Universidad De Las Fuerzas Armadas ESPE

Grupo #12

Integrantes:

Luca de Veintemilla

Kevin Vargas

Fecha: 09-06-2022

NRC: 4698

2022-2023

UniTest

Clases

Login.h

```
#ifndef LOGIN_H_INCLUDED
#define LOGIN_H_INCLUDED

#include <iostream>
#include <string>
#include <cstdlib>
#include <conio.h>
#include <string.h>
#include <windows.h>
#include <fstream>
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>

#define USER2 "admin"
#define PASS2 "admin"
#define USER  ""
#define PASS ""
#define ENTER 13
#define BACKSPACE 8

#include "Ingreso.h"
#include "Validacion.h"
#include "Menu.h"
#pragma once

using namespace std;

struct usuario
{
    string cedula;
    string apellido;
    string nombre_usuario;
    string nombre;
    string contrasena;
}persona[30];


int crear_usuario();
void superadmin();
void login ();
void usuarionormal();
//
int contadordeletras(string cedula_);
bool comprobador(string cedula_);



void superadmin(){
    int user=0;
    int op;
```

```
    system("cls");
    cout<<"Registro de nuevo usuario"<<endl;
    system("cls");
    crear_usuario();
    fflush(stdin);
    login();
}


void usuarionormal()
{
    int cont=0;

    if(cont == 0){
        Menu menu;
        cont = menu.mostrar_menu();
        //cont++;
    }else{
        login();
    }
}


void login ()
{
    string usuario, password;

    int contador = 0;
    bool ingresa = false;

    do {
    system("cls");
        cout << "\t\t\tLOGIN DE USUARIO" << endl;
        cout << "\t\t\t----------------" << endl;
        cout << "\n\tUsuario: ";
        getline(cin, usuario);

        char caracter;

        cout << "\tPassword: ";
        caracter = getch();

        password = "";

        while (caracter != ENTER) {

        if (caracter != BACKSPACE) {
            password.push_back(caracter);
            cout << "*";
        } else {
            if (password.length() > 0) {
                cout << "\b \b" ;
                password = password.substr(0, password.length() - 1);
            }
        }

        caracter = getch();
```

```
        }


        if (usuario == USER  && password == PASS) {
            ingresa = true;
            fflush(stdin);
            superadmin();
        } else {
            ingresa = false;
        }

        if ((usuario == USER2 && password == PASS2)){
            fflush(stdin);
            usuarionormal();
        }else{
            ingresa = false;
        }


        for(int i=0; i<=30; i++){
            if ((persona[i].nombre_usuario == usuario &&
persona[i].contrasena==password))
            {
                fflush(stdin);
                usuarionormal();
            }else{
                ingresa = false;
            }
        }
    } while (ingresa == false && contador < 3);

    if (ingresa == false) {
        cout << "\n\tUsted no pudo ingresar al sistema. ADIOS" <<
endl;
    }

    cin.get();
}



int crear_usuario()
{
    int i=0;
    int j=0;
    char prueba[30];
    int band=false;
    Validacion validar;
    string respuesta;

        cout<<"*****REGISTRO DE USUARIOS*****"<<endl;
        char nombre[100];
        strcpy(nombre, validar.ingresoString("Ingrese su nombre y
apellido\n"));
        string str(nombre);
        persona[i].nombre = nombre;
        //cin>>persona[i].nombre;
```

```
        cout<<endl;


        //cout<<"Ingrese su cedula"<<endl;
        char cedula[100];
        do{
        strcpy(cedula, validar.ingresoEntero("Ingrese la cedula \n"));
        string apl(cedula);
        persona[i].cedula = cedula;
        if(comprobador(cedula) == true){ //HECHO
            cout << "\nHa pasado mas de 10 caracteres o ha escrito
menos de 10\n";
        }
        else if(comprobador(cedula) == false){
            cout << "\nTiene los 10 caracteres, por favor
continue...\n";
        } //HECHO
        cout<<endl;
        }while(comprobador(cedula)==true);



        cout<<"Ingrese su nombre de usuario"<<endl;
        cin>>persona[i].nombre_usuario;
        cout<<"Ingrese la contrasena"<<endl;
        do{
        cin>>persona[i].contrasena;
        strcpy(prueba,persona[i].contrasena.c_str());
           if( prueba[i] == '1' || prueba[i] == '0' || prueba[i] ==
'2' || prueba[i] == '3' || prueba[i] == '4' || prueba[i] == '5' ||
prueba[i] == '6' || prueba[i] == '7' || prueba[i] == '8' || prueba[i]
== '9' )
            {
                band=true;
                break;
            }else{
                 cout<<"Ingreso no valido"<<endl;
                 //cin>>persona[i].contrasena;
                 }
        }while(band==true);
        i++;
        cout<<"Desea ingresar otro usuario"<<endl;
        cout<<"Presione n para terminar"<<endl;
        system("cls");
}



    int contador(string cedula_){
        string cedula;
        cedula = cedula_;
        int numerodeletras;
        numerodeletras = cedula.size();
        return numerodeletras;
    }

    bool comprobador(string cedula_){
```

```
        bool comprobar = false;
        string copiadecedula = cedula_;
        if(contador(copiadecedula) > 10){
            comprobar = true;
        }
        else if(contador(copiadecedula) < 10){
            comprobar = true;
        }
        return comprobar;
    }


#endif // LOGIN_H_INCLUDED
```

Menu.h

```
#include<stdlib.h>
#include"Lista.h"
#include "Persona.h"
#include "Tools.h"
#pragma once


using namespace std;
class Menu{
        public:
                Menu();
                int mostrar_menu();
                int opciones_menu();
   private:
      string dim;
      Ingreso ingreso;
      Validacion validar;
};
Menu::Menu(){
}
int Menu::opciones_menu(){
        int opc;
   bool gameover = false;
   char puntero = '->';
   int coord_y;


        cout<<"\t--Opciones de compra--"<<endl;
        //Gotoxy(coord_x,coord_y);
        printf("\t");
        cout<<"1. Hacer pedido de compra"<<endl;//Insertar inicio
        printf("\t");
        cout<<"2. Comprar desde la lista de ventas"<<endl;//(eliminar desde la lista de ventas)
        printf("\t");
        cout<<"3. Eliminar una compra"<<endl;
        printf("\t");
        cout<<"4. Imprimir la lista"<<endl;
        printf("\t");
        cout<<"5. Exportar datos de compra .txt"<<endl;
        cout<<"-----------------------------------------------"<<endl;
```

```
            cout<<"\t--Opciones de venta--"<<endl;
            printf("\t");
            cout<<"6. Insertar un nodo en la lista de venta"<<endl;
            printf("\t");
            cout<<"7. Vender una unidad existente"<<endl;
            printf("\t");
            cout<<"8. Eliminar de la lista de venta"<<endl;
            printf("\t");
cout<<"9. Imprimir la lista la lista de venta"<<endl;
            printf("\t");
            cout<<"10. Exportar datos de venta .txt"<<endl;
            printf("\t");
            cout<<"11. Crear Marquesina"<<endl;
            printf("\t");
            cout<<"12. Crear Marquesina"<<endl;
            printf("\t");
            cout<<"12. Salir"<<endl;
            //dim = ingreso.leer("Seleccione una opcion: ", 1);
            //istringstream(dim)>>opc;

while(!gameover)
{

   Gotoxy(7,coord_y );

   cout<<" ";

   if(coord_y<=0)
   {
      coord_y++;
   }
   if(coord_y>=15)
   {
      coord_y--;
   }

   if(kbhit())
   {
      switch(getch())
      {
      case 72: //arriba
         coord_y--;
         break;

      case 80: //abajo
         coord_y++;
         break;

      case 13: //enter
         opc = coord_y;
         cout<<opc<<endl;
         gameover = true;
         break;
      }
   }
```

```
        Gotoxy(7,coord_y );
        cout<<puntero;
        cout<<"";
    }

            return opc;
}
int Menu::mostrar_menu(){
    bool mouse = true;
    int opcion = 0;
            int opc,tam ;
            Lista<Persona> *lista_compra = new Lista<Persona>();
    Lista<Persona> *lista_venta = new Lista<Persona>();
    Lista<Persona> *lista = new Lista<Persona>();
    int salida = 0;

            do{
                    system("cls");

        /*
        int coord;
        coord = place();
        Sleep(1000);
        */

                    opc = opciones_menu();
                    switch(opc)
                    {
                            case 1:{
            system("cls");

            char nombre[100];
                                    char celular[100];
                                    char modelo[100];
                                    char color[100];
            float precio;


                                    //cout<<"Ingrese el nombre"<<endl;
            strcpy(nombre, validar.ingresoString("Ingrese su nombre y apellido \n"));
            cout<<endl;
                                    //memcpy(_nombre, *nombre, 50);

                                    cout<<"Ingrese el celular (Marca)"<<endl;
                                    gets(celular);
                                    cout<<"Ingrese el modelo"<<endl;
                                    gets(modelo);
                                    //cout<<"Ingrese el color"<<endl;
                                    //gets(color);
                                    strcpy(color, validar.ingresoString("Ingrese el color \n"));
            cout<<endl;
                                    cout<<"Ingrese precio"<<endl;
                                    precio = validar.convertirFlotante();

                                    Persona *temp=new
Persona(nombre,celular,modelo,color,precio);
```

```
                                    //dim = ingreso.leer("\t--Insertar un nodo al final--\nIngrese un
dato:", 1);
                                            //istringstream(dim)>>dato;
                                            cout<<"\t--Insertar un celular al final:"<<endl;
                                            lista_compra->insertarUltimo(*temp);
                                            lista_compra->imprimir();
                                            fflush(stdin);
                                            break;
                                    }

                                    case 2:{//eliminar desde la lista de ventas
            system("cls");
                                       int dato;
                                       lista_venta->imprimir();
                                       cout<<"Eliminar"<<endl;
            if(lista_venta->vacia() == false){
               dim = ingreso.leer("\nElija el celular que desea comprar: ", 1);
               istringstream(dim)>>dato;
               Persona temp = lista_venta->copiar(dato);
               lista_compra->insertarUltimo(temp);
            }else{
               cout<<"La lista esta vacia"<<endl;
            }
            fflush(stdin);
                                       //lista_venta->imprimir();
                                            break;
                                    }

                                    case 3:{//Eliminar el nodo X de la LISTA DE COMPRA
                                       system("cls");
                                       int dato;
                                       lista_compra->imprimir();
                                       cout<<"Eliminar"<<endl;

                                       if(!lista_compra->vacia()){
               dim = ingreso.leer("\nElija el celular que desea eliminar: ", 1);
               istringstream(dim)>>dato;
               lista_compra->borrar(dato);
               lista_compra->imprimir();
                                       }else{
               cout<<"No hay articulos que eliminar"<<endl;
                                       }
            fflush(stdin);

                                            break;
                                    }


                                    case 4:{//Imprimir lista LA LISTA DE COMPRA
                                       system("cls");
                                            lista_compra->imprimir();
                                            fflush(stdin);
                                            break;
                                    }

                                    case 5:{//Exportar lista LA LISTA DE COMPRA .txt
                                       system("cls");
```

```
                                    string nombre = "ListaCompra";
            lista_compra->exportartxt(nombre);
            fflush(stdin);
                                            break;
                                    }


                                    /*-------------------------------------------------*/

                                    case 8:{//Insertar nodo al final de la lista de venta  crear venta
                                       system("cls");
                                       char nombre[100];
                                            char celular[100];
                                            char modelo[100];
                                            char color[100];
                                            float precio;

                                            //cout<<"Ingrese el nombre"<<endl;
                                            //gets(nombre);

                                            strcpy(nombre, validar.ingresoString("Ingrese su nombre y
apellido \n"));
            cout<<endl;

                                            cout<<"Ingrese el celular (Marca)"<<endl;
                                            gets(celular);
                                            cout<<"Ingrese el modelo"<<endl;
                                            gets(modelo);

                                            //cout<<"Ingrese el color"<<endl;
                                            //gets(color);

                                            strcpy(color, validar.ingresoString("Ingrese el color \n"));
            cout<<endl;

                                            cout<<"Ingrese precio"<<endl;
                                            precio = validar.convertirFlotante();

                                            Persona *temp=new
Persona(nombre,celular,modelo,color,precio);
                                       //dim = ingreso.leer("\t--Insertar un nodo al final--\nIngrese un
dato:", 1);
                                            //istringstream(dim)>>dato;
                                            cout<<"\t--Insertar un celular al final:"<<endl;
                                            lista_venta->insertarUltimo(*temp);
                                            lista_venta->imprimir();
                                            fflush(stdin);
                                            break;
                                    }

                                    case 9:{//Eliminar el nodo X de la LISTA DE COMPRA
            system("cls");
                                       int dato;
                                       lista_compra->imprimir();
                                       cout<<"Eliminar"<<endl;

            if(lista_venta->vacia() == false){
```

```
   dim = ingreso.leer("\nElija el celular que desea vender: ", 1);
   istringstream(dim)>>dato;
   Persona temp = lista_compra->copiar(dato);
   lista_venta->insertarUltimo(temp);
}else{
   cout<<"No hay articulos a la venta"<<endl;
}
                                    //lista_venta->imprimir();
                                    fflush(stdin);
                                            break;
                                }

                                case 10:{//Eliminar lista LA LISTA DE VENTA

system("cls");
                                    int dato;
                                    lista_venta->imprimir();
                                    cout<<"Eliminar"<<endl;

                                            if(!lista_venta->vacia()){
   dim = ingreso.leer("\nElija el celular que desea eliminar: ", 1);
   istringstream(dim)>>dato;
   lista_venta->borrar(dato);
                                            }else{
   cout<<"\n\t--La lista esta vacia-"<<endl;
                                            }

                                    lista_venta->imprimir();
                                    fflush(stdin);
                                            break;

                                }

                                case 11:{//Imprimir lista LA LISTA DE VENTA

                                    system("cls");
                                            lista_venta->imprimir();
                                            fflush(stdin);
                                            break;

                                }

                                case 12:{//exportar LA LISTA DE VENTA

system("cls");
                                    string nombre = "ListaVentas";
                                    lista_compra->exportardat(nombre);
fflush(stdin);

                                            break;

                                }

                                case 13:{//
                                    system("cls");
                                            lista->marquesina();
                                            break;
```

```
                                        }

                                        case 14:{//
                                           system("cls");
                                                cout<<"--Gracias por usar mi programa--"<<endl;
            mouse = false;
            fflush(stdin);
            salida++;
                                                break;

                                        }
                                        default:{
                                                cout<<"Opcion incorrecta"<<endl;
                                                break;
                                        }

                                }
                                system("pause");
            }while(opc != 14);

   return salida;

}
```

Ingreso.h

```
#include <iostream>
#include "Validacion.h"
#pragma once
using namespace std;
// Clase que gestiona el ingreso de datos
class Ingreso {

public:
            string leer(string,int);
};
string Ingreso::leer(string mensage,int tipo) {
            Validacion validacion;
            string entrada;
            cout << mensage << endl;
            cin >> entrada;
            while (validacion.validar(entrada, tipo)) {
                        cout << "Valor no valido reingrese, ingrese solo numeros" << endl;
                        cin >> entrada;
            }
            return entrada;
}
```

Lista.h

```
#include <iostream>
#include <sstream>
#include <fstream>
#include <string>
#include "Ingreso.h"
#include "Tools.h"
```

```
#include "Persona.h"
#pragma once
using namespace std;

template<class DATO>
class Lista{
            private:
               template<class DATON>
               class Nodo {
      public:
         Nodo(const DATON dat, Nodo<DATON> *sig, Nodo<DATON> *ant) : dato(dat), sig(sig),
ant(ant) {}
         //int info;
         DATON dato;
         Nodo<DATON> *sig;
         //Nodo *sig;
         Nodo<DATON> *ant;
         //Nodo *ant;
      };

                        //Nodo *raiz;
                        Nodo<DATO> *raiz;

            public:
                        Lista() : raiz(NULL){}
      ~Lista();
      void insertarPrimero(const DATO dat);
      void insertarUltimo(const DATO x);
      bool vacia();
      void imprimir();
      int cantidad();
      void borrar(int pos);
      DATO copiar(int pos);
      void exportartxt(string);
      void exportardat(string);
      void marquesina();
};



template<class DATO>
Lista<DATO>::~Lista()
{
   if (raiz != NULL) {
      Nodo<DATO> *reco = raiz->sig;
      Nodo<DATO> *bor;
      while (reco != raiz)
      {
         bor = reco;
         reco = reco->sig;
         delete bor;
      }
      delete raiz;
   }
}

template<class DATO>
```

```
void Lista<DATO>::insertarPrimero(const DATO x)
{
   Nodo<DATO> *nuevo = new Nodo<DATO>(x,NULL,NULL);
   nuevo->dato = x;
   if (raiz == NULL)
   {
      nuevo->sig = nuevo;
      nuevo->ant = nuevo;
      raiz = nuevo;
   }
   else
   {
      Nodo<DATO> *ultimo;
      ultimo = raiz->ant;
      nuevo->sig = raiz;
      nuevo->ant = ultimo;
      raiz->ant = nuevo;
      ultimo->sig = nuevo;
      raiz = nuevo;
   }
}

template<class DATO>
void Lista<DATO>::insertarUltimo(const DATO x)
{
   Nodo<DATO> *nuevo = new Nodo<DATO>(x,NULL,NULL);;
   nuevo->dato = x;
   if (raiz == NULL)
   {
      nuevo->sig = nuevo;
      nuevo->ant = nuevo;
      raiz = nuevo;
   }
   else
   {
      Nodo<DATO> *ultimo;
      ultimo = raiz->ant;
      nuevo->sig = raiz;
      nuevo->ant = ultimo;
      raiz->ant = nuevo;
      ultimo->sig = nuevo;
   }
}

template<class DATO>
bool Lista<DATO>::vacia()
{
   if (raiz == NULL)
      return true;
   else
      return false;
}

template<class DATO>
void Lista<DATO>::imprimir()
{
   if (!vacia()) {
```

```
        Nodo<DATO> *reco = raiz;
        do {
           cout<<reco->dato  <<"-";
           reco = reco->sig;
        } while (reco != raiz);
        cout << "\n";
     }
}

template<class DATO>
int Lista<DATO>::cantidad()
{
   int cant = 0;
   if (!vacia())
   {
      Nodo<DATO> *reco = raiz;
      do {
         cant++;
         reco = reco->sig;
      } while (reco != raiz);
   }
   return cant;
}

template<class DATO>
void Lista<DATO>::borrar(int pos)
{
   if(!vacia()){
      if (pos <= cantidad())
      {
         if (pos == 1)
         {
            if (cantidad() == 1)
            {
               delete raiz;
               raiz = NULL;
            }
            else
            {
               Nodo<DATO> *bor = raiz;
               Nodo<DATO> *ultimo = raiz->ant;
               raiz = raiz->sig;
               ultimo->sig = raiz;
               raiz->ant = ultimo;
               delete bor;
            }
         }
         else {
            Nodo<DATO> *reco = raiz;
            for (int f = 1; f <= pos - 1; f++)
               reco = reco->sig;
            Nodo<DATO> *bor = reco;
            Nodo<DATO> *anterior = reco->ant;
            reco = reco->sig;
            anterior->sig = reco;
            reco->ant = anterior;
            delete bor;
```

```
            }
        }

    }else{
        cout<<"La lista esta vacia"<<endl;
    }
}

template<class DATO>
DATO Lista<DATO>::copiar(int pos)
{
    try{
        if(!vacia()){
            if (pos <= cantidad())
            {
                if (pos == 1)
                {
                    if (cantidad() == 1)
                    {
                        DATO copia = raiz->dato;
                        raiz = NULL;
                        delete raiz;
                        return copia;
                    }
                    else
                    {
                        Nodo<DATO> *bor = raiz;
                        Nodo<DATO> *ultimo = raiz->ant;
                        raiz = raiz->sig;
                        ultimo->sig = raiz;
                        raiz->ant = ultimo;
                        DATO copia = bor->dato;
                        delete bor;
                        return copia;
                    }
                }
                else {
                    Nodo<DATO> *reco = raiz;
                    for (int f = 1; f <= pos - 1; f++)
                        reco = reco->sig;
                    Nodo<DATO> *bor = reco;
                    Nodo<DATO> *anterior = reco->ant;
                    reco = reco->sig;
                    anterior->sig = reco;
                    reco->ant = anterior;
                    DATO copia = bor->dato;
                    delete bor;
                    return copia;
                }
            }
        }
    }catch(char e){
        cout<<"La lista esta vacia"<<endl;
    }

}
```

```
template<class DATO>
void Lista<DATO>::exportartxt(string nombreArchivo)
{
   string nombre = nombreArchivo + ".txt";
   fstream archivo(nombre);

   if(!archivo.is_open()){
      archivo.open(nombre, ios::out);
   }

   if (!vacia()) {
      Nodo<DATO> *reco = raiz;
      do {
         archivo<<reco->dato<<"-";
         reco = reco->sig;
      } while (reco != raiz);
      archivo<<endl;
   }

   cout<<"Exportado con exito"<<endl;

   archivo.close();
}

template<class DATO>
void Lista<DATO>::exportardat(string nombreArchivo)
{
   string nombre = nombreArchivo + ".dat";
   fstream archivo(nombre);

   if(!archivo.is_open()){
      archivo.open(nombre, ios::out);
   }

   if (!vacia()) {
      Nodo<DATO> *reco = raiz;

      do {
         archivo<<reco->dato->getNombre()<<"-";
         reco = reco->sig;
      } while (reco != raiz);
      archivo<<endl;
   }

   cout<<"Exportado con exito"<<endl;

   archivo.close();
}

template<class DATO>
void Lista<DATO>::marquesina()
{
int a=1,b,c=0;
char nombre[50];
cout <<"Ingrese caracteres para crear la marquesina (MAXIMO 25 Caracteres)";
```

```
Gotoxy(1,1);


		do
		{
			c++;
			nombre[c]=getche();
		}while(nombre[c]!=13);


		while(!kbhit())
			{
				cout <<"PRESIONE CUALQUIER TECLA PARA SALIR DEL EFECTO"<<endl;
				for(b=1;b<=c;b++)
				{

					if(a+b>=110) {
						cout <<""<<endl;
						Gotoxy(a+(b-109),1);
						}

					else {
						cout <<""<<endl;
						Gotoxy(a+b,1); }

					cout<<nombre[b];

				}

		Sleep(50);
		Gotoxy(1,1);
 system("cls");
a++;

if(a==110) { a=1; }
}

}
```

Persona.h

```
#ifndef PERSONA_H_INCLUDED
#define PERSONA_H_INCLUDED

#include <string.h>
#include <iostream>
using namespace std;

class Persona
{
		private:
			char *nombre;
			char *celular;
			char *modelo;
			char *color;
```

```
                float precio;

        public:

                Persona(char _nombre[],char _celular[],char _modelo[], char _color[], float
_precio);
                //~Persona();
                Persona &operator=(const Persona &c)
                {
                        if(this!=&c)
                        {
                                nombre = new char[strlen(c.nombre)+1];
                                strcpy(nombre,c.nombre);

                                celular = new char[strlen(c.celular)+1];
                                strcpy(celular,c.celular);

                                modelo = new char[strlen(c.modelo)+1];
                                strcpy(modelo,c.modelo);

                                color = new char[strlen(c.color)+1];
                                strcpy(color,c.color);

                                precio =c.precio;
                        }
                        return *this;
                }
                bool operator>(const Persona &d) const{
                        return strcmp(nombre,d.nombre)>0;
                }
                bool operator<=(const Persona &d) const{
                        return strcmp(nombre,d.nombre)<=0;
                }
                bool operator!=(const Persona &d) const{
                        return strcmp(nombre,d.nombre);
                }
                bool operator<(const Persona &d) const{
                        return strcmp(nombre,d.nombre)<0;
                }

                //Getters
                char *getNombre(){
                return nombre;}
                char *getCelular(){return celular;}
                char *getModelo(){return modelo;}
                char *getColor(){return color;}
                float getPrecio(){return precio;}


};

Persona::Persona(char _nombre[],char _celular[],char _modelo[], char _color[], float _precio){
        nombre = new char[strlen(_nombre)+1];
        strcpy(nombre,_nombre);
        celular = new char[strlen(_celular)+1];
        strcpy(celular,_celular);
        modelo = new char[strlen(_modelo)+1];
```

```
        strcpy(modelo,_modelo);
        color = new char[strlen(_color)+1];
        strcpy(color,_color);
        precio=_precio;
}
//~Persona::Persona(){
//        delete[] nombre;
//}


ostream& operator <<(ostream &os,Persona a)
{
        os<<a.getNombre()<<"\t";
        os<<a.getCelular()<<"\t";
        os<<a.getModelo()<<"\t";
        os<<a.getColor()<<"\t";
        os<<a.getPrecio()<<endl;
        return os;
}

#endif // PERSONA_H_INCLUDED
```

Tools.h

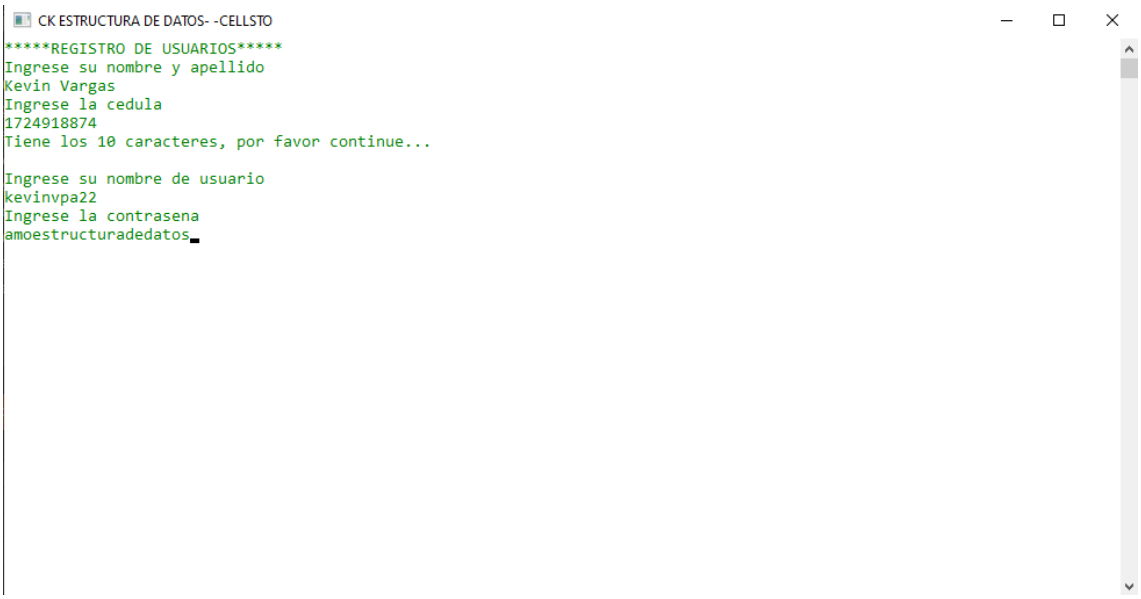
```
#ifndef TOOLS_H_INCLUDED
#define TOOLS_H_INCLUDED

#include <iostream>
#include <conio.h>
#include <time.h>
#include <stdlib.h>
#include <windows.h>
#include <vector>
#include <winuser.h>
#include <conio.h>

using namespace std;

void Gotoxy(int x, int y){
   HANDLE hcon;
   hcon = GetStdHandle(STD_OUTPUT_HANDLE);
   COORD dwPos;
   dwPos.X = x;
   dwPos.Y = y;
   SetConsoleCursorPosition(hcon,dwPos);
}

int place(){
   int coords;
   POINT cursor;
   GetCursorPos(&cursor);
   coords = cursor.y;

   cout<<"y"<<coords<<endl;
   return coords;
}
```

```
#endif // TOOLS_H_INCLUDED
```

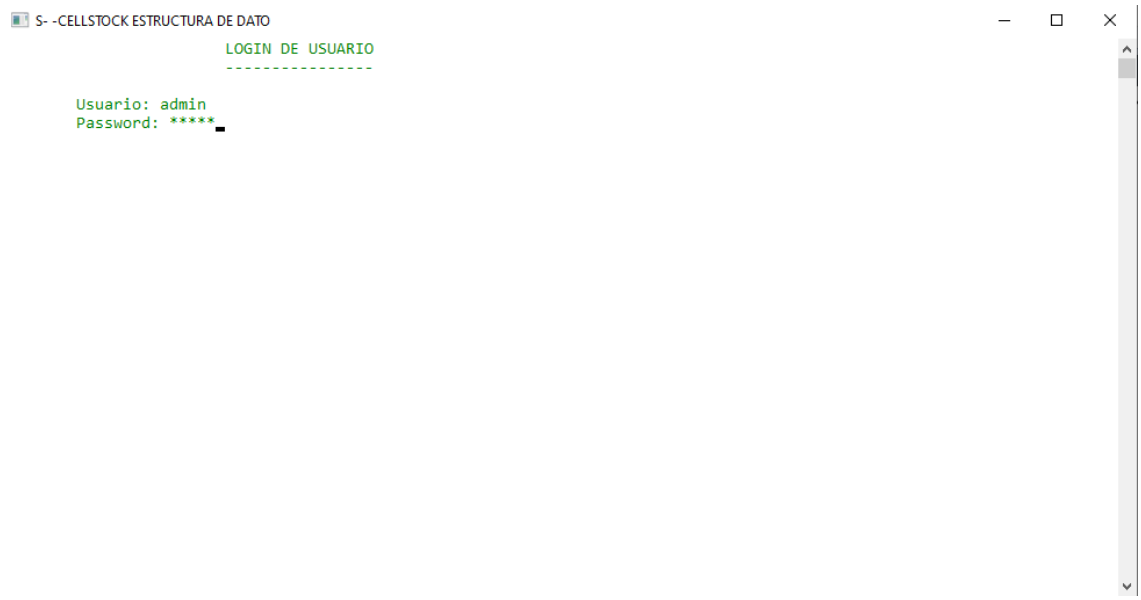
Opciones de compra

Capturas programa corriendo:

Registro de usuario

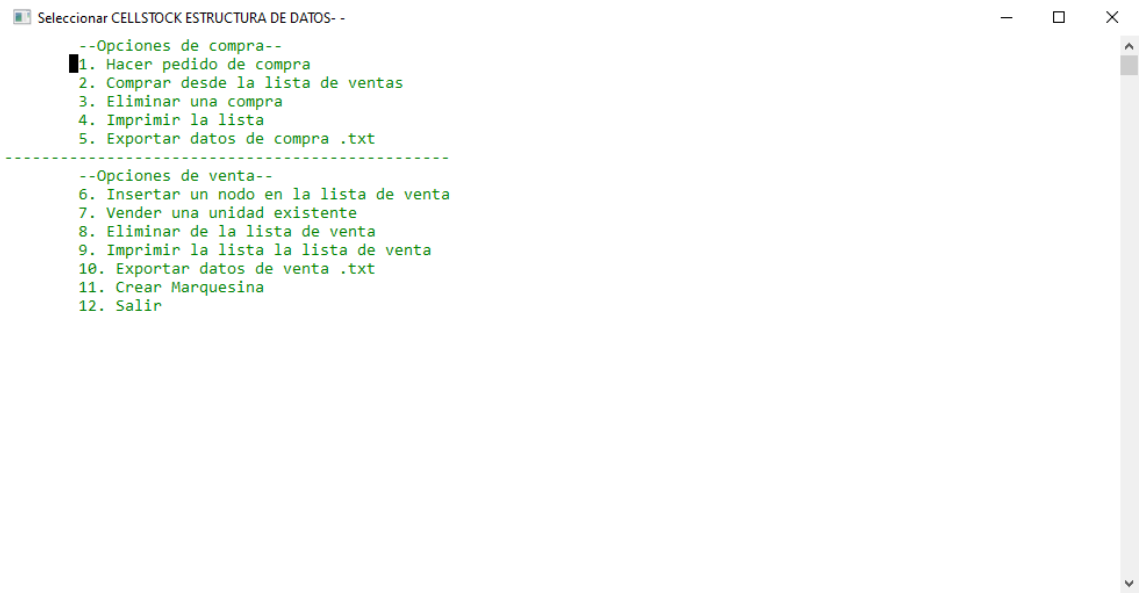


Login

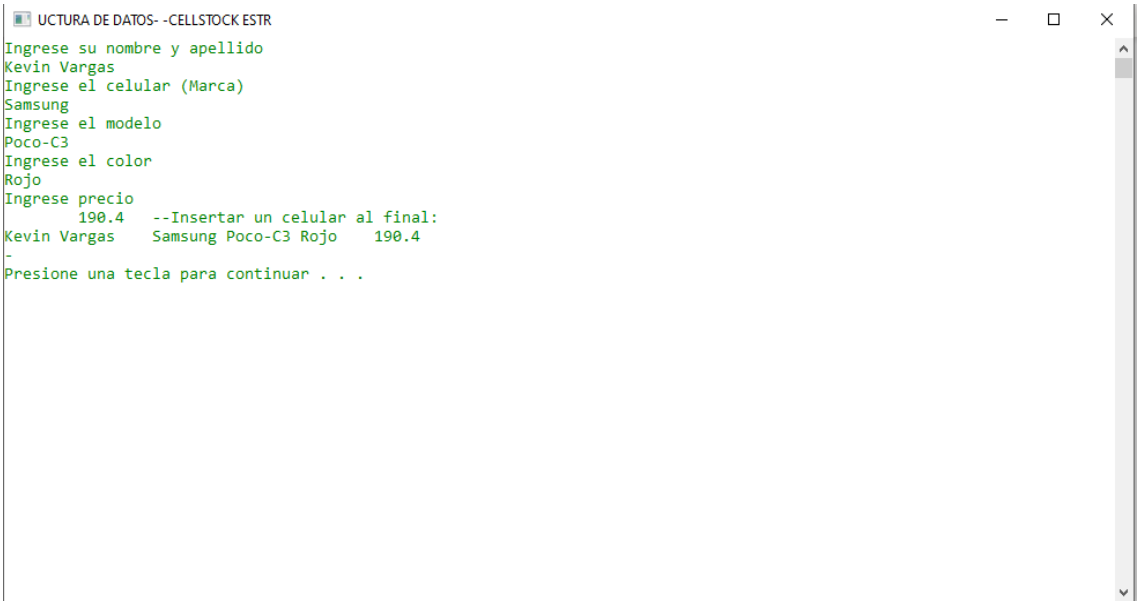


Imagen

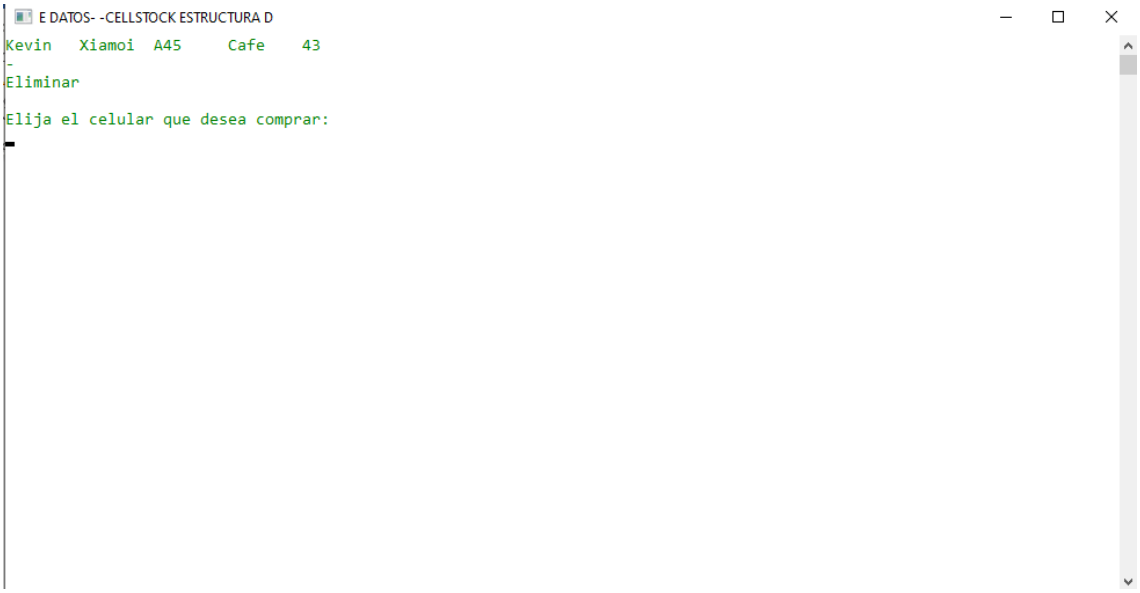
Menu



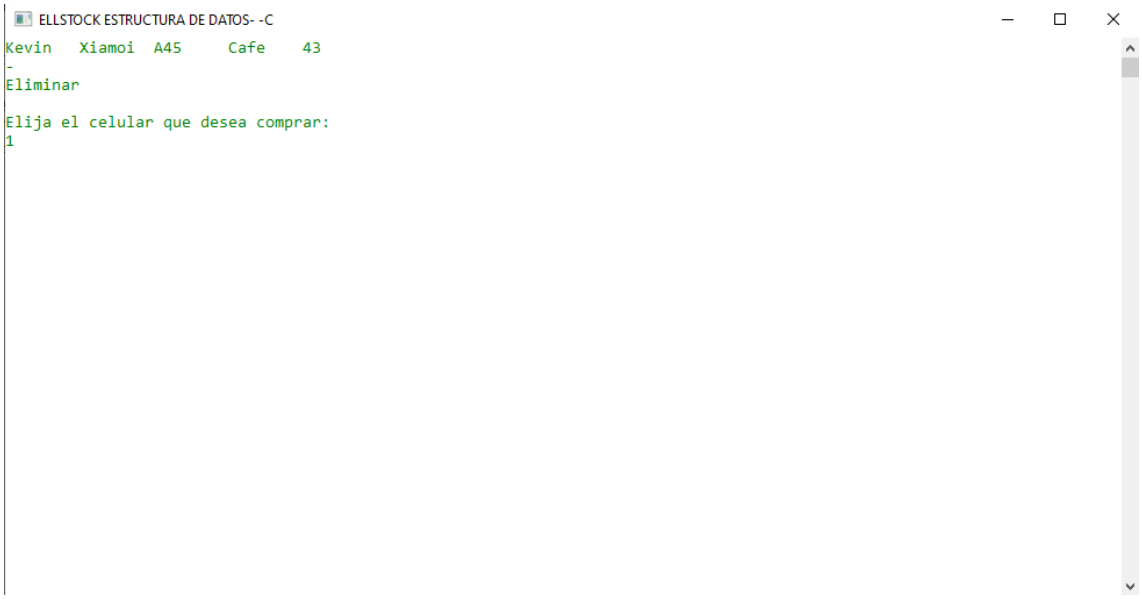
Hacer pedido de Compra



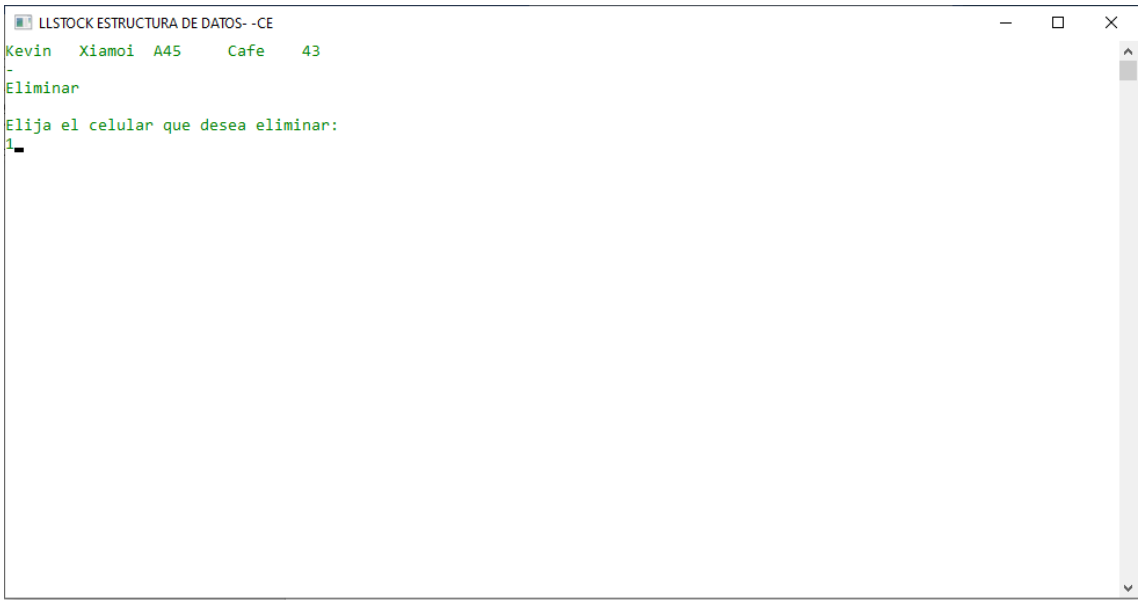
Comprar desde la lista de Ventas



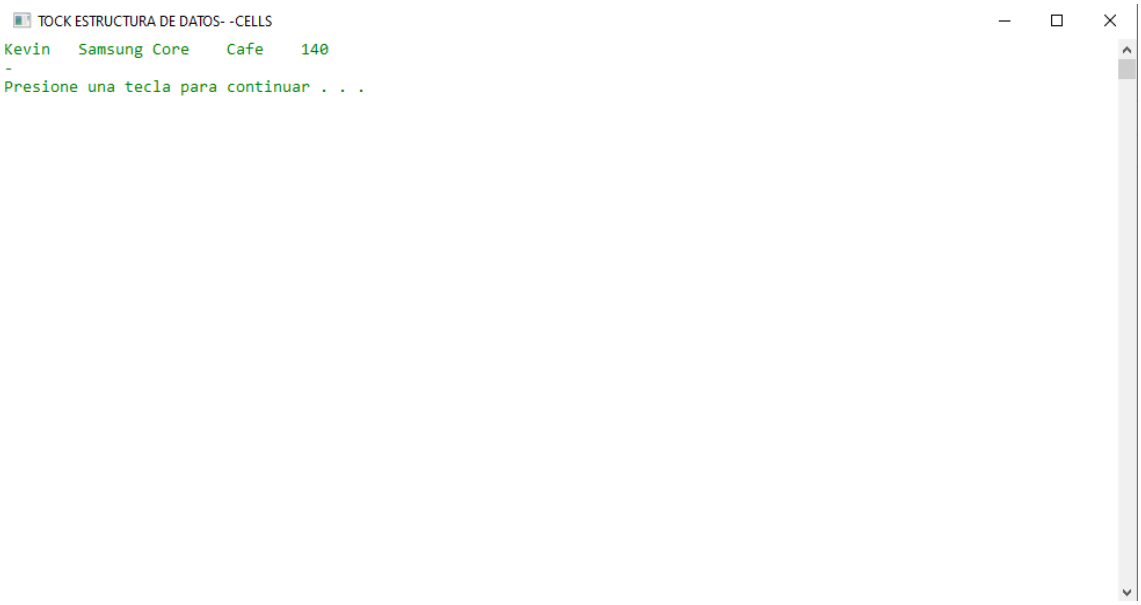
Comprar desde la lista de ventas



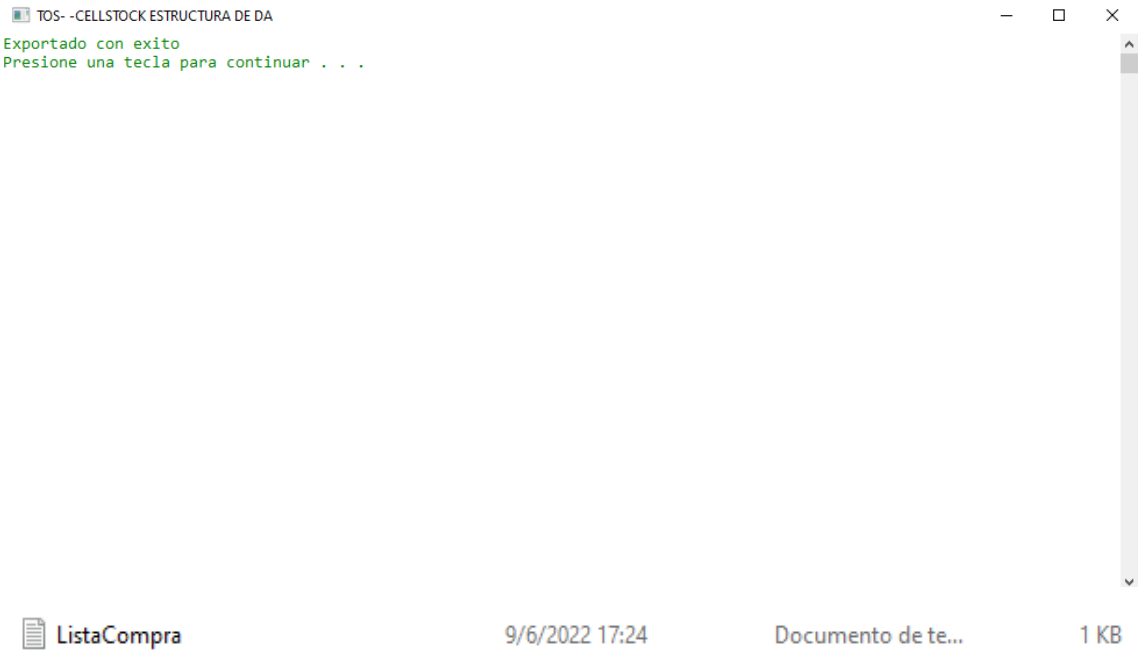
Eliminar compra



Imprimir lista

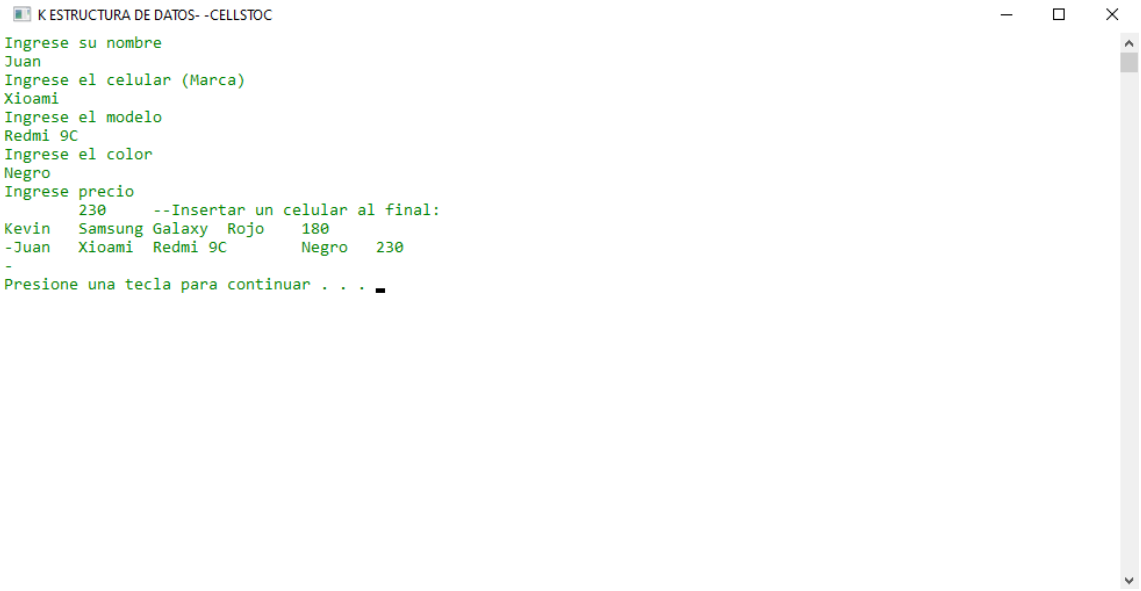


Exportar datos .txt

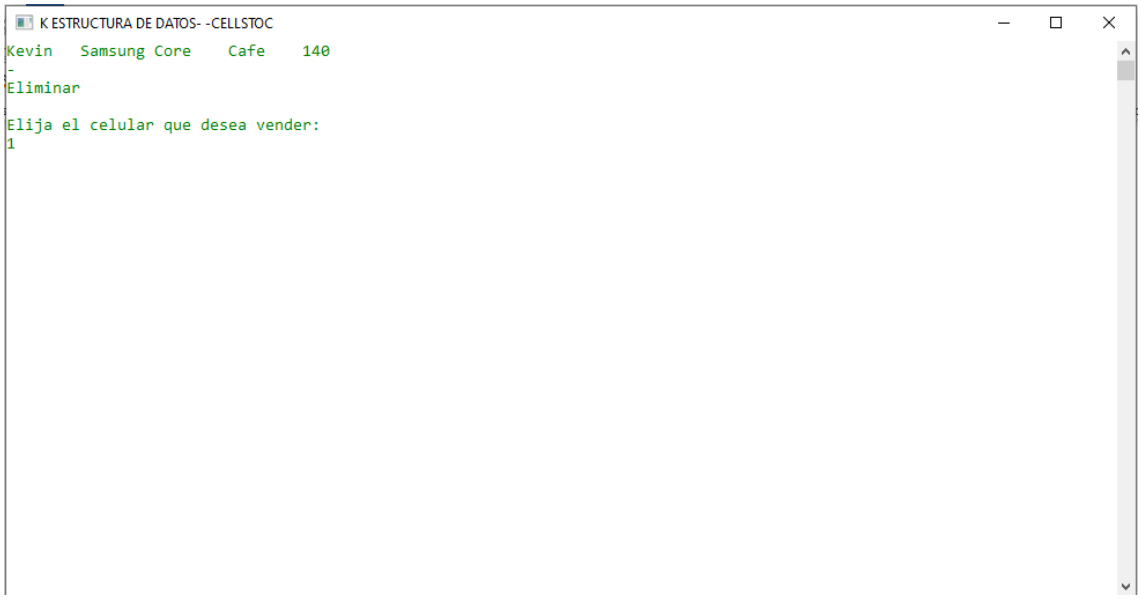


Opciones de venta

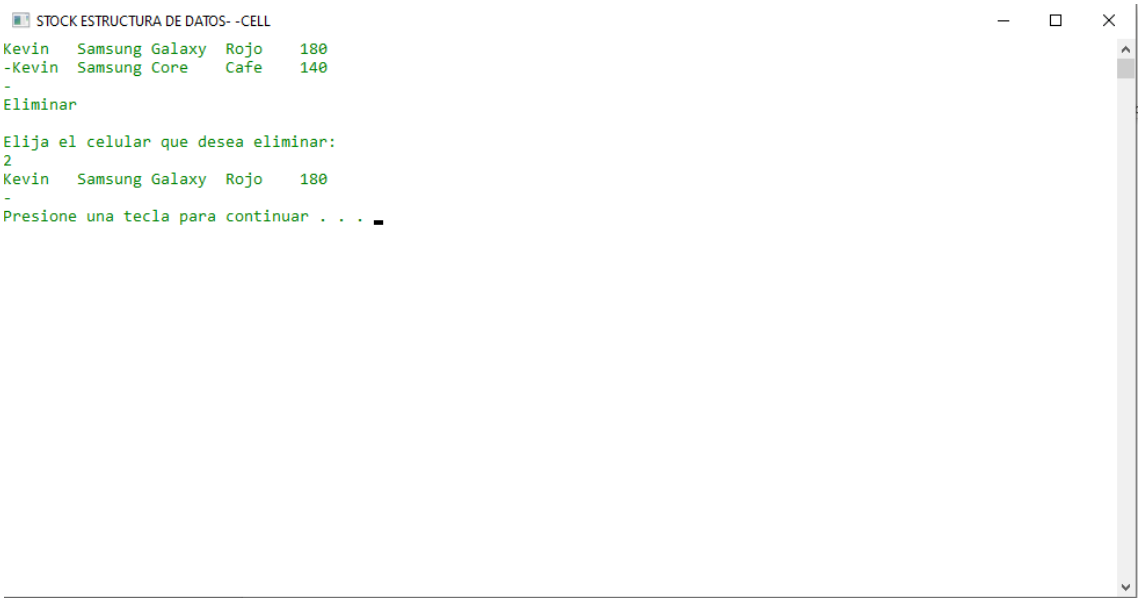
Insertar nodo a un listo de venta



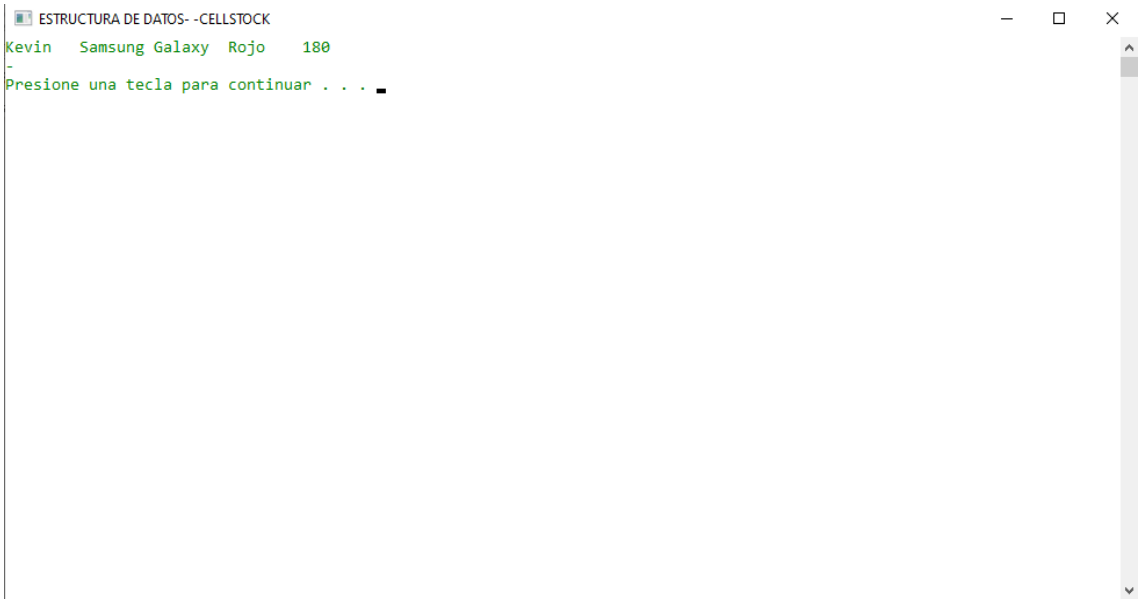
Vender una unidad existente



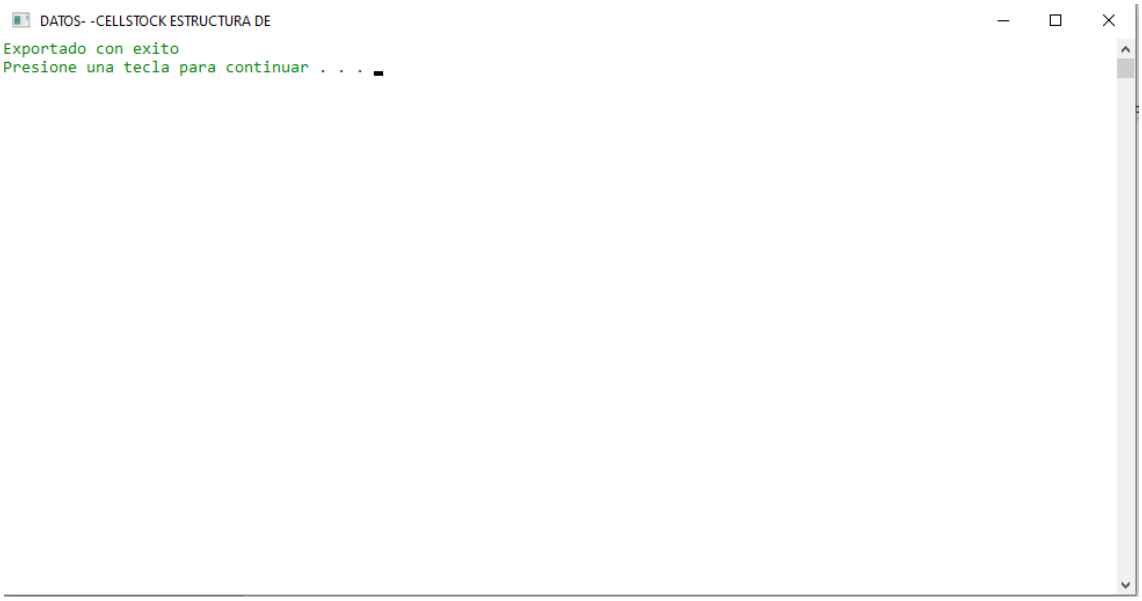
Eliminar de la lista de venta



Imprimir la lista de venta



Exportar datos de venta .txt



Crear marquesina