|  |  |  |
| --- | --- | --- |
|  | UNIVERSIDAD NACIONAL EXPERIMENTAL POLITÉCNICA  “ANTONIO JOSÉ DE SUCRE”  VICERECTORADO PUERTO ORDAZ  DEPARTAMENTO DE INGENIERÍA ELECTRÓNICA  SECCIÓN DE COMPUTACIÓN – **COMPUTACION III** | logo_departamentologo_departamentologo_departamento |

**PRACTICA N° 1**

|  |
| --- |
| Objetivos. |

* Repasar los conceptos básicos de punteros, funciones y archivos.
* Aprender los conceptos básicos del uso de PILAS, como estructuras de datos dinámicas.

|  |
| --- |
| Base Teórica. |

El sistema de manejo de archivos de C++ controla las operaciones de Entrada/Salida o de Lectura/Escritura por medio de streams. Un **stream** o ***ArchivoLogico*** es una especie de canal a través del cual fluyen los datos utilizado desde y hacia los dispositivos estándar conectados a la PC. Ejemplos de **streams** son ***cin*** y ***cout*** que permiten leer del teclado y mostrar por pantalla (***cin*** » x; - ***cout*** « x;).

Por definición, un archivo es una colección de datos guardados en algún medio de almacenamiento no volátil o sea que la información se mantiene aun cuando se “apague” o desconecte el dispositivo (Disco Duro, CD, Pendrive, SD card, etc). Estos archivos se pueden manejar como streams.

* Para declarar un ArchivoLogico se usa el formato: ***tipo <NombreArchivoLogico>***;

donde: ***tipo*** se usa ***ofstream*** para ***Salida y ifstream*** para ***Entrada***

***NombreArchivoLogico*** cualquier identificador valido en c++

* Para ABRIR un ArchivoLogico se usa el formato: ***ArchivoLogico.open(ArchivoFisico,modo);*** :

Dónde: ***ArchivoLogico*** es una variable previamente declarada según la indicación anterior.

***ArchivoFisico*** nombre según el formato que usa Windows. Debe incluir ruta de acceso.

***modo in, out, app, ate, trunc, binary.*** Se debe colocar delante ***ios::modo***

A continuacion se muestran algunos ejemplos de manejo de archivos de texto. Por razones de espacio el codigo no incluye las librerias y el main(), pero se aclara que deben estar presentes,

* **Escritura/Lectura de un archivo que contiene valores enteros**

|  |  |
| --- | --- |
| **Ejemplo 1**.- Crear y almacenar en un archivo de texto(A1.txt) una matriz de M filas y N columnas.   * El ***ArchivoLogico*** se crea con la declaración ***ofstream ASal;.*** * La apertura del archivo ***ASal.open("A1.txt"***); conecta Archivos ***Logico*** y ***Fisico***. Si la operación falla, la variable ***ArchivoLogico*** se hace NULL y el programa se detiene enviando un mensaje. En la declaracion ***ASal*** es de salida (***out***), luego no se requiere colocar el modo.   La escritura se hace a traves del ***ArchivoLogico***. Mediante el operador << (***Asal*<<*x;***), se envía la información al ***ArchivoFisico***, igual que ***cout<<x*** *envia a pantalla;* |  |

**Ejemplo 2.-** Leer y mostrar en pantalla el archivo de texto(A1.txt) creado en el paso anterior.

|  |  |
| --- | --- |
| * El ***ArchivoLogico*** se crea con la declaración ***ifstream ASal;.*** Luego se chequea si tuvo éxito. * En este caso se supone que se conoce el número de filas y de columnas, entonces se sabe cuántos valores se van a leer. Mediante un lazo doble se efectúan las lecturas requeridas (fil\*col) y se muestra en pantalla la matriz completa. * La operación de lectura se hace mediante el operador >> (***Aent>>x;***), igual que ***cin>>x;*** |  |

* **Escritura/Lectura de un archivo que contiene un conjunto de registros.**

|  |  |
| --- | --- |
| **Ejemplo** **3**.- La siguiente tabla contiene la información del inventario de una librería. Escribir un algoritmo que permita almacenar en un archivo el inventario del negocio.    Leer y almacenar en un archivo (A2.txt) un conjunto de registros que contienen los campos mostrados en la tabla de arriba.  El registro se define: |  |
| struct t1  { int cod, cant;  float precio;  char titulo[30];  };  t1 libro; | Esta estructura debe estar definida ANTES de comenzar el programa mostrado.  El programa lee cada registro del teclado y lo envia al archivo físico. |

La escritura se hace ***ASal<<libro.cod<<" ";.*** Se agrega un espacio para separar los campos.Los operadores(***setw, right, left***, etc) pueden ser utilizados para organizar los datos de manera que se obtenga una salida mas ordenada cuando se abra el archivo fisico con un editor de texto.

**Ejemplo 4.-** Lectura del archivo del ejemplo 3.

|  |  |
| --- | --- |
| Cada registro es una cadena de caracteres que termina con el salto de linea (\n). Los campos deben separarse de alguna manera, con blancos, con comas, con punto y coma o cualquier carácter fácilmente identificable.  La lectura del campo titulo del registro libro archivo tiene detalles similares a los que se presentan con la lectura de una cadena que contiene espacios en blanco. Es necesario por lo tanto usar getline.  **AEnt.getline(libro.titulo,30)**  La instrucción ***getline*** lee una cadena incluyendo espacios en blanco hasta que consigue el salto de línea '\n', que en este caso se agrega al pulsar la tecla <enter> o sea cuando se termina de insertar el titulo. |  |

El formato de **getline** tiene un tercer parámetro que corresponde al carácter que termina la lectura de la cadena, cuando se leen menos caracteres que loe especificados en el segundo parámetro.

* **Escritura/Lectura de un archivo que contiene cadenas de caracteres (palabras).**

**Ejemplo 5.-** programa que copia un archivo. El programa lee un archivo de texto llamado "license.txt", lo muestra en pantalla y luego copia en otro archivo "copia\_license.txt" la información leída, obteniéndose una copia igual al original.

El programa lee la información letra por letra, la muestra en pantalla y la envía a otro archivo (A2.txt).

Al mostrar la información en pantalla cuenta las líneas, buscando el carácter que indica salto de línea (’\n’) y produce una pausa cada 50 líneas.

La lectura puede hacerse letra por letra, o por palabras o por líneas completas. Una forma de hacer estas lecturas se muestra en los segmentos de código que se muestran en la tabla de la siguiente página, a la izquierda.

En este problema no se sabe cuántos registros tiene el archivo original por lo tanto se procede a leer hasta que se llegue al final del archivo (EOF). Una función ***ArchivoLogico***.***eof***() devuelve 1 si se alcanzó el fin del archivo y devuelve 0 en caso contrario. !A1.eof() es cierto cuando A1.eof()=0.

|  |  |
| --- | --- |
|  |  |

**DESARROLLO DE LA PRACTICA**

|  |  |
| --- | --- |
| Se tiene una tabla de datos como se muestra a la derecha la cual será suministrada como un archivo de texto (lista3.txt) junto con esta práctica.  Escribir un programa tipo menú que contenga las siguientes opciones:   1. Cargar datos 2. Mostrar datos cargados 3. Ordenar datos por expediente 4. Cargar datos en una pila 5. Opción adicional 6. Salir  * Oopción [1] debe leer el archivo y almacenarlo en un arreglo de registros. * Oopción [2] debe mostrar el arreglo de registros. * Opcion [3] debe ordenar el arreglo usando una pila. * Oopción [4] debe leer el arreglo y cargar los datos en una estructura tipo pila. * Oopción [5] será entregada en la práctica y estará relacionada con la pila creada en la opción 4. |  |

En la página siguiente se muestra un programa que permite leer el archivo de texto y mostrarlo por pantalla. Si así lo desea puede usarlo en el programa solicitado o puede escribir su versión personal.

#include<iostream.h>

#include<iomanip.h>

#include<fstream.h>

struct T1

{ int N;

string exp, grupo, nombre;

};

void main()

{ ifstream A1;

T1 registro;

A1.open("lista3.txt");

if(!A1)

{ cout<<"Error al abrir archivo";

system("pause");

return;

}

A1>>registro.N;

A1.ignore(1);

getline(A1,registro.exp,'\t');

getline(A1,registro.grupo,'\t');

getline(A1,registro.nombre);

while(A1.eof()==0)

{ cout<<setw(3)<<left<<registro.N;

cout<<setw(13)<<registro.exp;

cout<<setw(4)<<registro.grupo;

cout<<setw(24)<<registro.nombre<<endl;

A1>>registro.N;

A1.ignore(1);

getline(A1,registro.exp,'\t');

getline(A1,registro.grupo,'\t');

getline(A1,registro.nombre);

}

A1.close();

system("pause");

}