Universidad Autónoma del Estado de México

**Centro Universitario UAEM Atlacomulco**

**“Practica de archivos secuenciales”**

**Elías Edgardo Segundo Antonio**

**Unidad de Aprendizaje:**

**Organización de archivos**

**Fecha de entrega:**

**21 de marzo de 2017**

# Resumen

En esta práctica se cubrieron todas las operaciones que se pueden realizar con los archivos directos, se abarco la creación, altas, bajas lógicas y físicas.

El lenguaje usado para desarrollarlo es C++, por ello todos los archivos fueron creados y manipulados usando la clase fstream, ademas de ello se incluyó el uso de estos registros secuenciales con el uso de las clases de C++.

# Objetivo

El objetivo de esta práctica es generar un programa en C++ con las funciones necesarias para crear, borrar, actualizar y mostrar registros con las características de un archivo directo, sin embargo, para la opción de borrado se deberá incluir tanto la versión de borrado física como la versión lógica.

# Antecedentes

## Archivos secuenciales

Es la forma básica de organizar un conjunto de registros, que forman un archivo, utilizando una organización secuencial. En un archivo organizado secuencialmente, lo registros quedan grabados consecutivamente cuando el archivo se utiliza como entrada. En la mayoría de los casos, los registros de un archivo secuencial quedan ordenados de acuerdo con el valor de algún campo de cada registro. Semejante archivo se dice que es un archivo ordenado; el campo, o los campos, cuyo valor se utiliza para determinar el ordenamiento es conocido como la llave del ordenamiento. Un archivo puede ordenarse ascendente o descendentemente con base en su llave de ordenamiento.



Ilustración Archivo secuencial

## Ventajas

Los archivos secuenciales proveen la mejor utilización de espacio y son rápidos cuando los registros son accesados secuencialmente.  
  
Los archivos con poca volatilidad, gran actividad y tamaño variable son altamente susceptibles de ser organizados secuencialmente.  
  
La ventaja más importante de la técnica de organización secuencial de archivos es la capacidad de acceso al "siguiente" registro rápidamente: Mientras que el patrón de acceso a un archivo secuencial se conforme al ordenamiento de registros en el archivo, los tiempos de acceso serán muy buenos. Sin embargo, si el patrón de acceso al programa no se conforma al patrón de ordenamiento de los registros, entonces la eficiencia del programa puede ser terrible.

## Desventajas

El acceso a un registro es pobre, la localización de un determinado registro no se puede hacer individualmente no de manera rápida, y el acceso aleatorio es impráctico.

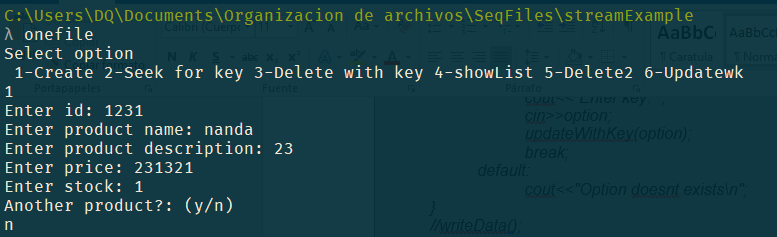
Además, en los archivos secuenciales la dirección de registro está implícita y están vulnerables a fallas del sistema.

# Desarrollo

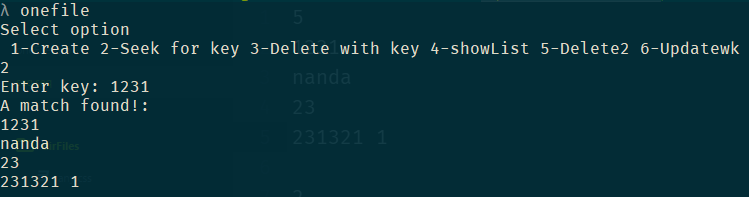
A continuación, se muestra el código fuente desarrollado en la práctica:

|  |
| --- |
| Seq.cpp |
| *#include<bits\stdc++.h>*  *using namespace std;*  *struct product{*  *int price;*  *int stock;*  *int id;*  *string name;*  *string description;*  *};*  *stack <product> p;*  *vector<product> products;*  *int seekForKey(int key)*  *{*  *for(product x: products)*  *{*  *if(key==x.id)*  *{*  *cout<<"A match found!: \n";*  *cout<<x.id<<"\n"<<x.name<<"\n";*  *cout<<x.description<<"\n"<<x.price<<" "<<x.stock<<"\n\n";*  *cout<<"-----------------";*  *return true;*  *}*  *}*  *return false;*  *}*  *void writeData(stack<product> p)*  *{*  *ofstream seq ("seq.seq");*  *seq<<p.size()<<"\n";*  *while(!p.empty())*  *{*  *product x= p.top();*  *p.pop();*    *seq<<x.id<<"\n";*  *seq<<x.name<<"\n";*  *seq<<x.description<<"\n";*  *seq<<x.price<<" "<<x.stock<<"\n\n";*  *}*  *}*  *void printProducts(vector<product> p)*  *{*  *for(product x:p)*  *{*  *if(x.id!=-1)*  *{*  *cout<<x.id<<"\n"<<x.name<<"\n";*  *cout<<x.description<<"\n"<<x.price<<" "<<x.stock<<"\n\n";*  *}*    *}*  *}*  *void readProducts()*  *{*  *ifstream inFile ("seq.seq");*  *int dataSize;*  *string trash;*  *inFile>>dataSize;*  *getline(inFile,trash);*  *for(int i=0;i<dataSize;i++)*  *{*  *product aux;*  *inFile>>aux.id;*  *getline(inFile,trash);*  *getline(inFile,aux.name);*  *getline(inFile,aux.description);*  *inFile>>aux.price>>aux.stock;*  *getline(inFile,trash);*  *p.push(aux);*  *products.push\_back(aux);*  *}*  *inFile.close();*  *//printProducts(products);*  *}*  *int searchIndex(int key)*  *{*  *for (int i = 0; i < products.size(); i++)*  *{*  *if(products[i].id==key)return i;*  *}*  *return -1;*  *}*  *void update(vector<product> products)*  *{*  *while(!p.empty()) p.pop();*  *for(product x:products)*  *p.push(x);*  *writeData(p);*  *}*  *void deleteElement(int key)*  *{*  *while( !p.empty() ) p.pop();*  *int index= seekForKey(key);*  *if(index!=-1)*  *{*  *products.erase(products.begin()+index);*    *for (int i = 0; i < products.size(); i++)*  *{*  *p.push(products[i]);*  *}*  *cout<<"\n"<<p.size();*  *writeData(p);*  *cout<<"\n Element <"<< key<<"> deleted!";*  *}*  *else cout<<"Product doesnt exists";*  *}*  *void deleteWithTrash(int key)*  *{*  *for(int i=0;i<products.size();i++)*  *{*  *if(products[i].id==key)*  *{*  *products[i].id=-1;*  *products[i].name="trash";*  *products[i].description="trash";*  *products[i].price=-1;*  *products[i].stock=-1;*  *}*    *}*  *update(products);*  *writeData(p);*  *}*  *void createProducts()*  *{*  *product p1;*  *fstream seq("seq.seq",fstream::app);*  *string another;*  *do{*  *cout<<"Enter id: ";*  *cin>>p1.id;*  *for(product x: products)*  *{*  *while(x.id==p1.id)*  *{*  *cout<<"Duplicated key, please type another: ";*  *cin>>p1.id;*  *}*  *}*  *getchar();*  *cout<<"Enter product name: ";*  *getline(cin,p1.name);*  *cout<<"Enter product description: ";*  *getline(cin,p1.description);*  *cout<<"Enter price: ";*  *cin>>p1.price;*  *cout<<"Enter stock: ";*  *cin>>p1.stock;*  *products.push\_back(p1);*  *p.push(p1);*  *cout<<"Another product?: (y/n) \n";*  *cin>>another;*    *} while(another!="n");*  *writeData(p);*  *seq.close();*  *}*  *void updateWithKey(int key)*  *{*  *int index=searchIndex(key);*  *getchar();*  *cout<<"Enter new name: ";*  *getline(cin,products[index].name);*  *cout<<"Enter new description: ";*  *getline(cin,products[index].description);*  *cout<<"Enter new price: ";*  *cin>>products[index].price;*  *cout<<"Enter new stock: ";*  *cin>>products[index].stock;*  *update(products);*  *writeData(p);*  *}*  *int main()*  *{*  *readProducts();*  *int option;*  *cout<<"Select option\n 1-Create 2-Seek for key 3-Delete with key 4-showList 5-Delete2 6-Updatewk \n";*  *cin>>option;*  *switch(option)*  *{*  *case 1:*  *createProducts();*  *break;*  *case 2:*  *cout<<"Enter key: ";*  *cin>>option;*  *seekForKey(option);*  *break;*  *case 3:*  *cout<<"Enter key: ";*  *cin>>option;*  *deleteElement(option);*  *break;*  *case 4:*  *printProducts(products);*  *break;*  *case 5:*  *cout<<"Enter key: ";*  *cin>>option;*  *deleteWithTrash(option);*  *break;*  *case 6:*  *cout<<"Enter key: ";*  *cin>>option;*  *updateWithKey(option);*  *break;*  *default:*  *cout<<"Option doesnt exists\n";*  *}*  *//writeData();*  *}* |

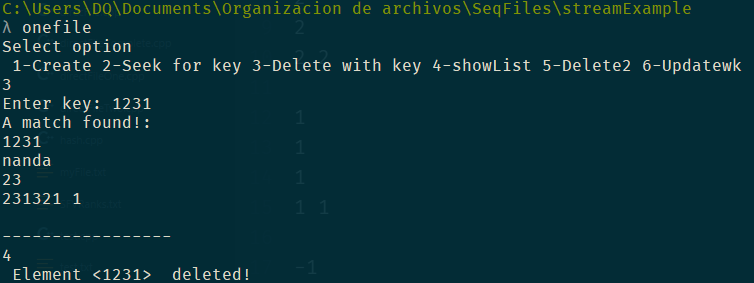
## Capturas de pantalla



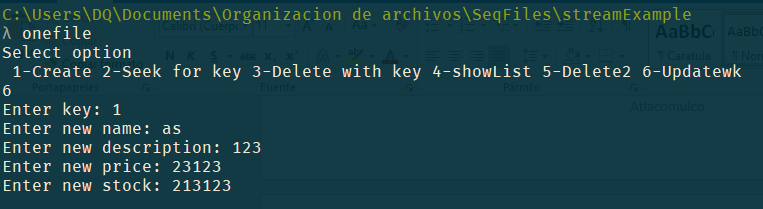
Creación del archivo



Búsqueda por clave



Borrado con clave



Actualizando datos con clave

# Conclusiones

Se concluirá algunas de las características vistas en los archivos realizados

* Los archivos secuenciales son una manera muy fácil de implementar programas que acceden a ficheros, sin embargo, su tiempo de ejecución, el rendimiento deja mucho que desear.
* Se tiene que ingresar el contenido miembro por miembro al fichero, una desventaja fuerte frente a los archivos directos.
* En contra parte esta implementación realmente es sencilla cuando se utiliza en proyectos de guardado pequeños.

# Bibliografía

|  |  |
| --- | --- |
| [1] | B. Garcia, Programacion en manejo de objetos, Mexico: Macrobit, 2006. |
| [2] | M. Lomis, Estructuras de datos y organizacion de archivos, Prentice Hall, 1991. |