Actividad 01 - Repaso de Programación

Roberto Haro González

Seminario de solución de problemas de algoritmia

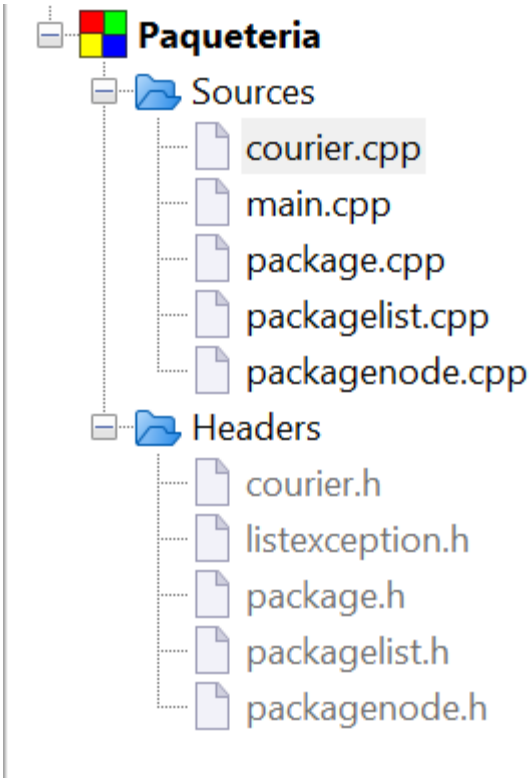
Lineamientos de evaluación

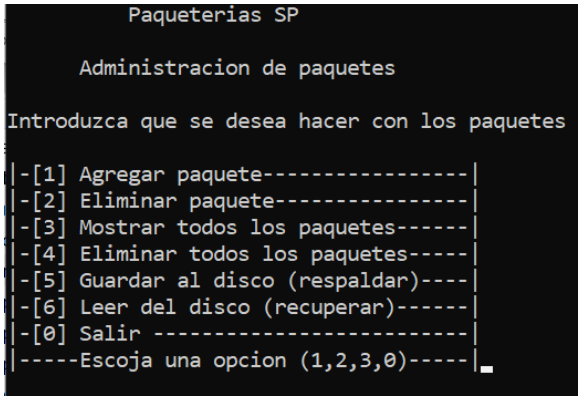
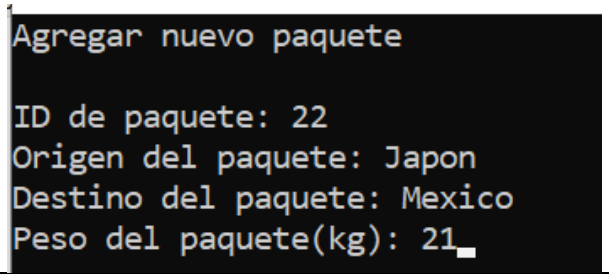
Aquí deberás escribir los lineamientos de evaluación descritos en la actividad, señalando o marcando aquellos lineamientos que estás cumpliendo. Ejemplo:

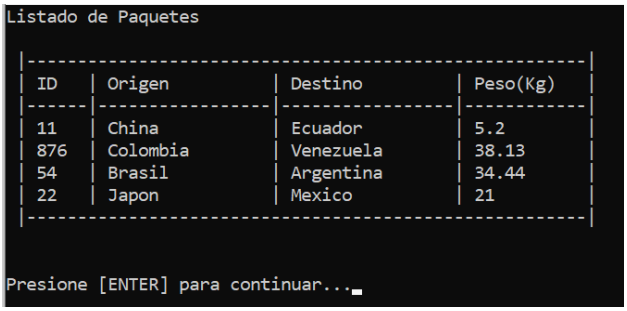
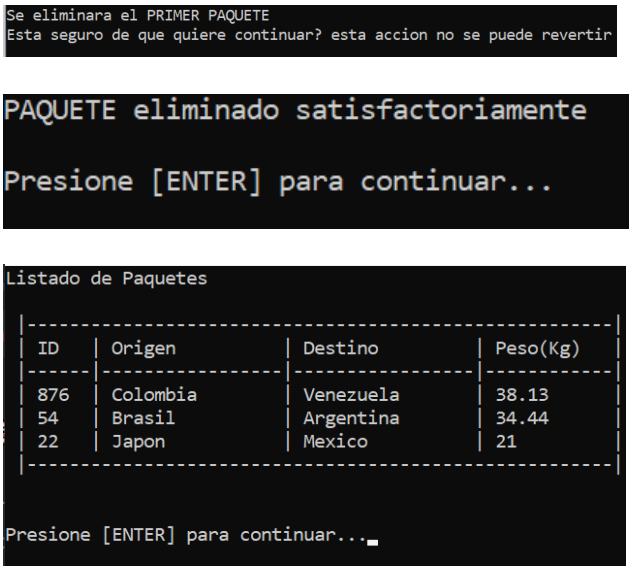
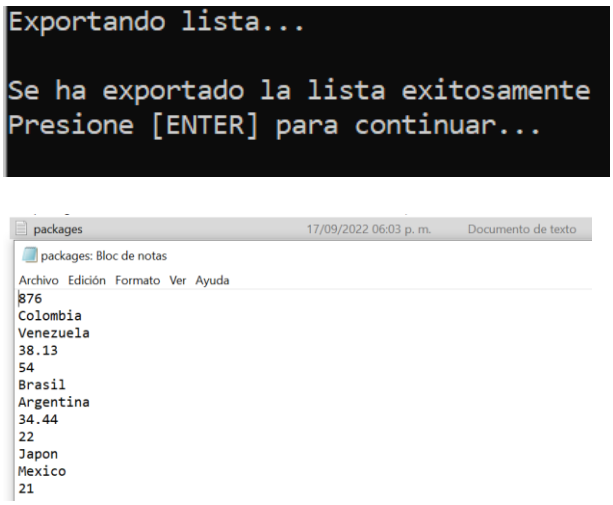
☑ El reporte está en formato Google Docs o PDF.
☑ El reporte sigue las pautas del Formato de Actividades.
☑ Se muestra código y captura de pantalla para agregar un objeto paquete dentro de la lista de la clase Paquetería.
☑Se muestra código y captura de pantalla para eliminar un objeto de la lista de paquete en la clase Paquetería.
☑Se muestra código y captura de pantalla para mostrar la información de toda la lista de paquete en la clase Paquetería.
☑ Se muestra código y captura de pantalla para guardar la lista de paquete en un archivo .txt.
☑ Se muestra captura del pantalla del contenido del archivo .txt.
☑ Se muestra código y captura de pantalla de la opción Recuperar.

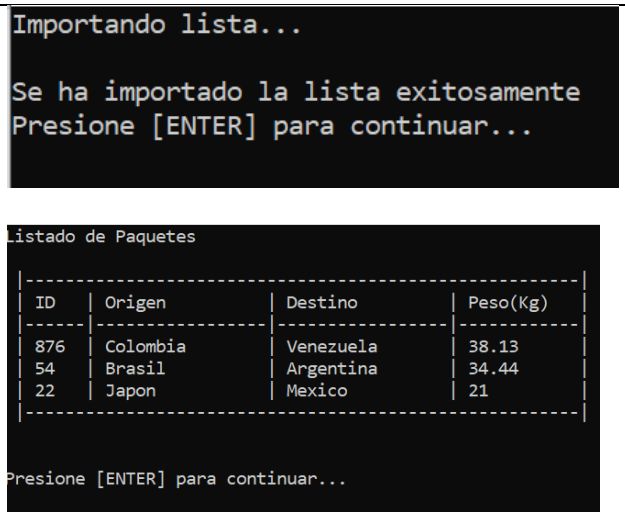
Desarrollo

En esta primera actividad analicé cuales iban a ser las clases necesarias para crear mi lista simplemente ligada para trabajar con el sistema de paquetería, entre ellos se encuentran los paquetes, la lista, los nodos para la lista, la paquetería en si misma cuyo nombre me parecía que era “Courier” en ingles y por ultimo el nodo, en los headers también tenemos la clase exception pero simplemente está ahí para controlar ciertos errores que se puedan dar con la lista por problemas de memoria entre algunos otros.
Ya había trabajado anteriormente con listas ligadas sin embargo me tuve que adaptar en todo lo que era el menú de la paquetería y la sobrecarga de operadores para que el respaldo y la recuperación de archivos fuera exitosa, así como crear métodos nuevos para la lista ya que anteriormente he agregado elementos al final o borrado en la posición que yo quisiera, sin embargo, ahora todo se manejaba desde la primera posición.



Menú principal		El menú cuenta con las opciones requeridas y se demostrara cada una de sus funciones (Eliminar todos es una manera practica de eliminar la lista para poder recuperarla sin la necesidad de cerrar el programa).
Agregar paquete		Aquí podemos ver que se ingresan los valores de los atributos del objeto paquete.

Información general de todos los paquetes		Loa valores se guardan exitosamente y se ven representados en una tabla.
Eliminación de paquetes		El paquete es eliminado de la lista y no se ve representado en la tabla desplegable.
Guardado de la lista		La lista es guardada en un archivo txt en la misma carpeta del proyecto o ejecutable.

Recuperación de la lista	 	La lista es recuperada después de eliminar la lista por completo o cerrar el programa, se puede apreciar como los datos siguen presentes.

Conclusiones

Uno de los problemas con los que me encontré fueron que de buenas a primeras la lista no funcionaba, pero debido a que la lista que era mandada en el main la declaré como apuntador, la solución fue quitar el “*” y mejor lo asigné como una nueva lista y dentro de la paquetería.

Otro de los asuntos fue la recuperación de los archivos debido a que trate de recuperarlos por medio de la sobrecarga del istream dentro de la clase paquete, y cada atributo lo guardaba como si cada atributo fuera un objeto, me di cuenta por un trabajo que había realizado antes y todo marchó bien.

Referencias

Con profesor Alfredo Gutiérrez tuve que trabajar durante el semestre pasado con el guardado y recuperación de la información de mis archivos, así como las listas ligadas.

Código

Agregar un objeto paquete en la lista

```
void Courier::addPackage(){

    string myStr;

    PackageNode* pos;
    Package myPackage;

    system("cls");

    cout << "Agregar nuevo paquete" << endl << endl;

    cout << "ID de paquete: ";
    getline(cin, myStr);

    myPackage.setId(myStr);

    pos = packageListRef->findData(myPackage);

    if(pos != nullptr){
        cout << "ID ya registrada";

        pause();

        return;
    }

    cout << "Origen del paquete: ";
    getline(cin, myStr);

    myPackage.setOrigin(myStr);

    cout << "Destino del paquete: ";
    getline(cin, myStr);

    myPackage.setDestiny(myStr);

    cout << "Peso del paquete(kg): ";
    cin >> myStr;

    myPackage.setWeight(myStr);


    try{
        packageListRef->insertFirst(myPackage);
    }catch (ListException ex){
        cout << "Ha ocurrido un problema: ";
        cout << ex.what() <<endl;
```

```
        cout << "Si vuelve a ocurrir llame al gerente" << endl << endl;

        return;
    }

    cout << endl << "Paquete agregado de manera exitosa";
    pause();
}

void PackageList::insertFirst(const Package& e){

    PackageNode* aux(new PackageNode(e));

    if(aux == nullptr){
        throw ListException("Memoria no disponible, tratando de insertar");
    }

    aux->setNext(anchor);
    anchor = aux;

}
```

Se tiene que enviar un objeto de tipo paquete que ingresamos y enviamos por medio de una referencia a la lista, se declara un apuntador con la dirección del paquete y si dicho auxiliar es nulo quiere decir que no hay memoria disponible y lanzara un error, si se pasa entonces se realiza un religado donde se asigna a la posición siguiente del ancla (aquella ancla cuando esta nula indica que la lista esta vacía, y cuando hay elementos siempre apunta al primer elemento de la lista) y ahora el ancla apunta al auxiliar que es nuestro primer elemento.

Eliminar un objeto de la lista de paquete en la clase paquetería

```
void Courier::deleteOnePackage(){
    char op;
    do{
        system("cls");
        cout << "Se eliminara el PRIMER PAQUETE" << endl;
        cout << "Esta seguro de que quiere continuar? esta accion no se puede
revertir (s/n)";
        cin >> op;
        cin.ignore();
        op = toupper(op);
        if (op == 'N'){
            cout << "No se eliminara el PAQUETE..." << endl << endl;
            pause();
            return;
        }
        else if(op == 'S'){
            packageListRef->deleteData();
            system("cls");
            cout << "PAQUETE eliminado satisfactoriamente" << endl << endl;
            pause();
        }
        else{
```

```
            cout << "Escoja una opcion valida..." << endl;
            pause();
        }
    }while(op != 'S');
}
void PackageList::deleteData(){
    PackageNode* aux;

    aux = anchor;
    anchor = anchor->getNext();

    delete aux;
}
```

Para eliminar se reasigna el ancla al siguiente elemento de la lista y el auxiliar que era la primera posición es eliminada, para hacer una eliminación se requiere una confirmación por parte del usuario.

Mostrar la información de toda la lista

```
void Courier::showAllPackages(){
    system("cls");
    cout << "Listado de Paquetes" << endl << endl;

    cout << packageListRef->toString();

    cout << endl << endl;
    pause();
}

string PackageList::toString()const{

    PackageNode* aux(anchor);
    string result;

    result += " |-------------------------------------------------------|";
    result += "\n";
    result += " | ID   | Origen          | Destino         | Peso(Kg)   |";
    result += "\n";
    result += " |------|-----------------|-----------------|------------|";

    result += "\n";
    while(aux != nullptr){
        result += aux->getData().toString();
        result += "\n";

        aux = aux->getNext();
    }
    result += " |-------------------------------------------------------|";
    result += "\n";
    return result;
}
string Package::toString(){
    string result;
    string aux;
```

```
    result += " | ";

    aux = id;
    aux.resize(4,' ');
    result += aux;

    result += " | ";

    aux = origin;
    aux.resize(15,' ');
    result += aux;

    result += " | ";

    aux = destiny;
    aux.resize(15,' ');
    result += aux;

    result += " | ";

    aux = weight;
    aux.resize(10,' ');
    result += aux;

    result += " | ";

    return result;
}
```

Aquí hacemos uso de 2 métodos con nombre to string, se encargan de convertir los elementos de la lista en texto por medio de un recorrido de lista que llamamos desde nuestra referencia a la lista e imprimiendo en pantalla con un cout.

Guardar la lista

```
void Courier::writeToDisk(){
    system("cls");

    cout << "Exportando lista..." << endl << endl;

    try{
        packageListRef->writeToDisk("packages.txt");
    }catch(ListException ex){
        cout << "Surgio un problema" << endl;
        cout << ex.what() << endl;

        pause();

    return;
    }

    cout << "Se ha exportado la lista exitosamente" << endl;
```

```
    pause();
}

void PackageList::writeToDisk(const std::string& fileName){
    ofstream myFile;

    myFile.open(fileName, myFile.trunc);

    if(!myFile.is_open()){
        string msg("No se pudo abrir el archivo ");
        msg+= fileName;
        msg += " para escritura, PackageList::writeToDisk";
        throw ListException(msg);
    }

    PackageNode* aux(anchor);

    while(aux != nullptr){
        myFile << aux->getData() << endl;

        aux = aux->getNext();
    }

    myFile.close();
}

ostream& operator << (ostream& os, Package& p){

    os << p.id << endl;
    os << p.origin << endl;
    os << p.destiny << endl;
    os << p.weight;

    return os;
}
```

El guardar el archivo depende del ifstream para abrir el archivo y de ofstream con una sobrecarga del mismo en la clase de package y se va realizando un recorrido de lista que gracias a la sobrecarga puede ir guardando todos los elementos de cada objeto sin mayor complicación.

Recuperar

```
void Courier::readFromDisk(){
    system("cls");

    cout << "Importando lista..." << endl << endl;

    try{
        packageListRef->readFromDisk("packages.txt");
    }catch(ListException ex){
        cout << "Surgio un problema" << endl;
        cout << ex.what() << endl;

        pause();
```

```
    return;
    }

    cout << "Se ha importado la lista exitosamente" << endl;
    pause();
}

void PackageList::readFromDisk(const std::string& fileName){
    ifstream myFile;

    myFile.open(fileName);

    if(!myFile.is_open()){
        string msg("No se pudo abrir el archivo ");
        msg+= fileName;
        msg += " para lectura, PackageList::readFromDisk";

        throw ListException(msg);
    }

    deleteAll();

    Package myPackage;
    PackageNode* lastNode(nullptr);
    PackageNode* newNode;

    while(myFile >> myPackage){
        if((newNode = new PackageNode(myPackage)) == nullptr){
            myFile.close();

            throw ListException("Memoria no disponible, readFromDisk");
        }

        if(lastNode == nullptr){
            anchor = newNode;
        }
        else{
            lastNode->setNext(newNode);
        }

        lastNode = newNode;
    }

    myFile.close();
}
istream& operator >> (istream& is, Package& p){
    getline(is, p.id);
    getline(is, p.origin);
    getline(is, p.destiny);
    getline(is, p.weight);

    return is;
}
```

La recuperación de los datos depende por su parte de istream para guardar y del ifstream igualmente para abrir el archivo, con istream tambien hay una sobrecarga de operadores para que pueda leer cada 4 lineas y asignar los valores que se encuentre en un objeto nuevo de la clase package, al recuperar los datos los guarda y les asigna un nodo y empieza a ligar los elementos nuevamente en una lista para su uso en el programa.

Main

```
#include <iostream>
#include <cstdlib>
#include <string>
```

```
#include "courier.h"
#include "packagelist.h"

int main(){
    new Courier(new PackageList);
}
```

Finalmente el main, este se encarga de crear la lista y mandársela a una nueva paquetería desde la cual vamos a manipular todas las funciones de la lista.