0)
{
temp=1;
setcolor(0);
setfillstyle (SOLID_FILL,15);
outtextxy (20,470,"Relleno y No. de Color : ");
*color=getch();
color_relleno=atoi(color);
bar(20,470,640,480);
setfillstyle (SOLID_FILL,color_relleno);
setcolor (color_relleno);
}
```

```
if (relleno==1)
{
temp=0;
setcolor(0);
outtextxy (20,470,"Relleno desconectado");
delay(1000);
}
proc_term=1;
relleno=temp;
break;
}
/*************************************************************************/

/*****************************Proceso de circulo*************************/
if (opcion=='c' || opcion=='C')
{

// especifica el centro//
if (reg.r_bx==1)
{
x1=reg.r_cx;
y1=reg.r_dx;
reg.r_bx=0;
gcir[i][f].x1=x1;
gcir[i][f].y1=y1;
}

//especifica el radio//
if (reg.r_bx==2)
{

setcolor (color_relleno);
r=abs(reg.r_cx-x1)-7;
gcir[i][f].r=r;
gcir[i][f].relleno=color_relleno;
f++;
if (relleno==1) pieslice (x1,y1,0,360,r);
else circle (x1,y1,r);
proc_term=1;
reg.r_bx=0;
```

```
}

}

/*****************************************************************************
/*****************************Proceso de linea*****************************/
if (opcion=='l' || opcion=='L')
{

// establece 1er punto//
if (reg.r_bx==1)
{
x1=reg.r_cx;
y1=reg.r_dx;
reg.r_bx=0;
glin[i][b].x1=x1;
glin[i][b].y1=y1;
}

// establece 2o punto//
if (reg.r_bx==2)
{

setcolor(color_relleno);
x2=reg.r_cx;
y2=reg.r_dx;
line (x1,y1,x2,y2);
glin[i][b].x2=x2;
glin[i][b].y2=y2;
gcir[i][b].relleno=color_relleno;
b=b++;
proc_term=1;
reg.r_bx=0;
}

}

/*****************************************************************************/

/****************************Proceso de rectangulo************************/
```

```
if (opcion=='r' || opcion=='R')
{

// establece 1er punto//
if (reg.r_bx==1)
{
x1=reg.r_cx;
y1=reg.r_dx;
reg.r_bx=0;
rectangu[i][c].x1=x1;
rectangu[i][c].y1=y1;
}

// establece 2o punto//
if (reg.r_bx==2)
{

x2=reg.r_cx;
y2=reg.r_dx;
rectangu[i][c].x2=x2;
rectangu[i][c].y2=y2;
gcir[i][c].relleno=color_relleno;
c++;
if (relleno==0)rectangle (x1,y1,x2,y2);
else bar(x1,y1,x2,y2);
proc_term=1;
reg.r_bx=0;
}

}
/*************************************************************************/


/*****************************Proceso de elipse*****************************/
if (opcion=='e' || opcion=='E')
{

// establece el centro//
if (reg.r_bx==1)
{
```

```
x1=reg.r_cx;
y1=reg.r_dx;
reg.r_bx=0;
gellipse[i][d].x1=x1;
gellipse[i][d].y1=y1;
}
// establece el radio x- y//
if (reg.r_bx==2)
{

setcolor (color_relleno);
x2=reg.r_cx;
y2=reg.r_dx;
gellipse[i][d].x2=x2;
gellipse[i][d].y2=y2;
gcir[i][d].relleno=color_relleno;
d++;
if (relleno==0) ellipse (x1,y1,0,360,abs(x1-x2),abs(y1-y2));
else fillellipse (x1,y1,abs(x1-x2),abs(y1-y2));
proc_term=1;
reg.r_bx=0;
}


/**********Este procedimiento es la estructura donde se trabajara*************/

void formato(void);
{
setfillstyle(1,8);
bar(0,0,640,480);
setfillstyle(1,15);
bar(0,450,640,480);
setcolor (0);
settextstyle (SMALL_FONT,HORIZ_DIR,4);
outtextxy (10,460,"C=CIRCLE L=LINE R=RECTANGLE E=ELIPSE F=COLOR Q=QUIT");

}

/***************Este procedimiento es la portada del programa*****************/
```

```
void presenta(void);
{
clrscr();
graficos();
cleardevice();
vent1(5,5,630,473,3);
put_text(60,80,"CECYT 1  GONZALO  VAZQUEZ VELA ",2.8,BLACK);
put_text(70,200,"ACADEMIA DE ISTEMAS DIGITALES ",2.5,BLACK);

delay (T);
put_text(100,300," México",2,RED);
delay (T);

vent1(5,5,639,473,3);
clrscr();
graficos();
cleardevice();
vent1(5,5,630,473,3);
put_text(60,80,"INSTRUCCIONES",1,BLACK);
put_text(60,100,"CUANDO QUIERAS UTILIZAR UNA DE LA OPCIONES DE LA BARRA",1,BLACK);
put_text(60,120,"PRESIONA EL BOTON DE LA PRIMERA LETRA, DESPUES CON EL",1,BLACK);
put_text(60,140,"BOTON IZQUIEDO DEL RATON ESPECIFICARA EL CENTRO DE LA FIGURA",1,BLACK);
put_text(60,160,"Y CON EL DERECHO EL EJE DE LAS Y o X, Y ASI DIBUJARA ",1,BLACK);
put_text(60,180,"<Presione cualquier tecla para continuar>",1, BLACK);
getch();
}

/*************************************************************************/



/*******************Procedimiento que inicializa raton**********************/
void rata(void);
{
reg.r_ax=1;
intr(0x33, &reg);
} }
/*************************************************************************/
}
```

}}}}

Este es otro EJEMPLO:

```
#include <dos.h>
/*#include <math.h> */
#include <conio.h>
#include <stdio.h>
#include <stdlib.h>
/*#include <stdarg.h>*/

#include <graphics.h>


int GraphDriver;
int GraphMode;
double AspectRatio;
int MaxX, MaxY;
int MaxColors;
int ErrorCode;

static int r = 8;
struct viewporttype vp;
int pausa, x=600, y=550, ulx, uly, lrx, lry, size, i, ancho, alto, cambio;

void Inicializar(void)
{

GraphDriver = DETECT;
initgraph( &GraphDriver, &GraphMode, "c:\\tc\\BGI" );
ErrorCode = graphresult();
if( ErrorCode != grOk ){
printf(" Graphics System Error: %s\n", grapherrormsg( ErrorCode ) );
exit( 1 );
}
}

void imagen (void *rect)
```

```
{
/* putimage(x, y, rect, XOR_PUT);
delay(pausa);
putimage(x, y, rect, XOR_PUT);*/

cambio = random( r );
if ((cambio/2) % 2 != 0 )
cambio = -1 * cambio;
x = x + cambio;
cambio = random( r );
if ((cambio/2) % 2 != 0 )
cambio = -1 * cambio;
y = y + cambio;

if (vp.left + x + ancho - 1 > vp.right)
x = vp.right-vp.left-ancho + 1;
else
if (x < 0)
x = 0;
if (vp.top + y + alto - 1 > vp.bottom)
y = vp.bottom-vp.top-alto + 1;
else
if (y < 0)
y = 0;
}

void Figura1(void)
{
static int Xin = 100;
static int Yin = 50;


void *rect1,*rect2;

getviewsettings( &vp );

bar(Xin, Yin, 20, 10);

ulx = Xin-(20*4);
uly = Yin-40;
```

```
lrx = Xin+(20);
lry = Yin+(20);
ancho = lrx - ulx + 1;
alto = lry - uly + 1;
size = imagesize(ulx, uly, lrx, lry);

rect1 = malloc( size );
rect2 =malloc( size );
getimage(ulx, uly, lrx, lry, rect1);
putimage(ulx, uly, rect1, XOR_PUT);

pausa = 100;

while ( !kbhit () ) {
putimage(x, y, rect1, XOR_PUT);
delay(pausa);
putimage(x, y, rect1, XOR_PUT);
imagen( rect2 );
putimage(x, y, rect2, XOR_PUT);
delay(pausa);
putimage(x, y, rect2, XOR_PUT);
imagen( rect1 );
}
free( rect1 );
free( rect2 );
}

main ()
{
Inicializar();
Figura1();
}
```

Conexión y programación con el puerto paralelo

Las comunicaciones en paralelo se realizan mediante la transferencia simultánea de todos los bits que constituyen el dato (byte o palabra). Presentan la ventaja de que la transmisión puede ser más rápida. Sin embargo, las comunicaciones en paralelo no pueden ser implementadas para grandes distancias debido a que no es viable la conexión física de todas las líneas necesarias.
Las comunicaciones en paralelo propiamente dichas no han sido normalizadas, lo que sí se reconoce es la norma Centronic:, *para la conexión del PC a la impresora, mediante el envío simultáneo de 8 bits de datos (un byte), además de un conjunto de líneas de protocolo* (handshake o intercambio). *La operación más frecuente en la que interviene el puerto paralelo del PC es en el envío de datos a la impresora.*
Los antiguos circuitos integrados que se incluían en las tarjetas de interface del puerto paralelo no permitían la recepción de datos, sólo estaban diseñados para el envío de información al exterior. Las versiones recientes de estas tarjetas de interface de puertos paralelo sí ***permiten la recepción de datos*** *y dan la posibilidad, por ejemplo, de* ***intercambiar información entre PC a través del puerto paralelo****, siempre que se utilice el software adecuado.*
La ***norma Centronics*** *hace referencia a las características de la conexión entre un interface de puerto paralelo y una impresora. Las líneas son* latcheadas, *esto es, mantienen siempre el último valor establecido en ellas mientras no se cambien expresamente y los niveles de tensión y de corriente coinciden con los niveles de la lógica TTL, cuyos valores típicos son:*

-Tensión de nivel alto: 5 V.
-Tensión de nivel bajo: 0 v.
-Intensidad de salida máxima: 2.6 mA.
-Intensidad de entrada máxima: 24 mA.

La norma Centronics establece el nombre y las características de 36 líneas eléctricas para la conexión entre el PC y la impresora.

En realidad, para la transferencia de las señales de datos y de control a través de la tarjeta de interface paralelo ***sólo*** *se requieren* ***18 líneas****, las restantes son líneas de masa que se enrollan alrededor de los cables de señal para proporcionarles* ***apantallamiento*** *y protección contra interferencias. Por esto, las citadas tarjetas suelen incorporar un conector hembra DB-25, mientras que prácticamente todas las impresoras incorporan un conector hembra tipo Centronics macho de 36 pines.*

Los cables comerciales para la conexión paralelo entre el PC y la impresora tienen una longitud de 2 metros, aunque no es recomendable que tengan una longitud superior a 5 metros si se desea una conexión fiable y sin interferencias.

En la siguiente tabla se describen todas las líneas del estándar Centronics, con indicación de su denominación y el número de pin que le corresponde, tanto en el conector tipo Centronics de 36 pines como en el conector

DB25. En esa tabla se indica que las 8 líneas correspondientes a los bits de datos (D0 a D7) son líneas de sal ida, pues así lo establece el estándar Centronics, sin embargo y sobre todo en las implementaciones más recientes, la circuitería asociada al interface del puerto paralelo puede ser tal que las líneas de datos pueden ser leídas desde el PC y, por tanto, ser consideradas como líneas ***bidireccionales****, aunque sea en determinadas condiciones y con el software adecuado.*
Descripción de los pines del Puerto paralelo:

N.º de pin		Señal	Sentido	Descripción
DB-25	DB-36		PC-PRN	
1	1	/STR	→	STROBE. Validación del dato (activa a nivel bajo). Un nivel L indica a la impresora que el dato es válido.
2	2	D0	→	Bit 0 de datos.
3	3	D1	→	Bit 1 de datos.
4	4	D2	→	Bit 2 de datos.
5	5	D3	→	Bit 3 de datos.
6	6	D4	→	Bit 4 de datos.
7	7	D5	→	Bit 5 de datos.
8	8	D6	→	Bit 6 de datos.
9	9	D7	→	Bit 7 de datos.
10	10	/ACK	←	ACKNOWLEDGE. (Activa a nivel bajo). Un nivel L indica que la impresora está en disposición de recibir un nuevo dato.
11	11	BSY	←	BUSY. Ocupada. Un nivel H indica que la impresora está ocupada y no puede recibir datos.
12	12	PAP	←	PAPER END. Sin papel. Un nivel H indica que la impresora se ha quedado sin papel.
13	13	OFON	←	ON LINE. Conectada. Un nivel H indica que la impresora está conectada y en línea.
14	14	/ALF	→	AUTO LINE FEED. Cambio de línea automático (Activa a nivel bajo). Un nivel L indica a la impresora que cuando reciba un retorno de carro debe hacer también un cambio de línea automáticamente.
15	32	/ERR	←	ERROR. (Activa a nivel bajo). Un nivel L indica que se ha producido un error en la impresora (buffer lleno, impresora fuera de línea, etc.).
16	31	/INI	→	INITIALIZE PRINTER. Inicialización (Activa a nivel bajo). Un nivel L inicializa o provoca un reset en la impresora (si la impresora lo admite).
17	36	/DSL	→	SELECT. (Activa a nivel bajo). Un nivel L selecciona o pone on line la impresora (si la impresora lo admite).
18-25	19-30 33	Masa		Referencia de tensión para las señales.
—	16	0V		—
—	17	Chasis		Conexión al chasis del equipo.
—	18	+Vcc		Tensión de +5 V.
—	34,35	—		No utilizada.

210

El puerto paralelo en un PC

Todos los ordenadores tipo PC están equipados, al menos, con una tarjeta de interface paralelo, frecuentemente junto a un interface serie. Como sistema operativo, el DOS puede gestionar hasta cuatro interfaces de puertos paralelo, LPT1 , LPT2 , LPT 3 y LPT4, además, reserva las siglas PRN como sinónimo del LPT 1, de modo que puede ser tratado como un archivo genérico. En el byte ***0040:0011*** *del BIOS almacena el número de interfaces de puertos paralelo que se hayan instalado en el equipo. La dirección de entrada/salida de cada uno de los puertos paralelo y el número de puertos instalados en un PC se muestra en la pantalla inicial de arranque del equipo es frecuente, casi estándar que las direcciones de los dos primeros puertos paralelo sean las siguientes:*

LPT1 = 0x378 Hexadecimal
LPT2 = 0x278 Hexadecimal

Las tarjetas del puerto paralelo tiene una estructura muy simple; consta de tres registros*:* ***de control,*** *de* ***estado*** *y de* ***datos****. Todas las señales que intervienen en el puerto tienen asociado un bit en uno de esos registros, de acuerdo con las funciones asignadas a cada línea en particular.*

El registro de datos

Es de tipo latch de 8 bits, que puede ser leído y escrito desde el procesador. Es el registro donde el procesador, en operaciones de salida (OUT), pone el dato que se quiere enviar a la impresora y su ***dirección*** *coincide con la* ***dirección base del puerto paralelo*** *(* ***0x 378*** *en LPT 1).*

En la Figura 32.1 se muestra la distribución de los bits de este registro y los pines asociados a cada uno de ellos en el conector DB-25.

REGISTRO DE DATOS (D)							
7	6	5	4	3	2	1	0
D7	**D6**	**D5**	**D4**	**D3**	**D2**	**D1**	**D0**
Pines⟶	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**

Figura 32.1. El registro de datos del puerto paralelo en el PC y los pines correspondientes en el conector DB-25.

El registro de estado

El registro de estado indica la situación actual de la impresora conectada al puerto, de acuerdo con los niveles de tensión que tengan las líneas
ACK, BSY, PAP y OF/ON , lo que permite controlar el comportamiento de la impresora.
Se trata de un ***registro de entrada*** *(Lectura) de información, su dirección se obtiene sumando 1 a la dirección base del puerto (****0x379*** *en LPT1).*

Error
es 11,10,12,12,15

REGISTRO DE ESTADO (S)							
7	6	5	4	3	2	1	0
/BSY	/ACK	PAP	OFON	ERR	X	X	X
(*)							
Pines → 9	8	7	6	5	4	3	2

Figura 32.2. El registro de estado del puerto paralelo en el PC y los pines correspondientes en el conector DB-25.

*Los bits de este registro se designan según se muestran en la Figura 32.2, en la que el símbolo «****/****» delante del nombre del bit indica que es* activo a nivel bajo.
Pero el bit 7 además (/ BSY) del registro de estado (bit 7) es invertido por el hardware y, por tanto, la línea tiene un nivel complementado al que aparece en ese bit.
El significado que tienen los bits de este registro es el siguiente:

Si el bit 7 (/BSY Busy) está a O, significa que la impresora está ocupada (buffer de impresión lleno, procesando información, pendiente de inicializar, etc.).

El bit 6 (/ACK Acknoledge) indica que se ha producido una transferencia correcta: cuando del puerto paralelo se transfiere un byte a la impresora, la impresora activa la línea ACK de reconocimiento del carácter y, como consecuencia, el bit ACK del registro de estado pasa a nivel bajo; cuando el bit ACK está a nivel alto, significa que la impresora está ocupada y no se pueden realizar envíos.

El bit 5 (PAP Paper) si está a 1, señala que la impresora ***no*** *dispone de papel.*
Metodología de Programación, Programación en C, Aplicaciones electrónicas 6 /16
Técnicas de Programación 3ª Parte: Programación del puerto paralelo

El bit 4 (OF/ON Line Off) indica cuando está a 1, que la impresora no está en línea.
El bit 3 (ERR) si está a 0, indica que se ha producido un error de impresora (mal funcionamiento, falta de papel, impresora fuera de línea ...).

Los bits 0,1 y 2 no se utilizan.

El registro de control

*El **registro de control** permite controlar las transferencias de información con la impresora, y puede ser escrito y leído desde el microprocesador. Es un registro de entrada/salida cuya dirección se obtiene sumando 2 a la dirección base del puerto (0x37A en L PT 1). Los bits de este registro se designan en la Figura 32.3, donde el símbolo «/» delante del nombre del bit indica que es* activo a nivel bajo.

REGISTRO DE CONTROL (C)							
7	6	5	4	3	2	1	0
X	X	X	IRQ	DSL	/INI	ALF	STR
				(*)		(*)	(*)
				17	16	14	1 ← Pines

Figura 32.3. El registro de control del puerto paralelo en el PC y los pines correspondientes en el conector DB-25.

*El símbolo (*) indica que los bits STR, ALF y OSL del registro de control son invertidos por el hardware con relación a las líneas correspondientes al cable de conexión, por lo que el nivel de los bits 0,1 y 3 del registro es complementado con relación a las líneas correspondientes.*
El significado que tienen los bits de este registro es el siguiente:

El bit 4 (IRQ) es el que permite controlar la generación de interrupciones de tipo hardware desde el puerto paralelo. Si este bit está a 1, el interface paralelo puede generar la petición de interrupción IRQ7 (en LPT1), que se corresponden con las interrupción 0x0Fh respectivamente del procesador 80X86. Esta petición de interrupción se produce cuando se da una transición H L en la línea ACK.

El bit 3 (DSL) : La mayoría de las impresoras paralelo IBM-compabiles, no utilizan esta línea y son activadas con un pulsador de on-line. El bit 2 (INI) produce una inicialización de la impresora (es poco utilizado).

Si el bit 1 (ALF) está a nivel alto, la impresora produce automáticamente un cambio de línea *(LF) cada vez que recibe un* retorno de carro *(CR).*

El bit 0 (STR) controla la línea que permite validar el dato existente en el registro de datos. La puesta a 1 del bit STR genera un impulso corto que indica a la impresora que el carácter del registro de datos es válido y debe ser aceptado. Así pues, cada vez que se precise enviar un carácter, no basta con ponerlo en el registro de datos, sino que hay que hacer un reset en el bit STR del registro de control y validar el dato volviendo a poner un 1 en ese bit.

Los bits 5, 6 y 7 no se utilizan.

Entradas y salidas por el puerto paralelo

Al hablar de operaciones de entrada y salida por el puerto paralelo no debe olvidarse que, inicialmente, este elemento se desarrolló de acuerdo con el están dar Centronics con el fin, casi exclusivo, de que el PC pudiese enviar datos en paralelo a la impresora conectada, no se pensó en la posibilidad inversa: que el PC pudiese recibir datos a través de ese puerto.
Las operaciones de entrada y salida de información a través del puerto paralelo en el PC las realizaremos ***gestionando el puerto paralelo en el nivel de registros****, es decir, programando directamente los circuitos integrados o chips que constituyen la tarjeta de interface, lo cual permitirá aprovechar al máximo todas las posibilidades que ofrezca realmente el hardware de la tarjeta de interface.*

Características E/S

Cuando usamos el puerto paralelo para otro cometido distinto al original, solo podemos hablar de 12 líneas de salida de información desde el ordenador:

pines del 2 al 9 registro de datos
pines 1,14, 16 y 17 registro de control

15 líneas de entrada al mismo:

pines 2-9registro de datos
pines 10,11,12,13 y 15 registro de estado

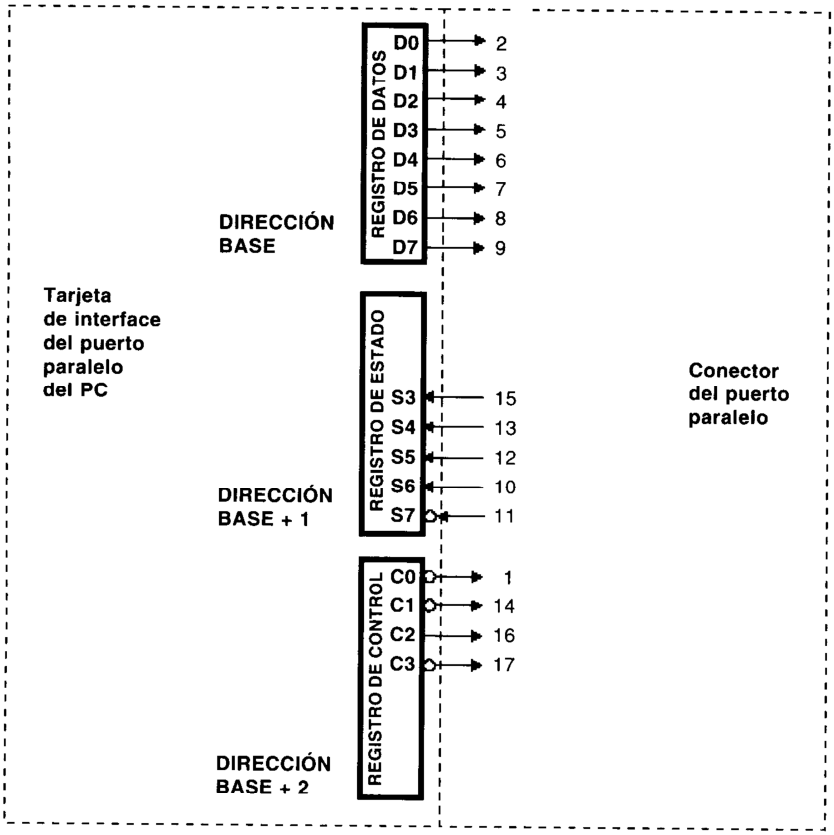


Figura 32.4. Líneas en el conector DB-25 del puerto paralelo.

Esto hace del puerto paralelo un interface de comunicación con el exterior bastante flexible.

*El **registro de estado** es de sólo lectura. Cuando se **lee** este registro, lo que se recibe es el estado lógico de los pines 10, 11, 12, 13 y 15 del conector DB-25 (el bit S 7 contiene el complemento del estado de la línea). Los tres bits de menor peso (SO-S2) no se utilizan y, habitualmente, se encuentran a nivel alto*

*El **registro de control** es parcialmente bidireccional. Cuando se **escribe** en los cuatro bits de menor peso (C0 - C3) lo que se hace es establecer el nivel lógico de los pines C 2 de forma directa y C0, C 1 y C 3 de forma complementada.*

Los tres bits de mayor peso (C 5 C 7) no se utilizan.
De forma experimental, se ha podido comprobar que, sólo en algunas tarjetas de interface paralelo, el bit C 6 del registro de control influye en la configuración, de modo que si C 6 = 0 las I líneas de datos se configuran como ENTRADAS y si C 6 =1 las líneas de datos se configuran como SALIDAS. Otras tarjetas, sobre todo si son bidireccionales, no permiten el cambio de nivel de ese bit.

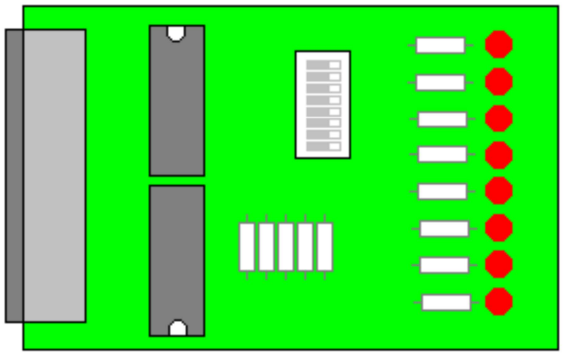
*El **registro de datos** es de tipo latch de lectura y de escritura, de modo que cuando se realiza una operación de escritura (OUT) el dato se carga en los bits correspondientes y las líneas asociadas del conector tienden a alcanzar la tensión correspondiente a ese estado.*

*En algunas ocasiones las líneas de datos de la tarjeta de interface paralelo (Centronics) son bidireccionales, pero la **etapa de salida** se ha construido mediante buffers con transistores en **colector abierto**. En este caso, el hecho de que las operaciones de entrada y salida se hagan por las mismas líneas, condiciona notablemente el proceso de lectura, ya que con esa configuración electrónica de las líneas de datos (D0 - D7), los valores lógicos leídos **dependerán del nivel lógico presente en el registro** y* **del valor de tensión en la línea** *(que no tienen por qué coincidir) de acuerdo con lo mostrado en la Tabla 32.2.*

Tabla 32.2. Niveles lógicos y de tensión en el puerto paralelo.

Nivel lógico de los bits en el registro de datos	Nivel de tensión en los pines del conector	Nivel lógico leído en una operación de lectura del puerto paralelo
L	L	L
L	H	L
H	L	L
H	H	H

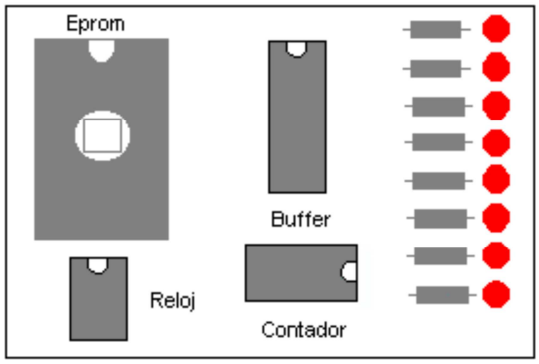
*A la vista de la Tabla 32.2, lo que se deduce es que, **si se va a realizar una operación de lectura** sobre el puerto paralelo, lo que se va a* **leer realmente** *es la operación **AND lógica entre el nivel lógico del registro y el nivel lógico de la línea**, lo que implica que, **si se desea realizar una lectura** real del estado de las líneas, deberá **escribirse antes el dato 0xFF** en el registro de datos del puerto paralelo.*



La tarjeta de E/S que se va a utilizar es un simulador de impresora, donde se pueden activar los 8 bits de datos y realizar lecturas del registro de control.
Las instrucciones en Turbo C que dan acceso a los puertos están en la librería ***conio.h*** *y son, para* escribir, ***outp (****puerto ,dato****)****, y para* leer ***inp (****puerto****).***
La tarjeta consta de 2 búferes para aislar las conexiones desde al PC y el exterior. Cada Led representa 1 bit del registro de datos, siendo el Led superior el correspondiente a D0 y el inferior al D7. Debemos pensar que si tenemos 8 bits de datos, cada uno de ellos se encendería de acuerdo a la potencia de 2 que nos da su peso, así los datos 1,2,4,8,16,32,64 y 128, encenderían cada uno de los Leds D0 a D7. Si queremos activar el Led de D3 escribiríamos la siguiente instrucción en C:

```
outp ( 0x378, 8)
↑instrucción ↑puerto ↑dato
```

Uso de la tarjeta como secuenciador
Si pensamos en la estructura real de un circuito digital que realice una secuenciación veremos que se asemeja a este diagrama:



En una memoria Eprom guardamos los datos y mediante un contador fijo vamos leyendo las distintas posiciones. Este circuito simple tiene el inconveniente poder realizar sólo una secuencia programada, aunque con alguna lógica adicional podemos incluir varios programas y seleccionarlos a posteriori. Vemos que es necesario un circuito contador y también presenta el problema de modificar el programa de la secuenciación, que debe ser extraída la memoria, borrarla completamente, y regrabar el nuevo programa y todos los anteriores.
Si utilizamos una tarjeta de E/S nos podemos olvidar de la Eprom e implementarla mediante una matriz. Podemos declarar una matriz de caracteres, cuyos valores están dentro del rango de 8 bits y se puede inicializar la matriz con el programa o secuencia que necesitemos. Luego mediante bucles del tipo ***for*** *o* ***while*** *se puede recorrer la matriz, e incluso variar el recorrido y el programa en función de ciertas condiciones.*

Unos ejemplos:

- **Recorrido del puerto:**

```
#include <conio.h>
void main (void)
{ char i=0;
for (i=0;i<256;i++)
outp (0x378,i);
}
```

- **Parpadeo de todos los leds de forma simultánea durante 10 veces:**

```
#include <conio.h>
void main (void)
{ char i;
for (i=0;i<10;i++)
outp (0x378,255);
outp (0x378,0);
}
```

- **Encendido secuencial ascendente – descendente**

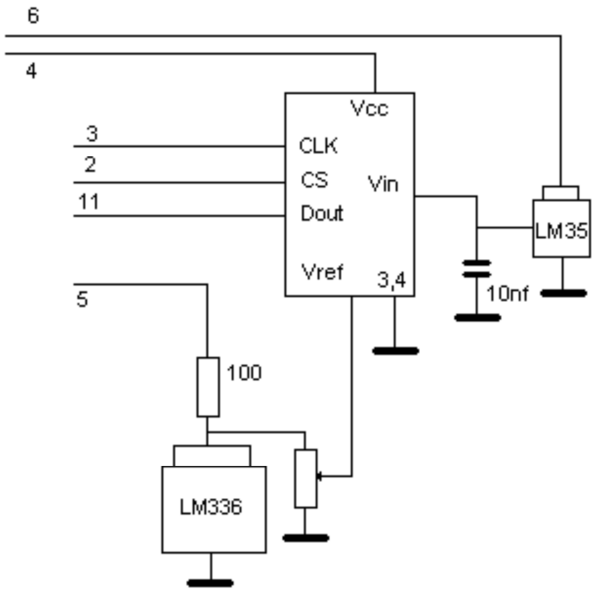
```
#include <conio.h>
void main (void)
{ char datos [ ]={1,2,4,8,16,32,64,128};
char i;
outp (0x378,0);
for (i=0,i<8,i++) // Recorrido ascendente
outp (0x378,datos[i]);
outp (0x378,0);
for (i=7,i<0,i--)
outp (0x378,datos[i]); // Recorrido descendente
}
```

Cálculo y conversión de señales analógicas

Medir una temperatura con un PC es relativamente fácil. Necesitamos un sensor que traduzca los grados centígrados a un voltaje y seguidamente convertir ese voltaje en una secuencia de ceros y unos, lo que se conoce como conversión analógica/digital (AD). El problema es que, al igual que ocurre con muchos ordenadores, en un portátil de 1,5 Kg. y 25 cm. de largo no se puede insertar ninguna tarjeta comercial, lo que obliga a una profunda reflexión para encontrar soluciones. Para complicar más las cosas, tampoco se dispone de ninguna fuente de alimentación a mano, a excepción de las baterías, Las cuales no podemos utilizar. Finalmente debemos proporcionar unos valores fiables y útiles. Así, un conversor AD de 10 bits, nos proporciona, en un rango de 0 a 100 grados, una resolución de una décima de grado, aunque también dependerá de los componentes.

Como el problema más importante es el de la alimentación del circuito, lo trataremos en primer lugar. No es del todo cierto que en el puerto paralelo no exista alimentación, dado que cada una de las líneas de salida produce una débil señal de nivel TTL. En términos electrónicos, 5 voltios para el nivel alto mientras que una señal a nivel bajo estaría casi a 0v. En cualquier caso el nivel 1 debe proporcionar ***al menos 3,8v****, por lo que el primer requisito del diseño. es que todos sus componentes puedan funcionar con una tensión superior a 3,5v. e interior a 5v. Un segundo requisito, más crítico, es el del consumo. La señal que aparece en cada línea compatible TTL es ciertamente débil y no admite cargas que impliquen consumos apreciables pero podemos tomar algo más de 1 mA de la salida como un valor seguro. En vez de utilizar un convertidos AD en paralelo utilizaremos uno que nos proporciona una salida de datos en serie porque un convertidor en paralelo, además de complicar el diseño, obligaría a disponer de 10 líneas de entrada de datos, mientras que en el convertidor en serie solo se necesitan,* ***dos de salida (selección y reloj) y uno de entrada (datos).***

Esquema del circuito:



Uno de los posibles convertidores que cumplen todos nuestros requerimientos es el LTC 1092 de Linear Technology. *Su funcionamiento es el siguiente:*
El integrado LTC 1092 comienza su ciclo cuando la línea de selección (CS) se pone a cero. Después de unos 2 ciclos de estabilización el integrado proporciona una conversión por aproximaciones sucesivas, para después transmitir el resultado mediante 10 pulsos de reloj (1 por bit). Esta señal será generada ordenador. La forma de hacerlo es fácil. Se pone una de las salidas del puerto paralelo alto, se espera un poco, después se vuelve a poner a nivel bajo y volvemos a esperar otro poco; y así sucesivamente. En medio de este proceso leemos el valor que nos proporciona el convertidor. La frecuencia de muestreo debe ser entre 10KHz y 500Khz. El resto de las líneas del puerto paralelo se situarán a nivel alto para que todas ellas en paralelo puedan alimentar al circuito.
Otros componentes:
Todos los convertidores AD necesitan una tensión de referencia que es aquella que equivale al fondo de escala del convertidor. Si el circuito es de 10 bits podemos alcanzar hasta 210-1= 1023□Vref. y 0□□0voltios. Mediante

el integrado LM336 de National generamos una tensión estable y prácticamente inmune a cambios de temperatura centrada en 2.49v y gracias a una resistencia ajustable regulamos a 1 voltio. (Técnicamente el circuito puede ajustarse hasta medir tensiones de 2.5v de fondo de escala). Ya sólo nos queda por describir el último componente, que es el sensor de temperatura. Se trata del LM35 que ofrece una tensión /ºC de 0.01 voltio, así 0º □□0 voltios y 25º □□0,25 voltios. Como el fondo de escala es de 1 voltio la temperatura máxima es de 100 grados.
Ahora se requiriere de un programa en el PC con el fin de transformar las señales analógicas en digitales. Como podemos ver en listado adjunto podemos direccionar el puerto paralelo realizando una operación de salida (OUT) en puerto correspondiente, que por defecto es 0x378h para el LPT1
Recordemos que ***todas las salidas deben ponerse a nivel alto, exceptuando aquellas donde van a existir variaciones.*** *La programación del chip LTC1092 es relativamente sencilla.*

```
read_ad_one_mtm10()
    {
    int base=0x378,val=0xff;
    int q,n,b=512,tot=0;

    #define LOW outp(base,val-3);fwait(10);
    #define HIGH outp(base,val-1);fwait(10);

    outp(base,val-2); fwait(10);

    LOW; HIGH; LOW; HIGH;

    /* conversión
    for (q=0; q<10; ++q) {
         LOW; HIGH;
    }
    */

    /* recogida de datos */
    for (q=0;q<10;++q) {
         LOW; HIGH;
         n=(inp(base+1)&128);
         n=(n)?b:0;
         tot+=n;
         b>>=1;
    }
    return tot;
}
```

Si seguimos el esquema, el bit 0 del puerto corresponde con la señal CS mientras que el bit 1 corresponde con la señal de reloj. Para la entrada de datos se utiliza el bit 7 del registro de estado (pin 11). Tenemos dos macros, denominadas HIGH y LOW que determinan el estado del reloj.
El algoritmo opera de la siguiente forma a la hora de realizar una conversión: en primer lugar, con el reloj en estado bajo, procedemos a activar la línea CS, que debe pasar del estado alto de reposo al bajo de comienzo de

actividad. Deberá mantenerse en este estado bajo hasta terminar el ciclo, momento en el cual se alzará. Seguidamente, se produce un ciclo de reloj, es decir , se pone el reloj a nivel bajo y acto seguido a nivel alto. En este ciclo no se realiza ninguna operación. A partir de este momento, todo está preparado para recibir los bits uno a uno. En cada ciclo de reloj leeremos el puerto de entrada recibiendo un bit. Dicho de otra manera, subimos el bit de reloj, lo bajamos y leemos un bit en el puerto. Esta operación la realizaremos diez veces, obteniendo así diez bits, de forma que nos llega ***primero el de mayor peso*** *(D9) [en el programa b=512 porque son 10 bits] y finalmente el de menor (Do).*
El resultado obtenido puede verse influenciado por inestabilidades fruto de la electrónica, por lo que es recomendable el empleo de un filtro que estabiliza la lectura. En general, las medidas ofrecidas por el tienen una gran estabilidad, aunque es posible que en ocasiones se pueda producir algún "ruido" externo que estropee el valor de la señal.
Para minimizar estos problemas, se suele emplear algún algoritmo que obtenga la media de tres o cuatro lecturas consecutivas, lo que suaviza la respuesta eliminando picos indeseados.

RAP 2:
Diseña funciones específicas para resolver la problemática planteada.

Estructura de una función.

El formato de una definición de una función es:

Tipo_de_valor_devuelto Nombre_de_función*(*lista_de_parametros*)*

{

declaraciones e instrucciones.

return expresión*;*

}

Donde:

Tipo_de_valor_devuelto *: Es el tipo de datos del resultado que la función devuelve a la función invocadora. El* Tipo ***void*** *indica que la función no devuelve ningún valor. Si no se especifica el* Tipo_de_valor_devuelto*, el compilador supone por default que es de tipo* ***int****.*

Nombre_de_función *: Es cualquier nombre valido para la función.*

lista_de_parámetros : *es una lista separada por comas que contiene las declaraciones de los parámetros que recibe la función al ser llamada. Si una función no recibe ningún valor, la* lista_de_parámetros *es* ***void*** *o simplemente se deja en blanco*. *Es necesario indicar el tipo de cada parámetro de las lista de parámetros de una función.*

Las declaraciones e instrucciones *entre llaves forman el cuerpo de la función.*

"No es posible definir una función dentro de otra bajo ninguna circunstancia."

Si la función devuelve un resultado, la instrucción return expresión*; le devuelve el valor de la expresión a la función invocadora.*

Prototipo de una función.

Un prototipo de una función le dice al compilador el nombre de la función, el tipo de datos que esta devuelve, el numero, tipo y orden de los parámetros que dicha función espera recibir. El compilador utiliza los prototipos para hacer validas las funciones. Los prototipos deben declararse junto a las librerías y antes de definir la función ***main()****.*

Ejemplo: programa que encuentra al mayor de tres numeros.

```
#include<stdio.h>/* declaración de las librerías */
#include<conio.h>

int maximo(int, int, int);    /* prototipo de la funcion*/
int main()                                                    /* funcion principal */
{
  int a, b, c;

  clrscr();
  printf(" teclea tres numeros :");
  scanf("%d", &a);
  scanf("%d", &b);
  scanf("%d", &c);
```

```
  maximo(a,b,c);     /* a, b, y c son argumentos para la llamada de la funcion maximo */
  printf("El mayor de los tres numeros es : %d", maximo());
  getch();
  return 0;
}

int maximo(int x, int y, int z)    /* función */
{                              /* x, y,  z, son parámetros para la definición de la función máximo */
  int max = x;

  if( y > max)
          {
           max = y;
          }
  if( z > max )
          {
            max = z;
          }
  return max;
}
```

Paso de argumentos a una función.

Dos maneras de invocar funciones son mediante ***llamada por valor y llamada por referencia****.*

Cuando se pasa un argumento utilizando una llamada por valor*, se hace una copia del argumento, la cual se pasa a la función llamada. Los cambios a la copia no afectan el valor de la variable original.*

Con una llamada por referencia*, el invocador le da a la función llamada la capacidad de acceder al argumento original y modificar la información.*

Un parámetro de referencia es un alias de su argumento correspondiente. Para indicar que el parámetro de una función se pasa por referencia, simplemente se pone un ampersand (&) después del tipo del parámetro en el prototipo de la función.

Ejemplos.

```
int XOR(int &);/* llamada por referencia */

int NOT(int);    /* llamada por valor */
```

Invocación de una función.

Para invocar una función, solo hay que hacer mención de ella e indicar los argumentos que utilizara, si es que utiliza.

Ejemplo:

Función1(x); / invocación de una función de un argumento */*

Función2(x,y,z); / invocación de una función de tres argumentos. */*

Función3(); / invocación de una función que no utiliza argumentos */*

Ejemplo.

El siguiente programa muestra todos los puntos explicados en este capitulo.

```
/* Programa que calcula el cuadrado de un numero */
/* se hace uso de funciones llamadas por referencia y por valor */

#include <stdio.h>      /* librerias */
#include <conio.h>

int cuadrado1(int);     /* prototipo de la función llamada por valor */
```

```
void cuadrado2(int &);/*prototipo de la función llamada por referencia*/

int main(void)           /* definicion del cuerpo de la función main() */
{
 int x = 2, z = 4;        /* declaracion de variables */

 clrscr();
 printf(" x = %d ", x);
 cuadrado1(x);          /* invocacion de la función cuadrado1*/
 printf("\n valor de x despues de la función cuadrado1 x = %d ", x);
 printf("\n z = %d ", z);
 cuadrado2(z);          /* invocacion de la función cuadrado2*/
 printf("\n valor de z despues de la función cuadrado2 z = %d ", z);
 getch();
 return 0;              /* devolucion de valor */
}

int cuadrado1(int a)    /* definicion de la función llamada por valor */
{
 a *= a;
 return a;     /* devolucion de expresion */
}

void cuadrado2(int &cref)  /* definicion de la función llamada por
referencia */
{
```

```
 cref *= cref;}
```

La corrida para este programa es la siguiente:

x = 2

valor de x después de la función cuadrado1 x = 2

z = 4

valor de z después de la función cuadrado2 z = 1

Como puede observarse, explícitamente se ve la demostración de los conceptos de llamada por valor y por referencia. Puede apreciarse como la variable x que utiliza una función por valor no se ve alterada y que la variable z que utiliza una función por referencia si se ve afectada directamente. Esto es por que en la funcion cuadrado1 la variable "a" es solo una copia de x y en la funcion cuadrado2 la variable "a" es la misma variable z.

FUNCIÓN Y PROCEDIMIENTOS

PASAR POR VALOR y POR REFERENCIA

1. ***Una variable***

a. Por valor

- ***Prototipo***

long *suma(****long*** *a,****long*** *b)****;***

- ***Definicion***

long *suma (****long*** *a,****long*** *b)*

{

*Long suma =0****;***

Suma = a+b;

return (suma);

}

- ***Llamada***

*Resultado = suma(var1, var2)****;***

b. Por referencia

- ***Prototipo***

void *intercambio(****long****a,****long****b)****;***

- ***Definicion***

void *intercambio(****long****a,****long****b)*

{

```
    *a ^= *b;

    *b ^= *a;

    *a ^= *b;
}
```

- ***Llamada***

```
Intercambio(&var1, &var2);
```

2. *Array unidimensional-los arrays se deben pasar por referencia a las funciones*

Así, cuando queremos pasar como argumento un vector, no es necesario hacer copia de todo él en la lista de parámetros: basta pasar como parámetro la dirección del primer elemento del vector. La función podrá acceder a todos los elementos del vector mediante operatoria de punteros o mediante índices.

Habitualmente, al pasar un array o matriz, será necesario pasar, como otros parámetros, la dimensión o dimensiones de ese array o matriz.

La llamada de la función usará el nombre del vector como argumento, ya que como dijimos al presentar los arrays y las cadenas, el nombre de un array o cadena, en C, indica su dirección: decir nombre_vector *es lo mismo que decir* &nombre_vector[0].

Existen tres formas de declarar un parámetro formal que va a recibir un puntero a un vector:

1 - tipo nombre_funcion(tipo vector[dimensión]); *es decir, declarando el parámetro como un vector dimensionado.*

Para llamar a una función:

```
nombre_funcion(nom_vector);

nombre_funcion(&nom_vector[0]);
```

ATENCIÓN – Si en el prototipo y en la definición de la función hemos especificado array con número de elementos fijo, solo podemos pasarle array que tenga la misma longitud, en caso contrario puede dar error o funcionar incorrectamente el programa.

2 - tipo nombre_funcion (tipo vector[]); *es decir, declarando el parámetro como un vector sin tamaño determinado.*

Para llamar a una función:

nombre_funcion(nom_vector);

nombre_funcion(&nom_vector[0]);

Es este caso el numero de elementos del array el indiferente para la función. Se debería pasar como otro parámetro.

3 - tipo nombre_funcion (tipo*); *es decir, declarando el parámetro como un puntero.*

Para llamar a una función:

nombre_funcion(nom_vector);

nombre_funcion(&nom_vector[0]);

Actividades

Competencia particular 4

UTILIZA FUNCIONES PARA CADA UNO DE LOS EJERCICIOS

*1. Crea una array de 10 elementos de tipo **short**. Rellena lo con valores desde el teclado. Imprime en la pantalla la dirección de memoria de cada elemento y el valor que contiene en el formato indicado:*

&nom_array[0] esta en dirección de memoria: 5467234 y tiene valor 5

&nom_array[1] esta en dirección de memoria: 5467236 y tiene valor 5

.

&nom_array[9] esta en dirección de memoria: 5467252 y tiene valor 5

Observa el intervalo que hay entre las direcciones.

2. Crea un array de elementos, rellénelo con valores leídos desde el teclado. Calcula e imprime la suma y el producto de todos sus elementos en el formato indicado:

Elemento nº 1 vale 4

Elemento nº 2 vale 5

LA SUMA DE LOS … ELEMENTOS ES: 45

EL PRODUCTO DE LOS … ELEMENTOS ES: 345

3. Crea un array, rellénalo con valores desde leidos desde el teclado. Copia sus elementos al revés en otro array e imprime los elementos de los dos arrays en el formato indicado:

Nom_array1	*Nom_array2*
1	*8*
2	*7*
4	*6*
5	*5*
6	*4*
7	*2*
8	*1*

4. Crea un array, rellenalo con valores desde el teclado. Imprime sus elementos indicando para cada uno si es par o impar en el formato indicado:

Nom_array

Elemento nº 1 vale 4 – es par

Elemento nº 2 vale 5 – es impar

.

Elemento nº … vale 56 – es par

5. Crea un array, rellénalo con valores leídos desde el teclado. Imprime sus elementos calculando para cada uno su factorial en el formato indicado:

Nom_array

Elemento nº 1 vale 4. El factorial de 4 es 24

Elemento nº 2 vale 5. El factorial de 5 es 120

.

Elemento nº … vale 12. El factorial de 12 es ¿?

6. Crea un arra(arreglo)y, rellenalo con valores desde el teclado. Imprime sus elementos indicando para cada uno si el valor es un numero perfecto en el formato indicado:

Nom_array

Elemento nº1 vale 28 – es perfecto

Un numero es perfecto cuanto es igual a la suma de sus divisores excepto el mismo. Por ejemplo:

Los divisores de 28 son: 1, 2, 4, 7,14. La suma de sus divisores es: 1+2+4+7+14 = 28. El un numero perfecto.

7. Crea un arreglo . El número de elementos de dicho arreglo lo indicará el usuario. Rellénalo con valores desde el teclado. Imprime aquellos de los valores que son múltiplos de 9. Un número es multiplo de 9 si la suma de sus cifras es 9 o múltiplo de 9.

Ej. 99 es multiplo por que 9+9 = 18, que a su ves 1+8 = 9

8. Crea un array, rellenalo con valores leídos desde el teclado. Calcula la suma y la media de los elementos que se encuentran en las posiciones pares e imprime estos elementos y sus valores correspondientes.

9. Crea un arreglo (array), rellenalo con valores leídos desde el teclado. Imprime el numero de veces que aparece un numero indicado por el usuario, la posición en el array.

Por ejemplo:

Si el array tiene 5 elementos con los siguientes valores: 1, 2,4,2,1

Y el usuario quiere saber cuantas veces aparece el 2, en la pantalla hay que imprimir la respuesta en el siguiente formato:

El numero 2 se repite 2 veces.

Las posiciones que contienen el valor 2 son: 1,3.

10. Crea un array de numero elementos indicado por el usuario. Rellenalo con valores leídos por el teclado e imprime los caracteres correspondientes en el formato indicado:

Nomb:array

Elemento nº 1 vale 65, el carácter correspondiente es A

.

11.- Desarrolla un programa que encuentre el mayor de tres números. Utiliza funciones y paso de argumentos.

12.- Desarrolla un programa que calcule la hipotenusa de un triangulo rectángulo cuando se dan los otros dos lados. Utiliza funciones y paso de argumentos.

13.- Desarrolla un programa que tome dos enteros y determine si el segundo es múltiplo del primero. Utiliza funciones y paso de argumentos.

14.- Desarrolla un programa que acepte una serie de datos enteros y los pase uno por uno por una función que determine si cada valor que se evalué es par o impar. Utiliza funciones y paso de argumentos.

15.- Desarrolla un programa que tome un valor entero y lo devuelva con sus dígitos en reversa. Por ejemplo el numero 7631, el resultado será 1367. Utiliza funciones y paso de argumentos.

16.- Complementa la siguiente tabla referente a instrucciones en modo grafico

INSTRUCCION	***FUNCION***	***SINTAXIS***
getmaxx()		
getmaxx()		
Setcolor		
Setbkcolor		
Cleardevice		
Putpixel		
Line		

Lineto		
Moveto		
Rectangle		
Circle		
Arc		
Ellipse		
Polly		

17.-Implemente un programa que tenga como función dar la oportunidad al usuario de implimir de 0º a 360º las seis funciones trigonométricas o bien los logaritmos de una función. El programa debe presentar recurrencia.

19.- Modifique el programa anterior para que esta secuencia se realizase de forma ininterrumpida hasta oprimir una tecla.

20.- Mediante secuencias de control haga un programa cuya salida sea una figura geométrica en pantalla , la cual deberá dar el aspecto de irse alejando en la pantalla.

21.- Implemente un programa por medio de una elipse que de cómo salida una figura que avance en pantalla simulando un "pacman".

22.-Haga un programa que de la opción de mover una circunferencia de 4 direcciones a través del teclado.

23.-Realice un programa que muestre en la pantalla un as de corazones

24.- Diseñe un programa que dibuje en pantalla una pieza de un sistema con animación

25.- Utilice el menú de ayuda de "BORLAND" para ejecutar un ejemplo de aplicación de las funciones getimage y putimage, realizando las adecuaciones correspondientes.

En dicho programa deberá incluir los comentarios de cada una de las líneas.

26.- Realice un programa en donde se vea la aplicación de las instrucciones getimage y putimage.

27.- Implemente un programa que muestre através del puerto paralelo una cuenta de 0 a 10 en binario con intervalos de 20 seg entre ellos y termine.

28.- En base al programa la función del programa anterior realiza una animación en modo grafico que simule la salida de los leds en pantalla.

29.- Diseñe un programa para encender de 1 en 1 los led´s conectados al puerto paralelo en orden acendente y desendente con retardos de 2 segundos, 1 segundo y 0.5 segundos.

30.- Diseñe un programa ue realice un contador de 0 a 100 en binario y lo muestre en los led´s conectados al puerto paralelo con un retardo de 0.5 segundos.

31.- Diseñe un programa que muestre en un diplay de 7 segmentos conectado al puerto paralelo un contador de 0 a 9 con intervalos de 2 segundos entre ellos.

Apéndices

Apéndice 1

El conjunto de caracteres ASCII

Valor SCII	Carácter	Valor ASCII	Carácter	Valor ASCII	Carácter	Valor ASCII	Carácter
000	NUL	032	blanco	064	@	096	`
001	SOH	033	!	065	A	097	a
002	STX	034	“	066	B	098	b
003	ETX	035	#	067	C	099	c
004	EOT	036	5	068	D	100	d
005	ENQ	037	%	069	E	101	e
006	ACK	038	&	070	F	102	f
007	BEL	039	‘	071	G	103	9
008	B S	040	(	072	H	104	h
009	HT	041	)	073	1	105	i
010	L F	042	*	074	j	106	j
011	VT	043	+	075	K	107	k
012	F F	044	,	076	L	108	1
013	C R	045	-	077	m	109	m
014	so	046	.	078	N	110	n
015	si	047	/	079	0	111	0

016	DLE	048	0	080	p	112	p
017	DC1	049	1	081	Q	113	q
018	DC2	050	2	082	R	114	r
019	DC3	051	3	083	S	115	5
020	DC4	0521	4	084	T	116	t
021	NAK	053	5	085	u	117	u
022	S YN	054	6	086	v	118	v
0 2 3	ETB	055	7	087	w	119	w
024	CAN	056	8	088	x	120	x
025	EM	057	9	089	y	121	y
026	SUB	058	:	090	Z	122	z
027	ESC	059	;	091	[	123	1
021	FS	160	<	092	\	124	1
029	GS	061	=	093	]	125	}
030	RS	062	>	094	^	126	~
031	Us	063	¿	095	-	127	DEL

APÉNDICE 2 Algunas funciones de Librería

Función	Tipo	Propósito	Archivo *include*
abs (i)	int	Retorna el valor absoluto de i.	stdlib. h
acos (d)	double	Retorna el arco coseno de d.	math. h
asin(d)	double	Retorna el arco seno de d.	math. h
atan (d)	double	Retorna la arco tangente de d.	math. h
atan2(d1,d2)	double	Retorna la arco tangente de dl/d2.	math.h
atof(s)	double	Convierte la cadena s a una cantidad en doble precisión.	stdlib.h
atoi(s)	int	Convierte la cadena s en un entero.	stdlib.h
atol(s)	long	Convierte la cadena s en un entero largo.	stdlib.h
calloc(u1,u2)	void*	Reserva memoria para un array de u i elementos, cada uno de longitud u2 bytes. Retorna un puntero al principio del espacio reservado.	malloc. h, 0 stdlib.h
ceil(d)	double	Retorna un valor redondeado por exceso al siguiente entero mayor,	math.h
cos(d)	double	Retorna el coseno de d,	math.h
cosh(d)	double	Retorna el coseno hiperbólico de d.	math.h
difftime(11,12)	double	Retorna la diferencia de tiempo 11 - 12, donde 11 y 12 representan el tiempo transcurrido después de un tiempo base (ver la función time).	time.h
exit(u)	void	Cerrar todos los archivos y buffers y terminar	stdlib.h

		el programa. (El valor de u se asigna en la función para indicar el estado de terminación.)	
exp(d)	*double*	*Elevar* e *a la potencia d* (e = *2.7182818 es la base del sistema de logaritmos naturales [Neperianos]).*	*math.h*
fabs(d)	*double*	*Retorna el valor absoluta de d*	*math.h*
fclose(f)	*int*	*Cierra el archivo f. Retorna un 0 si el archivo se ha cerrado con éxito.*	*stdio.h*
feof(f)	*int*	*Determina si se ha encontrado un fin de archivo. Si es así, devuelve un valor distinto de cero; en otro caso devuelve 0.*	*stdio.h*
fgetc(f)	*int*	*Leer un carácter del archivo f.*	*stdio.h*
fgets(s,i,f)	*char**	*Leer una cadena s, con i caracteres del archivo f.*	*stdio.h*
floor(d)	*double*	*Retorna un valor redondeado por defecto al entero menor más cercano.*	*math.h*
fmod(d1,d2)	*double*	*Retorna el resto de di/d2 (con el mismo signo que di).*	*math.h*
fopen(s1,s2)	*File**	*Abre un archivo llamado si, del tipo s2. Retorna un puntero al archivo.*	*stdio.h*
f printf (f, .	*int*	*Escribe datos en el archivo f (el resto de los argumentos son complicados)*	*stdio.h*
fputc(c,f)	*int*	*Escribe un carácter en el archivo f.*	*stdio.h*
fputs	*int*	*Escribe una cadena en el archivo f.*	*stdio.h*
fread(s,i1,i2,f)	*int*	*Leer i2 elementos, cada uno de tamaño il*	*stdio.h*

		bytes, desde el archivo f hasta la cadena s.	
free(p)	*void*	*Liberar un bloque de memoria reservada cuyo principio está indicado por p.*	*malloc.h.0*
scanf (f,....)	*int*	*Leer datos del archivo f (el resto de los argumentos son complicados)*	*stdio.h*
fseek(f,1,f)	*int*	*Mover el puntero al archivo f una distancia de 1 bytes desde la posición i (i debe representar el principio del archivo, la posición actual del puntero o el fin del archivo).*	*stdio.h*
ftell(f)	*long int*	*Retorna la posición actual del puntero dentro del archivo f.*	*stdio.h*
fwrite(s,il,i2,f)	*int*	*Escribir i2 elementos, cada uno de i 1 bytes desde la cadena a hasta el archivo f.*	*stdio.h*
getc(f)	*int*	*Leer un carácter del archivo f.*	*stdio.h*
getchar(void)	*int*	*Leer un carácter desde el dispositivo de entrada estándar.*	*stdio.h*
gets(s)	*char**	*Leer una cadena desde el dispositivo de entrada estándar.*	*stdio.h*
isalnum.(c)	*int*	*Determina si el argumento es alfanumérico. Retorna un valor distinto de cero si es cierto; en otro caso devuelve 0*	*ctype.h*
¡salpha(c)	*int*	*Determina si el argumento es alfabético. Retorna un valor distinto de cero si es cierto; en otro caso devuelve 0.*	*ctype.h*

isascii(c)	*int*	*Determina si el argumento es un caracter ASCII. Retorna un valor distinto de cero si es cierto; en otro caso devuelve 0.*	*ctype.h*
iscntrl(c)	*int*	*Determina si el argumento es un carácter ASCII de control. Retorna un valor distinto de cero si es cierto; en otro caso devuelve 0.*	*ctype.h*
isdigit(c)	*int*	*Determina si el argumento es un dígito decimal. Retorna un valor distinto de cero si es cierto; en otro caso devuelve 0.*	*ctype.h*
isgraph(c)	*int*	*Determina si el argumento es un carácter gráfico ASCII imprimible (hex Ox2l-Ox7e, octal 041-176). Retorna un valor distinto de cero si es cierto; en otro caso devuelve 0.*	*ctype.h*
islower(c)	*int*	*Determina si el argumento es una minúscula. Retorna un valor distinto de cero si es cierto, en otro caso devuelve 0.*	*ctype. h*
isodigit(c)	*int*	*Determina si el argumento es un dígito octal. Retoma un valor distinto de cero si es cierto, en otro caso devuelve 0.*	*ctype. h*
¡sprint (c)	*int*	*Determina si el argumento es un carácter ASCII imprimible (hex Ox2O-Ox7e, octal 040-176). Retorna un valor distinto de cero si es cierto; en otro caso devuelve 0.*	*ctype. h*

ispunct(c)	*int*	*Determina si el argumento es un carácter de puntuación. Retorna un valor distinto de cero si es cierto, en otro caso devuelve 0.*	*ctype. h*
isspace(c)	*¡lit*	*Determina si el argumento es un espacio en blanco. Retorna un valor distinto de cero si es cierto; en otro caso devuelve 0.*	*ctype. h*
isupper(c)	*int*	*Determina si el argumento es una mayúscula. Retoma un valor distinto de cero si es cierto; en otro caso devuelve 0.*	*ctype. h*
isxdigit(c)	*int*	*Determina si el argumento es un dígito hexadecimal. Retoma un valor distinto de*	*ctype. h*
-		*cero si es cierto; en otro caso devuelve 0.*	
labs (1)	*long int*	*Retorna el valor absoluto de 1.*	*math. h*
log (d)	*double*	*Retoma el logaritmo natural de d.*	*math. h*
loglO(d)	*double*	*Retoma el logaritmo (en base 10) de d.*	*math. h*
malloc(u)	*void**	*Reserva u bytes de memoria. Retoma un puntero al principio del espacio reservado.*	*malloc.h, 0 stdlib.h*
pow(d1, d2)	*double*	*Retoma di elevado a la potencia d2.*	*math. h*
printf	*int*	*Escribe datos en el dispositivo de salida estándar (los argumentos son complicados, ver Apéndice G).*	*stdio.h*
putC (c, f	*int*	*Escribe un carácter en el archivo f.*	*stdio.h*
putchar(c)	*int*	*Escribe un carácter en el dispositivo de salida estándar.*	*stdio.h*

puts(5)	*int*	*Escribe una cadena en el dispositivo de salida estándar.*	*stdio.h*
rand(void)	*int*	*Retoma un valor aleatorio positívo.*	*stdlib.h*
rewind(f)	*void*	*Mueve el puntero al principio del archivo f.*	*stdio.h*
scanf(...	*int*	*Leer datos desde el dispositivo de entrada estándar (los argumentos son complicados; ver Apéndice G).*	*stdio.h*
sin(d)	*double*	*Retorna el seno de d.*	*math.h h*
sinh(d)	*double*	*Retorna el seno hiperbólico de d.*	*math.h*
sqrt(d)	*double*	*Retoma la raíz cuadrada de d.*	*math.h*
srand(u)	*void*	*Inicializa el generador de números aleatorios.*	*stdlib.h*
atrcmp(s1,s2)	*int*	*Compara dos cadenas lexicográficamente, Retorna un Y alor negativo si 9 1 < 9 2 ; 0 si al y s2 son idénticas y un valor positivo si al > 92.*	*string.h*
trcpy(s1,s2)	*int*	*Compara dos cadenas lexicográficamente, considerar mayúsculas o minúsculas. Retorna un valor negativo sí si< s2: u si 91 y 92 son idénticas y un valor positivo el > 92.*	*string.h*
strcpy (s 1,s2)	*char**	*Copia la cadena 9 2 en la cadena 9 1.*	*string.h*
strlen(s)	*int*	*Retorna el número de caracteres en la cadena e.*	*string.h*
strset(s, c)	*char**	*Pone todos los caracteres de 9 en c (excluyendo*	*string.h*

		el carácter nulo de¡ Final \O).	
system(s)	*int*	*Pasa la orden 9 al sistema operativo, Retorna 0 si la orden se ejecuta correctamente; en otro caso devuelve un valor distinto de cero, típicamente -1).*	*stdlib.h*
tan(d)	*double*	*Retorna la tangente de d.*	*math h*
tanh(d)	*double*	*Retorna la tangente hiperbólica de d.*	*math.h*
time(p)		*long int Retorna el número de segundos transcurridos después de un tiempo base designado.*	*time.h*
toascii(c)	*int*	*Convierte el valor del argumento a ASCII.*	*ctype.h*
tolower(c)	*int*	*Convierte una letra a minúscula.*	*ctype.h o stdlib.h*
toupper(c)	*int*	*Convierte una letra a mayúscula.*	*ctype.h.o stdlib.h*

Algunos aspectos importantes

c denota un argumento de tipo carácter.

d denota un argumento de doble precisión.

f denota un argumento archivo.

i denota un argumento entero.

l denota un argumento entero largo.

p denota un argumento puntero.

g denota un argumento cadena.

u denota un argumento entero sin signo.

**La mayoría de los compiladores comerciales de C vienen acompañados de muchas más funciones de biblioteca.*

Consulte el manual de referencia de su compilador particular para una información más

detallada de las funciones anteriores y para tener una lista de las funciones .

Apéndice 3

Mensajes de error

Cuando un programa es compilado en BORLANDC, se encarga de revisar la sintaxis de las instrucciones una a una, por lo que de existir un error de sintaxis, se mostrara un mensaje de advertencia, indicando lo siguiente:

1. *Donde se localizo el error (En el programa Fuente o en el Linker).*
2. *Numero de línea donde probablemente se encuentre el error.*
3. *El error en cuestión.*

Este apéndice es un resumen de los mensajes más comunes de error.

La razón de existir de este apéndice es porque tanto el lenguaje como los mensajes de error son en "INGLES". Motivo por el que desde ahora se te exige aprender el idioma en cuestión.

Mensajes:

Lo siguiente es un compendio de los mensajes más usuales a los que te enfrentaras:

Undefined Symbol_main() in module c0.ASM

Este error es indicativo de que a tu programa le falta la funcion main(), función indispensable en todo programa que ejecutes. Verifica que tu programa tenga esta funcion.

Declaration sintaxis error.

Este error indica que la instrucción o función declarada tiene un error de sintaxis. Verifica que este bien escrita.

Function 'printf' should have a prototype.

Este error indica que no ha sido declarada la libreria donde se encuentra el código de la función **printf()**. Verifica que tengas la linea **#include <stdio.h>** en el encabezado de tu programa.

Function 'clrscr' should have a prototype.

Este error indica que no ha sido declarada la librería donde se encuentra el código de la función **clrscr()**. Verifica que tengas la línea **#include <conio.h>** en el encabezado de tu programa.

Function 'getch' should have a prototype.

Este error indica que no ha sido declarada la librería donde se encuentra el código de la función **getch().** Verifica que tengas la línea **#include <conio.h>** en el encabezado de tu programa.

Function should return a value.

Esta advertencia indica que tu funcion main() devolverá un valor int, char, double o cualquier otro que hayas definido. Verifica que tu programa tenga la instrucción **return valor;**. Donde valor es sustituido por un numero o variable (ejemplo: return 0;)

Unexpected }

Este error indica que una de las llaves no se esperaba encontrar. Verifica el numero de llaves que abren y las que cierran, deben coincidir en numero, igual numero de llaves que abren es el mismo numero de llaves que cierran.

Statement missing ;

Este error indica que una instrucción no se termino correctamente. Verifica que todas las instrucciones terminen con punto y coma (;). Recuerda que las estructuras de control no llevan punto y coma.

Unterminated string or Character constant.

Este error indica que una cadena de caracteres o un caracter no se definió bien. Verifica que todas las cadenas de caracteres de los **printf() o scanf()** esten bien terminados con comillas ("…..").

Compound statement missing }

Este error indica que una llave de cierre falta. Verifica que las funciones tienen su llave de apertura y cierre ({ }).

Unable to open include file 'stdio'

Este error indica que el archivo **'stdio'** no puede abrirse. Verifica que este bien escrito el nombre del archivo en la librería **#include <stdio.h>**

Declaration terminated incorrectly

Este error indica que una función no fue bien definida. Verifica que las funciones "**definidas"** no tengan punto y coma (;)

Type mismatch in redeclaration of 'funcion()'

Este error indica que la declaración de la función que creaste no es igual a la que definiste en el programa. Verifica que la declaración de la función en el encabezado sea igual a la definición de la función en el cuerpo del programa.

Undefined Symbol 'x'

Este error te indica que la variable 'x' no la declaraste. Verifica que en la seccion de declaracion de variables hayas declarado la variable que se te indica, asi como tambien verificar si es variable local o global.

Case statement missing :

Este error te indica que el "case" posee un error. Verifica que la definición del case sea la indicada. **Ejemplo: case 1: case 'a':**

Expression sintax

Este error te indica que la sintaxis de la expresión esta mal. Verifica que este bien escrita la expresion.

Do-while statement missing ;

Este error te indica que al utilizar el Do-while, no finalizaste con punto y coma. Verifica la estructura de tu Do-while y recuerda que el while lleva punto y coma. Ejemplo: do{…….}while(….);.

Possibly incorrect assignment

Esta advertencia te indica que posiblemente en una condición realizas una incorrecta asignación. Verifica todas tus condiciones, recuerda que para comparación y asignación hay diferentes operadores. (<=, >=, ==, =!, <, >, =, &&, ||).

) expected

Este error indica que una instrucción no cerro su parentesis. Verifica que todas las instrucciones tengan sus parentesis de apertura y cierre.

Bibliografia.

- *Apuntes desarollados por Ing. Miguel Garcia julian e Ing.Dimitri Cab Cordero Plan 93*
- *Brian W. Kernighan y Denis M. Ritchie, "El lenguaje de programacion C", Prentice Hall.*
- *Deitel y Deitel, "C++ Como programar", Pearson Prentice Hall.*
- *Técnicas de programación Cap 32 2 Ed. McGraw Hill*
- Revista PC World, Octubre'94. Sección de programación
- *www.abcdatos.com* *"Tutoriales gratuitos"*
- *www.elrincondec.com* *"Programas publicos de C"*

NOTA: El desarrollo de la internet ha permitido que sea posible la difusión de programas de dominio publico hechos en lenguaje C. Por lo que es posible encontrar una amplia gama de paginas WEB dedicadas a la exposición de programas o tutoriales dedicados al lenguaje C que pueden ser de mucha utilidad en el aprendizaje autodidacta.