Programacion C

Introducción :

Hola mi nombre es Álvaro y os voy a conducir en la introducción a la programación en el Lenguaje C o también conocido como "Lenguaje de programación de sistemas" desarrollado en el año 1972 por Dennis Ritchie para UNIX un sistema operativo multiplataforma.

El lenguaje C es del tipo lenguaje estructurado como son PASCAL, FORTRAN, Basic. Sus instrucciones son muy parecidas a otros lenguajes incluyendo sentencias como if, else, for, do y while...

Aunque C es un lenguaje de alto nivel (puesto que es estructurado y pose sentencias y funciones que simplifican su funcionamiento) tenemos la posibilidad de programar a bajo nivel (como en el ASSEMBLER tocando los registros memoria etc.).

Para simplificar el funcionamiento de el lenguaje C tiene incluidas librerías de funciones que pueden ser incluidas haciendo referencia la librería que las incluye es decir que si queremos usar una función para borrar la pantalla tendremos que incluir en nuestro programa la librería que tiene la función para borrar la pantalla.

Ventajas: La programación en C tiene una gran facilidad para escribir código compacto y sencillo a su misma vez.

En el lenguaje C no tenemos procedimientos como en otros lenguajes solamente tenemos funciones los procedimientos los simula y esta terminante mente prohibido escribir funciones, procedimientos y los comandos en mayúscula todo se escribe en minúsculas (a no ser las constantes ☺)

Los archivos en C se escriben en texto puro de ASCII del Dos si se escribe en WORD por ejemplo el mismo incluye muchos códigos no entendidos por el compilador y generara errores; una vez escrito se debe pasar a compilar el archivo; los archivos tienen 2 Extensiones **archivo.C** que es el archivo a compilar el que contiene todas los procedimientos funciones y código de nuestro programa y **archivo.h** que es las librerías que contienen las funciones de nuestro programa.
(NOTA: El compilador genera Archivos con extensión .EXE)

Cada instrucción que pasemos a poner en C va segida de un punto y coma para decirle al compilador que hasta ahí llega la instrucción simula un Enter del teclado.

Ej. ***clrscr();*** /* borra la pantalla */

Estructura de Programación en "C"

Encabezados **<ficheros.h>** en el se almacenan las funciones que trae el lenguaje propio o las funciones nuestras se les llaman Librerías.

Las funciones de **C** dan mucha potencia al programador en podemos realizar los programas con mucha potencia y facilidad de código.

Constantes simbólicas en el se definen las constantes de nuestro programa es lo único que se escribe en mayúsculas **ej.: #define ENERO 31**

Nota:

(Las constantes son variables que conservan el mismo valor todo el programa una vez que incluimos una constante en nuestro programa el compilador va a sustituir el texto contenido en la constante por el texto del mismo.)

Cabecera de funciones es la llamadas a las funciones que se encuentran el mismo programa y al final del.

Función principal MAIN() en el se desarrollara todo el código del programa las llamadas a funciones procedimientos etc. (es una función que sé autoejecuta cuando se compila el programa).

Muestra :

```
#include <librería.h> /* librería que incluye las funciones */

# void mifuncion();

Main()    /* Comentario  */
  {
        variables locales
        .....
        .....
        función();
        .....
        ....
 }

void mifuncion()
{
   .....
   .....
}
```

Ejemplo de un programa que muestra por pantalla “Hola mundo” y espera que se presioné una tecla.

```
/* Archivos de cabezera de funciones. */
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>

void main() /* comienzo del programa función principal.*/
{
 clrscr();   // Borra la pantalla
 printf("Hola Mundo"); // Imprime Hola Mundo
 getch(); // espera que se presioné una tecla.
}
```

La misma función main() puede recibir 2 parámetros de la línea de comando (ej. Dos) como el numero de parámetro (int argc) o como el texto de cada parámetro (int argv[]) la función del main() quedaría así.
Dentro de los argumentos pasados por la línea de comando el parámetro numero 0 contiene el nombre del archivo**.**

```
main ( int *argc, char *argv[])
{
  .... instrucciones  - ....
}
```

Y en nuestro programa yo podría detectar los comandos pasados de la línea de comando.
Este ejemplo muestra por pantalla todos los parámetros pasados por la línea del dos

```
// Inclusión de librerias
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
main( int *argc, char *argv[]) //parámetros de la línea de comando
{
  int cont = 0;
  clrscr();
  for (cont = 0; cont <= argc ; cont++)
     {
     printf("Parametro [%d] - Texto [ %s] ",cont, argv[cont]);
     }
 getch();
}
```

En el mismo ejemplo incluimos palabras reservadas que veremos mas adelante como son for.

Comentarios , variables y constantes.

Nota:

Para incluir en nuestro programa comentarios muy necesario para que otros programadores entiendan lo que quisimos hacer ☺ y nosotros mismos el con el tiempo.

```
/* Esto es un comentario */

/* esto es un comentario
   en varias
   lineas
*/
```

En C++ los comentarios se pueden hacer en una linea con //
(//en c++ esto es un comentario)

Tipo de Variables en C

Nombre	Tipo	Bits	Tamaño
Unsigned char	Carácter	8	0 a 255
Char	Carácter	8	-128 a 128
Enum	Enumeracion	16	-32,768 a 32,767
Unsigned int	Numero	16	0 a 65,535
Sort int	Numero	16	-32,768 /32,767
Int	Numero	16	-32,768 /32,767
Unsigned long	Numero	32	0 to 4,294,967,295
Long	Numero	32	-2,147,483,648
Float	Numero	32	3.4 * (10**-38)
Double	Numero	64	1.7 * (10**-308)
Long double	Numero	80	3.4 * (10**-4932)

Definición :

Para la definición de variables de debe poner primero el tipo de variable a definir y luego el nombre de la variable (si queremos inicializarla podemos a continuación poniendo = valor) y si queremos definir mas de una variable del mismo tipo podemos poniendo una coma a continuación.

Eje: `int contador;  /* define una variable con el nombre contador de tipo entero */`

`int cont = 0 ; /* inicializando cont con 0 */`

`Int cont , cont2, contn = 0; /* definición de mas de una variable */`

```
Float científica = 10.1
Int cont = cont +1;
Int cont = cont2;
```

Incrementar y Decrementar Variables

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

```
Cont++;  /*esta es una forma de hacer cont = cont+1 */
```

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

```
Char saludo[5] = “hola”;
Printf(“nombre %s”, saludo);
```

Dentro de la variable saludo se encuentra contenido el texto “hola” y en la posición 1 de la cadena la letra “H” en la posición 2 “o” y así sucesivamente
Cuando se define una cadena en la posición final de la cadena se pone \0 que significa el final mismo. Cuando nosotros trabajemos con cadenas podemos preguntar si es el final de la cadena (\0)

Cadena saludo

Posición 1	Posición 2	Posición 3	Posición 4	Posición 5
H	O	L	A	\0

La posición 5 de la cadena tiene el carácter de fin de linea que indica que la cadena llega hasta ahí.

De esta misma manera yo puedo imprimir el segundo carácter de la cadena
Para la asignación de texto se puede hacer así en la inicialización pero en el programa no se puede asignar texto de esta manera se debe inicializar cadenas con las funciones strcpy, no puedo hacer como en clipper, pascal, cobol a= “hola “ sino que se debe dar el valor de la cadena con strcpy(a,”hola”)

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

```
Char saludo[5] = “hola”;
Printf(“ %c”,saludo[2]);  /* resultado : “o” */
```

Este ejemplo pide texto por medio del teclado y va contabilizando lo que se le va ingresando.

```
#define ENTER  13 // defino unos tipos de constantes para el programa.
#define ESCAPE 27


void main()
{
 char texto[80];  // definición de variables e inicialización
 int cont  = 0;
 int contx = 0;
 int letra = 0;

 clrscr();
 while ((letra != ENTER) && (letra != ESCAPE))
  {
   letra = getch();
   gotoxy(cont,10);
   printf("%c",letra);
   gotoxy(20,12);
   printf("Cantidad de letras :%d",cont);
   texto[cont++] = letra;
  }
}
```

TRABAJANDO CON ESTRUCTURAS

También se pueden definir tipo de variables con varias variables dentro llamada estructuras de variables (muy precido a los registros Grande DBASE...) dentro de un registro nosotros podemos agrupar un conjunto de variables bajo un mismo nombre por ejemplo tengo una ficha de cliente donde tiene nombre, dirección, teléfono ; en la misma ficha yo podría preguntar por cliente.nombre o cliente.telefono etc.

Tipo de definición

```
Struct  nombre {
                Tipo variable1
                Tipo variable2
                Tipo variablex
               };
struct   nombre nombre_dela_variable;
```

Bueno para que sirve esto: Si tenemos que definir un nuevo tipo de variable como por ejemplo una variable que contenga nombre, teléfono, y edad en la misma podría definirlo así.

```
struct cliente {
        char nombre[];
        char telefono[10];
        int edad;
                } ;

struct cliente clientes[20];
..
...
```

dentro de nuestro programa...

```
if cliente[1].edad == 10    // si el cliente numero 1 tiene 10 años entonces mostrar
    printf("Muy joven");
```

Otra de las ventajas puede ser para guardar el contenido de los registros de clientes dentro de un archivo, nosotros solamente tendríamos que volcar el registro a el archivo y listo o para levantarlo tendríamos que leer del archivo de registro.

Constantes :

```
        #define nombre  texto_deremplazo
```

Las constantes son variables que se mantienen todo el programa con un mismo valor. Cada vez que nosotros llamamos a una constante lo que hacemos es remplazar la constante por su valor.
No se debe poner ; después de la definición.
Ej.:

```
#define  MAXLINEAS 24
#define  MINOMBRE "Alvaro"
#define  ARCHIVO "C:\autoexec.bat"
```

y cuando en mi programa yo haga por ejemplo printf("Autor : %s",MINOMBRE);
se sustituirá el contenido de la constante por el contenido de la misma
printf("Autor : %s","Alvaro"); se pueden definir constantes de tipo macro que se tocara en el próximo capitulo.

Un ejemplo que para los amantes de turbo pascal les puede gustar

```
// Ejemplo realizado para clase de definicion de varialbes.
#include <stdio.h> // inclusión de librerías.
#include <stdlib.h>
#include <conio.h>

// definición de constantes.
#define BEGIN {
#define END }
#define WRITELN( texto ) printf("%s\n",texto)
#define READLN(variable) gets(variable)
#define PAUSA getch()
```

```
void main()
BEGIN
  char var[255];
  clrscr();
  WRITELN("hola");
  READLN( var );
  WRITELN( var);
  PAUSA;
END
```

Cuando yo definí como constantes BEGIN el texto que sustituye a la palabra va a ser "{" y lo mismo para END "}"
Pero cuando definí WRITELN y READLN lo puse entre parentesis con un parametro que se lo paso a printf() como parametro. Esto es una macro simple se pueden hacer macros mas complejas como por ejemplo

```
/* Inclusion de librerias de funciones de C */
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>

//definicion de una macro simple.
#define PAUSA printf("PAUSA"); getch();
/* esta macro tiene 2 comandos printf() mostrar en pantalla

        getch() espera que se precione una tecla,
*/

//definicion de una macro compuesta.
#define max(a,b) ((a > b) ? (a) : (b));

/* Esta macro compara el contenido de a y b
   si a es > b entonces retorna a
   si no retorna b
   en seudo codigo es
    A mayor a B entonces retorna A si no B
*/

/* Funcion principal */
void main()
{
 int res= 0, var1, var2;
 clrscr();                        // borrar la pantalla
 printf("Ingrese numero 1:");     // mostrar texto
 scanf("%d",var1);                // capturar var1
 fflush(stdin);                   // limpio el buffer del teclado.
 printf("\nIngrese numero 2;");   // mostrar texto
 scanf("%d",var2);                // capturar var2
 fflush(stdin);                   // limpio el buffer del teclado.
 res = max(var1, var2);           // realizar ecuacion llamando a la macro
 printf("\nEl mayor es %d",res); // imprimir resultado
 PAUSA;                           // llamar a la macro de pausa.
}
```

Operadores Lógicos

Valor	Operador
Suma	+
Resta	-
División	%
Multiplicación	*
IGUAL (EQUAL)	= =
O (OR)	&&
NO (NOT)	!
DISTINTO	!=
Y (AND)	||
MAYOR	>
MENOR	<
Mayor =	>=
Menor =	<=
Operador para comparación de bits	&

Ejemplos

```
(numero == 10)
(letra != 'p')
(tecla != ENTER)
((numero > 10 ) ||  (numero < 100)) /* numeros entre 10 y 100*/
((a >= b) || (c !=d))
```

Este ejemplo repite el movimiento de una pelotita hasta que se precione una tecla
Contiene 2 funciones como son

```
/*
   Pelotita que rebota por la pantalla hasta que se precione una tecla

*/

#include <stdio.h>
#include <conio.h>
#include <string.h>
#include <stdlib.h>
#include <ctype.h>
#include <dos.h>

#define ARR 1
#define ABA 0
#define DER 1
#define IZQ 0
#define VEL 35

int x = 0;
int y = 0;
int s_x = 1;
int s_y = 1;
```

```
void main()
{
  clrscr();
  while ( !kbhit() ) /* si no se preciona una tecla kbhit debuelve 0 */
   {
    delay(VEL);   /* demora en segundos */
    /* wherex() = > debuelve la posicion del cursor en la columna
horizontal
       wherey() = > debuelve la posicion del cursor en la columna
vertical
    */
    gotoxy(wherex()-1 , wherey() ); printf(" ");
    if (s_x == DER)
     {
      if ( (x+1) != 80 )
      x++;
      else
      s_x = IZQ;
     }
    else
     {
      if ( ( x-1 ) != 0 )
      x--;
      else
      s_x = DER;
     }

    if (s_y == ARR)
     {
      if ( (y+1) != 24)
      y++;
      else
      s_y = ABA;
     }
    else
     {
      if ( (y-1) != 0)
      y--;
      else
      s_y = ARR;
     }
    gotoxy(x,y); printf("*");
  }
}
```

Función clrscr()

Esta función borra la pantalla es igual que el CLS de dos ;)

Es la abreviación de Clear Screen, atención esta función puede dar error cuando no se incluya la librería stdlib.h

```
#include <stdio.h>   /* librería de funciones */

main()
        {
          clrscr();  /* borrar la pantalla */
          printf("Mi primer programa"); /* muestra en pantalla el texto */
          printf("Para borrar la pantalla precione una tecla");
          getch();
          clrscr();
        }
```

Función printf()

Esta funcion muestra texto en pantalla a partir de la posicion actual del cursor tiene varios modificadores que luego vamos a mostrar pero la estructura del es printf("texto ..... [mascaras ....]"[,var1,var2,var3.....]);

Vea tambien **putc** que permite mostrar caracteres y hasta imprimirlos

Las mascaras en el comando printf son

Mascara	Resultado
%d	Muestra decimal
%o	Muestra en octal
%x	Muestra entro hexadecimal (minúsculas)
%X	Muestra entro hexadecimal (mayúscula)
%f	Notación científica (10E20)
%c	Carácter
%s	Modificador de cadena (string)
\n	Otro renglón (ENTER)
\t	Tabulador
\\	Barra de separación.
\a	Beep por parlante

Ejemplo 1.2

```
#include <stdio.h>
main()
{
  clrscr();  /*borrar pantalla*/
  printf("\n \n \t \t Muestra de PrintF"); /* mostrar texto en
                              pantalla en el separacion  renglon
                              con 2 tabulaciones*/
  printf("Numero %d",128); resultado Numero 128
  printf("Carácter %c",128); resultado Carácter "A"
  printf("Cadena %s","HOLA MUNDO");
  printf("el numero %d en notacion cientifica %f",384,384);
  printf("\n %c %c %c ",'F','I','N');
}
```

complicando un poco 8=O
dentro de un printf yo puedo implicar una cuenta por ejemplo para mostrar el resultado de una suma puedo hacer

```
printf("Suma : %d ", 1+2);
```

perfecto mostrara 3 de resultado, además puedo hacer

```
a = 1; b = 2;
printf ("Ecuación  ", ((a+b ) /2) );
```

cuando a y b son variables de tipo entero y también podemos incluir un if lógico pero mostremos primero un if común si a es mayor que b entonces mostraremos a sino b
Ejemplaso…

```
if (a > b )
  printf(" %d ", a);
else
  printf(" %d", b);
```

pero podemos hacerlo en el printf todo junto **;=**)

```
  printf( " %d ", ( a>b) ? (a) : (b));
```

en seudo-código seria si a mayor que b mostrar a sino b
también podemos hacerlo con texto por ejemplo

```
   printf( "%s", (a > b )? ("A mayor que B") : ("B mayor que A"));
```

Función GotoXY(x,y)
Para poder posesionar el cursor en algún lugar de la pantalla se debe usar el comando Gotoxy (Columna , Linea)

Ejemplo 1.3

```
....
....
gotoxy(10,10);
printf("Resultado %d",varresultado);
```

Ejemplo 1.4 :

```
/*      Este sensillo programa captura los dos valores de tipo numerico del teclado
y los guarda en las variables valor1 y valor2 por ultimo los suma y el resultado lo
guarda en la variable resultado y por ultimo lo muestra en pantalla.
*/
      #include <conio.h>
      #include <stdlib.h>
      #include <stdio.h>
      main()
      {     int valor1, valor2, resultado; /* definicion de
variables */
```

```
          clrscr();
          gotoxy(10,10);
          printf("ingrese el primer numero :");
          scanf(" %d ",&valor1);   /*captura del teclado y lo
guarda la variable*/
          gotoxy(10,12);
          printf("ingrese el segundo numero :");
          scanf(" %d",&valor2);
          resultado = valor1 + valor2);
          gotoxy(10,14);
          printf("Resultado : %d", resultado);
     }
```

Funcion SCANF()

Scanf permite capturar datos de el teclado y guardarlos en una variable ; el tipo de variables admitidas en el es cualquier tipo el formato de scanf es Scanf(" %modificador ",&var1,&var2 .....);

El modificador que se utiliza es el mismo para printf()
Pero una vez captura la variable y se debe limpiar el buffer de entrada con la funcion ffpush(stdin);

Ahora veremos el mismo programa pero usando mas potencia de C y una pequeña introduccion a las funciones.

```
/* inclusión de librerías */
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>

/* definicion de variables */
int sumando1, sumando2 =0; /* sumando1 y sumando2 tienen valor 0
*/

main()
{
 clrscr();
 printf("Ingrese sumando1 y sumando2 ");
 scanf("%d %d",sumando1,sumando2);
 ffpush(stdin);
 printf("el resultado es :,%d",sumando1+sumando2);
}
```

/* y ahora el mismo programa con una función inclusión de librerías */

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
int suma( int var1 , int var2); /* definición de la cabecera de la
funciona/
main();
```

```
{
      int xvar1, xvar2,resultado = 0
      clrscr();
      printf("\n ingrese el valor1 y valor2");
      scanf("%d %d",xvar1,xvar2);
      resultado = suma(xvar1, xvar2);
      printf("el resultado es :%d",resultado);
}
/* desarrollo de la función */
int suma( int var1, int var2)
      {return var1+var2}
```

Introducción a las funciones:

Una Función es una parte del código independiente del programa que se llama dentro de él, como dijimos anteriormente C no tiene procedimientos solamente funciones los procedimientos los simula con funciones que no devuelven nada (Void) la estructura de la funciones en C y es los elementos mas fuertes del lenguaje.
El desarrollo de una función es
Tipo_de_valor_retorno nombreFunción(lista de argumentos)
{
 declaracion de variables locales a la función
 codigo ejecutable
 return (exprecion) // optativo
}

las variables de nuestras funciones son locales es decir que solamente viven cuando se esta ejecutando la llamada a la función; el código ejecutable son las sentencias que el programa utiliza para funcionar y el valor de retorno es el resultado que va a devolver la misma función (si se dice que una función retorna un entero en el return se debe poner un entero como valor de devolución
Ej: esta función de ejemplo cuenta cuantos caracteres existen dentro de una variable que se le pase como parámetro.

```
Int cuentacaracteres( char cadena[] )
{
 int cont =0; // defino una variable local para la función
 for (cont=0; cadena[cont]!= '\0'; cont++)
  {
   ;;
  }
 return (cont);
}
```

analicemos la función expuesta primero decimos que la función devuelve un int entero y el nombre de la función es **cuentacaracteres** y como argumento se le pasa una cadena. Dentro de la función se define una variable local que solamente funciona para esta función y es de tipo entero después recorro la cadena pasada

como parámetro buscando el fin de línea (‘\0’) cuando lo encuentro retorno el valor de cont que tiene la cantidad de caracteres que contiene la cadena. ;)
Y dentro de nuestro programa se debe definir las funciones de esta manera.

```
Valor_de_retorno  nombrefuncion( lista de argumentos);

Main()
{   ....
   ....
}
valor_de_retorno nombrefuncion( lista de argumentos)
{
  variables locales
 codigo ejecutable ...
 ...
 ...
 return valor_de_retorno
};
```

ejemplo de una funcion que dibuja un cuadrado en la pantalla solamente con pasarle las cordenadas de x,y -x1,y1 esta función no devuelve nada.

```
/*
  Esta funcion dibuja un cuadrado simple y un cuadrado doble
  especificando las cordenadas de x y ,x1 ,y1

*/

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

void cuadro ( int x, int y, int x1 , int y1 ,int tipo)
{
 int cont, lin_hor, lin_ver, sup_izq, sup_der, inf_izq, inf_der;
 if (tipo ==1)
   {
    lin_hor =196; lin_ver =179;
    sup_der =218; sup_izq =191;
    inf_der =192; inf_izq =217;
   }
 else
   {
    lin_hor =205; lin_ver =186;
    sup_der =201; sup_izq =187;
    inf_der =200; inf_izq =188;
   }
 for (cont=x ;cont <=x1 ;cont++)
  {
   gotoxy(cont,y ); cprintf("%c",lin_hor);
   gotoxy(cont,y1); cprintf("%c",lin_hor);
  }
```

```
 for (cont=y ;cont <=y1 ;cont++)
  {
   gotoxy(x, cont); cprintf("%c",lin_ver);
   gotoxy(x1,cont); cprintf("%c",lin_ver);
  }
 gotoxy(x ,y ); cprintf("%c",sup_der);
 gotoxy(x1,y ); cprintf("%c",sup_izq);
 gotoxy(x ,y1); cprintf("%c",inf_der);
 gotoxy(x1,y1); cprintf("%c",inf_izq);
}

main()
{
 clrscr();
 cuadro(1,1,80,24,1);
 cuadro(5,5,75,19,2);
}
```

Ejemplo 2.1:
/* este ejercicio devuelve el cuadrado de cualquier numero que se le pase como parámetro */

```
/* inclusión de librerías */
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>

int  suma3( int valor ); /* funcion que devuelve el valor +3 */

int cuadrado(int valor); /* funcion que devuelve el numero al
cuadrado */

main()
{
  int resultado = cuadrado(5);
 clrscr();
 printf("Cuadrado de %d es %d",5,resultado");
 /*también puede ser así printf("el cuadrado de %d es
%d",5,cuadrado(5));*/
}

int suma3( int valor )
{     int resultado = valor +3
     valor = valor +3;
       return resultado;
}
```

```
int cuadrado(int valor) /* la misma funcion pero anidando comandos */
{    return (valor * valor); }
```

esta funcion no devuelve nada pero coloca el texto en las pociciones pasadas como parametros.

```
void mostxy(int pox, posy, char texto[40])
{
 gotoxy(posx,posy);
 printf("%s",texto);
}
```

Esta función ubica el texto que le especificamos en el centro de la pantalla.

```
/* Funcion que sentra el texto en el medio de la pantalla */
void centrar(int linea , char texto[] )
{
 int medio=40-strlen(texto)/2;
 gotoxy(medio,linea);
 printf("%s",texto);
}
```

Nota :

Cuando se pasa como parámetro cadenas se debe definir como un arreglo de caracteres bajo un mismo nombre el cual va contener la cadena de caracteres. Podemos crear librerías de funciones que se llamen haciendo una inclusion de las mismas en nuestro programa por ejemplo crearemos un archivo Librería.H que lo que va a contener son nuestras funciones y dentro de nuestro programa haremos una inclusión de la misma para que las funciones sean llamadas desde la librería. Ej.

Dentro de librería.H

```
Int suma3( int valor)
{
 return (valor+3);
}
```

dentro de el archivo fuente EjLib1.c

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#include "Librería.H"

void main()
{
 int numero = 5;
```

```
  clrscr();
  printf("Numero sin sumar %d", numero);
  numero = suma3(numero);
  printf("Numero sumado 3 [ %d] ",numero);
  getch();
}
```

Funciones con recursividad :

Las funciones en C además tienen las propiedades de auto llamarse a si misma hacerse recursiva esto puede ser útil para aquellas funciones de (ordenamiento de archivos y índices)

Esta función se auto-ejecuta hasta que se realice toda la sumatoria de los números.

```
int sumatoria( int numero)
{
  if (numero !=0)
    { return ( numero  + sumatoria(numero-1) ); }
 }

main()
{
  clrscr();
  printf("Sumatoria de %d, es igual a %d ", 5 , sumatoria(5));
  getch();
}
```

TEMA 3 Sentencias de comparación y repeticiones

Instrucción IF

La sentencia if permite comparar 2 valores entre si. Por medio de la condición cuando se cumple (TRUE) entra a realizar la accion1 si no pasa a cumplir la accion2

```
 if ( condición_logica )
   { ....
    acción 1
     ....
   }
else
  { .....
    acción 2
    .....
  }
```

Ejemplo:

Este ejemplo compara el contenido de la variable var con NUMGRANDE una constante definida con el valor 100.

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

#define NUMGRANDE 100

main()
{
int var =0;
 printf("Ingrese un numero :\a "); /* \a para beep del
parlante*/
 scanf("%3d",var);
 if (var <= NUMGRANDE )
    {printf("Numero mayor a 100");}
 else
   {printf("numero menor a 100");}
 getch(); /* solamente haci espera que se presione una tecla
*/
}
```

Sentencia SWITCH

La sentencia switch puede comparar una variable con mas de un posible resultado Si ustedes tubieron el gusto de programar en clipper la sentencia es muy parecida a Case.

```
Switch (variable)
{
  case comparacion1 : .....
                      accion...
                      break
  .....
  case comparacion n : .....
                      accion...
                      break
  default  : ..... /* en caso de que no se cumpla ninguna...*/
                      accion...
                      break
}
```

Ejemplo aplicado con switch.-

....
....

```
Printf(“Seleccione una Opcion 1)agregar \n 2)Eliminar \n 3)salir:”)
scanf(“%d”,opcion);
switch opcion
{
 case 1 : printf(“selecciono agregar”);
          break;
 case 2: printf(“Eliminar”);
          break;
 case 3: printf(“Salida del programa”);
          exit(0); /* sale del programa y envia el mensaje de errorlevel 0 al DOS */
default :
          printf(“Opcion no valida”);
          break;
}
.....
```

vea la sección MENUPOP funciones

REPETICIONES

Es el mas usado en su sintaxis puesto que es muy parecido al de otros lenguajes como clipper pascal visual basic....

While primero evalúa la condición y no se cumple entra en el ciclo While hasta que la condición se cumpla.

```
While (condición)
{
  ....
  sentencias
  ....
};
```

Do primero entra cumple 1 vez las sentencias en el bucle y sigue cumpliendo hasta que se cumpla la condición.

```
Do
{
  ....
  ....
  sentencias
  ....
  ....
} while (condicion);

  /*
     Inclusion de librerias del programa.
```

```
    Ejemplo While.C del directorio.
*/
 #include <stdio.h>
 #include <conio.h>
 #include <stdlib.h>

 /* Definicion de constante del programa */
#define ENTER 13
#define ESC   27

char tecla =' ';


void main()
 {
  clrscr();
  do
  {
    while ((( tecla = getche()) != ENTER) && (tecla !=ESC));
    printf("\n"); /* salta al proximo renglon */
  } while (tecla != ESC);
 }
```

SENTENCIA FOR

La sentencia for o para permite ejecutar un conjunto de sentencias evaluando la condición central del for.

```
For ( variable = valor inicial ; condición  ; incremento)
{...
 ....
  sentencias....
  ...
 ....
}
```

este ejemplo muestra el contenido de la variable cont 10 veces.

```
for (cont =0 ; cont <= 10 ; cont++)
  {
    printf(“ %d ”,cont);
  };

/*
  Ejemplo que dibuja * en toda la pantalla.
*/

/* Inclusion de librerias */
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
```

```
/* deficicion de constantes */
#define LARGO 80
#define COL   24

/* inicialisacion y definicion de variables */
int contx = 0;
int conty = 0;

void main()
{
 clrscr();
 for ( conty = 0; conty != COL ; conty++ )
  {
   for ( contx = 0 ; contx != LARGO ; contx++ )
    { printf("*"); }
  }
}
```

Función variable_char = GETCH();

Permite capturar que tecla se presiono y guardarlo en una variable de tipo char, esta funcion no muestra que tecla se presiono es decir sin ECHO a pantalla.

Función variable_char = getche();

Permite capturar lo mismo que getch pero muestra el contenido a pantalla.

Ej.:

/* este ejemplo captura teclas normales y teclas de funciones como F1,F2,....y muestra la letra y su codigo ascii +300 */

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

#define ESC 27

char tecla( void )
{
   char tec;
   tec = getch();
   if (tec == 0)
        return tec = getch() +300 ;
   return tec;
}

main()
{
 char tecl;
 while ((tecl = tecla()) != ESC )
  {
  printf("Tecla : %c  Valor Ascii %d ",tecl,tecl);
  };
}
```

Función Gets(varialbe_string)

Permite capturar texto del teclado y guardarlo en la una variable de tipo String

Ejemplo 3.2

```
Char nombre[20] =""; /* defino la variable nombre de 20 caracteres. */
....
....
printf("Ingrese el nombre :");  /* muestro por pantalla */
gets(nombre);                   /* lo guardo en la variable nombre
printf("El nombre ingresado es ::%20s",nombre); /* lo muestro por pantalla*/
....
....
```

Función SCANF("modificador",variable)

La función scanf permite capturar desde un numero hasta un texto pasando como parámetro primero el modificador que es igual que el de printf() y luego el nombre de la variable se pueden poner mas que una variable y mas que un modificador.

```
int numero;
Int código;
Char nombre[20];
....
....
printf("Ingrese el numero  :");  /* muestro por pantalla */
scanf("%d",numero);                 /* lo guardo en la variable nombre
printf("El numero ingresado es ::%20s",nombre); /* lo muestro por pantalla */
....
....

scanf("%d  %20s", código, nombre);
```

Funciones Strcpy, Strcat, strlen, strcmp

Strcpy (variable, texto),

Almacena la dentro de la variable el texto pasado como parametro.

Variable = texto.

Strcat (destino , texto)

Encadena el texto dentro de la variable destino con el texto pasado como parametro.

Desitno = destino + texto

Int strlen(variable)

Devuelve la cantidad de caracteres que contiene la variable pasada como parametro.

Strcpy(Var, “hola”)

Res = strlen(var) res = 4

Int strcmp(variable1, variable2)

Compara el contenido de las dos variables y devuelve tres posibles resultados cuando son iguales devuelve 0 .- cuando la variable1 es mayor a la variable2 devuelve menor a 0 y cuando es inverso mayor a 0.

```
/*
      Programa de Ejemplo 3.3 para Manual.Doc

      Federico Rena  - Montevideo , Uruguay

      E-Mail feforena@yahoo.com

      Programa de ejemplo de strcpy, strlen, strcmp, strcat
*/


#include <stdio.h>          //inclusion de librerias.
#include <conio.h>
#include <string.h>
#include <stdlib.h>


void main()
{
 char cadena1[20]; /*definición de variables */
 char cadena2[20];
 char resulta[40] = "";
 int  res =0;

 clrscr(); /* borrar la pantalla */
 printf("\n\n\t Ingrese la Cadena 1:");
 scanf("%s",cadena1);
 printf("\t Ingrese la Cadena 2:");
 scanf("%s",cadena2);

 /* strlen ( variable ) devuelve el largo de una cadena */
 printf("\n la cadena 1 tiene [%d] letras",strlen(cadena1));
 printf("\n la cadena 2 tiene [%d] letras",strlen(cadena2));

 /* strcpy copia el texto como parametro en la variable destino */
 strcpy(resulta, cadena1);
 printf("\n Resultado : %s",resulta);

 /* strcat(destino , cadena) inserta dentro
    de la cadena destino la cadena parametro */
 strcat(resulta,cadena2);
```

```
 printf("\n\n las cadenas unidas :%s \n\n\t",resulta);

 res = strcmp(cadena1,cadena2);
 if (res == 0)
  { printf("Las cadenas son iguales"); }
 else
  {
   if (res > 0)
    {printf("La cadena 1 es mayor"); }
   else
    {printf("La cadena 2 es mayor"); }
  };

 getch(); /* espera que se presione una tecla (pausa)*/
 }


/* este es el desarrollo de la función strlen */
/* donde '\0' es el final de cadena */
int strlen( char cadena )
{
 int cont = 0;
 for (cont = 0 ; cadena[cont] != '\0' ; cont++)
     ;;
return cont;
}
```

Otras funciones de texto pueden ser feol() esta función detecta si es el final de la línea por ejemplo podemos tener una función que lea toda una línea de corrido
Funciones a nivel de caracteres.

Isalpha(c)	Define si c es un carácter
Isupper(c)	Define si c es un carácter en mayúscula
Islower(c)	Define si c es un carácter en minúscula
Isdigit(c)	Define si c es un dígito (de 0 a 9)
Isalnum(c)	Define entre c si es numero o carácter
Isspace(c)	Define si C es un carácter de espacio (código 32 del ASCII)
Toupper(c)	Convierte un carácter en minúscula en mayúscula
Tolower(c)	Convierte un carácter mayúscula en minúscula
isprint	Devuelve si es un carácter imprimible en pantalla o impresora

Esta funcion pasa todo un texto de minúscula a mayusculas

```
#include <conio.h>
#include <string.h>
char cadena[80] = "";
char letra;
int  cont       =  0;
void main()
{
 clrscr();
 printf("Ingrese cadena :");
 gets(cadena);
```

```
 /* pasar la cadena a minuscula */
 for (cont = 0; cont != strlen(cadena); cont++)
  {
   if isalpha( cadena[cont])
    {
     cadena[cont] = toupper(cadena[cont]);
    }
  }
}
```

Ahora mostraremos un ejemplo de una función de menú pop up lindo suena pero lindo de entender, búscale la vuelta que se la encontrar.

```
/* Librerias que incluye el programa */
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <conio.h>
#include <dos.h>


/* Definicion de constantes del programa */
#define screen (* screen_ptr )
#define  INICIO         379
#define  FINAL          371
#define  ARRIBA         372
#define  ABAJO          380
#define  IZQUIERDA      375
#define  DERECHA        377
#define  ESCAPE          27
#define  ENTER           13
#define  SI               1
#define  NO               0

/* Definicion de constantes y variables de la funcion */

struct pop_datos{
              int x;             /* posision de x */
              int y;             /* posision de y */
              char prompt[20];   /* mensaje de prompt */
              char mensaje[30];  /* datos en la linea de mensajes */
            };


/* Esta funcion devuelve el valor de la tecla que se preciono. */
int tecla( void );

/* Esta funcion centra el texto que le pasemos en la pantalla */
void centrar(int linea ,char texto[]);

/* Funcion que resalta la segunda letra de texto */
void r_letra(int x, int y, char cadena[]);

/* Esta funcion muestra el prompt en inverso */
void inverso_r_letra(int x, int y, char cadena[]);
```

```
/* Funcion pone en las cordenadas x ,y el texto que se le indique */
void xyt(int x, int y, char texto[]);

/* Funcion restablece los attributos de color de la pantalla  */
void norm( void );

/* Definicion de llamada a funciones. */
int teclaok( int valor , int valor_actual );

/* Pop_Menu Vertical y con movimiento  */
int menu_v(int cant, int o_actual, struct pop_datos apop[]);

/* Pop_Menu Horizontal y con movimiento  */
int menu_h(int cant, struct pop_datos apop[]);


/*-------------------------------------------------------------------------
*/
/*                CODIGO DE LAS FUNCIONES                  */
/*                ***********************                  */
/*                                                  */
/*-------------------------------------------------------------------------
*/

/* Funcion que debuelve el valor de la tecla que se preciono */
int tecla(void)
{
  int valor = getch();
  if (valor == 0)
     valor = getch()+300;
  return(valor);
}

/* Funcion que sentra el texto en el medio de la pantalla */
void centrar(int linea , char texto[] )
{
 int medio=40-strlen(texto)/2;
 gotoxy(medio,linea);
 printf("%s",texto);
}

/* Funcion que resalta la segunda letra de texto */
void r_letra(int x, int y, char cadena[])
{
 int ax = wherex();
 int ay = wherey();
 gotoxy(x,y);
 textcolor(LIGHTGRAY);
 cprintf("%s",cadena);
 textcolor(BLUE);
 gotoxy(x+1,y);
 cprintf("%c",cadena[1]);
 textcolor(LIGHTGRAY);
 gotoxy(ax,ay);
};
```

```
/* Esta funcion muestra el prompt en inverso */
void inverso_r_letra(int x, int y, char cadena[])
{
 gotoxy(x,y);
 textbackground(WHITE);
 textcolor(BLACK);
 cprintf("%s",cadena);
 textcolor(BLUE);
 gotoxy(x+1,y);
 cprintf("%c",cadena[1]);
 textcolor(LIGHTGRAY);
 textbackground(BLACK);
};

void xyt(int x, int y, char texto[])
{
 gotoxy(x,y); cprintf("%s",texto);
}

void norm( void )
{
 textcolor( LIGHTGRAY );
 textbackground( BLACK );
}


int menu_v(int opciones, int o_actual ,struct pop_datos apop[])
{

 /* Definicion de Variables */
 int cont;
 int valor;
 int opcion_elegida = 0;

 /* Codigo de la funcion  */

 for (cont=1; cont<=opciones; cont++)
     r_letra(apop[cont].x, apop[cont].y, apop[cont].prompt);

 inverso_r_letra(apop[o_actual].x, apop[o_actual].y,
apop[o_actual].prompt);

 do
 {


   valor = tecla();
    for (cont=1; cont<=opciones; cont++)
      r_letra(apop[cont].x, apop[cont].y, apop[cont].prompt);

   switch( valor )
   {
     case DERECHA : if (o_actual == opciones)
                    {    o_actual = 1; }
                else
                    {    o_actual++;   }
                break;
```

```
     case IZQUIERDA: if (o_actual == 1)
                       {    o_actual = opciones; }
                     else
                       {    o_actual--;    }
                     break;

     case INICIO   : o_actual = 1;
                     break;

     case FINAL    : o_actual = opciones;
                     break;

     case ESCAPE   : valor = 27;
                     opcion_elegida = 0;
                     return 0;
                     break;

     case ABAJO    : valor = 27;
                     opcion_elegida = 200 +o_actual;
                     return 200+o_actual;
                     break;

     case ARRIBA   : valor = 27;
                     opcion_elegida = 300 +o_actual;
                     return 300+o_actual;
                     break;

     case ENTER    : valor = 27;
                     opcion_elegida = o_actual;
                     return o_actual;
                     break;


     default:
                    o_actual = teclaok(valor,o_actual);
                    break;
    }
    inverso_r_letra(apop[o_actual].x, apop[o_actual].y,
apop[o_actual].prompt);
  } while ( (valor != 27) );
  return opcion_elegida;
}


/* Menu Horizontal */
int menu_h(int opciones, struct pop_datos apop[])
{

 /* Definicion de Variables */
 int cont;
 int valor;
 int o_actual = 1;
 int opcion_elegida = 0;

 /* Codigo de la funcion  */
```

```
 for (cont=1; cont<=opciones; cont++)
     r_letra(apop[cont].x, apop[cont].y, apop[cont].prompt);

 inverso_r_letra(apop[o_actual].x, apop[o_actual].y,
apop[o_actual].prompt);

 do
 {
   valor = tecla();

    for (cont=1; cont<=opciones; cont++)
      r_letra(apop[cont].x, apop[cont].y, apop[cont].prompt);

   switch( valor )
   {
     case ABAJO    : if (o_actual == opciones)
                     {    o_actual = 1; }
                else
                     {    o_actual++;   }
                break;

     case ARRIBA   : if (o_actual == 1)
                     {    o_actual = opciones; }
                 else
                     {    o_actual--;   }
                 break;

     case INICIO   : o_actual = opciones;
                 break;

     case FINAL    : o_actual = 1;
                 break;

     case ESCAPE   : valor = 27;
                 opcion_elegida = 0;
                 return 0;
                 break;

     case IZQUIERDA: valor = 27;
                 opcion_elegida = 200;
                 return 200;
                 break;

     case DERECHA  : valor = 27;
                 opcion_elegida = 300;
                 return 300;
                 break;

     case ENTER    : valor = 27;
                 opcion_elegida = o_actual;
                 return o_actual;
                 break;


     default:
                o_actual = teclaok(valor,o_actual);
                break;
```

```
    }
    inverso_r_letra(apop[o_actual].x, apop[o_actual].y,
apop[o_actual].prompt);
  } while ( (valor != 27) );
}


/* esta funcion posisiona el cursor dentro de la posision */
int teclaok( int valx , int opc_act )
{
 return opc_act;
}

int resultadoopcion = 0;

struct pop_datos menu[5];

void main()
{
 menu[1].x = 1 ; menu[1].y = 7; strcpy( menu[ 1 ].prompt, "Administracion ");
 menu[2].x = 18; menu[2].y = 7; strcpy( menu[ 2 ].prompt, " Busqueda  ");
 menu[3].x = 29; menu[3].y = 7; strcpy( menu[ 3 ].prompt, " Imprecion ");
 menu[4].x = 41; menu[4].y = 7; strcpy( menu[ 4 ].prompt, " Setup  ");
 menu[5].x = 70; menu[5].y = 7; strcpy( menu[ 5 ].prompt, " Salir  ");

 clrscr();
 /* centro en el medio de la pantalla el titulo. */
 centrar(3,"Muestra de programa MENUPOP");
  /* espero que se precione una tecla. */

 /* llamada a la funcion de menupop */

 resultadoopcion = menu_v(5,1, menu);

 clrscr();
 printf("Opcion elegida %d", resultadoopcion);
 getch();
}
```

Punteros :

C permite el manejo de punteros a memoria de una forma fácil y ágil por medio de punteros podremos desarrollar funciones que no devuelvan un solo valor sino que devuelvan mas valores a su misma vez. solamente tenemos que tener en cuenta un par de cosas y podemos acceder a posiciones de memoria de las variables apuntado hacia ella. :->

& dirección hacia donde apunta el objeto o variable

* contenido de la dirección.

Fíjate en el ejemplo suma3 de introducción a las funciones
Ahora vamos a hacerlo con punteros ;=)

```
Void suma3( int *valor)
{
 *valor += 3; // valor = valor + 3
}
```

Una función que invierte el contenido de dos variables con punteros por supuesto

```
void invertir( int *val1, int *val2)
{
      int temp = *val1;
      val1 = *val2;
      val2 = temp;
}
```

O para realizar una función que devuelva una cadena en mayúscula o minúscula.

```
void mayúscula( char *cadena[])
{
 int cont;        /* definición de variables */
 char local[255]; /* variable de tipo local de texto */
 strcpy(local, cadena);
  for (cont = 0; local[cont] != '\0'; cont++)
  {
   local[cont] = toupper(local[cont]);
  };
 strcpy(cadena, local);
};

void minúscula( char *cadena[])
{
 int cont;        /* definición de variables */
 char local[255]; /* variable de tipo local de texto */
 strcpy(local, cadena);
  for (cont = 0; local[cont] != '\0'; cont++)
  {
   local[cont] = tolower(local[cont]);
  };
 strcpy(cadena, local);
};
```

Ahora usaremos un ejemplito mas de punteros y una función llamada QSORT que permite ordenar un arreglo en memoria es una función recursiva que se ejecuta hasta que se complete el ordenamiento. Este ejemplo ordena una cadena de nombres

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <conio.h>
int ordenar( const void *a, const void *b);
```

```
char lista[5][9] = { "Pepe    ", "Carlos  ", "Luis    ", "Andres  ",
          "Paco   " };

void
 main(void)
{
   int  x = 0;
   clrscr();
   printf("\n Lista sin Ordenar \n");
   for (x = 0 ; x < 5; x++);
       printf("%s\n",lista[x]);

   qsort((void *)lista, 5, sizeof(lista[0]), ordenar);

   printf("Lista Ordenada :\n");
   for (x = 0; x < 5; x++)
      printf("%s\n", lista[x]);

}

int ordenar( const void *a, const void *b)
{
   return( strcmp((char *)a,(char *)b) );
}
```

Complicando los punteros... ☺
Yo puedo apuntar un puntero a una posicion de memoria como la de video, y guardar las mismas para realizar una función que guarde y restaure la pantalla **"cool example"**

```
#include <stdio.h>
#include <conio.h>
#include <string.h>
#include <dos.h>
#include <stdlib.h>

#define screen (* screen_ptr )

/* Definicion de estructuras del programa para la pantalla */
typedef struct texel_struc {
                        unsigned char ch;   /* Caracter de pantalla */
                        unsigned char attr; /* attributo de pantalla */
                      } texel;

typedef texel screen_array[25][80];
screen_array far *screen_ptr = (screen_array far *) 0xB8000000L;

struct texel_struc pantalla[25][80];

/* Salva todo el contenido de la pantalla en memoria */
void salva_pantalla( void )
{
 int x, y;
 for ( x = 1 ; x <= 80; x++ )
   for ( y = 1 ; y <= 25 ; y++ )
      {
       pantalla[ y ][ x ].ch   = screen[ y ][ x ].ch;
```

```
       pantalla[ y ][ x ].attr = screen[ y ][ x ].attr;
      }
}

/* Restaura el contenido de la pantalla previamente salvada con
salva_pantalla */
void restaura_pantalla( void )
{
 int x, y;
 for ( x = 1 ; x <= 80; x++ )
   for ( y = 1 ; y <= 25 ; y++ )
      {
       screen[ y ][ x ].ch   = pantalla[ y ][ x ].ch;
       screen[ y ][ x ].attr = pantalla[ y ][ x ].attr;
      }
}

void main()
{
 /* lo que se guarda en memoria */
 clrscr();
 printf("\n\t\t Muestra de grabacion de pantalla ");
 printf("\n\t\t  precione una tecla para continuar .....");
 /* salvo la pantalla en memoria */
 salva_pantalla();
 getch();
 /* borro la pantalla y espero que se precione una tecla */
 clrscr();
 getch();
 /* restauro la pantalla guarda en memoria */
 restaura_pantalla();
 getch();
}
```

Más complicado punteros:

Cuando yo defino un arreglo, tengo 2 tipos los más usados, estáticos (son aquellos que nunca varían su tamaño no se puede modificar los mismos) y dinámicas (son aquellos que varían según él las necesidades del programa).

Estaticos:
n > max de memoria.
Arreglo estático tantos elementos como se definan

1	2	3	4	5

Desventajas

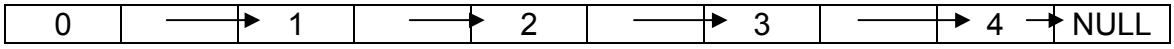
- No se puede modificar el tamaño del vector o arreglo
- Desperdicio de memoria al ser estático no se puede aprovechar al máximo

Ventajas

- Acceso directo hacia los elementos de la lista.

- Almacenamiento continuo de los datos. v[1],v[2],v[3],v[4]....

Dinamicos :



Desventajas :
- No es sencillo poder implementarlo tenes que pensarlo 8=)
- El acceso a las posiciones de memoria no es lineal
- Almacenamiento no continuo.

Ventajas
- Facilita a la asignación de memoria y la liberación de la misma.
- Mayor aprovechamiento de la memoria.

Manejo de archivos en C

Para el manejo de archivos en C tenemos que tomar en cuenta varias partes primero la parte de apertura , el modo de trabajo y luego cerrar el archivo (necesario para no perder datos o que el archivo quede en memoria corrupto).

Al abrir una archivo tenemos que tener en cuenta el modo de apertura del mismo si es de lectura, escritura, creación, binario etc.

Primero tenemos que definir la variable que va a contener en el la información de el comportamiento del archivo (si existe, si se produjo un error al grabar al recuperar etc.)
Las variables de tipo archivo se definen como punteros ej.

FILE *archivo

Esto solamente define una variable de tipo de archivo la información de el archivo se encuentra almacenada en un puntero hacia el mismo.

Archivo = fopen (“Nombre_fisico”, “modo”);

En el modo de apertura del archivo podemos tener en cuenta si el archivo se quiere crear (w) si el archivo es de solo lectura (r) modo binario (b) etc.

Modo de apertura de archivos.

R	Solo lectura
W	Escritura desde el comienzo del archivo (Crearlo tambien)
A	Escritura añadida al final del archivo.
r+	Lectura y escritura

W+	Escritura y lectura
Rb	Solo lectura de archivo Binario.
Wb	Escritura desde el comienzo de archivo binario
Ab	Escritura añadida al final del archivo binario

La función devuelve 2 posibles resultados a la apertura del archivo NULL si no se pudo realizar la tarea y otro que es distinto de 0 que si se realizo correctamente la apertura del archivo.
Hagamos un ejemplo para abrir un archivo y mostrar el contenido del mismo por pantalla.

.. Incluir archivos de cabecera.

```
FILE *archivo
char nom_archivo[] = "C:\Autoexec.bat";
char let; /* es el caracrter que se lee del archivo.*/
main()
{

  archivo = fopen("Autoexec.bat","r"); /* abro el archivo para
lectura. */
  if (archivo == NULL)  // compruebo si el archivo fue bien
abierto.
  {
    printf("Error al abrir el archivo");
   exit(1);
  }
else
  {
    while (!feof(archivo))  /* mientras no sea fin de archivo */
     {
        let = fgetc(archivo);
        printf("%c",let);
     };
  }
}
```

función feof(archivo)

Permite saber en el lugar del archivo esta el puntero si es al final o no, esta devuelve el resultado 0 si se encuentra en el final de archivo o #0 si se encuentra en cualquier otra parte del mismo.

Función int = fgetc(archivo)

Lee del archivo un carácter y lo almacena en una variable de tipo entero yo en el ejemplo anterior almacene el contenido de la variable dentro de una variable de tipo char esto se debe que almacene el valor ascii de la variable leída en la variable carácter.

Función fputc(carácter, archivo)

Permite agregar o modificar datos dentro del archivo siempre y cuando el archivo haya sido abierto para modificación y agregar datos al final.

```
.. incluir librerias...
FILE *arch
Char Texto[] = "Programa de prueba";
Int cont = 0;
Void main()
{
  if (( arch = fopen("prueba","w+")) == NULL)
   {
      printf("Error al crear el archivo");
      exit(1);
   }

  for (cont = 0; texto[cont] !=  '\n' ; cont++)
   {
     fputc(texto[cont], arch);
   }
 fclose(arch);
}
```

Encriptemos un archivo de texto juntando los ejemplo feof, fgetc, fputc la técnica para el mismo es leemos un carácter del archivo origen y a su valor ascii le agregamos 20 y ya esta encriptado el carácter después lo guardamos en un archivo destino.

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

#define MIARCHIVO "C:\autoexec.bat"
#define ENCRIPTADO "C:\autoexec.enc"
#define MASCARA 20

FILE *in, *out;
char letra, letra_enc =' ';

void main()
{
  // apertura y verificacion de archivos.
  if (( in = fopen(MIARCHIVO, "r")) == NULL )
    { printf("abrir el archivo a encriptar");
      exit(0);
    };
  if (( out = fopen(ENCRIPTADO, "w")) == NULL )
    {
      printf("Error al crear el archivo destino");
      exit(1);
    };

  while (!feof(in)) // mientras no sea final del archivo de entrada
   {
```

```
    letra = fgetc(in);  // leo la letra del archivo
    letra_enc = letra ^MASCARA; // encripto la letra
    fputc(letra_enc,out);  // la coloco en el archivo de salida
   }

  fclose(in);
  fclose(out);

}

/*
dentro del while podria haberme ahorado codigo y variables poniendo todo
en una misma
lienea

 while (!feof(in)
  {
   fputc(fgetc(in)+MASCARA , out);
  }

*/
```

Pequeño editor de texto que permite agregar texto de corrido hasta que se presione escape para salir sin grabar o Control –Z para grabar y salir

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

#define OR &&
#define AND ||
#define NOT !=

#define CRT_Z 26
#define ESC 27

FILE *temp, *arch;


/* esta función devuelve el valor ascii de una tecla presionada */
int tecla()
 {
  int itec = getch(); // tecla local.
  if ( itec == 0)
     { itec = getch() + 300; }
  return itec;
 };

void grabar()
{
 char nom_arch[255];
 char leido =' ';
 printf("\n Nombre del archivo a grabar :");
 gets(nom_arch);
 if (( arch = fopen(nom_arch,"w" )) == NULL)
```

```
  {
   printf("Error de grabacion se cancela el programa");
   exit(1);
  }
 else
  {
   temp = fopen("temp.tem","r");
   while (!feof(temp))
   {
    leido = fgetc(temp);
    fputc(leido,arch);
   };
  }
}


void main()
{
 char tec = ' ';
 // creo un espacio temporal para el uso de el archivo.
 if (( temp = fopen("Temp.tem","w")) == NULL )
  {
   printf("Error no hay espacion de retorno");
   exit(1);
  };

 while ( ( tec != ESC) && (tec != CRT_Z) )
  {
   tec = tecla();
   if ( (tec != ESC) OR (tec != CRT_Z) )
    { printf("%c",tec);
      fputc(tec,temp);
    };
   }
 fcloseall();
 if (tec == CRT_Z ) grabar();

 clrscr();
 printf("Gracias por usar EDITOR 1.0\n\n");
}
```

veamos otras funciones para el manejo de archivo .

int remove(const char *nombre_archivo)

Permite remover un archivo de disco pasado como parámetro el nombre del archivo, el resultado devuelto es 0 con éxito o <> de 0 si fallo en el intento.

Int rename(char *nombre_viejo, char nombre_nuevo)

Cambia el nombre de un archivo y le asigna un nombre nuevo al archivo es muy parecido con la función REN del dos. Los resultados devueltos son 0 si se realizo con éxito y <> 0 si ocurrió un error

FILE *tmpfile(void)

Crea un archivo temporal que se mantiene vivo hasta que se cierra el programa modo de apertura del archivo es “wb+” retorna el puntero NULL si no se puede crear el archivo temporal.

En la demostración del programa siguiente se usan las funciones de **putc** que permite especificar la salida
Void putc(carácter, salida)
Para salida existen
Salidas posibles

stdin	Estándar dispositivo de entrada
stdout	Dispositivo de salida (pantalla)
stdprn	Impresora
stderr	Estándar salida de error

Ejemplo de imprimir una linea por impresora y pantalla

```
/* Ejemplo de getc */
#include <stdio.h>
void main()
{
 char linea[] = "Muestra de imprecion por pantalla e impresora";
 int cont = 0;
 while (linea[cont]);
  {
   putc(linea[cont++],stdprn); // mando a la impresora
   putc(linea[cont],stdout);   // mando a la pantalla
  };
}
```

este ejemplo es de un programa para manejo de archivo secuencial que permite mostrar una archivo ascii por pantalla o impresora, pasando los parámetros correspondiente de la linea de comando del dos.
Cuando se preciona la tecla de control + c se autoejecuta una funcion muy util para controlar esta secuencia de teclas. Antes de ver el programa original se muestra un ejemplo de esta funcion simple.

```
#include <stdio.h> /* inclusion de librerias */
#include <dos.h>
#define ABORT 0    /* cosntante para el programa */
int c_break(void)
{   printf("se preciono Control-C . programa finalizado \n");
   return (ABORT);
}
void main(void)
{   ctrlbrk(c_break);
   for(;;)  /* bucle infinito */
   { printf("Looping... Press <Ctrl-Break> to quit:\n"); }
}
```

Comienzo del programa maestro. Espero os guste a mi me dio un trabajo impresionante O:=)

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <string.h>
#include <bios.h>
#include <dos.h>

#define A_BINARIO  "rb" /* Apertura en forma binaria y de lectura */
#define ENTER   13      /* Constantes del teclado */
#define ESCAPE  27
#define NULL     0
#define ABORT    0
                 // constantes para la impresora

#define IMP_OUT   0x01 //Impresora OK
#define IMP_ERRO 0x08 //I/O error
#define IMP_SLEC   0x10 //Selected
#define IMP_OUTP   0x20 //Sin Papel
#define IMP_DESC   0x40 //Desconosido error
#define IMP_NORW  0x80 //No responde
#define PUERTO    0 // Equivalente a LPT1
#define STATUS    2 // Para recopilar informacion del estado de la
impresora

/* Mensaje de ayuda del programa */
void help( void );

/* Chequea el estado de la impresora */
void chkimpresora( void );

/* permite cortar los listados con Ctl + C */
void breaklistado( void );


FILE *archivo;
char let_in, let_out;


void main( int argv, char *argc[] )
{
 clrscr();
 breaklistado(); /* Esta funcion se autoejecuta cuando se presiona
                CTRL + C  no importa en que parte del programa
                se este ejecutando */
 switch (argv)
  {
    case 3:
         if ( ( strcmp (argc[1],"-?"   ) == 0) ||
            ( strcmp (argc[1],"-help") == 0) ||
            ( strcmp (argc[1],"-h"   ) == 0) ||
            ( strcmp (argc[1],"-H"   ) == 0)  )
            { help(); /* llamada la funcion de ayuda */ }
         else
```

```
          {
           if ( ( archivo = fopen(argc[1],A_BINARIO) ) == NULL )
          {
           printf("\nError de apertura de archivo STOPED");
           printf("\nVerifique que el archivo [ %s ] exista ",argc[1]);
           printf("\nConsulte el manual de usuario con -? para ayuda");
          }
          else
           {
            if ( !( ( strcmp( argc[2],"-C") == 0) ||
                 ( strcmp( argc[2],"-c") == 0) ||
                 ( strcmp( argc[2],"-P") == 0) ||
                 ( strcmp( argc[2],"-p") == 0) ))
               {
                printf("El archivo fue abierto con exito\n");
                printf("pero tiene un error en los parametros\n");
                printf("Parametro incorrecto consulte la ayuda\n");
               }
             else
               {
                if ( (strcmp(argc[2],"-p") == 0) )
                 {
                  // mandar a imprimir
                  printf("Imprimiendo archivo .....");
                  while (!feof(archivo)) /* no fin de archivo */
                   {
                     let_in = fgetc(archivo); /* leo un caracter */
                     chkimpresora();
                     putc(let_in,stdprn);
                     /* putc permite redireccionar la salida por
                      el puerto seleccionado la impresora o la
                      pantalla
                     */
                   }
                   printf("OK");
                 }
                else
                 {    // Mostrar el archivo por pantalla

                  while (!feof(archivo)) /* no fin de archivo */
                   {
                           delay(10);
                     let_in = fgetc(archivo); /* leo un caracter */
                     putc(let_in,stdout);
                     /* para mostrar por pantalla no uso printf
                      uso putc que me permite especificar la
                      consola de salida de los datos */
                   }
                 };
               };
             fclose( archivo ); /* Cierro el archivo */

           }
          }
       break;
  default:
      printf("      Error no se paso los parametros correctos \n");
```

```
        printf(" Verifique el manual de usuario o consulte la ayuda
?\n");
        printf("      Ej: listar -?  o listar -Help  o listar -H\n");
        printf("      Vercion 1.0 Shaware FefoRena@hotmail.com\n");
        break;
  }
  getch();
}


/* Permite visualizar el texto por pantalla de ayuda */
void help( void )
{
 /* textcolor( color ) permite seleccionar un nuevo color de texto
    pero para poder visualizar el texto con nuevo color se debe ejecutar
    en vez de printf se usa cprintf
 */
 textcolor(BLUE);
 cprintf("\n             -= Programa Share de Listar  Vercion 1.0 =-");
 printf( "\n Este programa permite el listado de archivos por pantalla o
impresora");
 printf( "\n modo de uso:");
 printf( "\n      \Listar.Exe [Nombre de archivo] [Parametros] ");
 printf( "\n Parametros");
 printf( "\n\t-C => Mostralo por pantalla");
 printf( "\n\t-P => Mandarlo por la impresora");
 printf( "\n\t-? => Consultar la Ayuda");
 printf( "\n\t <Ctrl + Break> para cancelar el programa");
};

void chkimpresora( void )
{

  #define STATUS  2    /* printer status command */
  #define PORTNUM 0    /* port number for LPT1 */

   int status, abyte=0;
   status = biosprint(STATUS, abyte, PORTNUM);
   switch (status)
    {
     case 0x01 :
            cprintf("Error al imprimir : \n");
            printf (" Impresora apagada prendala y pruebe nuevamente");
            break;
     case 0x08 :
            cprintf("Error al imprimir : \n");
            printf (" No existe comunicacion con el printer I/O error");
            break;
     case 0x20 :
            cprintf("Error al imprimir : \n");
            printf(" Impresora sin papel coloque papel para continuar");
            printf(" Presione una tecla para continuar imprimiendo");
            getch();
            break;
     case 0x80 :
            cprintf("Error al imprimir : \n");
            printf(" La impresora no responde pruebe de nuevo");
```

```
            break;
   }
};

/* la funcion ctrbrk permite pasarle una funcion de comandos
   cuando se le llama .-  O:=) */
int c_break( void )
  {
    printf("\n ABORTANDO");
    printf("\n\n Programa abortado.!.");
    sound(7);   /* pequeñito sonido */
    delay(100);
    nosound();
    return(ABORT);
 };

void breaklistado( void )
{
  ctrlbrk( c_break );
}
```

Además de las funciones vistas tenemos también definidas macros para el momento de compilación como por ejemplo

Macro __LINE__ devuelve el numero de línea que se esta ejecutando.

Macro __FILE__ devuelve el nombre del archivo compilado

Macro __DATE__ fecha de compilación del programa

Macro __TIME__ hora de compilado el programa.

Para el manejo de estructuras no para el manejo de archivos secuenciales tenemos varias funciones diferentes dentro de C como por ejemplo la grabacion de registros levantar registros y posesionares dentro del archivo.

Función fwrite (&variable, tamaño , cantidad, puntero)

La función permite guardar datos dentro de un archivo en forma secuencial
Variable = tipo int, char, estructura, etc.
Tamaño = en byte que ocupa la variable ej. una variable tipo (int ocupa 2 byte)
Cantidad = cantidad que repetir la grabacion por ejemplo 1.
Puntero = archivo al cual grabar la estructura.

Función fread(&variable, tamaño, cantidad, puentero)

La función permite leer datos dentro de un archivo indicado y los parámetros son iguales al otro

Estas funciones graban datos o leen datos a partir de la posición del puntero dentro del archivo en adelante.

Funcion fseek(puntero, tamaño_variable * nposisiones, posision_referencia)
Permite el movimiento dentro del archivo a partir de una referencia de posición
Puntero = es el stream del archivo.

Tamaño de la variable * nposisiones = tamaño de la estructura * la cantidad de posiciones que queremos que se desplace hacia delante.
Estas son las constantes de posición dentro del archivo.

Constante	Valor	Posición dentro del archivo
SEEK_SET	0	Principio del archivo
SEEK_CUR	1	Posición actual
SEEK_END	2	Final de archivo

Función entero = SIZEOF(variable o estructura)
También en estos casos es muy útil usar una función que devuelva el tamaño de bytes una estructura o de una variable en vez de estar sumando los valores de cada elemento de la estructura.

Ejemplo si yo tengo una estructura con los siguientes campos
Struct registro {
Int numero;
Char nombre[10];
Char teléfono[10];
}
para saber el tamaño de esta estructura tendría que sumar

Variable	Tamaño
Int	2 byte
Char	1 byte * 10
Char	1 byte * 10
Total	22 byte

Si nosotros usamos la función sizeof no tenemos que sumar todos las variables solamente le pasamos el registro y ya esta. Esto nos evitaría problemas en nuestro programa cuando tengamos que modificar estructuras no tendremos que modificar cada vez que leamos un registro o escribamos el tamaño de los parámetros.

Ejemplo completo de un programa de muestra que permite agregar datos a una base de clientes con todas la funciones vistas y con muchas funciones escritas.

```
/* Ejemplos de funciones */
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
```

```
#include <string.h>
#include <ctype.h>
#include <dos.h>

/* defino un nuevo tipo de variable llamado registro el
      contiene los campos de la estructura */
struct registro {
            int numero;
             char nombre[10];
             char direcc[10];
             char telefo[10];
            };

/* uso el nuevo tipo de variable y defino registro */
struct registro r_cliente; /* registro para la datos */
struct registro e_cliente; /* registro para la edicion */


/* defino el puntero al archivo, ya abriendolo. */
FILE *archivo;

/* Funciones para el archivo */

/* esta funcion crea el archivo de 0 o sobre escribe el actual */
void crear_archivo( void )
{
 char tec = ' ';
 printf("Desea crear un nuevo archivo (S/N)");
 while ( (tec != 'S') && (tec != 'N') )
  {
   tec = toupper( getch() );
   /* toupper si es una letra minuscula la pasa a mayuscula */
  };
 if (tec == 'S')
   {
     if (( archivo = fopen("clientes.dat","w")) == NULL )
      { printf("el Archivo fue Creado con exito\a"); }
     else
      { printf("Error no se pudo crear el archivo"); };

   }
};


/* esta funcion abre el archivo pasando como parametro el directorio */
void abrir_archivo( void )
{
 if (( archivo = fopen("clientes.dat", "a+")) == NULL )
  {
    printf("\n\tError al intentar abrir el archivo\n");
    crear_archivo();
  };
};


void cerrar_archivo( void )
{
```

```
 fcloseall(); /* cerra todos los archivos abiertos */
};

/*    FUNCIONES de movimiento dentro del archivo.     */
/* funcion que se para en el primer registro de la base */
void ir_primero( void )
{
 fseek(archivo,sizeof(r_cliente), SEEK_SET);
};


/* funcion mover 1 adelante */
void ir_adelante( void )
{
 fseek(archivo,sizeof(r_cliente), SEEK_CUR);
};


/* funcion ir al ultimo registro */
void ir_ultimo( void )
{
 fseek(archivo,sizeof(r_cliente), SEEK_END);
};


/*funcion  lee un registro de la base */
void leer_registro( void )
{
  fread(&r_cliente, sizeof(r_cliente),1, archivo);
};


/* funcion que permite guardar un registro en la base */
void guardar_registro( void )
{
  fwrite(&r_cliente, sizeof( r_cliente),1,archivo);
};


int editar_registro( void )
{
 char tec =' ';
 clrscr();
 printf("\nNombre    :"); gets(r_cliente.nombre);
 printf("\nDireccion :"); gets(r_cliente.direcc);
 printf("\nTelefono  :"); gets(r_cliente.telefo);
 printf("\n Guardar Datos (S/N)");

 while ( (tec !='S') && (tec != 'N') )
  { tec = toupper( getch() ); };
 if (tec == 'S') return 1;
 else return 0;
};

int menu( void )
{
 char mopc;
```

```
 clrscr();
 printf("Menu de Opciones \n\t 1) Agregar \n\t 2) Modificar");
 printf("                 \n\t 3) Ver     \n\t 4) Salir");
 mopc = 0;
 while ( (mopc != '1') && (mopc != '2') && (mopc != '3') && (mopc != '4')
)
  {
   mopc = getch();
//   sound(1000); /* hacer un beep */
//   delay(300);
//   nosound();
  };
  switch (mopc)
   {
    case '1' : return 1;
    case '2' : return 2;
    case '3' : return 3;
    case '4' : return 4;
   }
 return 0;
};

void main()
{
 int tecl,bandera,cont,opc = 0; /* definicion de variables locales. */
 struct registro ir_cliente;


 clrscr();
 abrir_archivo();

 while (opc != 4)
  {
   opc = menu();
   switch (opc)
    {
     case 1: if ( editar_registro() == 1)
              guardar_registro();
             break;

     case 2: printf("\n\tCliente a Modificar :");
              bandera = 0;
             gets(ir_cliente.nombre);
             fseek(archivo,0,SEEK_SET);
             while (!feof(archivo) && (bandera != 1))
              {

               leer_registro();
               if ( strcmp(r_cliente.nombre, ir_cliente.nombre) == 0 )
                bandera = 1;
              }
              if (bandera == 1)
               {
                   clrscr();
               printf("\t\t Cliente Encontrado");
                   gotoxy(5,4);
               printf("Nombre    :%s",r_cliente.nombre);
```

```
                gotoxy(5,5);
              printf("Direccion :%s",r_cliente.direcc);
              gotoxy(5,6);
              printf("Telefono  ;%s",r_cliente.telefo);
                  gotoxy(7,2);
              gotoxy(16,4); gets(r_cliente.nombre);
              gotoxy(16,5); gets(r_cliente.direcc);
              gotoxy(16,6); gets(r_cliente.telefo);
              printf("Cambios realizados ...");
              guardar_registro();

              }
             else
              {
             printf("Cliente No encontrado");
             getch();
                }

            break;
     case 3: clrscr();
            printf("\t\tLISTADO DE CLIENTES\n");
            fcloseall();
              archivo = fopen("clientes.dat","a+");
            while (!feof(archivo))
             {
                cont = 0;
              printf("\t\tLISTADO DE CLIENTES\n");
              printf(" NOMBRE         DIRECCION          TELEFONO");
              while ( (cont != 20) && ( !feof(archivo)) )
             {
                    cont++;
                  leer_registro();
             printf("\n[ %20s ] [ %20s ][ %20s
]",r_cliente.nombre,r_cliente.direcc,r_cliente.telefo);
             };
                 gotoxy(20,23); printf("Precione una tecla para continuar
. . . .");
                 getch();
              };
              break;
    };
  };
 cerrar_archivo()
}
```

Ejecutando el aplicaciones para el dos.-

En esta seccion se tratar utilidades para el dos como por ejemplo la creación de un programa que diga si el disqute esta insertado en la disqutera ejecutar un comando desde un programa , buscar un archivo en el disco duro etc.

Veamos el siguiente programa que permite ejecutar un comando del dos desde el mismo programa util para crear aplicaciones y llamadas a otros programa, en el incluimos una nueva librería que es la de process.h en ella se encuentra unas

funciones interesantes como son EXIT que permite terminar el programa abrutamente retornando codigo de error al dos por si se usa archivos BATCH (archivos del viejo y querido dos)

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

char comando[255] = "";

void main()
{
 clrscr();
 printf("Ingrese el comando a ejecutar :");
 gets(comando);
 if ( strcmp(comando,"") == 0)
  {
    printf(" No hay comando a ejecutar !!");
    exit(1); // salio del programa y comunico al dos error 1
  }
 else
  {
    printf("Ejecutando %s\n",comando);
    system(comando);
    exit(0);  // salio del programa sin problemas
  }
}
```

El corazon del programa se encuentra en system que permite ejecutar el comando ej system("Dir"); es lo mismo que poner de la linea de comandos DIR

El sigiente programa muestra los archivos del directorio que le especifiquemos ademas muestra el tamaño, fecha, hora y atributos.

```
#include <stdlib.h>
#include <stdio.h>
#include <dir.h>
#include <conio.h>
#include <string.h>

#define NO !
void main(void)
{
   struct ffblk ffblk;
        /* La estructura siguiente contiene attributo, hora, fecha
           tama¤o (en long) se encuentra en la biblioteca DIR */
   int hay;
        /* Esta variable almacena si existen mas archivos     */
   char filtro[11];
        /* Filtro para filtrar los archivos                   */

   clrscr();
   printf("Ingrese el filtro de archivos a visualizar :");
```

```
   scanf("%s", filtro);
   /* busca el primer archivo que cumpla con la condicion */
   /* retorna 0 si no encontro ninguno y 1 si lo encontro */
   hay = findfirst(filtro,&ffblk,0);

   if ( hay )
      { /* si no se encuentran archivos a listar se sale
           del programa */
       printf("No se encontraron archivos a listar !!");
       getch();
       exit(1);
      }
   /* se mantiene listando mientras existan archivos que cumplan
      la condicion de filtro. */
   while ( NO hay )
   {
      printf("[ %11s ] [ %d %s ] \n",ffblk.ff_name,
           ( (ffblk.ff_fsize < 1024)  /* Tama¤o en byte o kbyte*/
             ? (ffblk.ff_fsize) : (ffblk.ff_fsize / 1024) ) );
      /* dentro de este printf inclui  un if logico (comenzando con las
       macritos 8=D espero les guste */


      hay = findnext(&ffblk);
   }
   getch();
}
```

MOUSE XD

La programación del mouse esta lindo por cierto puesto que podemos realizar funciones con mas potencia como menu que selecciones las opciones con un clic del mismo, marcar texto, crear juegos etc.
Primero veremos la libreira del mouse completa anda que si queres podes copiarte el texto a una archivo y te queda completa . No son muchas lineas pero es medio difícil de entenderla yo no la entiendo toda solamente la copie y la uso mejorando un poco las funciones :

```
/*
   Libreria para el manejo del mouse
   Creador Federico Rena
   Para Programadores Asociados    Montevideo, Uruguay
   Lenguaje C++

*/

typedef struct {
             int m1, m2, m3, m4;
             } mparams; /* parametros para la llamada de mouse */

void mouse( mparams *p); /* funcion llamada por todas las demas */
int m_inic( void );
void m_sicursor( void );
void m_nocursor( void );
void m_estado ( int *boton, int *xact, int *yact );
void m_posicion( int x, int y );
```

```
void m_botpres( int btn, int *bathora, int *bcuenta, int *x, int *y );
void m_hlimites( int izq, int der );
void m_vlimites( int arr, int abj );
/* void m_modograf( int hrefe, int vrefe , void far *cmasc ); */
void m_velocidad( int x, int y );
void m_subrutina( int ,void (far *)(void) );
void m_botlib(int ,int *,int *,int *,int * );
void m_texto(int, int, int );
void m_desplaz( int *,int * );
void m_nocond( int ,int ,int ,int );

/*********************************************************
 desarrolo de las rutinas de Raton.C
********************************************************* */

#include <dos.h>

mparams *cola, mp;
void mouse( mparams *p)
{
 union REGS mousreg;
 mousreg.x.ax = p->m1;
 mousreg.x.bx = p->m2;
 mousreg.x.cx = p->m3;
 mousreg.x.dx = p->m4;
 int86(0x33, &mousreg, &mousreg );
 /* Llamar a la interupcion del mouse */
 p->m1 = mousreg.x.ax;
 p->m2 = mousreg.x.bx;
 p->m3 = mousreg.x.cx;
 p->m4 = mousreg.x.dx;
}

int m_inic( void ) /* devuelve si el software esta intalado */
{
 union REGS mreg;

 struct SREGS segs;
 cola = &mp;
 if ( _osmajor < 2)
  return(0);/* Dos 2.0 o mayor para usar las rutinas */
 cola->m1 = cola->m2 = cola->m3 = cola->m4 =0;
 /* inicializar las variables a 0 */
 if (_osmajor >= 3)
    mouse(cola); /* estado devuelto por cola si cola ->m1;si 0,no
instalado */
 else
  {
    mreg.h.al=0x35; /* funcion para dos  para obtener el vector de
interrupcion /*
    mreg.h.al=0x33; /* nro. de la interrupcion del mouse */
    intdosx( &mreg, &mreg, &segs);
    if (segs.es==0 && mreg.x.bx ==0)
      cola->m1=0; /* si el valor de apertura a 0000:0000, no esta el
mouse */
    else
      mouse ( cola );
```

```
  }
  return(cola->m1);
}

void m_sicursor( void )
{
 /* hacer visible el puntero */
 cola->m1=1;
 mouse( cola );
}

void m_nocursor( void )
{
 /* hacer desaparecer el cursor */
 cola->m1=2;
 mouse( cola);
}

void m_estado( int *boton, int *actx, int *acty )
{
 /* devuelve el estado del mouse y la posicion */
 cola->m1=3;
 mouse(cola);
 *boton = cola->m2;
 *actx  = cola->m3;
 *acty  = cola->m4;
}

void m_posicion(int x, int y )
{
 /* coloca el mouse en x,y cord en pixels */
 cola->m1=4;
 cola->m3=x;
 cola->m4=y;
 mouse( cola );
}

void m_botpres( int boton, int *bahora, int *bcuenta, int *x ,int *y )
{
 /* obtener informacion de los botones precionados */
 cola->m1=5;
 cola->m2=boton;
 mouse( cola );
 *bahora = cola->m1;
 /* nro. de veces precionados desde la ultima llamada */
 *bcuenta = cola->m2;

 *x = cola->m3;
  /* posicion horizontal actual */

 *y=cola->m4;
  /* posicion vertical actual */
}

void m_botlib( int boton, int *bahora, int *bcuenta, int *x ,int *y )
{
 /* obtener informacion sobre los botones liberados */
```

```
 cola->m1=6;
 cola->m2=boton;
 /* boton para chequear 0=izq o 1=der */
 mouse( cola );
 *bahora =cola->m1;
 /* estado del boton ahora */
 *bcuenta=cola->m2;
 /* nro de veces librerado desde la ultima llamada */
 *x=cola->m3;
 /* posicion horiz la ultima vez liberado */
 *y=cola->m4;
 /* posicion vertical de ultima vez liberado */
}

void m_hlimites( int izq, int der )
{
 cola->m1=7;
 cola->m3=izq;
 cola->m4=der;
 mouse( cola );
}

void m_vlimites( int arr, int abj )
{
 cola->m1=8;
 cola->m3=arr;
 cola->m4=abj;
 mouse( cola );
}

void m_modotexto( int cursor, int mascpant, int masccurs )
{
 /* Establese el cursor en modo texto : por Hardware o software */
 cola->m1= 10;
 cola->m2= cursor;
 /* 0=cursor software : 1=cursor hardware */
 cola->m3= mascpant; /* si cursor software define la mascara de pantalla
*/
 cola->m4= masccurs; /* si cursor softwaree define la mascara del cursor
*/
 mouse( cola );
}

void m_desplaz(int *x, int *y )
{
  cola->m1=11;
  mouse( cola );
 *x=cola->m3;
 /* dist. horz. recorida desde la ultima vez */
 *y=cola->m4;
 /* dist. ver. reccorida desde la ultima vez */
}

void m_subrutina(int masc, void(far *func)(void))
{
  _AX=12;
  _ES=FP_SEG(func);
```

```
  _DX=FP_OFF(func);
  _CX=masc;
  geninterrupt( 0x33 );
}

void m_velocidad( int x, int y)
{
 /* establece la velocidad de relacion mickey/pixel */
 cola->m1=15;
 cola->m2=x;
 cola->m4=y;
 mouse( cola );
}

void m_nocond(int x1, int y1, int x2, int y2 )
{
 /* esconder el cursor si esta en la region cuando se llama a esta
funcion */
 _AX= 16;
 _SI= x2;
 _DI= y2;
 _CX= x1;
 _DX= y1;
 geninterrupt( 0x33);
}
/*    fin de la libreria de funciones del mouse */
```

/* Programa de ejemplo de la librería raton.h vamos todavia*/

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

/* inclusion de la libreria del mouse -)-)-) */
#include "c:\c\fede\raton.h"

/* estas variables son usadas para mostrar las posiciones de x,y*/
int x, y, boton;

void main()
{

 clrscr();
 if (!m_inic()) {printf("no hay raton"); exit(1);}
  /* formato del texto puede ser grafico tambien */
  m_modotexto(0, 0xfff, 0x7700 );
  /* posicion inicial del mouse */
  m_posicion(16,24);
  /* muestro el cursor en pantalla */
  m_sicursor();
  /* defino la velocidad de movieminto del mouse */
  m_velocidad(15,15);
 /* mientras no se precione una tecla */
```

```
 while (!kbhit())
  {
   gotoxy(10,24);
   m_estado(&boton,&x,&y);
   printf(" X = %d  : Y = %d : BOTON = %d ",x/8,y/8, boton);
  }

 /* oculta el mouse moviendose */
 m_nocursor();
}
```

y sin darse cuenta ya estan pasando a C++ veamos estas funciones de pantalla como son gettext y puttext que permiten guardar la pantalla en una variable y recuperarla luego

Funcion void GETTEXT(x, y, x1 , y1 , variable caracter)
Permite salvar la pantalla en memoria en una variable de tipo carácter en donde se van a almacenar
y poner otra ves la pantalla
PUTTEXT(x, y, x1 , y1 , variable)

estas dos funciones estan incluidas en conio.h la libreria por supuesto pero para c++ no se si para **c**

```
#include <conio.h>

char buffer[4096];

int main(void)
{
   int i;

   clrscr();
   textcolor(RED+ BLINK);
   for (i = 0; i <= 20; i++)
      cprintf("TEXTO #%d\r\n", i);
   gettext(1, 1, 80, 25, buffer);

   textcolor(WHITE);
   gotoxy(1, 25);
   cprintf("presione una tecla para borrar la pantalla.");
   getch();
   clrscr();
   gotoxy(1, 25);
   cprintf("presione una tecla para recuperar la pantalla");
   getch();
   puttext(1, 1, 80, 25, buffer);
   gotoxy(1, 25);
   cprintf("Presione una tecla para salir.");
   getch();
}
```