ACTIVIDAD 01 / / REPASO DE PROGRAMACIÓN

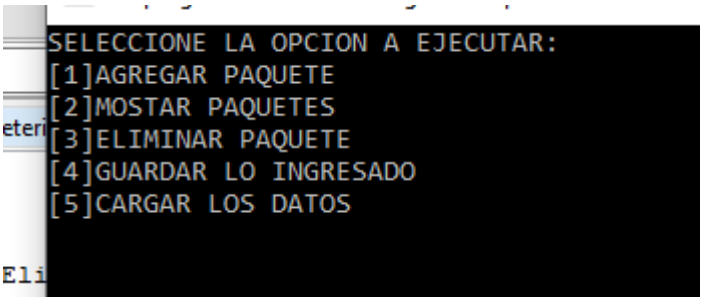
BERACOECHEA ROSALES JOSE FRANCISCO

SEMINARIO DE ALGORITMIA

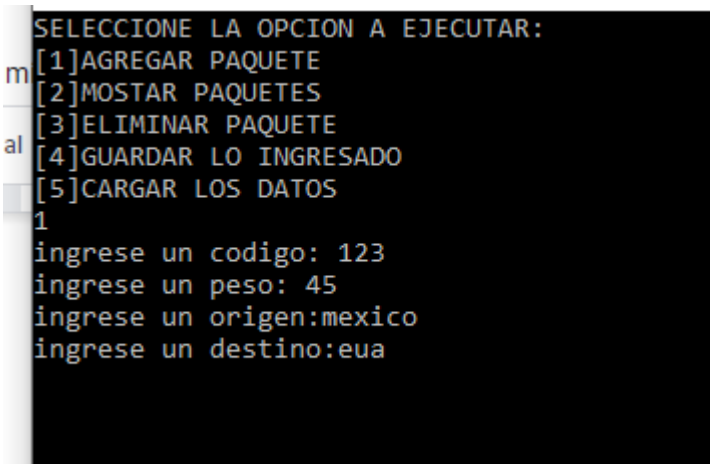
- [ ] El reporte está en formato Google Docs o PDF.
- [ ] El reporte sigue las pautas del Formato de Actividades .
- [ ] Se muestra código y captura de pantalla para agregar un objeto paquete dentro de la lista de la clase Paqueteria.
- [ ] Se muestra código y captura de pantalla para eliminar un objeto de la lista de paquete en la clase Paqueteria.
- [ ] Se muestra código y captura de pantalla para mostrar la información de toda la lista de paquete en la clase Paqueteria.
- [ ] 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:

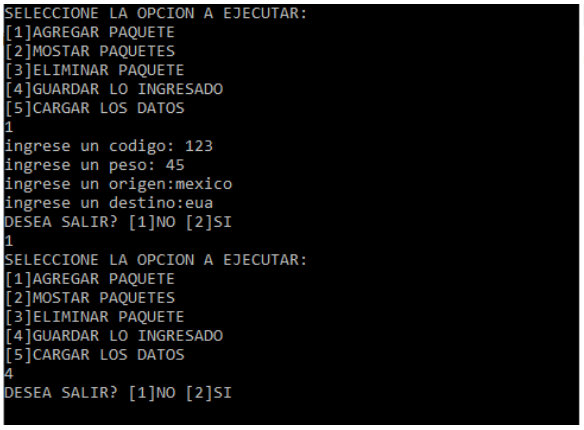
ingresamos al switch de opciones:



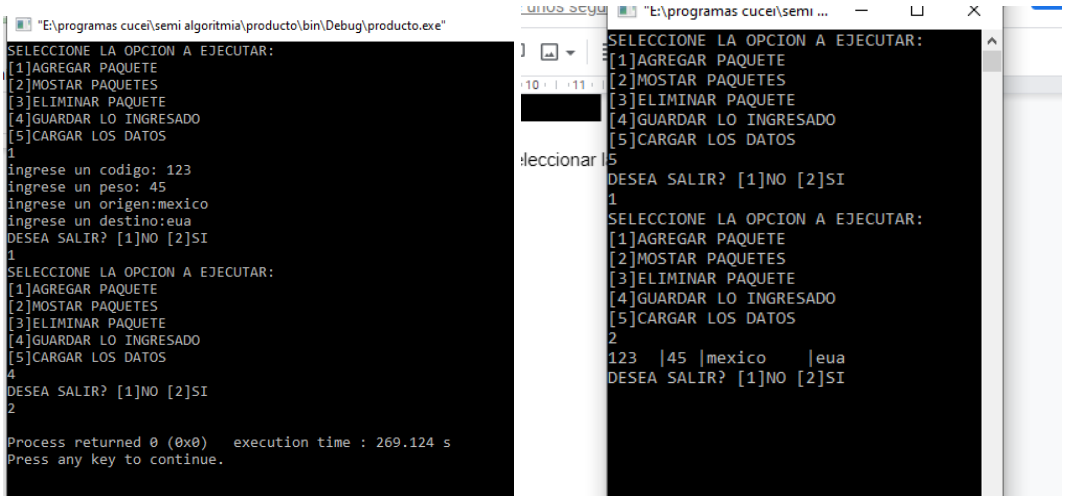
seleccionamos la opción 1 y agregamos un paquete:



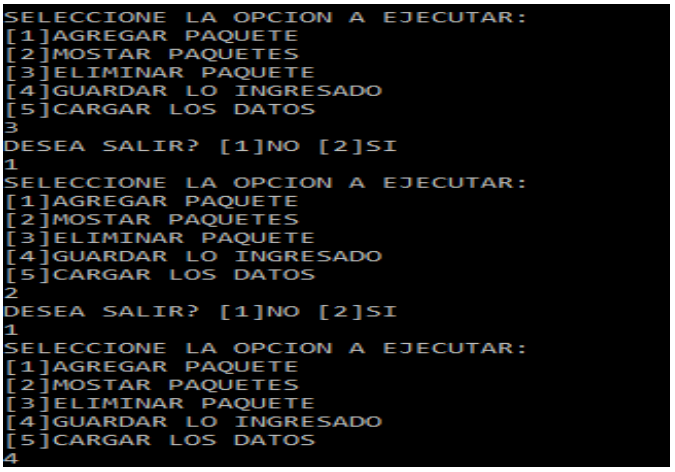
seleccionamos la opción 4 para guardar el paquete en el txt:



salimos y cerramos el programa para después seleccionar la opción 5 y cargar los datos



finalmente eliminamos el registro y guardamos los cambios



CONCLUSIÓN:

DESDE EL SEMESTRE PASADO HABÍA SUFRIDO CON EL MANEJO DE ARCHIVOS TXT, COSA QUE CON ESTA MINI PRACTICA AHORA TODO QUEDA MUCHO MÁS CLARO Y VEO QUE NO ERA TAN DIFÍCIL COMO LO PENSABA EN UN INICIO.

REFERENCIAS:

material proporcionado por el profesor así como conocimientos previos del semestre pasado

https://youtu.be/g3gtAhHgYcA

https://youtu.be/QqU8emKoiIA

CÓDIGO:

main:

```
#include <iostream>

#include<string>

#include"paqueteria.hpp"

#include"paquete.hpp"

using namespace std;

 Paquete p;

    CLista lista;


int main()

{

    int continuar;

    int codigo;

    int peso;

    string o;

    string d;
```

```
    int menu;
    do{
    cout<<"SELECCIONE LA OPCION A EJECUTAR: " <<endl<<"[1]AGREGAR PAQUETE\n[2]MOSTAR PAQUETES\n[3]ELIMINAR PAQUETE\n[4]GUARDAR LO INGRESADO\n[5]CARGAR LOS DATOS\n";
    cin>>menu;
    switch(menu){
       case 1:{
    Paquete* paquete=new Paquete;
    cout << "ingrese un codigo: ";
    cin>>codigo;
    paquete->setID(codigo);

    cout << "ingrese un peso: ";
    cin>>peso;
    paquete->setPeso(peso);

    cout << "ingrese un origen:";
    cin.ignore();
    getline(cin,o);
    paquete->setOrigen(o);

    cout << "ingrese un destino:";
    getline(cin,d);
    paquete->setDestino(d);

    lista.insertarInicio(paquete);
```

```
    break;
       }
    case 2:{
       lista.Imprime();
    break;
    }
    case 3:{
       lista.EliminarInicio();
       break;
    }
    case 4:{
       lista.Guardar();
    break;
    }
    case 5:{
       lista.Recuperar();
    }
    }
    cout<<"DESEA SALIR? [1]NO [2]SI"<<endl;
    cin>>continuar;
    }while(continuar != 2);
    return 0;
}
```

:

clase paquete:

```
#ifndef PAQUETE_HPP_INCLUED
#define PAQUETE_HPP_INCLUED

#include<iostream>
#include<string>

using namespace std;

class Paquete{
private:
    int id;
    int peso;
    string origen;
    string destino;

public:

    Paquete();
    Paquete(const Paquete&);

    virtual string toStringP()const;

    int getID()const;
    int getPeso()const;
    string getOrigen()const;
```

```
    string getDestino()const;

    void setID(const int&);
    void setPeso(const int&);
    void setOrigen(const string&);
    void setDestino(const string&);

    Paquete & operator =(const Paquete&);


};

Paquete::Paquete(){}


Paquete& Paquete::operator = (const Paquete& p) {
    id = p.id;
    peso= p.peso;
    origen = p.origen;
    destino = p.destino;

    return *this;
    }
string Paquete::toStringP() const
{
    string paquete;
```

```
    paquete+=to_string(id);
    paquete.resize(5,' ');
    paquete+="|";

    paquete+=to_string(peso);
    paquete.resize(9,' ');
    paquete+="|";

    paquete+=origen;
    paquete.resize(20,' ');
    paquete+="|";

    paquete+=destino;
    paquete.resize(35,' ');
    return paquete;
}

int Paquete::getID() const
{
    return id;
}

int Paquete::getPeso() const
{
    return peso;
}
```

```
string Paquete::getOrigen() const
{
    return origen;

}

string Paquete::getDestino() const
{
    return destino;

}

void Paquete::setID(const int &I)
{
    id =I;
}

void Paquete::setPeso(const int &P)
{
    peso =P;

}

void Paquete::setOrigen(const string &O)
{
```

```
    origen =O;
}

void Paquete::setDestino(const string &D)
{
    destino =D;
}

#endif // PAQUETE_HPP_INCLUED
```

clase paqueteria o lista:

```
#ifndef PAQUETERIA_HPP_INCLUED
#define PAQUETERIA_HPP_INCLUED

#include <iostream>
#include<fstream>
#include<string>
#include"nodos.hpp"

using namespace std;
class CLista{
private:
    CNodo Inicio;
    CNodo Final;
    CNodo *h;
```

```
public:
    CLista();
    ~CLista();
    void insertarInicio(Paquete* fData);
    void Vaciar();
    void Imprime();
    void Guardar();
    void Recuperar();

    bool isEmpty();

    int Size();

    Paquete* EliminarInicio();



};
CLista::CLista() {
  Inicio.pSiguiente = &Final;
  Final.pAnterior =& Inicio;

    }

CLista::~CLista() {
```

```
    }

void CLista::insertarInicio(Paquete* fData) {
     Inicio.insertarAdelante(fData);
    }

void CLista::Vaciar() {
   CNodo* p=Inicio.pSiguiente;
    CNodo* x;
    while (p != &Final) {
        x =p ;
        p = p->pSiguiente;
        delete x;
    }
    Inicio.pSiguiente = &Final;
    Final.pAnterior = &Inicio;
    cout<<"SE ELIMINO TODO CORRECTAMENTE"<<endl;
    }

void CLista::Imprime() {
    CNodo *p = Inicio.pSiguiente;
    while(p != &Final){
        cout<<p -> Contenido->toStringP()<< endl;
        p = p -> pSiguiente;
    }
    }
```

```
bool CLista::isEmpty() {
  if(Inicio.pSiguiente == &Final){
    return true;
  }
    return false;
   }

int CLista::Size() {
    CNodo *p = Inicio.pSiguiente;
 int nodos = 0;

     while(p != &Final){
          nodos++;
       p = p -> pSiguiente;
     }
 return nodos;
   }

Paquete* CLista::EliminarInicio() {
return Inicio.eliminarSiguiente();
 }

void CLista::Guardar(){
   ofstream archivo("datos.txt");
   if(archivo.is_open()){
```

```
            CNodo *p = Inicio.pSiguiente;
    while(p != &Final){
        archivo<<p -> Contenido->getID()<< endl;
        archivo<<p -> Contenido->getPeso()<< endl;
        archivo<<p -> Contenido->getOrigen()<< endl;
        archivo<<p -> Contenido->getDestino()<< endl;

        p = p -> pSiguiente;
    }

    }
    archivo.close();
}

void CLista::Recuperar(){
    ifstream archivo("datos.txt");

    if(archivo.is_open()){
        string linea;
        int codigo,peso;
        while(true){

        Paquete* P=new Paquete;

        getline(archivo,linea);//ID
        if(archivo.eof()){
```

```
            break;
        }
        codigo=stoi(linea);//conversion a entero
        P->setID(codigo);

        getline(archivo,linea);//peso
        peso=stoi(linea);
        P->setPeso(peso);

        getline(archivo,linea);//origen
        P->setOrigen(linea);

        getline(archivo,linea);//destino
        P->setDestino(linea);

        insertarInicio(P);
    }
    }

}

#endif // PAQUETERIA_HPP_INCLUED
```

clase nodos:

```
#ifndef NODOS_HPP_INCLUED
#define NODOS_HPP_INCLUED
#include"paquete.hpp"
 class CNodo{
 private:
    CNodo* pSiguiente;
     CNodo*pAnterior;
     Paquete* Contenido;
 public:

    void insertarAdelante(Paquete* cData);
    Paquete* eliminarSiguiente();
    friend class CLista;
 };

 void CNodo::insertarAdelante(Paquete* cData)
{
  CNodo* pNuevo = new CNodo;

  pNuevo->pSiguiente = this -> pSiguiente;
  pNuevo-> pAnterior = this;
  this -> pSiguiente = pNuevo;
  this -> pSiguiente -> pSiguiente -> pAnterior =pNuevo;
  pNuevo -> Contenido = cData;
}
```

```
Paquete* CNodo::eliminarSiguiente() {

    Paquete* x;
    CNodo* p=this->pSiguiente;
    x = p->Contenido;
    this->pSiguiente = this ->pSiguiente->pSiguiente;
    this->pSiguiente->pAnterior = this;



    delete p;
    return x;

}

#endif // NODOS_HPP_INCLUED
```