

FACULTAD DE CIENCIAS DE LA COMPUTACION.

ING. EN TECNOLOGIAS DE LA INFORMACION.

PROGRAMACION ORIENTADA

OBJETOS 1.

DOCUMENTO DE PRÁCTICAS

PROFESOR: MIGUEL RODRIGUEZ.

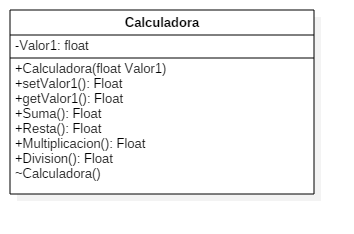
IRVING SAID MARTINEZ VERA

JOEL PATRICIO HILARIO.

EJERCICIO NUMERO 1

Q1. Write a program to implement four function calculator that can Add, Subtract, Multiply and Divide two numbers, taken from user as input. Also allow user to select the operation to be performed.

UML DEL PROBLEMA

  
CODIGO DEL PROBLEMA

#include <iostream>

#include <cmath>

using namespace std;

class Calculadora{

private:

float valor1;

public:

Calculadora(float v1){

this->valor1=v1;

}

Calculadora(){

valor1=0;

}

void setValor1(float v1){

v1=valor1;

}

float getValor1(){

return valor1;

}

float Suma(float valor2){

return valor1+valor2;

}

float Resta(float valor2){

return valor1-valor2;

}

float Multiplicacion(float valor2){

return valor1\*valor2;

}

float Divicion(float valor2){

return valor1/valor2;

}

};

int main(int argc, char \*argv[]) {

float x, y;

cout<<"Ingrese su primer valor: ";

cin>>x;

cout<<"\nIngrese su segundo valor: ";

cin>>y;

cout<<endl;

Calculadora obj1(x);

Calculadora obj2(y);

cout<<"----LA SUMA ES:----"<<endl

<<obj1.getValor1()<<"+"<<obj2.getValor1()<<"="

<<obj1.Suma(y)<<endl

<<"\n\n----LA RESTA ES:----"<<endl

<<obj1.getValor1()<<"-"<<obj2.getValor1()<<"="

<<obj1.Resta(y)<<endl

<<"\n\n----LA MULTIPLICACION ES:----"<<endl

<<obj1.getValor1()<<"\*"<<obj2.getValor1()<<"="

<<obj1.Multiplicacion(y)<<endl

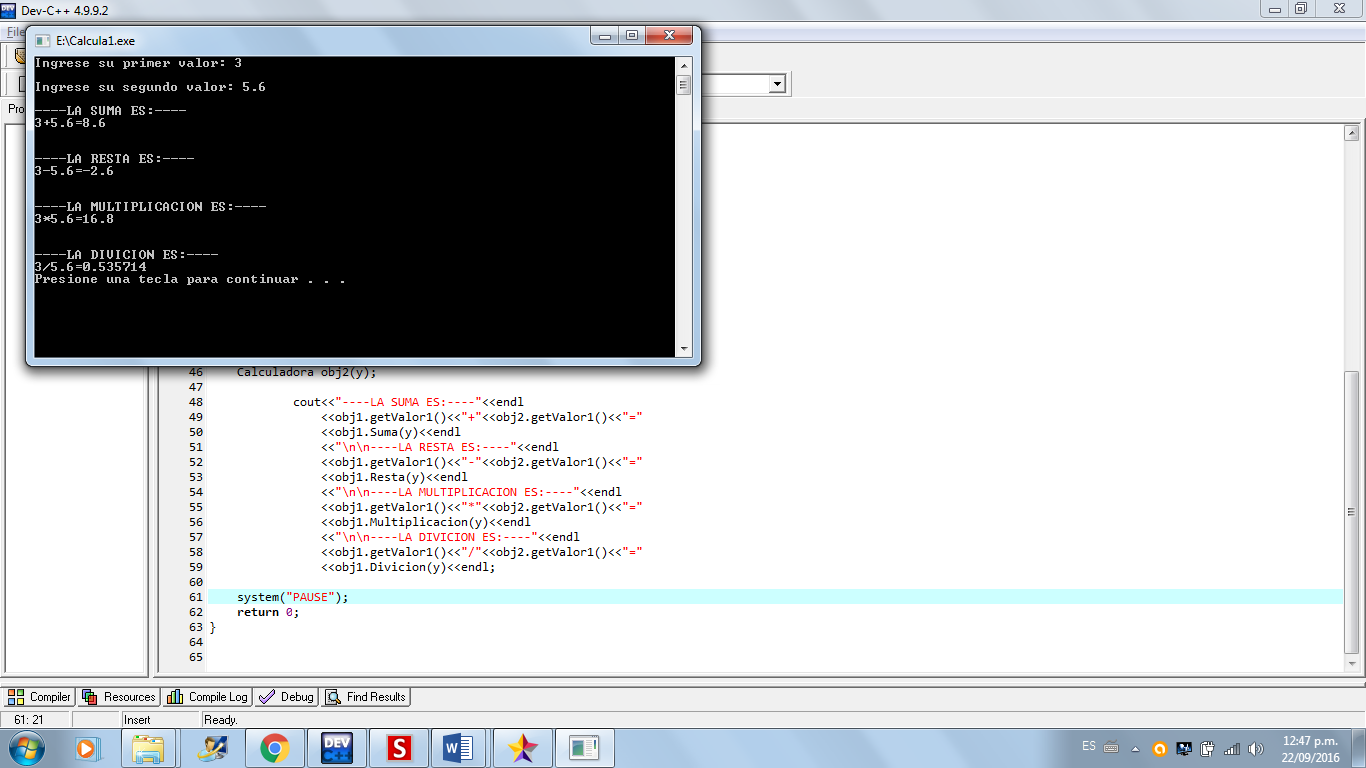
<<"\n\n----LA DIVICION ES:----"<<endl

<<obj1.getValor1()<<"/"<<obj2.getValor1()<<"="

<<obj1.Divicion(y)<<endl;

return 0;

}

RESULTADO

EJERCICIO NUMERO 2

**Exercise**  
**Object Oriented Programming Lab Session 02**  
***NED University of Engineering & Technology – Department of Computer & Information Systems Engineering*** 24 Q1.What is the difference between object and a class? Q2. What is the criterion for defining a functions **inline** or **out of line**?

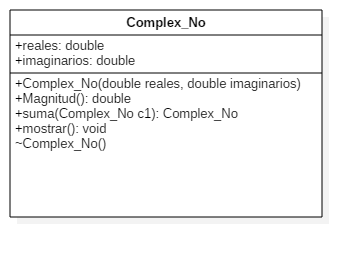
Q3. Define a class **Complex\_No** that has two member variables; **Real** and **Imaginary**. Also include following in the class • A parameterized constructor that takes Real and Imaginary values as argument. • A default constructor that assign zero to Real and Imaginary. • A copy constructor • A method **Display** that shows the value of complex number in appropriate format. • A method **Magnitude** that calculates the magnitude of complex number • A method **Add** that adds two complex numbers and return result; take one complex number as argument. Write a driver program to test your class

Q4. Define a class **Counter** having an attribute **value**. Provide a constructor that initializes value to zero. Also provide following methods:

• Increment (): that increment the value by one.

• Decrement (): that decrement the value by one.

UML DEL PROBLEMA



CODIGO DEL PROBLEMA

#include <iostream>

#include <math.h>

using namespace std;

class Complex\_No{

private:

double reales;

double imaginarios;

public:

Complex\_No(double reales, double imaginarios){

this->reales=reales;

this->imaginarios=imaginarios;

}

Complex\_No(){

reales=0;

imaginarios=0;

}

Complex\_No(double Complex\_No);

double Magnitud(){

double r;

r=sqrt( (reales\*reales)+(imaginarios\*imaginarios) );

return r;

}

Complex\_No suma(Complex\_No c1){

Complex\_No c2;

c2.reales=reales+c1.reales;

c2.imaginarios=imaginarios+c1.imaginarios;

return c2;

}

void mostrar(){

if(imaginarios>0){

cout<<reales<<"+"<<imaginarios<<"i";

}

else if(imaginarios<0){

cout<<reales<<imaginarios<<"i";

}

else{

cout<<reales<<"+0i";

}

}

};

int main(int argc, char \*argv[]) {

double x1, y1;

double x2, y2;

cout<<"Asigna los valores del Numero complejo 1: ";

cin>>x1>>y1;

cout<<"\nAsigna los valores del Numero complejo 2: ";

cin>>x2>>y2;

Complex\_No Comp1(x1,y1), Comp2(x2,y2);

cout<<"\n\nNumeros Complejos: "<<endl;

Comp1.mostrar();

cout<<endl;

Comp2.mostrar();

cout<<endl;

cout<<"\n\n---Magnitud---"<<endl;

Complex\_No r;

cout<<Comp1.Magnitud();

//r.mostrar();

cout<<endl;

cout<<Comp2.Magnitud();

//r.mostrar();

cout<<endl;

cout<<"\n\n---SUMA---"<<endl;

cout<<"(";

Comp1.mostrar();

cout<<") + (";

Comp2.mostrar();

cout<<")";

cout<<"=";

r=Comp1.suma(Comp2);

r.mostrar();

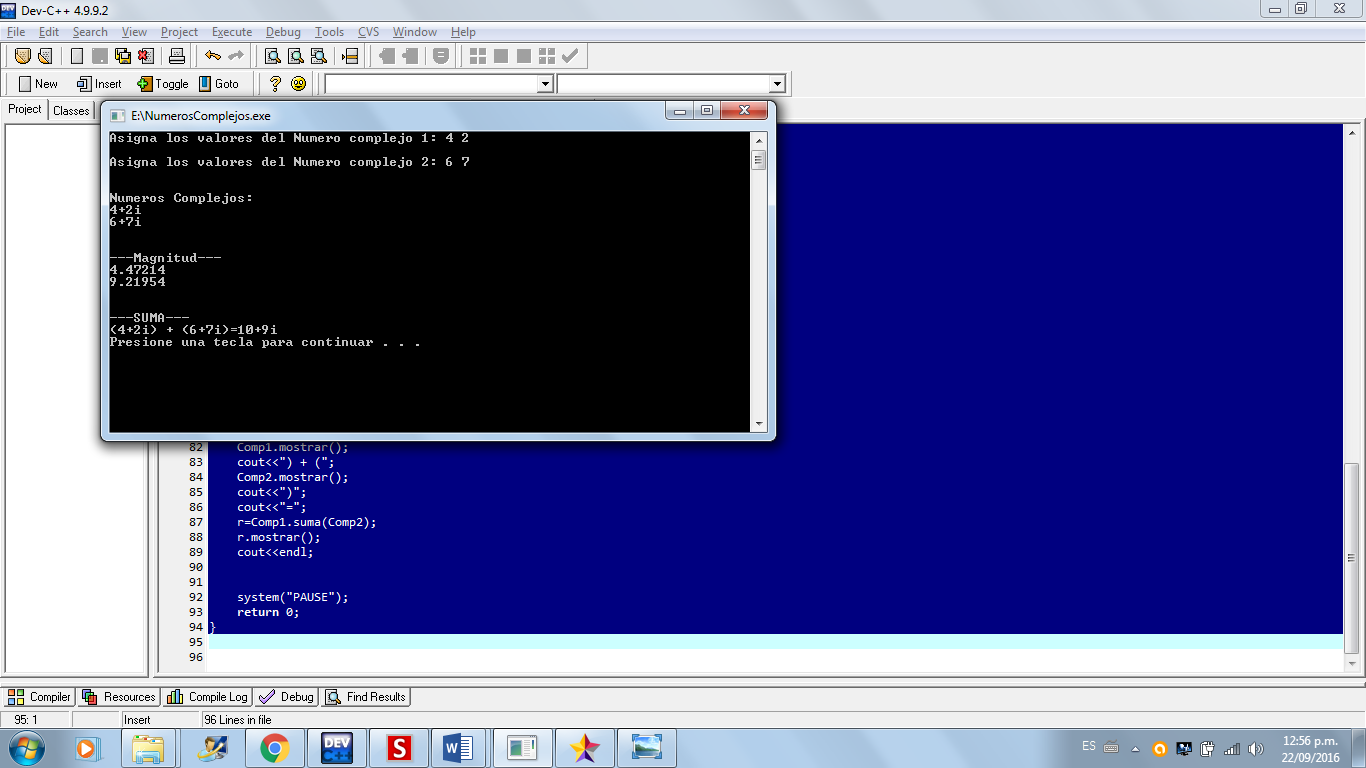
cout<<endl;

system("PAUSE");

return 0;

}

**RESULTADO**



EJERCICIO NUMERO 2.1

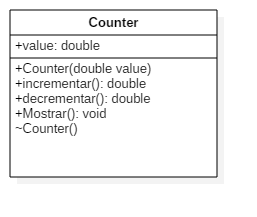
Q4. Define a class **Counter** having an attribute **value**. Provide a constructor that initializes value to zero. Also provide following methods:

• Increment (): that increment the value by one.

• Decrement (): that decrement the value by one.

Q5. Define a function **Reset** that takes a **Counter** type object as input and resets its **value** to zero. Make this function a friend of **Counter** class

UML DEL PROBLEMA



CODIGO DEL PROBLEMA

#include <iostream>

using namespace std;

class Counter{

private:

double value;

public:

Counter(double value){

this->value=value;

}

Counter(){

value=0.0;

}

double incrementar(){

return value+1;

}

double decrementar(){

return value-1;

}

void mostrar(){

cout<<value<<endl;

}

friend void reset(double v1, Counter &valor1){

valor1.value=v1;

}

};

int main(int argc, char \*argv[]) {

double var1;

cout<<"Valor de la primera variable: ";

cin>>var1;

cout<<endl;

Counter x(var1);

x.mostrar();

cout<<"Incremento: "<<x.incrementar()<<endl

<<"Decremento: "<<x.decrementar()<<endl;

cout<<endl;

double var2;

cout<<"Ingresa el nuevo valor: ";

cin>>var2;

cout<<endl;

reset(var2,x);

x.mostrar();

cout<<"Incremento: "<<x.incrementar()<<endl

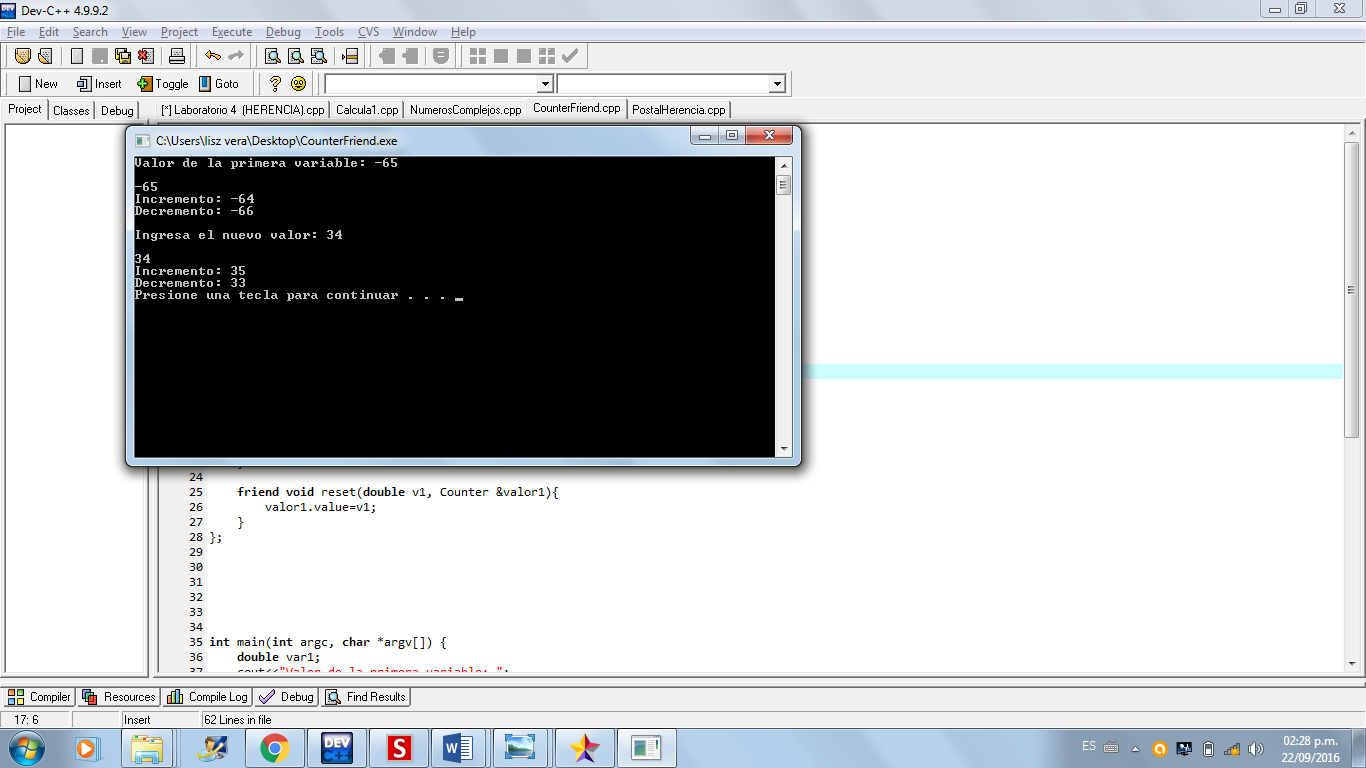
<<"Decremento: "<<x.decrementar()<<endl;

system ("pause");

return 0;

}

**RESULTADO**



LABORATORIO #3

**Exercise**

Q1. Write a program that takes the record of 10 students from user in an array and display all the records.  
[use Student class, defined in lab session 02]

Q2.Define a pointer to student class to access the contents of array defined in Q1. Allow user to search a record in array by means of Rollno

Q3. Develop a class to represent an integer array. The member variables include an integer to represent the size of array and an integer pointer to represent the address of the first element of the array. The user is allowed to create an array at runtime using this class. Include appropriate constructors (parameterized and copy). Also include a method that adds the contents of array.

Q4.Consider following code:  
class myclass  
{int data[2];  
public:  
int\* p;  
public:  
myclass()  
{p=data;}  
};  
int main()  
{ myclass\* cp;  
cp=new myclass[3];  
return 0;  
} How would you access the contents of *data* of each element of *myclass* array? Add code in the above program to do the following:  
a. Assign values to array *data* of each element of *myclass* array.

UML DEL PROBLEMA

LABORATORIO #4

Q4. Develop a class **Post** that has following attributes  
**Name**: a string  
**To** : a string that holds the reciever's address  
**StampCost**: a float that holds the value of postal stamp required The class should include following:

• A constructor that initializes StampCost to 1$ and To to empty address

• **Read():** a method that reads data member's values from user • **Print():** a method that displays data member's values on screen

• **TotalCost():** a method that returns stampCost value.

Develop another class **RegisteredPost** that inherits from **Post** class and has following additional attributes:  
**Weight:** a float that holds the weight of post  
**ReistrationCost:** a float that holds registration charges

Also include following in the class.

• A constructor that initializes weight to 20 gms and RegisterationCost to 10$

• **Read():** a method that reads data member's values from user • **Print():** a method that displays data member's values on screen

• **TotalCost():** a method that returns stampCost+RegistrationCost.

Q5. Develop a class **InsuredRgisteredPost** that inherits from **RegisteredPost** class and has additional attribute **AmtInsured** to hold the insured value of post. The class should include following:

• A constructor that initializes AmtInsured to 20$

• **Read():** a method that reads data member's values from user • **Print():** a method that displays data member's values on screen

• **TotalCost():** a method that returns StampCost+RegistrationCost+AmtInsured.

Q6. Use following driver program to test your classes  
#include <iostream>  
using namespace std;  
int main()  
{ InsuredRegisteredPost envelope;  
envelope.Read();  
cout<<"Post Details..";  
envelope.Print();  
return 0;  
}

Codigo del Programa

#include <iostream>

using namespace std;

class Estudiante{

protected:

string nombre;

int matricula;

double marks[5];

float porcentaje;

public:

Estudiante(){}

Estudiante(string nombre, int matricula, double marks, float porcentaje){

this->nombre=nombre;

this->matricula=matricula;

this->marks[5]=marks;

this->porcentaje=porcentaje;

}

void PedirValores(){

cout<<"Nombre Del Estudiante: ";

cin>>nombre;

cout<<"\nMatricula Del Estudiante: ";

cin>>matricula;

cout<<"\n---Ingresa los valores de Marks---"<<endl;

for(int i=1; i<=5; i++){

cout<<"Mark "<<i<<": ";

cin>>marks[i];

cout<<endl;

}

cout<<endl;

for(int i=1; i<=5; i++){

cout<<"Marks "<<i<<": "<<marks[i]<<endl;

}

}

double CalcularPorcentaje(){

double s;

for(int i=1; i<=5; i++){

s=s+marks[i];

}

porcentaje=(s/500)\*100;

return porcentaje;

}

double grado(){

return CalcularPorcentaje();

}

void mostrar(){

PedirValores();

cout<<"\n\n\n\n\n";

cout<<"Nombre: "<<nombre<<endl

<<"Matricula: "<<matricula<<endl;

CalcularPorcentaje();

cout<<"Porcentaje Cursado: "<<grado()<<"%"<<endl;

}

};

class EstudianteScience:public Estudiante{

public:

void PedirValores(){

Estudiante::PedirValores();

}

double CalcularPorcentaje(){

double s;

for(int i=1; i<=5; i++){

s=s+marks[i];

}

s=s+150;

porcentaje=(s/650)\*100;

return porcentaje;

}

double grado(){

return CalcularPorcentaje();

}

void mostrar(){

Estudiante::mostrar();

}

};

class EstudianteArt:public Estudiante{

public:

void PedirValores(){

Estudiante::PedirValores();

}

double CalcularPorcentaje(){

double s;

for(int i=1; i<=5; i++){

s=s+marks[i];

}

s=s+150;

porcentaje=(s/600)\*100;

return porcentaje;

}

double grado(){

return CalcularPorcentaje();

}

void mostrar(){

Estudiante::mostrar();

}

};

int main(int argc, char \*argv[]) {

Estudiante est1;

//est1.mostrar();

EstudianteScience est2;

est2.mostrar();

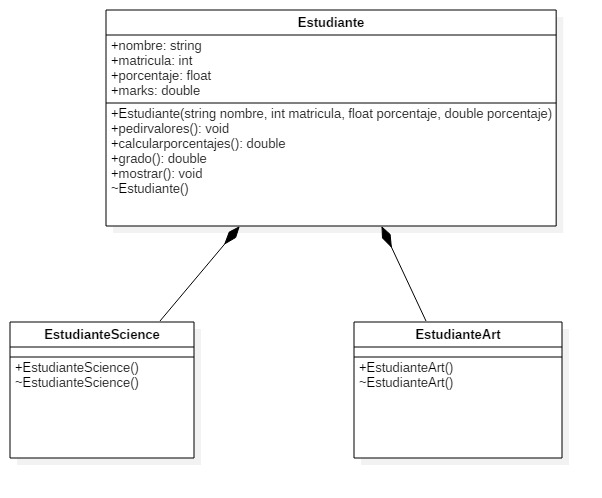
EstudianteArt est3;

est3.mostrar();

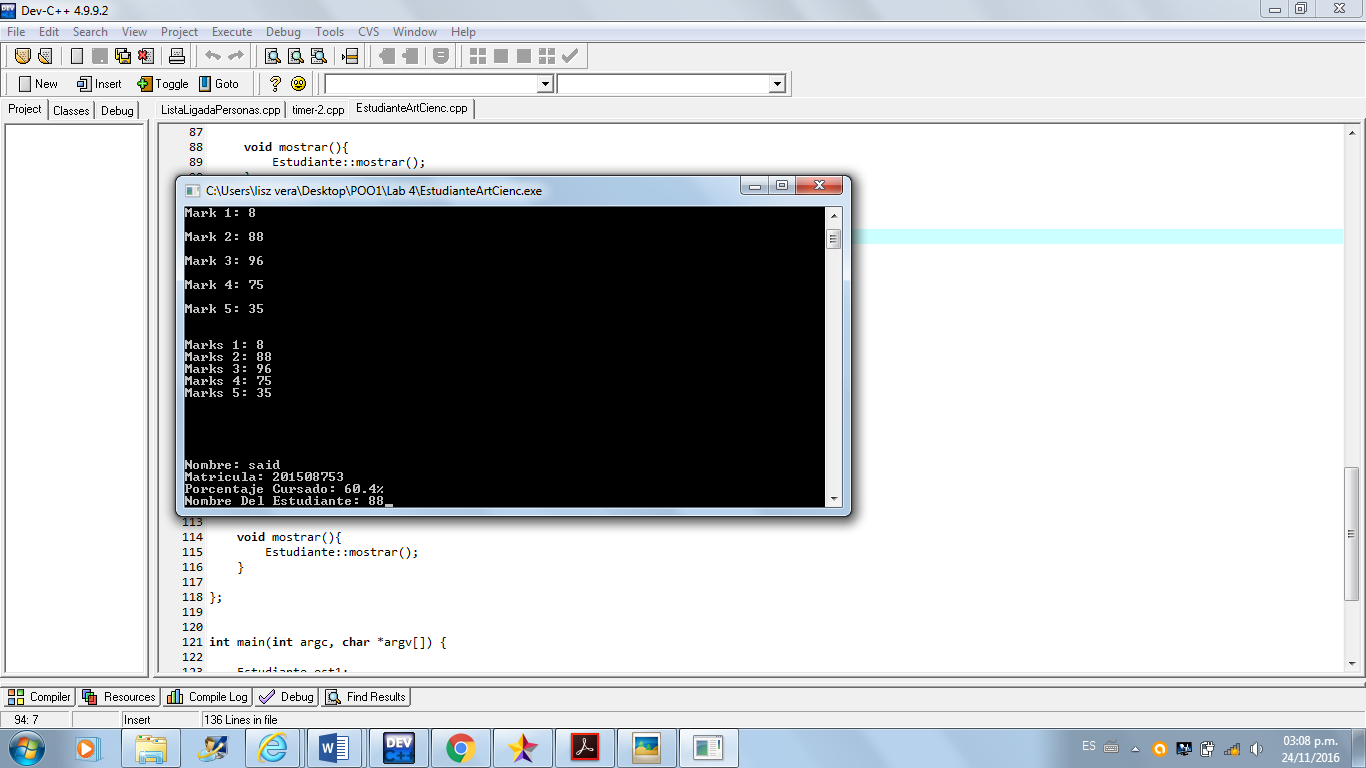
return 0;

}

UML



Pantallas



Codigo del Programa

#include <iostream>

using namespace std;

class Post{

protected:

string name;

string to;

float stampcost;

public:

Post(string name, string to, float stampcost){

this->name=name;

this->to=to;

this->stampcost=stampcost;

}

Post(){

stampcost =1.0;

to = "";

}

virtual void Read(){

cout<<"Ingresa Nombre: ";

cin>>name;

cout<<"\nIngresa La Direccion: ";

cin>>to;

cout<<"\nIngrese El Costo De La Stampilla: ";

cin>>stampcost;

cout<<endl;

}

virtual float Totalcost (){

return stampcost;

}

virtual void Print(){

//Read();

cout<<endl

<<"Nombre: "<<name<< endl

<<"Direccion: "<<to<< endl

<<"Costo Total: "<<Totalcost()<<"$"<< endl;

}

};

class registeredPost: public Post{

protected:

float weight;

float costReistration;

public:

registeredPost(string name, string to, float stampcost, float weight, float costReistration):Post(name, to, stampcost){

weight=weight;

costReistration=costReistration;

}

registeredPost(){

weight=20;

costReistration=10;

}

virtual void Read(){

Post::Read();

cout<<"\nPeso En Gramos: ";

cin>>weight;

cout<<"\nGastos De Registro: ";

cin>>costReistration;

cout<<endl;

}

virtual float Totalcost(){

return Post::Totalcost()+costReistration;

}

virtual void print(){

Post::Print();

cout<<endl

<<"Peso En Gramos: "<<weight<<"gr"

<<endl

<<"Gastos De Registros: "<<costReistration<<endl;

}

};

class insuredRgisteredPost: public registeredPost{

private:

double AmtInsured;

public:

insuredRgisteredPost(){

AmtInsured =20;

}

insuredRgisteredPost(string name, string to, float stampcost, float weight, float costReistration, double AmtInsured){

AmtInsured=AmtInsured;

}

void Read(){

registeredPost::Read();

cout<<"\nValor asegurado del correo: ";

cin>>AmtInsured;

cout<<endl;

}

float Totalcost(){

return registeredPost::Totalcost()+AmtInsured;

}

void print(){

registeredPost::Print();

cout<<endl

<<"Valor asegurado: "<<AmtInsured

<<endl;

}

};

int main (){

insuredRgisteredPost envelope;

envelope.Read();

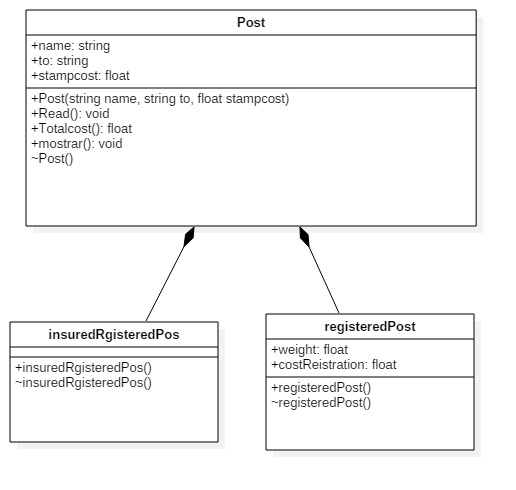
cout<<"Pos Datails.."<<endl;

envelope.Print();

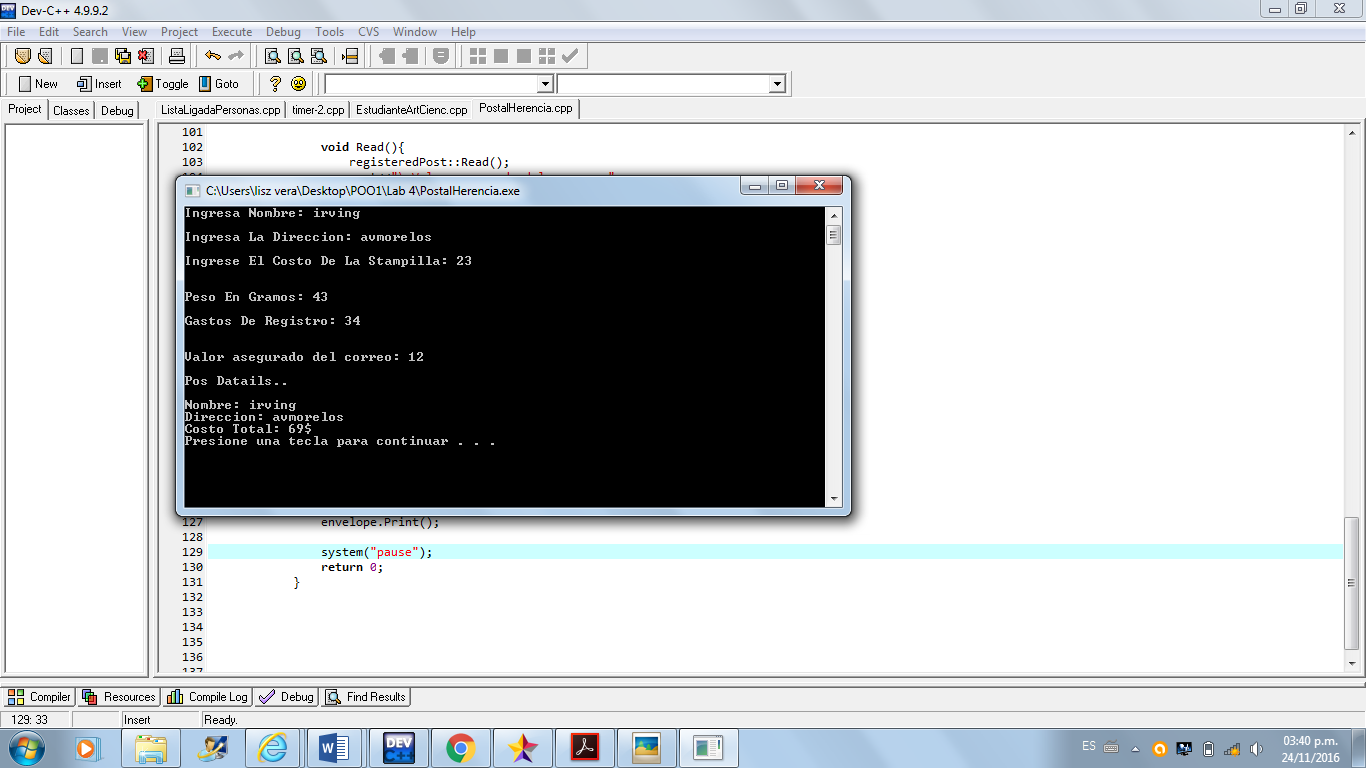
return 0;

}

UML



Pantallas



Codigo del Programa

#include <iostream>

using namespace std;

class ShopingMall{

protected:

string nombre;

string IDproduct;

public:

ShopingMall(string nombre, string IDproduct){

this->nombre=nombre;

this->IDproduct=IDproduct;

}

void setNombre(string n){

n=nombre;

}

void setIDproduct(string IDp){

IDp=IDproduct;

}

string getNombre(){

return nombre;

}

string getIDproduct(){

return IDproduct;

}

void Asignar(){

cout<<"Nombre del Producto: ";

cin>>nombre;

setNombre(nombre);

cout<<"\nID del Producto: ";

cin>>IDproduct;

setIDproduct(IDproduct);

cout<<endl;

}

void mostrar(){

cout<<"Nombre: "<<this->nombre<<endl

<<"ID del producto: "<<this->IDproduct<<endl;

}

};

class UtilityItems:public ShopingMall{

private:

double descuento;

public:

UtilityItems(string nombre, string IDproduct, double descuento):ShopingMall(nombre,IDproduct){

this->descuento=descuento;

}

void setDescuento(double d){

d=descuento;

}

double getDescuento(){

return descuento;

}

void Asignar(){

ShopingMall::Asignar();

cout<<"Cual es el descuento del producto: ";

cin>>descuento;

setDescuento(descuento);

cout<<endl;

}

void mostrar(){

ShopingMall::mostrar();

cout<<"Descuento: "<<this->descuento<<"%"<<endl;

}

};

class FoodItems:public ShopingMall{

private:

string FechaExpired;

public:

FoodItems(string nombre, string IDproduct, string FechaExpired):ShopingMall(nombre, IDproduct){

this->FechaExpired=FechaExpired;

}

void setFechaExpired(string fe){

fe=FechaExpired;

}

string getFechaExpired(){

return FechaExpired;

}

void Asignar(){

ShopingMall::Asignar();

cout<<"Ingresa Fecha de Expiracion "

<<endl<<"Ejem. : (31/01/2016): ";

cin>>FechaExpired;

setFechaExpired(FechaExpired);

cout<<endl;

}

void mostrar(){

ShopingMall::mostrar();

cout<<"Fecha de Expiracion: "<<this->FechaExpired<<endl;

}

};

int main(int argc, char \*argv[]) {

cout<<"Supermercado..."<<endl;

ShopingMall ven("xxx","0000");

ven.Asignar();

ven.mostrar();

cout<<"\nUtility Items..."<<endl;

UtilityItems ven1("xxx", "000", 0.0);

ven1.Asignar();

ven1.mostrar();

cout<<"\nFood Items..."<<endl;

FoodItems ven2("xxx","000", "00/00/0000");

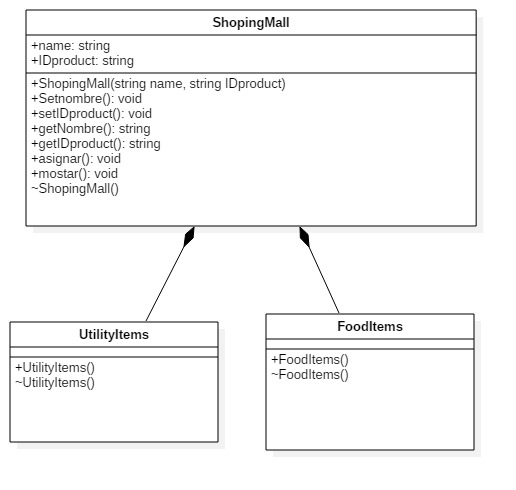
ven2.Asignar();

ven2.mostrar();

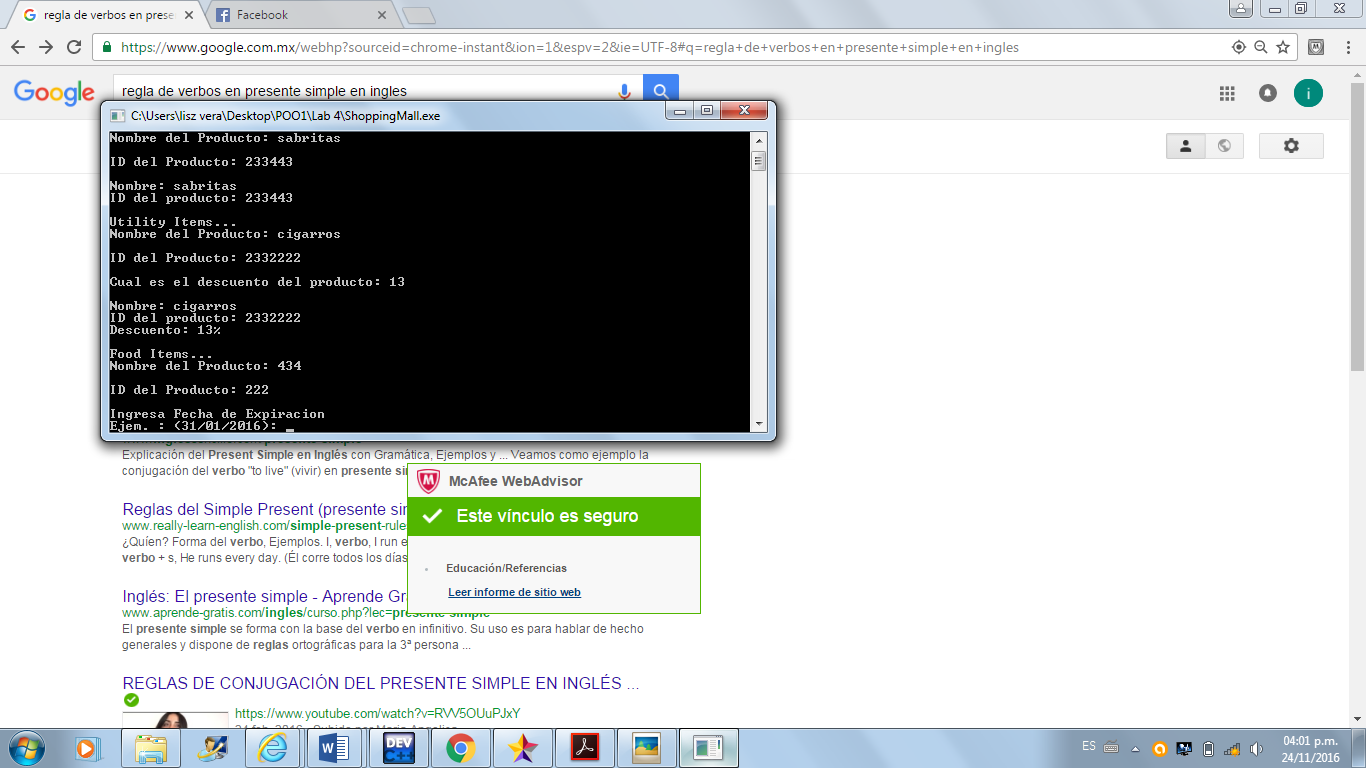
return 0;

}

UML



Pantallas



Laboratorio 5

Código del Programa

#include <iostream>

#include <windows.h>

//#include <conio2.h>

using namespace std;

class DateTime{

private:

int anio;

int mes;

int dia;

int hora;

int minuto;

int segundo;

public:

DateTime(){

anio=0;

mes=0;

dia=0;

hora=0;

minuto=0;

segundo=0;

}

DateTime(int anio, int mes, int dia, int hora, int minuto, int segundo){

this->anio=anio;

this->mes=mes;

this->dia=dia;

this->hora=hora;

this->minuto=minuto;

this->segundo=segundo;

}

void SetValue(int dia, int mes, int anio){

for( anio=anio; anio<=3000; anio++){

for(mes=mes; mes<=12; mes++){

for(dia=dia; dia<=30; dia++){

for(int ms=0; ms<100; ms++){

cout<<"Dia: "<<dia<<" /Mes: "<<mes<<" /Anio: "<<anio<<endl;

}

}

}

}

}

void SetValue(int dia, int mes, int anio, int hora){

for( anio=anio; anio<=3000; anio++){

for(mes=mes; mes<=12; mes++){

for(dia=dia; dia<=30; dia++){

for(hora=hora; hora<=24; hora++){

for(int ms=0; ms<100; ms++){

cout<<"Dia: "<<dia<<" /Mes: "<<mes<<" /Anio: "<<anio<<endl

<<"Hora: "<<hora<<endl;

}

}

}

}

}

}

void SetValue(int dia, int mes, int anio, int hora, int minuto){

for( anio=anio; anio<=3000; anio++){

for(mes=mes; mes<=12; mes++){

for(dia=dia; dia<=30; dia++){

for(hora=hora; hora<=24; hora++){

for(minuto=minuto; minuto<=59; minuto++){

for(int ms=0; ms<100; ms++){

cout<<"Dia: "<<dia<<" /Mes: "<<mes<<" /Anio: "<<anio<<endl

<<"Hora: "<<hora<<" /Minutos: "<<minuto<<endl;

}

}

}

}

}

}

}

void SetValue(int dia, int mes, int anio, int hora, int minuto, int segundo){

for( anio=anio; anio<=3000; anio++){

for(mes=mes; mes<=12; mes++){

for(dia=dia; dia<=30; dia++){

for(hora=hora; hora<=24; hora++){

for(minuto=minuto; minuto<=59; minuto++){

for(segundo=segundo; segundo<=59; segundo++){

for(int ms=0; ms<100; ms++){

cout<<"Dia: "<<dia<<" /Mes: "<<mes<<" /Anio: "<<anio<<endl

<<"Hora: "<<hora<<" /Minutos: "<<minuto<<" /Segundos: "<<segundo<<endl;

}

}

}

}

}

}

}

}

};

int main(int argc, char \*argv[]) {

DateTime tiempo;

//tiempo.SetValue(5,4,2001);

int opcion;

int Dia, Mes, Anio, Hora, Minutos, Segundos;

do{

cout<<"\n\n"

<<"\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*TIEMPO\*-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*"

<<endl

<<endl

<<"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"

<<endl

<<"1. ASIGNAR(Dia/Mes/Anio)"<<endl

<<"2. ASIGNAR(Dia/Mes/Anio/Hora)"<<endl

<<"3. ASIGNAR(Dia/Mes/Anio/Hora/Minutos)"<<endl

<<"4. ASIGNAR(Dia/Mes/Anio/Hora/Minutos/Segundos)"<<endl

<<"5. Salir\n==>"

<<"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"

<<endl

<<" Opcion--> ";

cin>>opcion;

cout<<endl;

switch(opcion){

case 1:

cout<<"ASIGNAR(Dia/Mes/Anio)"

<<endl<<"Dia: ";

cin>>Dia;

cout<<endl<<"Mes: ";

cin>>Mes;

cout<<endl<<"Anio: ";

cin>>Anio;

cout<<endl;

tiempo.SetValue(Dia, Mes, Anio);

break;

case 2:

cout<<"ASIGNAR(Dia/Mes/Anio/Hora)"

<<endl<<"Dia: ";

cin>>Dia;

cout<<endl<<"Mes: ";

cin>>Mes;

cout<<endl<<"Anio: ";

cin>>Anio;

cout<<endl<<"Hora: ";

cin>>Hora;

cout<<endl;

tiempo.SetValue(Dia, Mes, Anio, Hora);

break;

case 3:

cout<<"ASIGNAR(Dia/Mes/Anio/Hora/Minutos)"

<<endl<<"Dia: ";

cin>>Dia;

cout<<endl<<"Mes: ";

cin>>Mes;

cout<<endl<<"Anio: ";

cin>>Anio;

cout<<endl<<"Hora: ";

cin>>Hora;

cout<<endl<<"Minutos: ";

cin>>Minutos;

cout<<endl;

tiempo.SetValue(Dia, Mes, Anio, Hora, Minutos);

break;

case 4:

cout<<"ASIGNAR(Dia/Mes/Anio/Hora/Minutos/Segundos)"

<<endl<<"Dia: ";

cin>>Dia;

cout<<endl<<"Mes: ";

cin>>Mes;

cout<<endl<<"Anio: ";

cin>>Anio;

cout<<endl<<"Hora: ";

cin>>Hora;

cout<<endl<<"Minutos: ";

cin>>Minutos;

cout<<endl<<"Segundos: ";

cin>>Segundos;

cout<<endl;

tiempo.SetValue(Dia, Mes, Anio, Hora, Minutos, Segundos);

break;

case 5:

cout<<"ELIMINADO TU LINEA DE TIEMPO...";

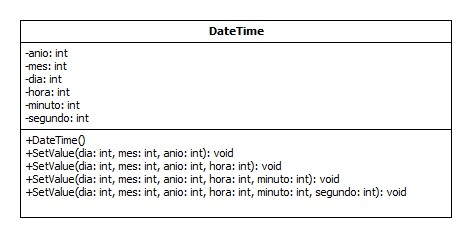
break;

}

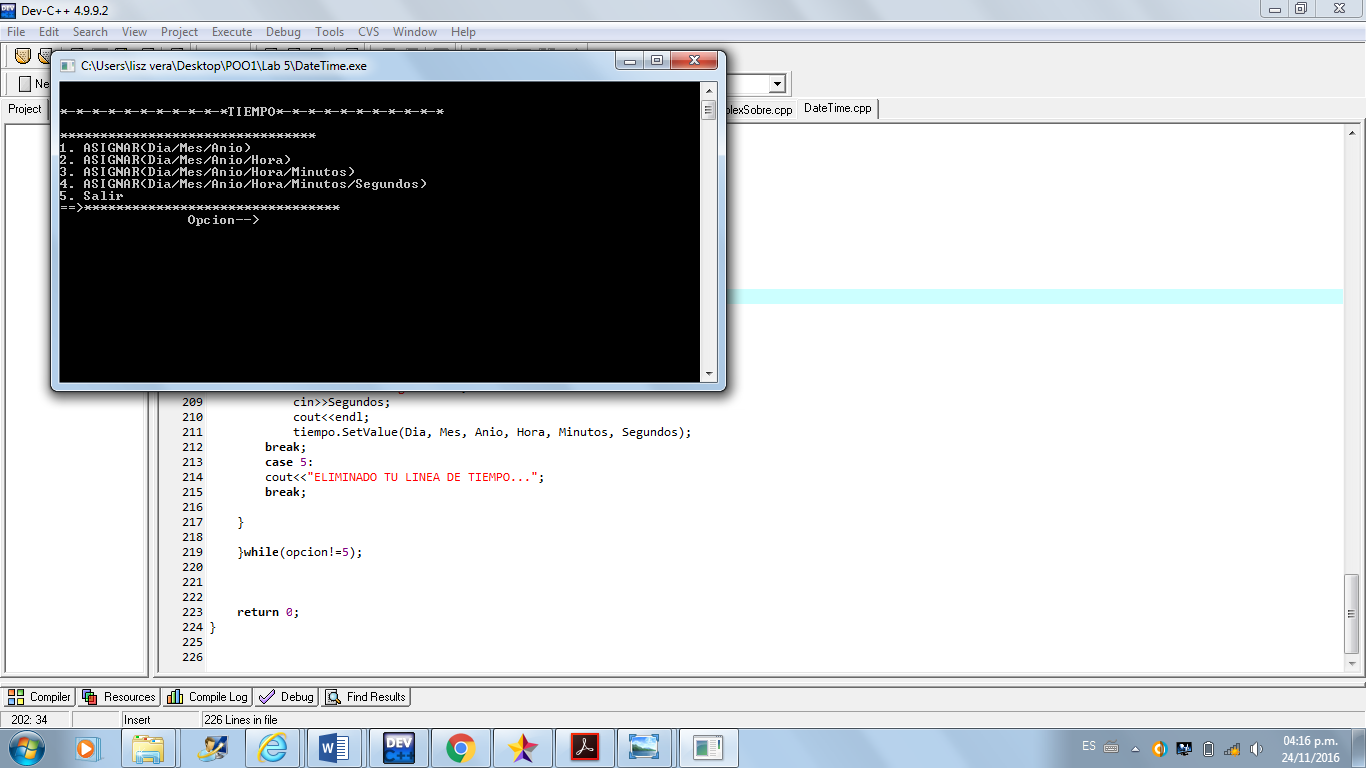
}while(opcion!=5);

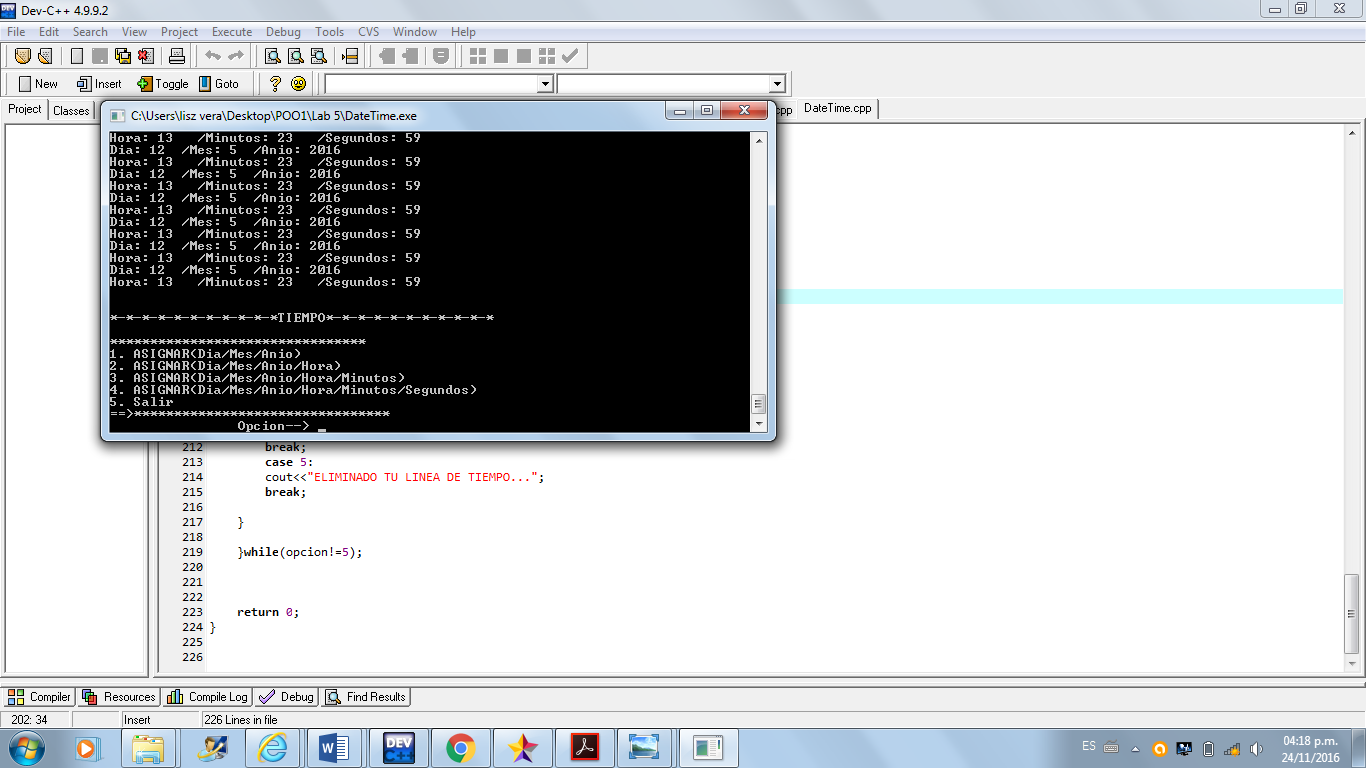
return 0;

UML



Pantallas





Código del Programa

#include <iostream>

using namespace std;

class abc{

public:

void func(){

//Es una Funcion abstracta y las funciones abstractas no se pueden instanciar

//=0;

}

};

class xyz:public abc{

public:

void func(){

cout << "this is function of xyz";

}

};

int main(){

abc obj1;

xyz obj2;

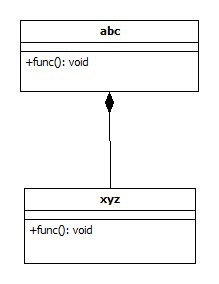
obj1.func();

obj2.func();

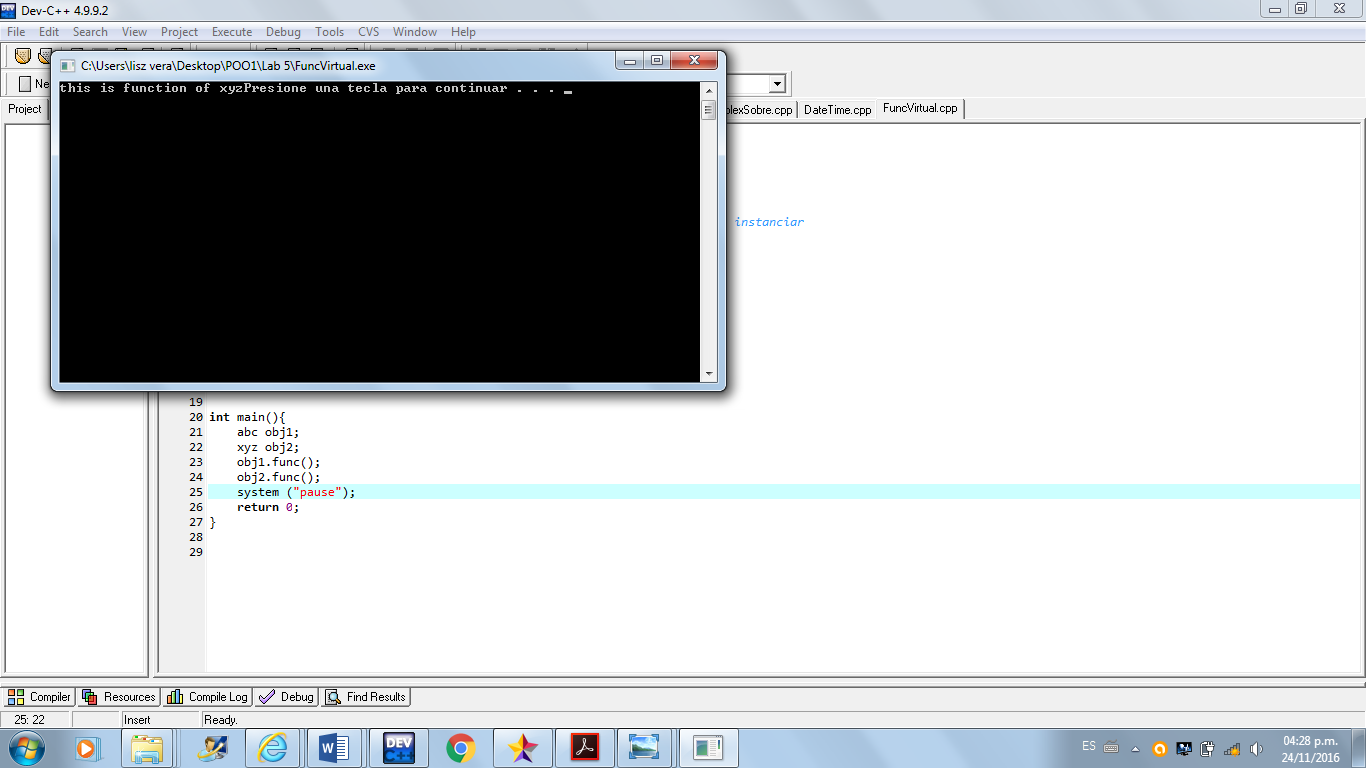
return 0;

}

UML



Pantallas



Laboratorio 6

Código del Programa

#include <iostream>

#include <math.h>

using namespace std;

class Complex\_No{

private:

double reales;

double imaginarios;

public:

Complex\_No(double reales, double imaginarios){

this->reales=reales;

this->imaginarios=imaginarios;

}

Complex\_No(){

reales=0;

imaginarios=0;

}

Complex\_No(double Complex\_No);

Complex\_No operator+(Complex\_No f){

Complex\_No s;

s.reales=reales+f.reales;

s.imaginarios=imaginarios+f.imaginarios;

return s;

}

Complex\_No operator-(Complex\_No f){

Complex\_No s;

s.reales=reales-f.reales;

s.imaginarios=imaginarios-f.imaginarios;

return s;

}

Complex\_No operator\*(Complex\_No f){

Complex\_No s;

s.reales=(reales\*f.reales)-(imaginarios\*f.imaginarios);

s.imaginarios=(reales\*f.imaginarios)+(imaginarios\*f.reales);

return s;

}

Complex\_No operator/(Complex\_No f){

Complex\_No s;

s.reales=( (reales\*f.reales)+(imaginarios\*f.imaginarios) )/pow(f.reales,2)+pow(f.imaginarios,2);

s.imaginarios=( (imaginarios\*f.reales)-(reales\*f.imaginarios) )/pow(f.reales,2)+pow(f.imaginarios,2);

return s;

}

Complex\_No &operator++(){

reales++;

imaginarios++;

return \*this;

}

Complex\_No &operator++(int){

reales++;

imaginarios++;

return \*this;

}

Complex\_No &operator--(){

reales--;

imaginarios--;

return \*this;

}

Complex\_No &operator--(int){

reales--;

imaginarios--;

return \*this;

}

bool operator>(Complex\_No f){

if(reales\*f.imaginarios>imaginarios \* f.reales){

return true;

}

else

return false;

}

bool operator<(Complex\_No f){

if(imaginarios\*f.reales<reales\*f.imaginarios){

return true;

}

else

return false;

}

bool operator==(Complex\_No f){

if(reales==f.reales && imaginarios==f.imaginarios){

return true;

}

else

return false;

}

double Magnitud(){

double r;

r=sqrt( pow(reales,2)+pow(imaginarios,2) );

return r;

}

void mostrar(){

if(imaginarios>=0){

cout<<"("<<reales<<" + "<<imaginarios<<"i)";

}

else{

cout<<"("<<reales<<imaginarios<<"i)";

}

}

};

int main(int argc, char \*argv[]) {

double x1, y1;

double x2, y2;

cout<<"Asigna los valores del Numero complejo 1: ";

cin>>x1>>y1;

cout<<"\nAsigna los valores del Numero complejo 2: ";

cin>>x2>>y2;

Complex\_No Comp1(x1,y1), Comp2(x2,y2);

cout<<"\n\nNumeros Complejos: "<<endl;

Comp1.mostrar();

cout<<endl;

Comp2.mostrar();

cout<<endl;

Complex\_No r;

cout<<endl<<endl<<"------SUMA------"<<endl;

Comp1.mostrar(); cout<<" + "; Comp2.mostrar(); cout<<" = ";

r=Comp1+Comp2;

r.mostrar();

cout<<endl<<endl<<"-----RESTA-----"<<endl;

Comp1.mostrar(); cout<<" - "; Comp2.mostrar(); cout<<" = ";

r=Comp1-Comp2;

r.mostrar();

cout<<endl<<endl<<"--MULTPLICACION--"<<endl;

Comp1.mostrar(); cout<<" \* "; Comp2.mostrar(); cout<<" = ";

r=Comp1\*Comp2;

r.mostrar();

cout<<endl<<endl<<"-----DIVICION-----"<<endl;

Comp1.mostrar(); cout<<" / "; Comp2.mostrar(); cout<<" = ";

r=Comp1/Comp2;

r.mostrar();

cout<<endl<<endl;

cout<<"Incrementando Prefijo..."<<endl;

Comp1.mostrar();

cout<<" ++ = ";

r=Comp1++;

r.mostrar();

cout<<endl<<endl;

cout<<"Incrementando Postfijo..."<<endl;

cout<<"++ ";

Comp2.mostrar();

cout<<" = ";

r=Comp2++;

r.mostrar();

cout<<endl<<endl;

cout<<"Decrementando Prefijo..."<<endl;

Comp1.mostrar();

cout<<" -- = ";

r=Comp1--;

r.mostrar();

cout<<endl<<endl;

cout<<"Decrementando Postfijo..."<<endl;

cout<<"-- ";

Comp2.mostrar();

cout<<" = ";

r=Comp2--;

r.mostrar();

cout<<endl<<endl;

if(Comp1>Comp2){

cout<<"El Numero Complejo: ";

Comp1.mostrar();

cout<<"Es Mayor Que El Complejo: ";

Comp2.mostrar();

cout<<endl<<endl;

}

else if(Comp1<Comp2){

cout<<"El Numero Complejo: ";

Comp2.mostrar();

cout<<"Es Mayor Que El Complejo: ";

Comp1.mostrar();

cout<<endl<<endl;

}

else if(Comp1==Comp2){

cout<<"Los Numeros Complejos Son Iguales"<<endl;

Comp1.mostrar();

cout<<" = ";

Comp2.mostrar();

cout<<endl<<endl;

}

cout<<"\n\n---Magnitud---"<<endl;

cout<<"| ";

Comp1.mostrar();

cout<<"| =";

cout<<Comp1.Magnitud();

cout<<endl<<endl;

cout<<"| ";

Comp2.mostrar();

cout<<"| =";

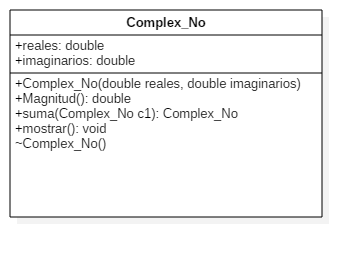
cout<<Comp2.Magnitud();

cout<<endl<<endl;

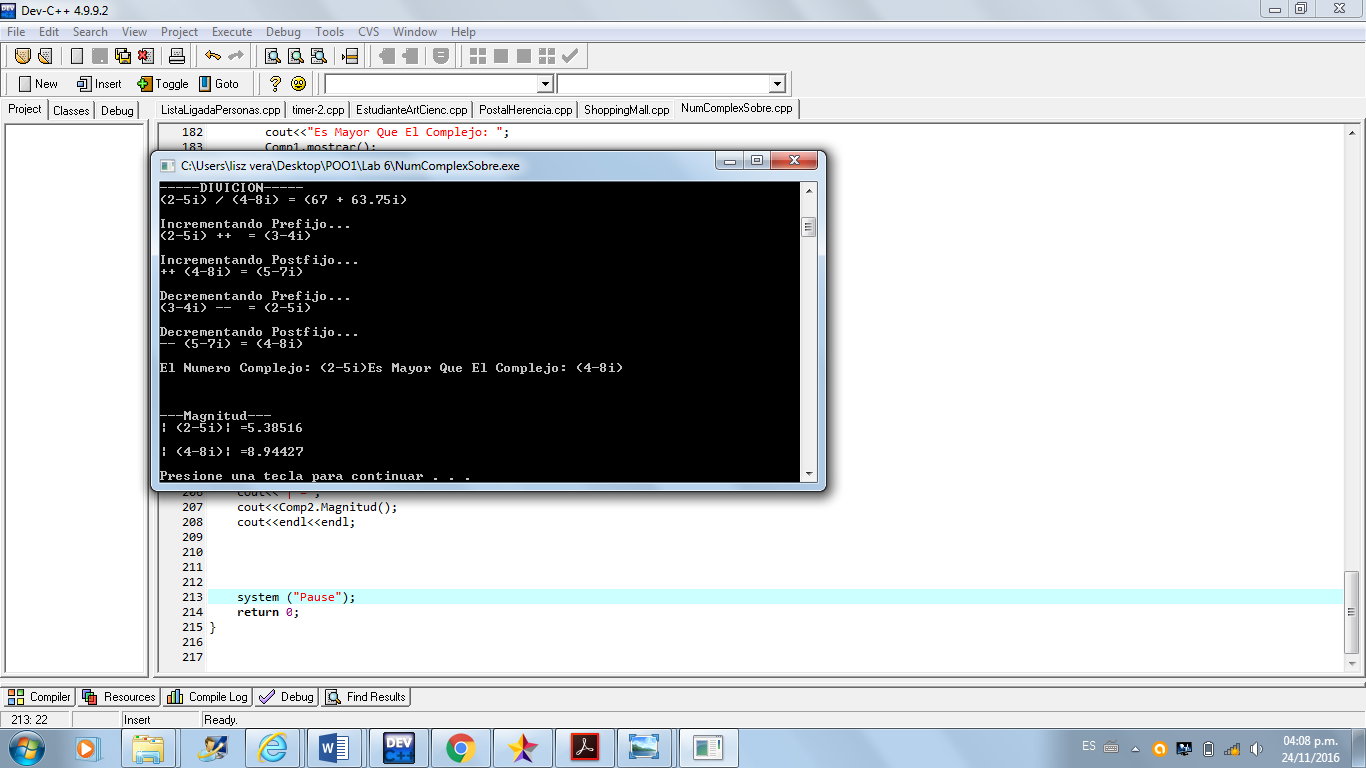
return 0;

}

UML



Pantallas



Codigo del Programa

#include <iostream>

using namespace std;

class Employee{

protected:

int NumberEmployee;

string NameEmployee;

float pagaTotal;

float deducciones;

float pagoBancario;

float salario;

int dia;

int mes;

int anio;

public:

Employee(){

NumberEmployee=0;

NameEmployee=" ";

pagaTotal=0.0;

deducciones=0.0;

pagoBancario=0.0;

salario=0.0;

dia=0;

mes=0;

anio=0;

}

Employee(int NumberEmployee, string NameEmployee, float pagaTotal, float deducciones, float pagoBancario, float salario, int dia, int mes, int anio){

this->NumberEmployee=NumberEmployee;

this->NameEmployee=NameEmployee;

this->pagaTotal=pagaTotal;

this->deducciones=deducciones;

this->pagoBancario=pagoBancario;

this->salario=salario;

this->dia=dia;

this->mes=mes;

this->anio=anio;

}

float salarioTFinal(){

pagaTotal=pagoBancario-deducciones;

salario=pagaTotal;

return salario;

}

void pedir(){

cout<<"Datos Del Empleado: "<<endl

<<"Numero ID: ";

cin>>NumberEmployee;

cout<<endl<<"Nombre: ";

cin>>NameEmployee;

cout<<endl;

//pedirFecha();

//pedirSalario();

}

void mostrar(){

cout<<"Nombre: "<<NameEmployee<<endl

<<"Numero ID: "<<NumberEmployee<<endl;

//mostrarFecha();

//mostrarSalario();

}

};

class Fecha:public Employee{

public:

Fecha(int dia, int mes, int anio):Employee(dia, mes, anio){

}

void pedirFecha(){

cout<<"Ingrese Fecha: "<<endl

<<"Dia: ";

cin>>dia;

cout<<endl<<"Mes: ";

cin>>mes;

cout<<endl<<"Anio: ";

cin>>anio;

cout<<endl;

}

void mostrarFecha(){

cout<<" Mostrando Fecha"<<endl

<<"(Dia, Mes, Anio)-> "

<<dia<<" / "<<mes<<" / "<<anio<<endl;

}

};

class PagoFinal:public Employee{

public:

PagoFinal(float pagaTotal, float deducciones, float pagoBancario, float salario):Employee(pagaTotal, deducciones, pagoBancario, salario){

}

void pedirSalario(){

cout<<"Ingrese Detalles De Salario: "<<endl

<<"Salario Bruto: ";

cin>>pagaTotal;

cout<<endl<<"Deducciones: ";

cin>>deducciones;

cout<<endl<<"Pago En Coins: ";

cin>>pagoBancario;

cout<<endl<<"Calculando..."<<endl

<<"Salario Total: "<<salarioTFinal()<<endl;

}

void mostrarSalario(){

cout<<"Salario: "<<salarioTFinal()<<endl;

}

};

int main(int argc, char \*argv[]) {

return 0;

}

UML

Pantallas

Programa de Tiempo (8)

#include<iostream>

using namespace std;

class Timer{

protected:

int hora,minuto,seg;

public:

Timer(){

hora=0;

minuto=0;

seg=0;

}

int incrementos(){

while(hora!=24){

cout<<"seg:"<<seg<<" minutos:"<<minuto<<" hora:"<<hora<<endl;

seg++;

if(seg==60){

seg=0;

minuto++;

if(minuto==60){

minuto=0;

hora++;

if(hora==24){

return hora;

}

}

}

}

}

};

class Excepcion:public Timer{

private :

int h;

public:

Excepcion():Timer (){

h=hora;

}

void excepcion(Timer s){

if(s.incrementos()==24){

cout<<"excepcion de desbordamiento de temporizador..."<<endl;

}

}

};

int main(){

Timer a;

Excepcion s;

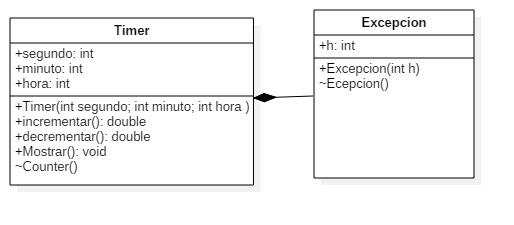
a.incrementos();

s.excepcion(a);

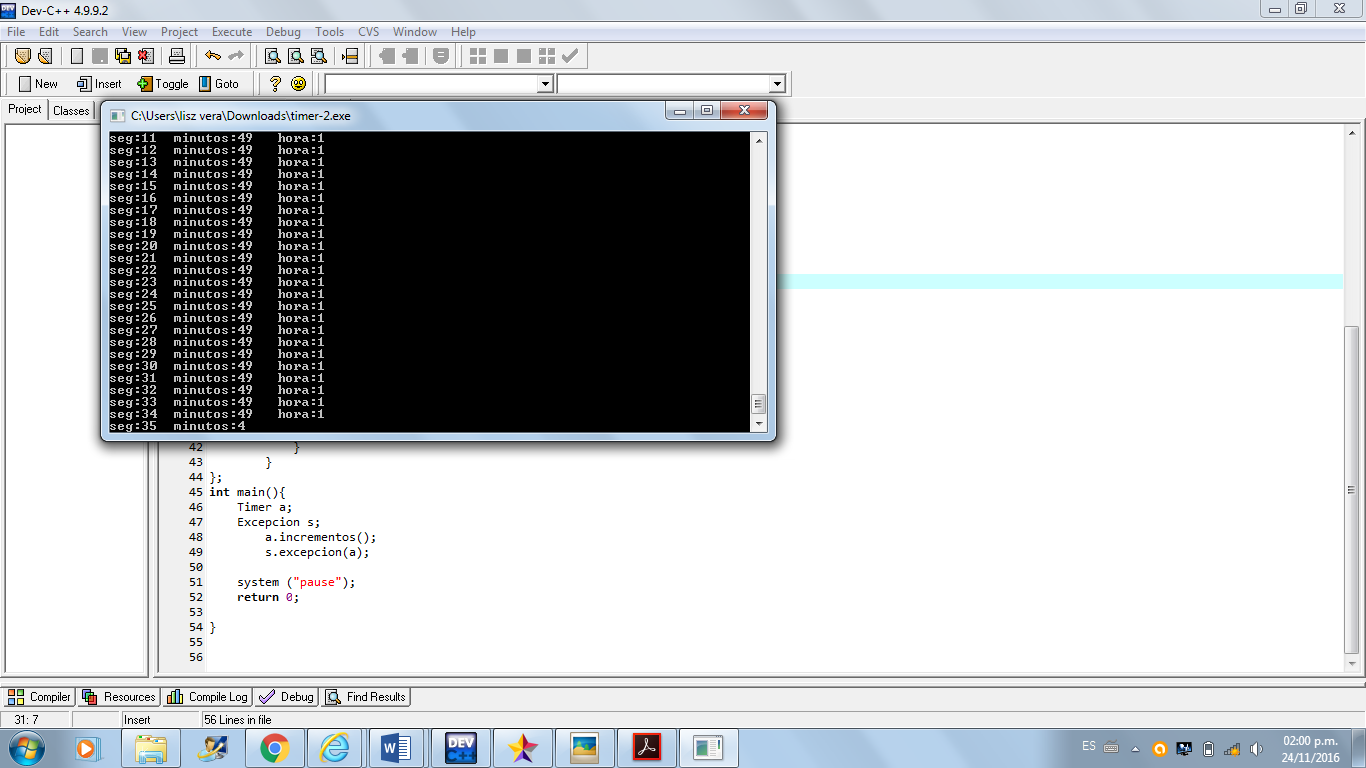
return 0;

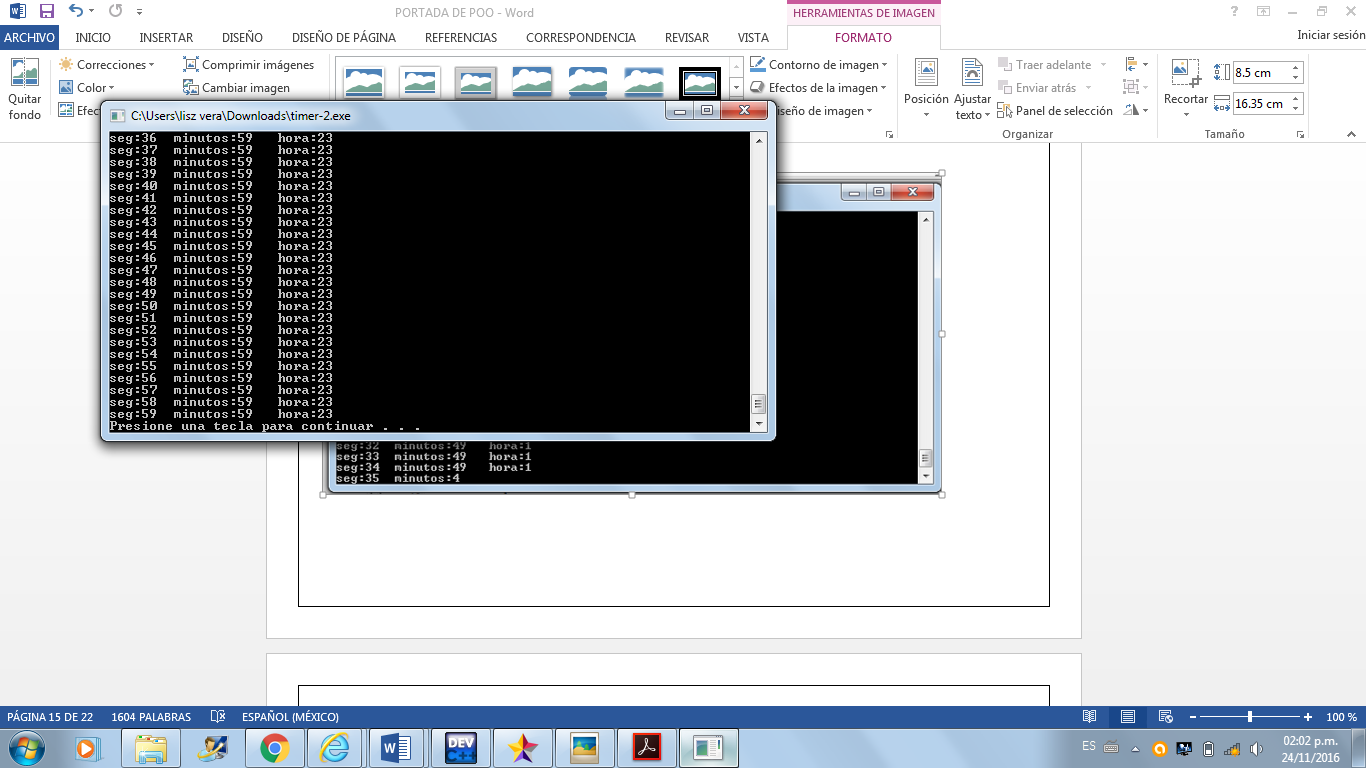
}

UML



Pantallas





Código del Programa

#include <iostream>

#include <fstream>

using namespace std;

class Texto

{

private:

int n;

string l;

public:

Texto()

{

n=1;

}

string ingresaFrase()

{

ofstream entrada("Frase.txt",ios::app);

cout<<"Introducir la frase u usar ! para terminar :"<<endl;

getline(cin,l);

entrada<<l;

entrada.close();

}

};

int main()

{

Texto t1;

t1.ingresaFrase();

return 0;

}

Listas Ligadas de Personas

Código

#include <iostream>

using namespace std;

class Person{

private:

string nombre;

string email;

int edad;

int peso;

string ciudad;

string direccion;

public:

int getEdad(){ return edad;}

string getCiudad(){return ciudad;}

string getNombre(){return nombre;}

Person(string nombre, string email, int edad,int peso, string ciudad, string direccion){

this->nombre=nombre;

this->email=email;

this->edad=edad;

this->peso=peso;

this->ciudad=ciudad;

this->direccion=direccion;

}

Person(){

nombre="";

email="";

edad=0;

peso=0;

ciudad="";

direccion="";

}

friend ostream& operator<<(ostream& out, const Person& p){// output

return out <<"Nombre: " << p.nombre<<endl

<<"Email: "<< p.email<<endl

<<"Edad: "<< p.edad<<endl

<<"Peso: "<< p.peso<<endl

<<"Ciudad: "<< p.ciudad<<endl

<<"Direccion " << p.direccion<<endl;

}

friend istream& operator>>(istream& in, Person& p){//input

cout <<"Nombre: ";

in>>p.nombre;

cout<<"Email: ";

in>>p.email;

cout<<"Edad: ";

in>>p.edad;

cout<<"Peso: ";

in>>p.peso;

cout<<"Ciudad: ";

in>>p.ciudad;

cout<<"Direccion";

in>>p.direccion;

return in;

}

void setEdad(int e){ edad=e;}

};

class Node{

public:

Person person;

Node \*sig;

Node \*ant;

public:

Node(Person p){

person=p;

sig=NULL;

ant=NULL;

}

};

class LinkedList{

private:

Node \*head;

Node \*tail;

public:

LinkedList(){

head=NULL;

tail=NULL;

}

Person getPerson(Node \*p){

return p->person;

}

void agregarPersonaAlFinal(Person p){

Node \*q=new Node(p);

if(head==NULL){

head=q;

tail=q;

}

else{

tail->sig=q;

tail=q;

}

}

void printList(){

Node \*p=NULL;

for(p=head;p!=NULL;p=p->sig){

cout<<p->person;

cout<<"--------------------------------------"<<endl;

}

}

Node \*buscarPerson(string nombre){

Node \*p=NULL;

for(p=head;p!=NULL;p=p->sig){

if(p->person.getNombre()==nombre){

return p;

}

}

return p;

}

void eliminarPersona(string nombre){

Node \*p=NULL;

Node \*q=NULL;

p=head;

q=head->sig;

if(head->person.getNombre()== nombre){

head=head->sig;

p->sig=NULL;

delete p;

cout<<"Persona Eliminada..."<<endl;

}else{

while(q!=NULL){

if(q->person.getNombre()==nombre){

break;

}else {

p=p->sig;

q=q->sig;

}

}

if(q!=NULL){//borrar el nodo q

p->sig=q->sig;

q->sig=NULL;

delete q;

cout<<"Persona Eliminada..."<<endl;

}else {

cout<<"La Persona NO Existe en la Lista"<<endl;

}

}

}

void modificarPersona(string nombre){

int newEdad;

Node \*p= buscarPerson(nombre);

if(p!=NULL){

cout<<" Como prueba solo modificara la edad: "<<endl;

cout<<"Ingresa la nueva edad: ";

cin>>newEdad;

p->person.setEdad(newEdad);

cout<<" El Nodo fue moficado... :)";

} else{

cout<<" :( La persona no Existe...";

}

}

float getPromedioEdades(){

Node \*p=head;

float suma=0.0;

int n=0;

for(p=head;p!=NULL;p=p->sig){

n++;

suma+=p->person.getEdad();

}

return suma/n;

}

};

int main(int argc, char \*argv[]) {

/\*int m;

cout<<"\t\t\t Elija una opcion \n \n" ;

cout<<"1 Agregar Nodo \n";

cout<<"2 Eliminar Nodo\n";

cout<<"1 \n";

cout<<"1 Salir\n";

cin>>m;

switch (m)

{

case 1:cout<<"Datos de la persona"<<endl;break;

case 2:cout<<"Persona que quiera eliminar"<<endl;break,

case 3:cout<<""<<endl;break;

case 4:cout<<"ya esta fuera"<<endl;break;

}\*/

Person per1("Carmen","Carm\_22@gmail.com",22,68,"Jalisco","Av San Miguel, Colonia: Los Fuertes, #434");

Person per2("Albert", "Beto\_45@gmail.com",45,80,"Puebla","Av Morelos, Colonia: San Cristobal, #654");

Person per3("Carlos","C\_27\_2015@gmail.com",27,70,"Jalisco","Av 5 de Mayo, Colonia: Buganbilias, #231");

Person per4("Juan", "Juan\_12@gmail.com",45,85,"Puebla","Av Emiliano Zapata, Colonia: La vista, #131");

Person per5("Saul","Saul\_52@gmail.com",52,87,"DF","Av 3 de Octubre, Colonia:Granjas del sur, #154");

Person per6("David", "Dab\_1889@gmail.com",34,90,"Puebla","Av Rodolfo Sanchez Taboada, Colonia:San Antonio, #222");

Person per7("Jesus", "Yisus\_87@gmail.com",44,76,"Puebla","Av Pancho Villa, Colonia:Los Mirasoles, #564");

/\*cin>>per2;

cout<<per1<<endl; SI FUNCIONA

cout<<per2;\*/

LinkedList LL;

LL.agregarPersonaAlFinal(per1);

LL.agregarPersonaAlFinal(per2);

LL.agregarPersonaAlFinal(per3);

LL.agregarPersonaAlFinal(per4);

LL.agregarPersonaAlFinal(per5);

LL.agregarPersonaAlFinal(per6);

LL.agregarPersonaAlFinal(per7);

LL.printList();

LL.eliminarPersona("Jesus");

cout<<"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;

LL.printList();

cout<<"Promedio de edades de las Personas de la Lista: "<<LL.getPromedioEdades()<<endl;

//LL.modificarPersona("Saul");

LL.printList();

system ("Pause");

return 0;

}

Pantalla

