UNIVERSIDAD AUTONOMA METROPOLITANA

Programación Avanzada

Práctica 4:

"Almacenamiento de Datos en Archivos Binarios"

Fecha de Elaboración:
Martes, 15 de julio del 2008

Fecha de Entrega:
Martes, 22 de julio del 2008

Contenido

AVISOS:

- No se reciben reportes de las prácticas después de la fecha de entrega.
- En la página se encuentra el formato del reporte para las prácticas en donde se describen cada uno de los apartados que deberá considerarse. Se deberá seguir dicha plantilla para el desarrollo del reporte de las prácticas.
- Enviar a la cuenta de correo un paquete zipeado que contendrá:
  - Códigos fuentes (extensión **.c**).
  - Códigos Objetos (extensión **.o**)
  - Ejecutables (sin extensión)
  - Archivos generados (extensión .**dat**)
  - En el reporte incluir por actividad (extensión **.doc)**:
    - **Desarrollo** de la actividad.
    - **Resultados** de la actividad.
- En el subject del correo escribir: ***Práctica 4 - <nombre completo>***.

1 Objetivos

1) Utilizar el diseño de programación ascendente y descendente en el enfoque de **programación modular** (enfoque "divide y vencerás") para la resolución de problemas mediante el uso de **módulos** que serán traducidos a funciones (y/o procedimientos) en el lenguaje C.
2) Hacer uso de **estructura o registros** en el lenguaje C, para definir nuevos tipos de datos.
3) Utilizar funciones de **escritura** a un archivo para **almacenar** la información que se tiene en una lista de registros en memoria a un **archivo binario**.
4) Utilizar funciones de **lectura** a un archivo para **recuperar** la información que se tiene en un **archivo binario** y almacenarla en una lista de registros en memoria.
5) Reutilizar y modificar algunos de los **algoritmos de búsqueda** y **algoritmos de ordenamiento** para trabajar con la lista de registros.

2 Introducción

En el desarrollo de esta práctica se requiere que el alumno realice cada una de las actividades que se le indica.

La reutilización de los algoritmos de búsqueda y ordenamiento se requiere efectuar una serie de modificaciones para su posible utilización.

2.1 Complementos

Para completar el reporte de la práctica el alumno deberá de efectuar una breve investigación de los siguientes puntos (máximo 3 párrafos por punto):

- Apertura, lectura, escritura y cierre, de archivos binarios.
- Utilización del operador **sizeof** para obtener el tamaño de un tipo de dato.
- Lectura y escritura de información con formato utilizando las funciones: **fread** y **fwrite**, respectivamente.

2.2 Ayuda complementaria

- Referencia de la bibliografía contenida en la planeación del curso de Programación Avanzada.
- Notas del Curso de Introducción a al Programación 08-I.
- Guía Básica para el uso de C.

3 Desarrollo

La sección del desarrollo de la práctica se divide en una serie de actividades que permiten englobar, desarrollar y alcanzar cada uno de los objetivos de la práctica. Es necesario leer con atención cada una de las actividades a desarrollar.

3.1 Antecedentes del manejo de archivos

3.1.1 Archivos binarios

Los archivos binarios permiten almacenar y recuperar la información en términos de bytes, respecto a la longitud del tipo de dato que se almacena o se desee leer.

A diferencia de los archivos de texto, que regularmente se almacena (o recupera) la información sin considerar el tipo de dato que se esta guardando (recuperando) en el archivo, los archivos binarios tiene la particularidad de especificar el tipo de dato y su longitud en bytes para almacenar (recuperar) dicho dato.

Generalmente es utilizado cuando se desea almacenar información que se tiene definido como un tipo de dato nuevo, bien sea una estructura o registro.

La extensión para nombrar un archivo binario es <nombre>.dat, y de acuerdo al modo de apertura del archivo es posible trabajar con el apuntador a archivo como un archivo binario.

3.1.2 Modos de apertura de un archivo binario

Modo de Apertura	Descripción
wb	Crea un archivo de escritura, si ya existe lo crea de nuevo.
w+b	Crea un archivo de escritura y lectura, si ya existe lo crea de nuevo.
ab	Abre o crea un archivo para escribir datos al final del mismo.
a+b	Abre o crea un archivo para leer o escribir datos al final del mismo.
rb	Abre o crea un archivo para escribir datos.
r+b	Abre o crea un archivo para escribir y leer datos.

3.1.3 Operador 'sizeof'.

El operador **sizeof** obtiene el tamaño de almacenamiento utilizado por una variable o un dato, de un tipo de dato ya sea básico o definido por el programador.

sizeof(2)	→	printf("\n%d", sizeof (2));
sizeof(1.23)	→	printf("\n%d", sizeof (1.32));
sizeof("Hola")	→	printf("\n%d", sizeof ("Hola"));
sizeof("Hola!")	→	printf("\n%d", sizeof ("Hola!"));
sizeof(AlJuan)	→	printf("\n%d", sizeof (AlJuan));

Aunque la sintaxis del operador es similar a la que se utiliza como función, es un operador por lo cual muchas veces se confunde en su utilización.

3.1.4 Función setbuf – Limpiar el buffer.

Al utilizar la función **setbuf** pasándole como parámetro el flujo de entrada **stdin**, que especifica el flujo de entrada estándar, limpia el búfer de comunicación que existente entre el elemento de entrada y el programa.

setfbuf(stdin, NULL);

Esta función es una alternativa al uso de la función **fflush(stdin)**, cuya funcionalidad es similar a la descrita con la función **setbuf**.

3.1.5 Función fwrite – Escritura de datos en un archivo binario.

La función **fwrite** se utiliza para escribir bloques de texto o de datos, estructuras, en un archivo.

fwrite (*Apuntador_Dato, Tamaño, # Datos, Apuntador_Archivo)

El primer parámetro (***Apuntador_Dato**) describe el apuntador de la dirección de memoria en donde se encuentra el dato a almacenar.

El segundo parámetro (**Tamaño**) específica la longitud en bytes que ocupan los datos del **Apuntador_Dato**, regularmente se hace uso del operador **sizeof** para calcular la longitud en bytes de los datos.

El tercer parámetro (**#Datos**) indica la cantidad de datos de tamaño **Tamaño** que serán escritos en el archivo. Si se trata de un solo dato el valor de **Tamaño** es 1, pero si se está almacenando un arreglo de datos es posible que el **Tamaño** sea variable.

El cuarto parámetro es el (**Apuntador_Archivo**) apuntador a archivo.

Ejemplos:

```
fwrite( &cont, sizeof(cont), 1, arch);
fwrite( &valorf, sizeof(valorf), 1, arch);
fwrite( &nombreArch, sizeof(nombreArch), 1, arch);
fwrite( &Arreglo[0], sizeof(Arreglo[0]), 3, arch);
fwrite( Arreglo, sizeof(int), 3, arch);
```

3.1.6 Función fread – Lectura de datos desde un archivo binario.

La función **fread** se utiliza para leer bloques de texto o de datos, estructuras, desde un archivo.

fread (*Apuntador_Dato, Tamaño, # Datos, Apuntador_Archivo)

El primer parámetro (***Apuntador_Dato**) describe el apuntador de la dirección de memoria en donde se guardara la información que se lea desde el archivo.

El segundo parámetro (**Tamaño**) específica la longitud en bytes que ocupan los datos del **Apuntador_Dato**, regularmente se hace uso del operador **sizeof** para calcular la longitud en bytes de los datos.

El tercer parámetro (**#Datos**) indica la cantidad de datos de tamaño **Tamaño** que serán leídos desde el archivo. Si se trata de un solo dato el valor de **Tamaño** es 1, pero si se va a leer un arreglo de datos es posible que el **Tamaño** sea variable.

El cuarto parámetro es el (**Apuntador_Archivo**) apuntador a archivo.

Ejemplos:

```
printf("\n\nInformacion Leida... ");
fread( &cont, sizeof(cont), 1, arch);
printf("\n\n Cont: %d", cont);
fread( &valorf, sizeof(valorf), 1, arch);
printf("\nvalorf: %f", valorf);
fread( &nombreArch, sizeof(nombreArch), 1, arch);
printf("\nnombreArch: %s", nombreArch);
fread( &Arreglo[0], sizeof(Arreglo[0]), 3, arch);
fread( Arreglo, sizeof(int), 3, arch);
```

3.2 Actividad 1 “Lectura y Escritura de Datos en un Archivo Binario”

Con el desarrollo de esta actividad se alcanzará los objetivos 1,2 y 4 de la práctica.

Efectué la implementación de las siguientes funciones dentro de un archivo (**Actividad1.c**):

AlmacenarInfoArchivoBianrio():

Dicha función debe de permitir almacenar los siguientes datos de acuerdo al tipo de dato en un archivo binario (con extensión .dat), considere este orden:

- Entero.
- Flotante.
- Carácter.
- Cadena.
- Arreglo de enteros.
- Estructura de Alumno (Nombre, matricula, promedio) – Visualizar la propuesta en la actividad 2.

AlmacenarInfoArchivoBianrio():

Dicha función debe de permitir recuperar los siguientes datos de acuerdo al tipo de dato desde un archivo binario (el mismo archivo que se creo con la función descrita anteriormente), considere el mismo orden de la función anterior. Se debe leer la información y desplegar su contenido:

3.2.1 Diseño Modular

3.3 Actividad 2 "Escritura de una Lista de Registros de Alumnos"

Con el desarrollo de esta actividad se alcanzará los objetivos 1, 2 y 3 de la práctica.

3.3.1 "Leer Datos de un Alumnos desde el Teclado"

Cree un archivo (**EscribirRegAL.c**), en el cual se define la estructura de un alumno, así como la definición del mismo.

```
#define TAMCAD 40
#define TAMAXALUMNOS 20


typedef char Cadena[TAMCAD];


//Declaración del tipo de dato alumno
struct   Alumno
{
  Cadena Nombre;
  int Matricula;
  float Promedio;
};

//Definición del nuevo tipo de dato
typedef struct Alumno AL;
```

3.3.2 Descripción del los módulos

Defina el diseño modular que defina las siguientes funciones:

AltaAlumnos(Alumnos: AL[], indiceAlumnos *Ent)

Descripción: Obtiene cada uno de los datos de los alumnos y se incorpora en la lista (arreglo) de alumnos utilizando la función de ***CapturaDatos()***.El valor del indiceAlumnos se modifica cada vez que se agrega un nuevo Alumno.

CapturaDatos (Alumnos: AL)

Descripción: Obtiene cada uno de los miembros (campos) de la estructura Alumno que es proporcionada por el usuario desde el teclado y se almacena en una localidad de memoria de la lista de Alumnos.

ImprimirListaAlumnos (Alumnos: AL[], indiceAlumnos *Ent)

Descripción: Imprime los miembros de cada uno de los alumnos que contienen la lista de Alumnos.

GuardarInfoAlumnosAB(Alumnos: AL, indiceAlumnos *Ent)

Descripción: Utilizando las funciones de apertura (creación) de archivos, se crea el archivo binario (**Datos.dat**) en el cual se almacenará la información de la lista de Alumnos. Una vez almacenados los datos, se cierra el archivo.

3.3.3 Diseño Modular EscribirRegAL

3.4 Actividad 3 "Sistema de Mantenimiento de Alumnos"

Con el desarrollo de esta actividad se alcanzará los objetivos 1, 2, 3, 4 y 5 de la práctica.

3.4.1 "Leer Datos de un Alumnos desde un Archivo Binario – Ordenamiento por Nombre"

Reutilice las funciones que sean necesarias del archivo fuente que se genero en la actividad anterior. Se utilizará el archivo binario creado en la actividad anterior, para acceder a los datos que se contienen dentro de esté.

Cargada la información de la lista de alumnos se prosigue en utilizar un algoritmo de ordenamiento, desarrollando algunas modificaciones para que se efectué el ordenamiento por el nombre del alumno.

3.4.2 Descripción del los módulos

Modifique el diseño modular que defina las siguientes funciones:

LeerinfoALAB (Alumnos: AL[], indiceAlumnos *Ent)

Descripción: Obtiene cada uno de los datos de los alumnos que fueron almacenados en un archivo binario (*.dat) utilizando las funciones de lectura desde un archivo para leer estructura a estructura. El valor del indiceAlumnos se modifica cada vez que se leer un nuevo Alumno desde el archivo binario.

OrdenarNombre(Alumnos: AL[], indiceAlumnos *Ent)

Descripción: Reutilizar el algoritmo de ordenamiento por selección para ordenar la lista de alumnos por el nombre los alumnos. Haga uso de funciones de para la manipulación de cadenas para el desarrollo de esta actividad.

Recuerde de redefinir la función de intercala para el cambio de registros de alumnos.

3.4.3 Diseño Modular SMAL

4 Reporte de las actividades.

Dentro del reporte se debe de incluir, por actividad:

- Las investigaciones necesarias para la elaboración de la práctica.
- El contenido de los archivos creados así la información de ejecución de los programas.
- Las conclusiones del uso de estructuras de datos, del uso de archivos binarios y de texto, así como la experiencia del uso de las funciones para escribir y leer datos de un archivo binario.