"2015. Año del Bicentenario Luctuoso de José María Morelos y Pavón"

Centro Universitario UAEM Zumpango

Ingeniería en Computación

Unidad de Aprendizaje:
Organización de Archivos

Unidad de Competencia III:
Reconocer y manejar archivos directos

.

MTE-MI. Rosa Erendira Reyes Luna

Agosto 2015

www.uaemex.mx

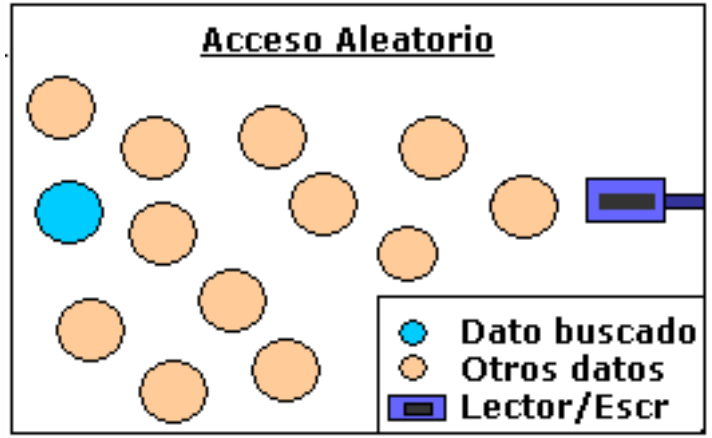
Propósito de la Unidad de Aprendizaje

Que el dicente adquiera la habilidad para aplicar las estructuras en memoria secundaria y la organización de archivos como herramienta para la implementación de los principales algoritmos aplicables a la creación de programas de cómputo y software que de solución a problemas de almacenamiento, procesamiento y acceso de información.

Introducción

- Un archivo de acceso aleatorio es aquel cuya información puede leer o escribir en cualquier orden, sin tener que leer o escribir toda la demás información del archivo. Por añadidura un archivo de acceso directo puede leerse o escribirse de manera indistinta, sin que requiera un modo específico de apertura para tales fines. (Perry, 1999)

- Los archivos directos explotan la capacidad de los discos para acceder directamente a cualquier bloque de dirección conocida. Como en los archivos secuenciales y secuenciales indexados, se requiere un campo clave en cada registro. Sin embargo, aquí no hay concepto de ordenamiento secuencial. (Rojas, 2014)



Unidad de Competencia III

Reconocer y manejar archivos directos

Contenido

1. Definición de Archivos
2. Características
3. Ventajas
4. Desventajas
5. Operaciones sobre Archivo
6. Conclusiones
7. Referencias

Condiciones para que un archivo sea de organización directa:

- Almacenamiento en un soporte direccionable.
- Cada registro debe contener un campo clave.
- Debe haber correspondencia entre los posibles valores de la clave y las direcciones existentes en el soporte.
- Conocer el número de registros que van a almacenar.

DIRECCIÓN ABSOLUTA O RELATIVA

- Es un soporte direccionable, cada posición se realiza por su dirección absoluta, número de pista y número de sector de disco. Los archivos directos manipulan direcciones relativas en lugar de absolutas esto hará al programa independiente de la posición absoluta del archivo en el soporte.

Características

- Registros de longitud fija
- Acceso directo a los registros
- Cantidad máxima de registros fija
- No hay acceso secuencial
- No existe ordenamiento
- Accede a los registros por medio de direcciones
- Uso de soporte direccionable.

(Perry, 1999)

Ventajas

- Los archivos pueden insertarse sin destruir otros datos del archivo.
- Para consultar un registro en específico no es necesario accesar a todos
- Tiempo de acceso optimo
- Los registros pueden actualizarse o eliminarse sin necesidad de reescribir el archivo completo

(Deitel & Deitel, 2004)

Desventajas

- Los datos pueden ser borrados o sobrescritos accidentalmente al menos que se tomen precauciones especiales
- Riesgo de pérdida de precisión y violación de seguridad
- Uso menos eficiente de espacio de almacenamiento
- Actualizar el archivo es más difícil que método secuencial

Operaciones

- Crear archivos
- Consultar registros
- Lecturas de registros
- Actualización
- Reorganización
- Borrado

Creación de un archivos directos en C

- La función **fwrite** transfiere un número específico de bytes que comienzan en una ubicación específica de la memoria hacia un archivo.

```
fwrite (&numero ,sizeof(int) ,1, ptrF);
```

Creación de un archivos directos en C

- La función **fread** transfiere un número específico de bytes desde la ubicación especificada en el archivo por medio del apuntador de posición del archivo

fprintf (ptrF,”%d”, numero);

- Aunque **fread** y **fwrite** leen y escriben datos, de tamaño fijo en lugar de formatos de tamaño variable, los datos que manipulan se procesas en formato fuente de la computadora. (Deitel & Deitel, 2004)

Escritura de un archivo directo

- La función **fseek** establece el apuntador de posición de archivo en una posición específica del archivo, luego **fwrite** escribe los datos. (Deitel & Deitel, 2004)

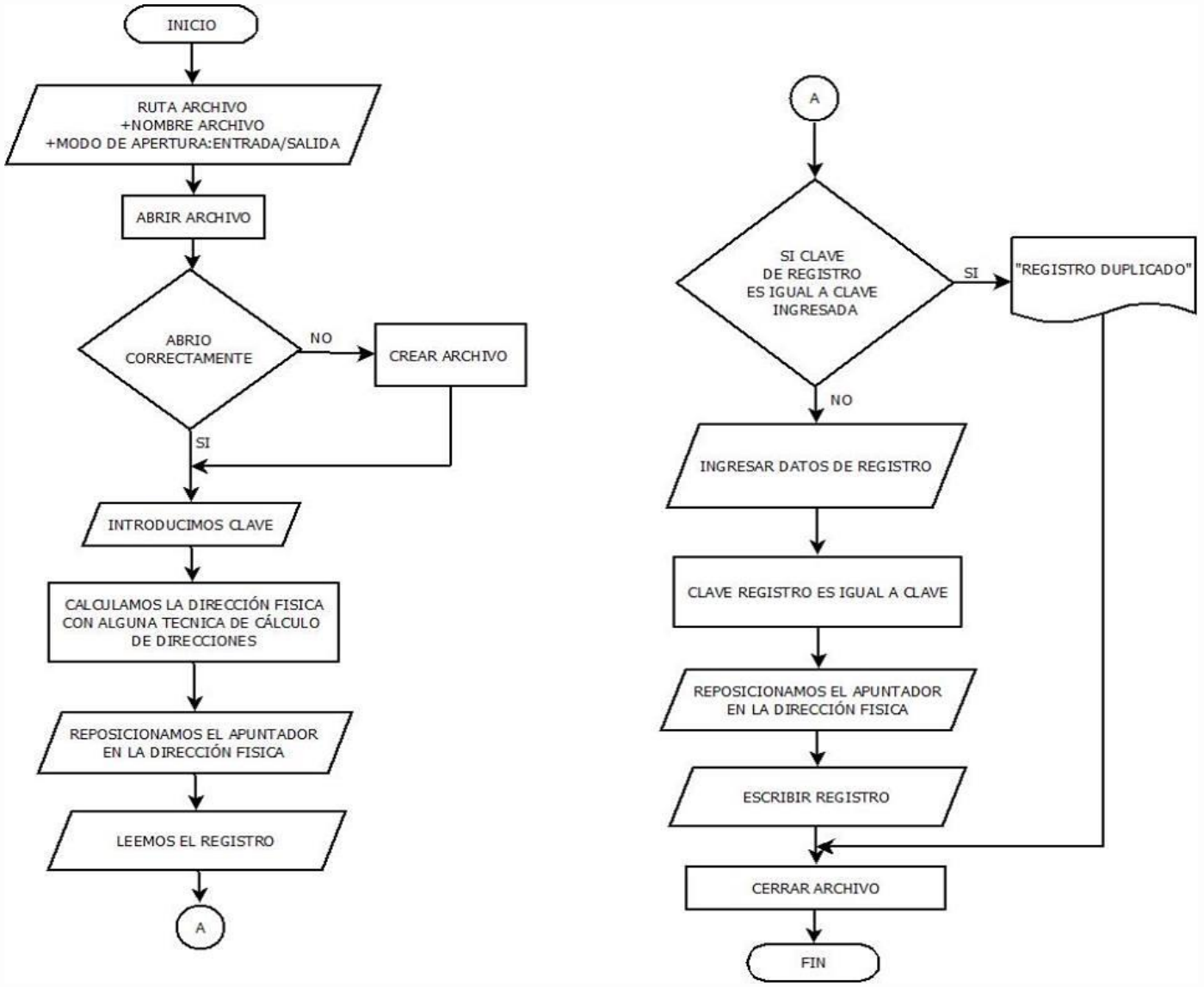
int fseek (FILE* flujo, long int desplazamiento, int en donde);

```
void ALTA_DIRECTA(void)
{
int no_prod; // Variable local para el numero de producto
clrscr();
cout << "\n\r ALTAS DE REGISTROS DE PRODUCTOS";
alias=fopen("PRODUCTO.DIR","rb+"); // Intenta abrir el archivo
//PRODUCTO.DIR
// en modo de lectura/escritura
if(alias==NULL)
alias=fopen("PRODUCTO.DIR","wb"); // Crea el archivo en caso de no existir
cout << "\n\n\n\rNúmero de producto: "; cin >> no_prod;
dir_fisica=no_prod*sizeof(Registro); // Calculo de la Dir. física
```

```
fseek(alias,dir_fisica,SEEK_SET); //Posicionar el apuntador del archivo
fread(&Registro,sizeof(Registro),1,alias);
// Lee el "Registro", de tamano=sizeof(Registro) del archivo "alias"
if(Registro.no_prod==no_prod)
{
cout << "\n\n\n\rRegistro duplicado !!!";
fclose(alias);
getch();
return;
}
cout << "\n\rDescripcion: "; gets(Registro.descrip);
cout << "\n\rCantidad : "; cin >> Registro.cantidad;
cout << "\n\rPrecio : "; cin >> Registro.precio;
do
```

```
{
cout << "\n\rGarantia : "; Registro.garantia=toupper(getche());
}while(Registro.garantia!='S' && Registro.garantia!='N');
Registro.no_prod=no_prod;
fseek(alias,dir_fisica,SEEK_SET); //Posicionar el apuntador del archivo
fwrite(&Registro,sizeof(Registro),1,alias); // Grabar el Registro completo
fclose(alias); // Cierra el archivo
cout << "\n\n\n\rProducto registrado !!!";
cout << "\n\r<<< Oprima cualquier tecla para continuar >>>";
getch();
return 0;
}
```

Escritura


INICIO
RUTA ARCHIVO
+NOMBRE ARCHIVO
+MODO DE APERTURA:ENTRADA/SALIDA
ABRIR ARCHIVO
ABRIO CORRECTAMENTE
NO
CREAR ARCHIVO
SI
INTRODUCIMOS CLAVE
CALCULAMOS LA DIRECCIÓN FISICA CON ALGUNA TECNICA DE CÁLCULO DE DIRECCIONES
REPOSICIONAMOS EL APUNTADOR EN LA DIRECCIÓN FISICA
LEEMOS EL REGISTRO
A
A
SI CLAVE DE REGISTRO ES IGUAL A CLAVE INGRESADA
SI
"REGISTRO DUPLICADO"
NO
INGRESAR DATOS DE REGISTRO
CLAVE REGISTRO ES IGUAL A CLAVE
REPOSICIONAMOS EL APUNTADOR EN LA DIRECCIÓN FISICA
ESCRIBIR REGISTRO
CERRAR ARCHIVO
FIN

Lectura de un archivo directo

- La función **fread** lee un número específico de bytes de un archivo en memoria.

```
fread(&cliente, sizeof (struct datosCliente), 1, ptrCf);
```

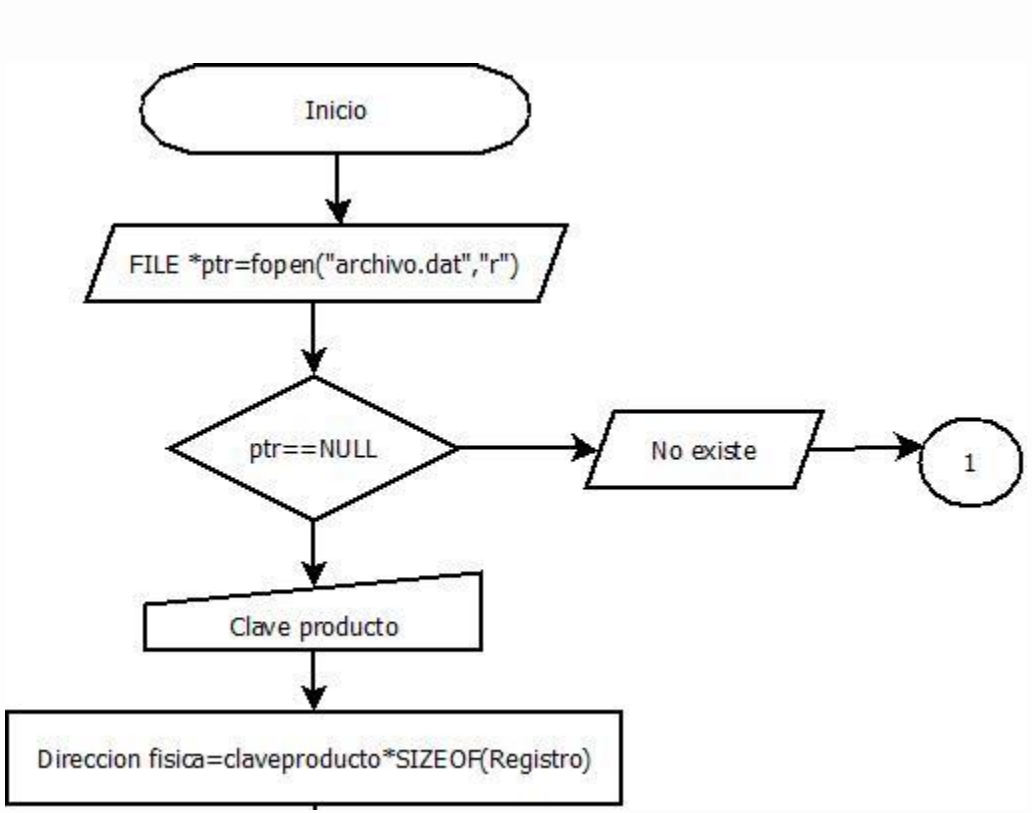
- La función **fread** puede utilizarse para leer varios elementos de tamaño fijo del arreglo, al proporcionar un apuntador al arreglo en el cual se almacenan los datos.

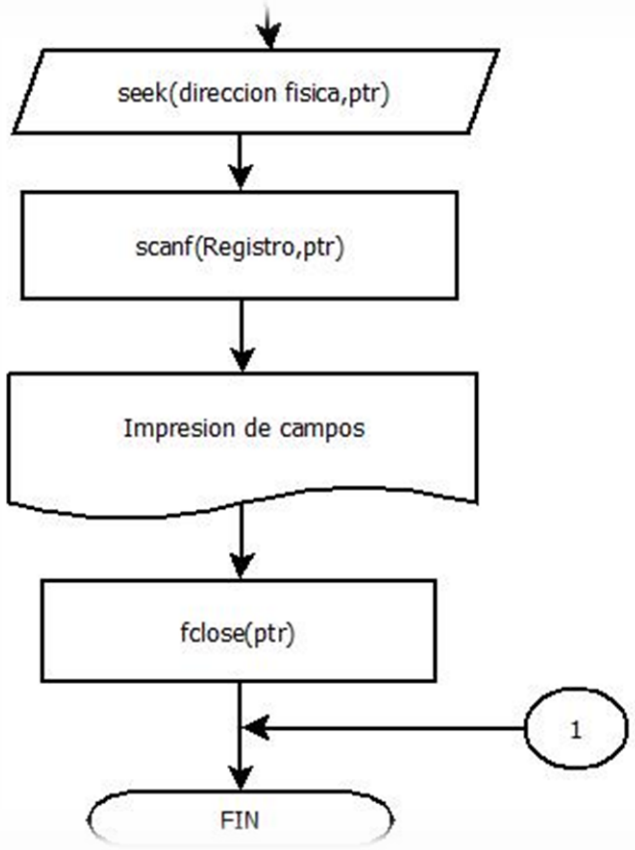
Lectura de un archivo directo

- La función **feof** determina cuando se alcanza el final del archivo y la función **fread** transfiere los datos desde el disco hasta la estructura. (Deitel & Deitel, 2004)

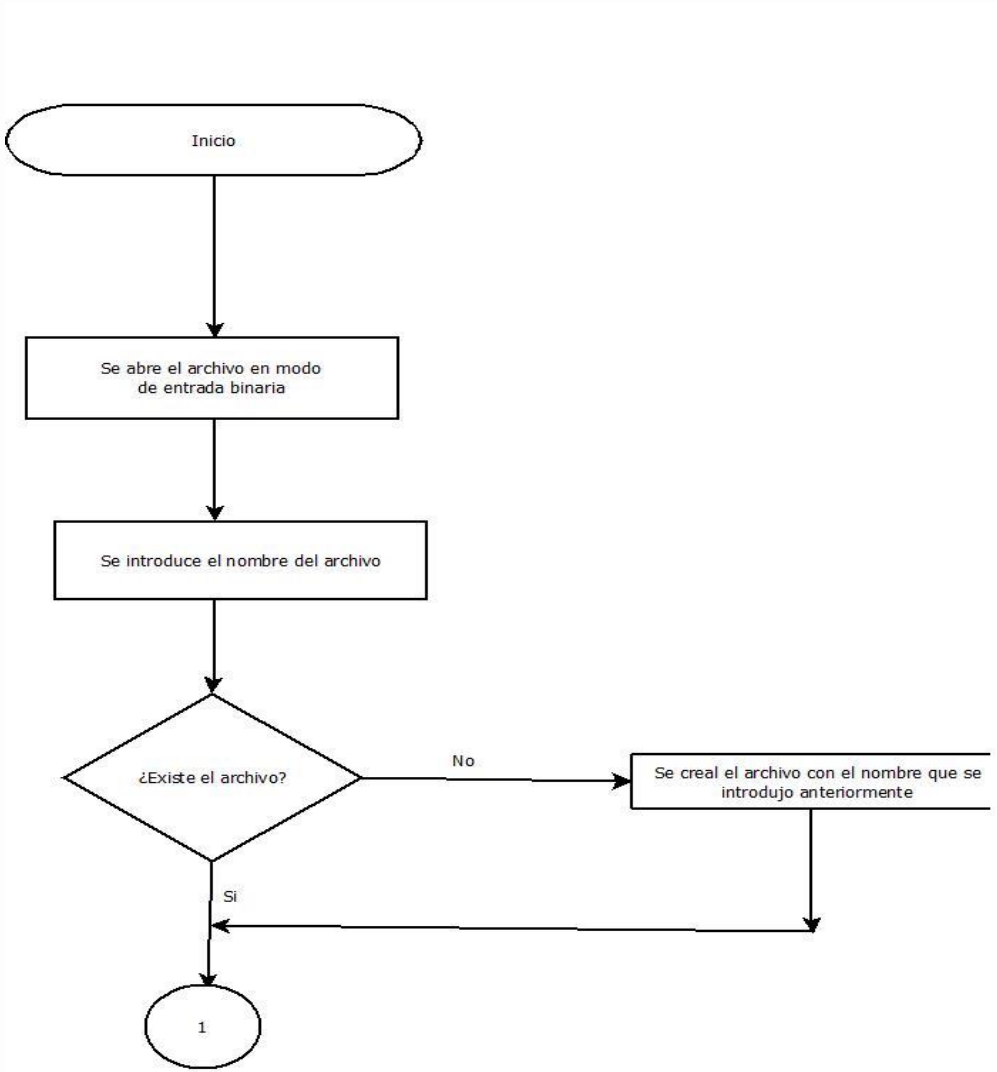
```
12
13 int main()
14 {
15    FILE *ptrCf; /* apuntador de archivo credito.dat */
16
17    /* crea datosCliente con información predeterminada */
18    struct datosCliente cliente = { 0, "", "", 0.0 };
19
20    /* fopen abre el archivo; si no se puede abrir, sale del archivo */
21    if ( ( ptrCf = fopen( "credito.dat", "rb" ) ) == NULL ) {
22       printf( "No pudo abrirse el archivo.\n" );
23    } /* fin de if */
24    else {
25       printf( "%-6s%-16s%-11s%10s\n", "Cta", "Apellido",
26          "Nombre", "Saldo" );
27
28       /* lee todos los registro del archivo (hasta eof) */
29       while ( !feof( ptrCf ) ) {
30          fread( &cliente, sizeof( struct datosCliente ), 1, ptrCf );
```

Lectura


Inicio
FILE *ptr=fopen("archivo.dat","r")
ptr==NULL
No existe
1
Clave producto
Direccion fisica=claveproducto*SIZEOF(Registro)


seek(direccion fisica,ptr)
scanf(Registro,ptr)
Impresion de campos
fclose(ptr)
1
FIN

Consulta


Inicio
Se abre el archivo en modo de entrada binaria
Se introduce el nombre del archivo
¿Existe el archivo?
No
Se creal el archivo con el nombre que se introdujo anteriormente
Si
1

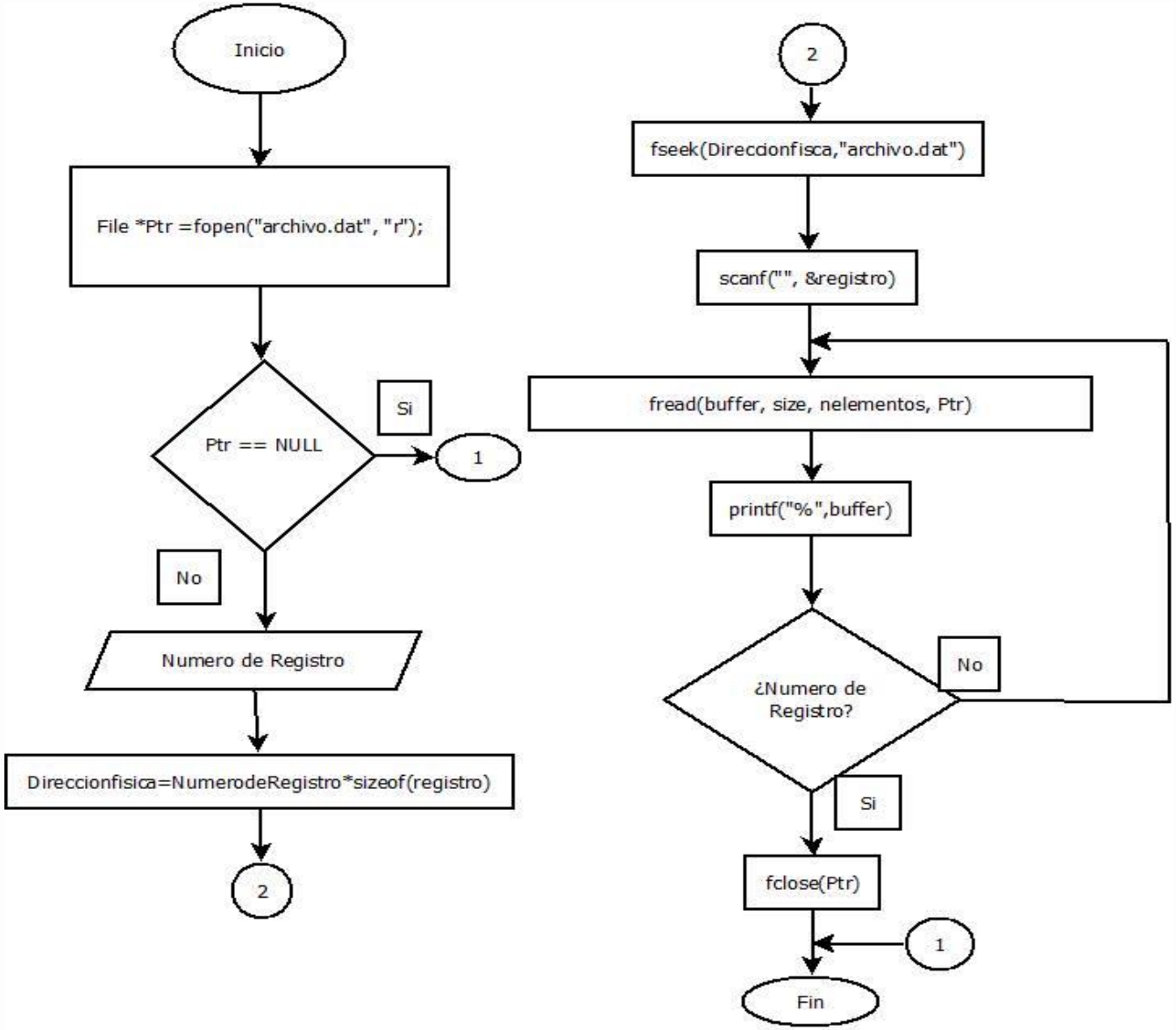

1
Se introduce la clave del registro a buscar
Se obtiene la direccion fisica apartir de la formula Dir. Fisica= Clave de registro * Tamaño del registro
Se coloca el puntero en la posicion que se obtuvo
Se imprime el registro
Se cierra el archivo
Fin

```
void CONSULTA_DIRECTA(void)
{
int no_prod; // Variable local para el numero de producto que desea consultar
clrscr();
cout << "\n\rCONSULTA DE REGISTROS DE PRODUCTOS";
alias=fopen("PRODUCTO.DIR","rb"); // Intenta abrir el archivo PRODUCTO.DIR en modo de solo lectura
if(alias==NULL)
{
cout << "\n\n\n\rNo existe el archivo !!!";
cout << "\n\r<<< Oprima cualquier tecla para continuar >>>";
getch();
return;
}
```

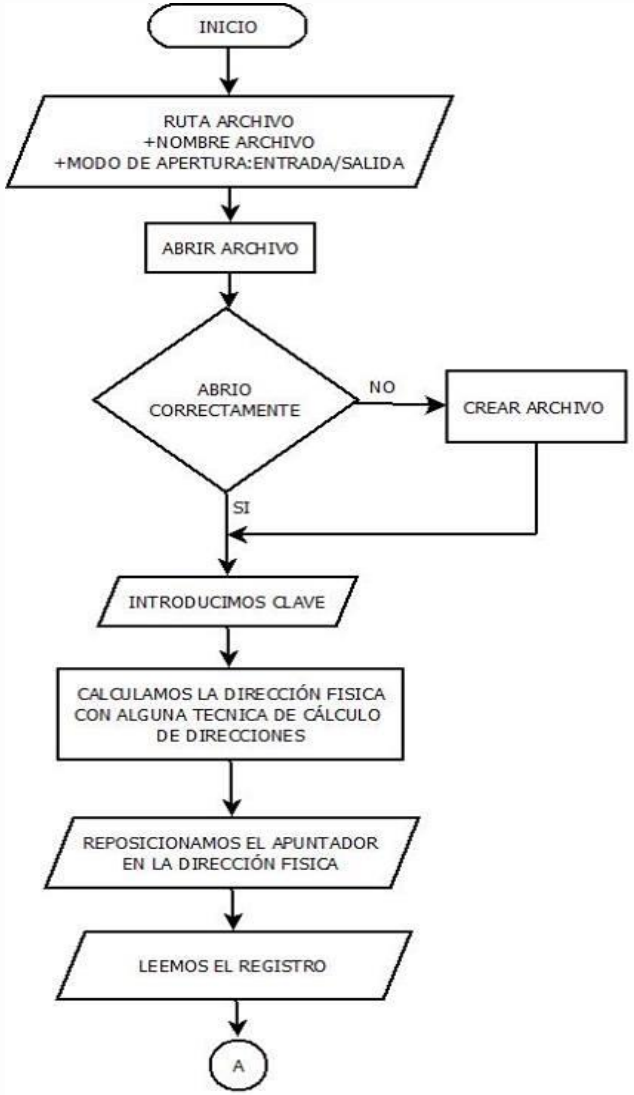
```
cout << "\n\n\n\rNumero de producto: "; cin >> no_prod;
dir_fisica=no_prod*sizeof(Registro); // Calculo de la dir. fisica
fseek(alias,dir_fisica,SEEK_SET); //Posicionar el apuntador del archivo
fread(&Registro,sizeof(Registro),1,alias);
// Lee el "Registro", de tamano=sizeof(Registro) del archivo "alias«
if(Registro.no_prod==no_prod)
{
cout << "\n\rNo Prod Descripcion Cantidad
Precio Garantia";
cout << "\n\r------------------------------------------------------------------------";
printf("\n\r%3d\t%30s\t%3d\t\t$%4.2f\t%c",Registro.no_prod,Registro.descr
ip,Registro.cantidad,Registro.precio,Registro.garantia);
}
```

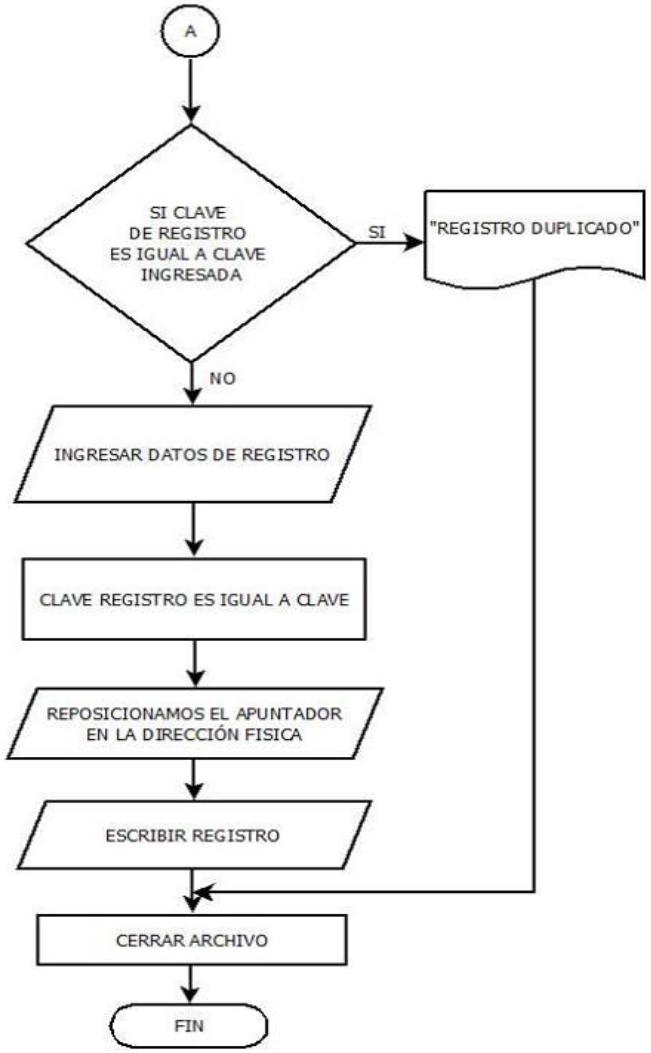
```
else
{
cout << "\n\n\n\rNo existe ese registro !!!";
}
fclose(alias);
cout << "\n\n\n\n\r<<< Oprima cualquier tecla para continuar >>>";
getch();
return;
}
```

Lectura Consecutiva

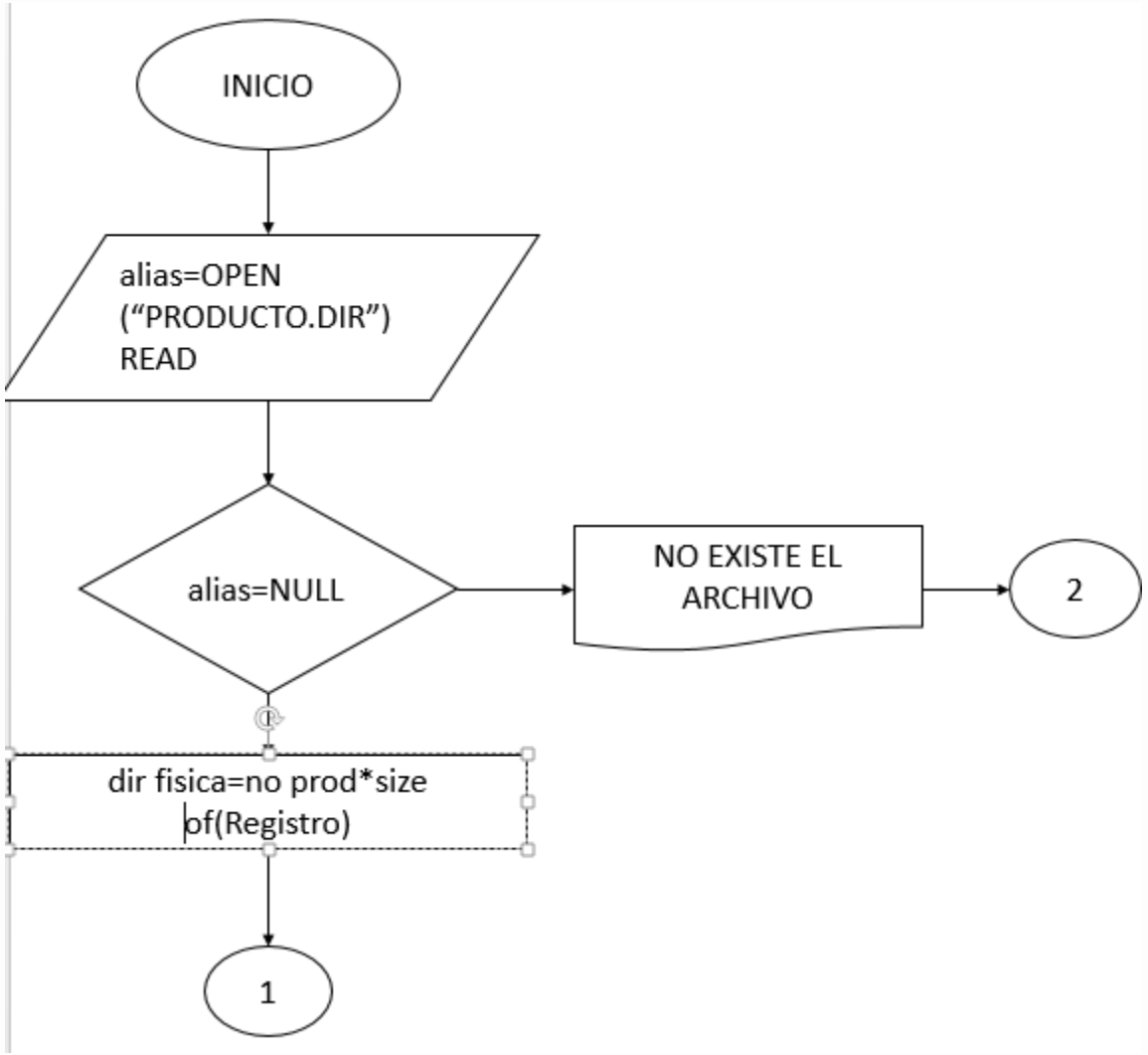


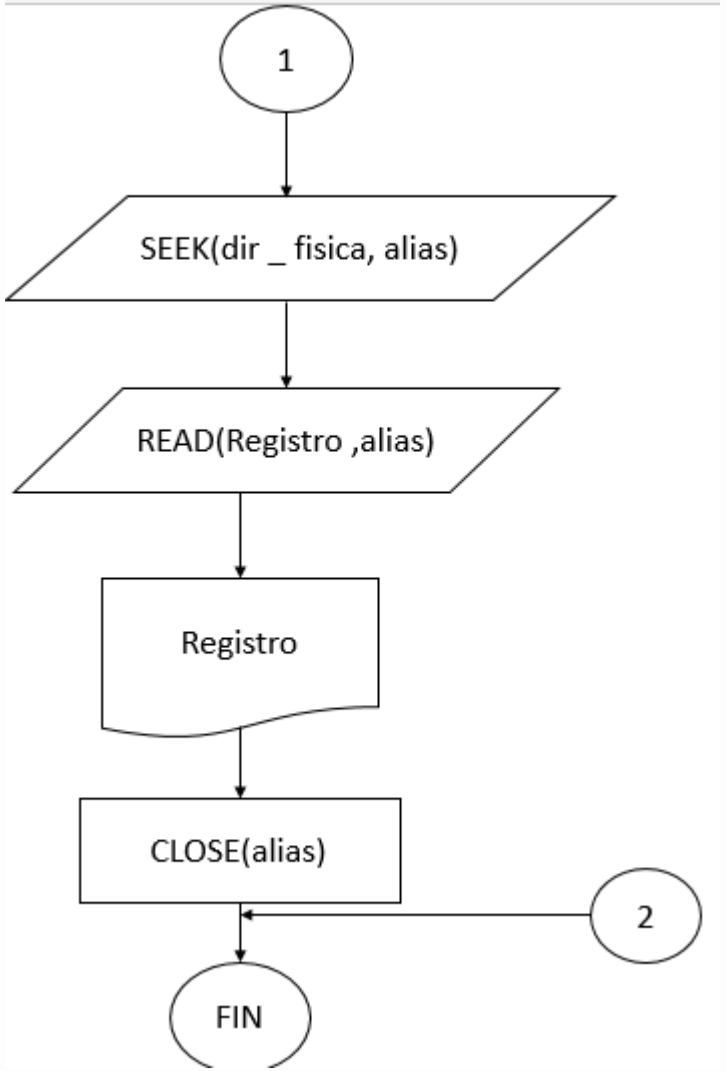
Lectura Exahustiva



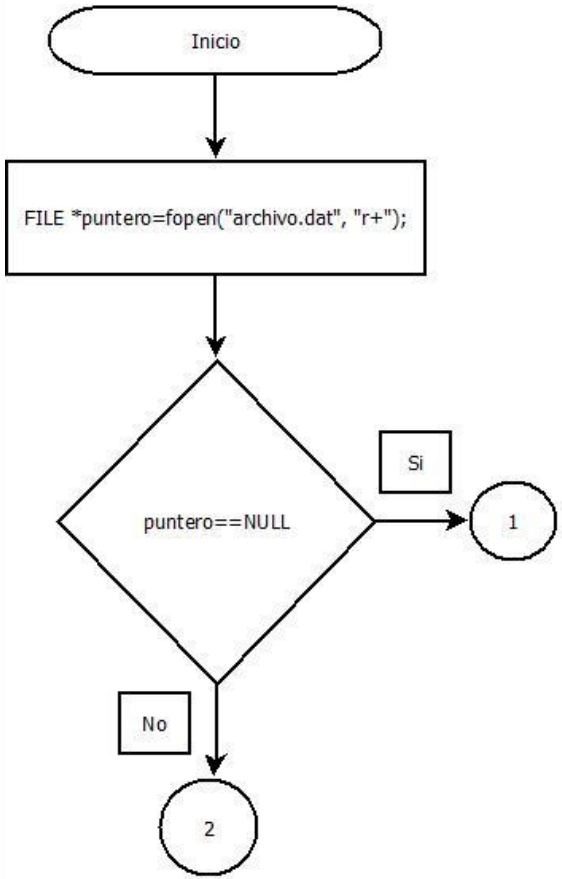


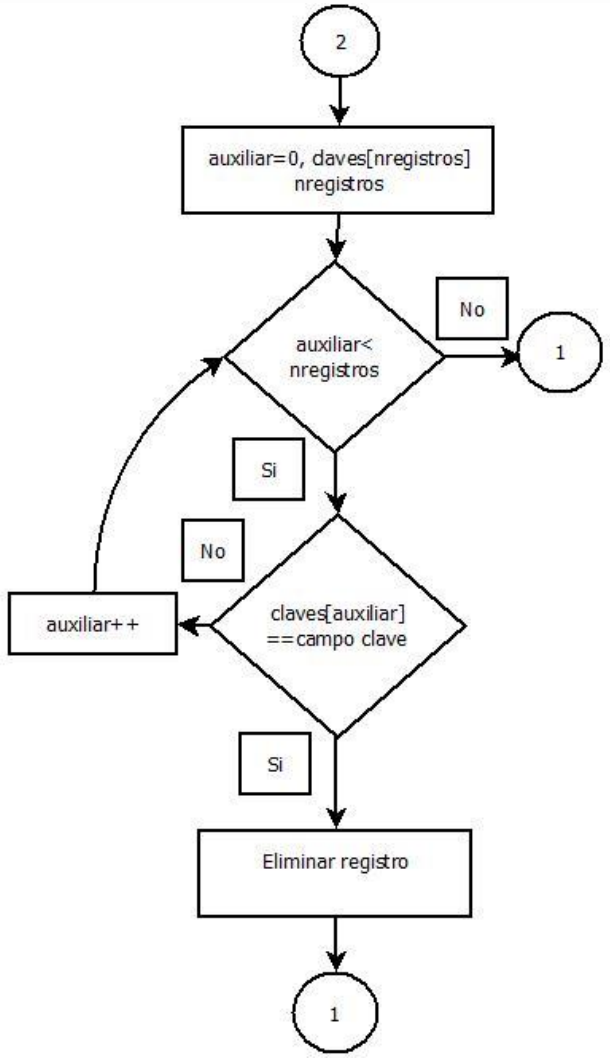
Lectura Ordenada

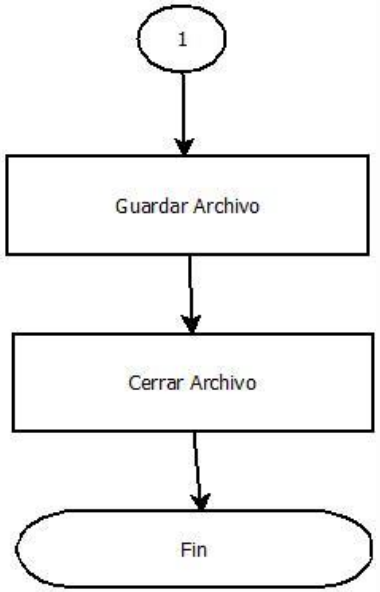




Borrado


Inicio
FILE *puntero=fopen("archivo.dat", "r+");
puntero==NULL
Si
1
No
2


2
auxiliar=0, daves[nregistros]
nregistros
auxiliar<
nregistros
No
1
Si
No
auxiliar++
claves[auxiliar]
==campo clave
Si
Eliminar registro
1


1
Guardar Archivo
Cerrar Archivo
Fin

```
void BAJA_LOGICA_DIRECTA(void)
{
int no_prod; // Variable local para el numero de producto que desea eliminar
char op;
clrscr();
cout << "\n\rBAJA LOGICA DE REGISTROS DE PRODUCTOS";
alias=fopen("PRODUCTO.DIR","rb+"); // Intenta abrir el archivo PRODUCTO.DIR
// en modo de lectura/escritura
if(alias==NULL) // Valida la existencia del archivo
{
```

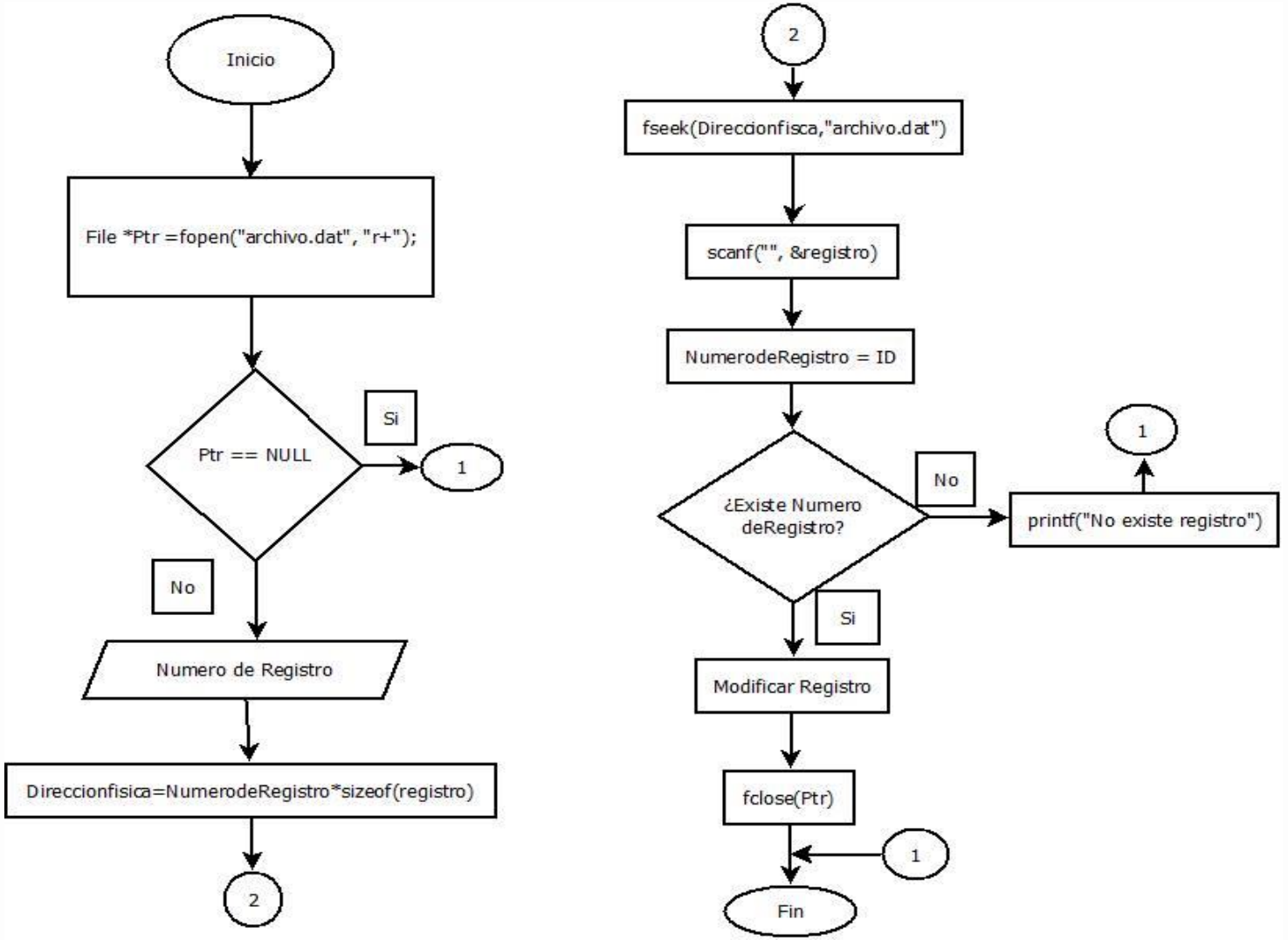
```
cout << "\n\n\n\rNo existe el archivo !!!";
cout << "\n\r<<< Oprima cualquier tecla para continuar >>>";
getch();
return;
}
cout << "\n\n\n\rNumero de producto: "; cin >> no_prod;
dir_fisica=no_prod*sizeof(Registro); // Calculo de la dir. fisica
fseek(alias,dir_fisica,SEEK_SET); //Posicionar el apuntador del archivo
fread(&Registro,sizeof(Registro),1,alias);
// Lee el "Registro", de tamano=sizeof(Registro) del archivo "alias"
```

```
if(Registro.no_prod==no_prod)
{
cout << "\n\rNo Prod Descripcion Cantidad
Precio Garantia";
cout << "\n\r-------------------------------------------------------------------------";
printf("\n\r%3d\t%30s\t%3d\t\t$%4.2f\t%c\n\n\n\n\r",Registro.no_prod,Regi
stro.descrip,Registro.cantidad,Registro.precio,Registro.garantia);
Registro.no_prod=0;
strcpy(Registro.descrip,"");
Registro.cantidad=0;
Registro.precio=0.0;
Registro.garantia=' ';
// Es necesario reposicionar el apuntador del archivo al principio del
// registro que desea modificar, ya que al leer un registro, el
// apuntador se posiciona en el registro siguiente
// La funcion ftell(alias) devuelve la posicion donde se encuentra el apuntador
```

```
do
{
cout << "\n\rEstá seguro que desea eliminar este registro? [S/N] -->";
op=toupper(getche());
}while(op!='S' && op!='N');
if(op=='S')
{
fseek(alias,dir_fisica,SEEK_SET); //Posicionar el apuntador del archivo
fwrite(&Registro,sizeof(Registro),1,alias); // Graba el registro con
// los nuevos campos
cout << "\n\n\n\rRegistro eliminado lógicamente !!!";
}
else
{
cout << "\n\n\n\r Registro NO eliminado !!!";
}
fclose(alias); // Cierra el archivo
```

```
cout << "\n\r<<< Oprima cualquier tecla para continuar >>>";
getch();
return;
}
else
{
cout << "\n\n\n\r No se encuentra ese registro !!!";
}
fclose(alias); // Cierra el archivo
cout << "\n\n\n\n\r<<< Oprima cualquier tecla para continuar >>>";
getch();
return;
}
```

Actualización



```
void MODIFICACION_DIRECTA(void)
{
int no_prod; // Variable local para el numero de producto que desea
modificar
clrscr();
cout << "\n\r MODIFICACIÓN DE REGISTROS DE PRODUCTOS";
alias=fopen("PRODUCTO.DIR","rb+"); // Intenta abrir el archivo PRODUCTO.DIR
// en modo de lectura/escritura
if(alias==NULL) // Valida la existencia del archivo
{
cout << "\n\n\n\r No existe el archivo !!!";
cout << "\n\r<<< Oprima cualquier tecla para continuar >>>";
getch();
return;
}
```

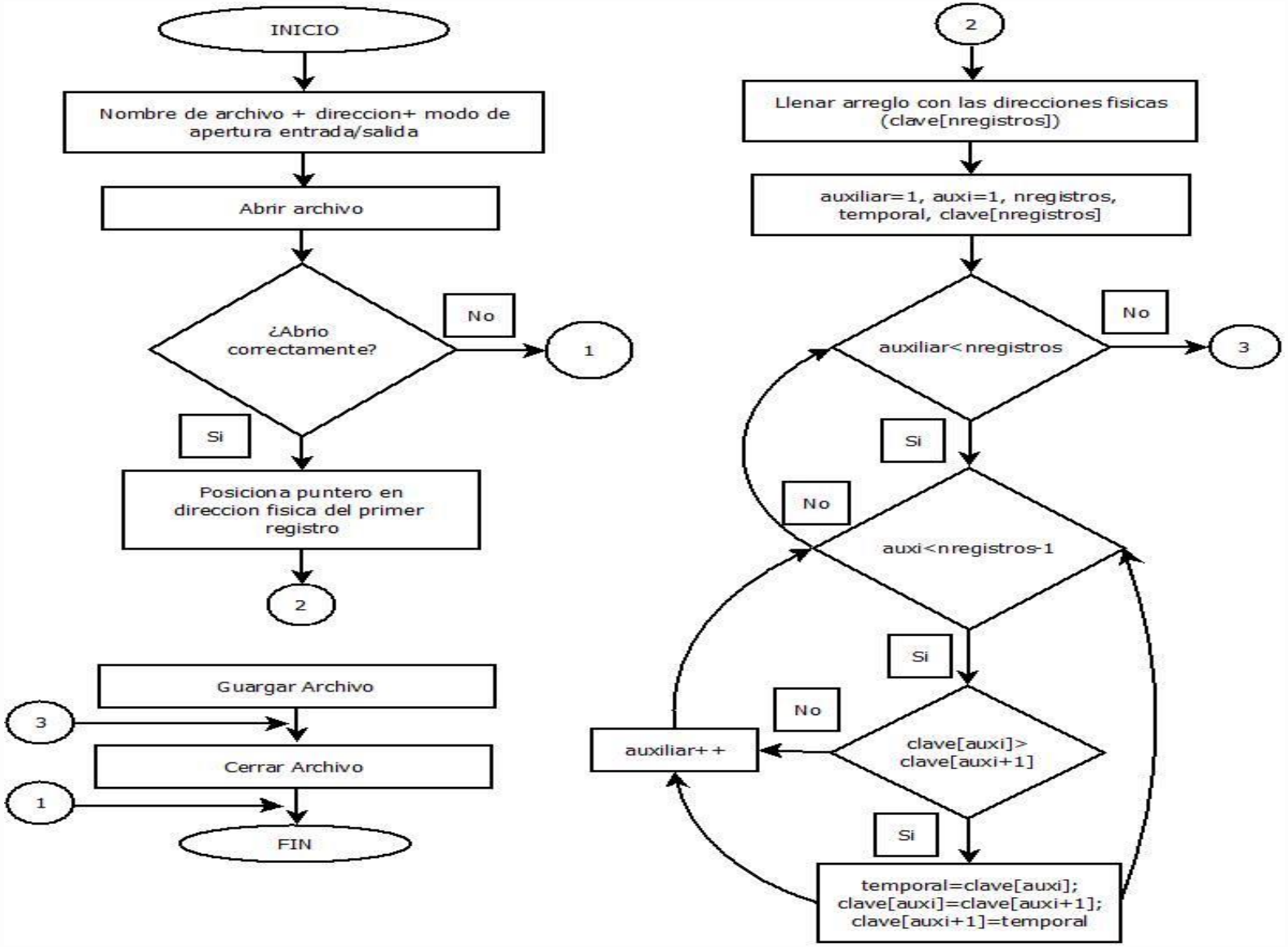
```
cout << "\n\n\n\rNumero de producto: "; cin >> no_prod;
dir_fisica=no_prod*sizeof(Registro); // Calculo de la Dir. fisica
fseek(alias,dir_fisica,SEEK_SET); //Posicionar el apuntador del archivo
fread(&Registro,sizeof(Registro),1,alias);
// Lee el "Registro", de tamano=sizeof(Registro) del archivo "alias"
if(Registro.no_prod==no_prod)
{
cout << "\n\rNo Prod Descripcion Cantidad
Precio Garantia";
cout << "\n\r------------------------------------------------------------------------";
printf("\n\r%3d\t%30s\t%3d\t\t$%4.2f\t%c",Registro.no_prod,Registro.descrip,Registro.cantidad,Re
gistro.precio,Registro.garantia);
```

```
cout << "\n\n\n\r Anote los nuevos datos ...";
cout << "\n\rDescripcion: "; gets(Registro.descrip);
cout << "\n\rCantidad : "; cin >> Registro.cantidad;
cout << "\n\rPrecio : "; cin >> Registro.precio;
do
{
cout << "\n\rGarantia : "; Registro.garantia=toupper(getche());
}while(Registro.garantia!='S' && Registro.garantia!='N');
// Es necesario reposicionar el apuntador del archivo al principio del registro que desea modificar, ya que al
//leer un registro, el apuntador se posiciona en el registro siguiente la funcion ftell(alias) devuelve la posicion
//donde se encuentra  el apuntador
```

```
fseek(alias,dir_fisica,SEEK_SET); //Posicionar el apuntador del archivo
fwrite(&Registro,sizeof(Registro),1,alias); // Graba el registro con los nuevos campos
fclose(alias); // Cierra el archivo
cout << "\n\n\n\rRegistro modificado !!!";
cout << "\n\r<<< Oprima cualquier tecla para continuar >>>";
getch();
return;
}
else
{
cout << "\n\n\n\rNo se encuentra ese registro !!!";
}
fclose(alias); // Cierra el archivo
cout << "\n\n\n\n\r<<< Oprima cualquier tecla para continuar >>>";
getch();
return;
}
```

```
cout << "\n\n\n\r Registro modificado !!!";
cout << "\n\r<<< Oprima cualquier tecla para continuar >>>";
getch();
return;
}
else
{
cout << "\n\n\n\r No se encuentra ese registro !!!";
}
fclose(alias); // Cierra el archivo
cout << "\n\n\n\n\r<<< Oprima cualquier tecla para continuar >>>";
getch();
return;
}
```

Reorganización



Conclusión

- Los archivos directo resultan ser apropiados para sistema de reservación de líneas aéreas, sistemas bancarios y otros sistemas de procesamiento de información que requieren acceso rápido a registros específicos; pues presentan un desempeño optimo en su forma de operación, además que comparado con un archivo secuencial la tarea de reescribir el archivo no es necesaria.

- Aunque si hablamos de eficiencia en cuanto almacenamiento, tenemos un gran problema pues por su misma forma de operación el desperdicio y mal uso de memoria es mucho.

Bibliografía

- Deitel, H., & Deitel, P. (2004). *Como programar en C /C++ y Java.* México: Pearson Educación.
- Perry, G. (1999). *Aprendiendo Visual Basic 6 en 21 días.* Argentina: Pearson Educación.
- Rojas, L. D. (13 de Octubre de 2014). *Universidad Interamericana para el Desarrollo.*
- Recuperado el 29 de septiembre de 2015, de Universidad Interamericana para el Desarrollo: http://moodle2.unid.edu.mx/dts_cursos_mdl/lic/TI/PE/S12/PE12_Lectura.pdf

GRACIAS

Organización de Archivos
Unidad de Competencia III
MTE-MI. Rosa Erendira Reyes Luna
rereyesl@uaemex.mx

GUÍA EMPLEO DE MATERIAL

Organización de Archivos

Unidad de Competencia III

GUÍA EMPLEO DE MATERIAL

Las primeras diapositivas muestran el propósito, justificación y objetivos de la unidad de aprendizaje. Se presentan para que el alumno identifique dichos elementos.

El contenido, conforme a la unidad de aprendizaje, maneja los temas de un menor a mayor grado de dificultad.

De la diapositiva 7 a la 11 describe los archivos directos a partir de sus características, ventajas, desventajas y las condiciones que debe cumplir el archivo para determinarlo esta organización.

GUÍA EMPLEO DE MATERIAL

De las dispositivas 13 a la 44 explican cada proceso para llevar a cabo las operaciones a través de los diagramas de flujos y códigos en lenguaje de programación C.