**Rational**

**main.cpp**

#include <iostream>

using namespace std ;

#include "Rational.h"

int main() {

Rational F1, F2, F4;

F1.setNum(1);

F1.setDen(2);

F2.setNum(1);

F2.setDen(4);

/\*

//Operaciones usando funciones.

cout << "WITH FUNCTIONS:" << endl;

//Math

F1.Add(F2);

F1.Subtract(F2);

F1.Multiply(F2);

F1.Divide(F2);

//Display

F1.DisplayFract();

F1.DisplayFloat();

//Copy Constructor\*/

Rational F3(F2);

cout << "\n--COPIED FRACTION--" << endl;

//F3.DisplayFract();

//cout << endl;

//Uso de operadores.

cout << "WITH OPERATORS:" << endl;

cout << "SUM: \n" << F1 << "+ " << F2;

F3 = F1 + F2;

cout <<"= " << F3 << endl;

cout << "SUBCTRACTION: \n" << F1 << "- " << F2;

F3 = F1 - F2;

cout <<"= " << F3 << endl;

cout << "MUTIPLICATION: \n" << F1 << "\* " << F2;

F3 = F1 \* F2;

cout <<"= " << F3 << endl;

cout << "DIVISION:\n" << F1 << "/ " << F2;

F3 = F1 / F2;

cout <<"= " << F3 << endl;

cout << F3 << endl;

return 0;

}

**Rational.h**

// Rational.h

#ifndef RATIONAL\_H\_

#define RATIONAL\_H\_

#include <stdio.h>

class Rational {

private:

int numerator;

int denominator;

public:

//Constructor & Destructor

Rational( int numerator = 0, int denominator = 1);

~Rational();

//Getters and Setters

int getNum() const;

int getDen() const;

void setNum(int a);

void setDen(int b);

//Math

void Add(Rational a1);

void Subtract(Rational s1);

void Multiply(Rational m1);

void Divide(Rational d1);

Rational GCD();

double CalculateFloat();

//Display

void DisplayFract() const;

void DisplayFloat();

//Copy Constructor

Rational(const Rational &aRational);

//Overloaded Operators

Rational operator + (const Rational &tempRational);// addition operator

Rational operator - (const Rational &tempRational);// subtraction operator

Rational operator \* (const Rational &tempRational);// multiplication operator

Rational operator / (const Rational &tempRational);// division operator

Rational &operator=(const Rational &);//assign operator

bool operator >(Rational &tempRational);// is greater than operator

bool operator <(Rational &tempRational);//is less than operator

bool operator ==(const Rational &);//is equal to operator

friend ostream &operator<<(ostream &, const Rational &tempRational);//cout operator

friend istream &operator>>(std::istream &, Rational &);//cin operator

};

#endif /\* Rational\_h \*/

**Rational.cpp**

// Rational.cpp

#include <iostream>

using namespace std;

#include "Rational.h"

//Overloaded Operators

double Rational::CalculateFloat() {

double q1 = (double)this->getNum();

double q2 = (double)this->getDen();

return q1 / q2;

}

Rational Rational::operator + (const Rational &tempRational) {

int cd = this->getDen() \* tempRational.getDen();

int Num1 = this->getNum() \* tempRational.getDen();

int Num2 = tempRational.getNum() \* this->getDen();

Rational R1(Num1 + Num2, cd);

R1.GCD();

return R1;

}

Rational Rational::operator - (const Rational &tempRational) {

int cd = this->getDen() \* tempRational.getDen();

int Num1 = this->getNum() \* tempRational.getDen();

int Num2 = tempRational.getNum() \* this->getDen();

Rational R1(Num1 - Num2, cd);

R1.GCD();

return R1;

}

Rational Rational::operator \* (const Rational &tempRational) {

int cdr = this->getDen() \* tempRational.getDen();

int Num5 = this->getNum() \* tempRational.getNum();

Rational R1(Num5, cdr);

R1.GCD();

return R1;

}

Rational Rational::operator / (const Rational &tempRational) {

int cdd = this->getDen() \* tempRational.getNum();

int Num6 = this->getNum() \* tempRational.getDen();

Rational R1(Num6, cdd);

R1.GCD();

return R1;

}

Rational & Rational::operator=(const Rational &tempRational) {

this->numerator = tempRational.getNum();

this->denominator = tempRational.getDen();

return \*this;

}

//"Is more than" operator

bool Rational::operator>(Rational &tempRational) {

bool flag = false;

if (this -> CalculateFloat() > tempRational.CalculateFloat()) {

flag = true;

}

return flag;

}

//"Is less than" operator

bool Rational::operator<(Rational &tempRational) {

bool flag = false;

if (this -> CalculateFloat() < tempRational.CalculateFloat()) {

flag = true;

}

return flag;

}

//"Is equal to" operator

bool Rational::operator==(const Rational &tempRational) {

bool flag = false;

if(tempRational.getNum() == this-> getNum() & tempRational.getDen() == this-> getDen()) {

flag = true;

}

return flag;

}

ostream &operator<<(ostream &output, const Rational &tempRational) {

output << "(" << tempRational.getNum() << "/" << tempRational.getDen() << ") ";

return output;

}///end operator

istream &operator>>(istream &input, Rational &tempRational) {

cout << "Entra el valor de la base: " << endl;

input >> tempRational.numerator;

cout << "Entra el valor de la altura: " << endl;

input >> tempRational.denominator;

return input;

}

//Constructor, Destructor, Getters and Setters

Rational::Rational( int numerator, int denominator): numerator(numerator), denominator(denominator)

{}

Rational::~Rational() {

//cout << "Rational object destroyed.\n";

}

//Getters and Setters

int Rational::getNum() const {

return numerator;

}

int Rational::getDen() const {

return denominator;

}

void Rational::setNum(int a) {

numerator = a;

}

void Rational::setDen(int b) {

denominator = b;

}

//Math

Rational Rational::GCD() {

int r = 1, a = this->getNum() , b = this->getDen();

if(a < 0) {

a \*= -1;

}

else if(b < 0) {

b \*= -1;

}

while(b > 0)

{

r = a % b;

a = b;

b = r;

}

this->setNum(this->getNum()/a);

this->setDen(this->getDen()/a);

return \*this;

}

void Rational::Add(Rational a1) {

int cd = getDen() \* a1.getDen();

int Num1 = getNum() \* a1.getDen();

int Num2 = a1.getNum() \* getDen();

cout << "SUM: (" << getNum() << " / " << getDen() << ") + (" << a1.getNum() << " / " << a1.getDen() << ") = (" << Num1 + Num2 << " / " << cd << ")"<< endl;

}

void Rational::Subtract(Rational s1) {

int cds = getDen()\* s1.getDen();

int Num3 = getNum() \* s1.getDen();

int Num4 = s1.getNum() \* getDen();

cout << "SUBTRACTION: (" << getNum() << " / " << getDen() << ") - (" << s1.getNum() << " / " << s1.getDen() << ") = (" << Num3 - Num4 << " / " << cds << ")"<< endl;

}

void Rational::Multiply(Rational m1) {

int cdr = getDen()\* m1.getDen();

int Num5 = getNum() \* m1.getNum();

cout << "MULTIPLICATION: (" << getNum() << " / " << getDen() << ") \* (" << m1.getNum() << " / " << m1.getDen() << ") = (" << Num5 << " / " << cdr << ")"<< endl;

}

void Rational::Divide(Rational d1) {

int cdd = getDen()\* d1.getNum();

int Num6 = getNum() \* d1.getDen();

cout << "DIVISION: (" << getNum() << " / " << getDen() << ") / (" << d1.getNum() << " / " << d1.getDen() << ") = (" << Num6 << " / " << cdd << ")"<< endl;

}

//Display

void Rational::DisplayFract() const {

cout <<"DISPLAY FRACTION: " << this->getNum() << " / " << this->getDen() << endl;

}

void Rational::DisplayFloat() {

cout <<"DISPLAY DECIMAL OF FRACTION: " << this->getNum() << " / " << this->getDen() <<" = " << this->CalculateFloat() << endl;

}

//Copy Constructor

Rational::Rational( const Rational &aRational) {

setNum( aRational.getNum() );

setDen( aRational.getDen() );

//cout << "Copied fraction using the copy constructor." << endl;

}

**Parking Ticket Sim**

**main.cpp**

#include<string>

#include"ParkedCar.h"

#include"ParkingMeter.h"

#include"PoliceOfficer.h"

#include<iostream>

using namespace std;

int main()

{

ParkedCar car;

ParkingMeter meter;

string aname, abadge, amake, amodel, alicensenumber, acolor;

int aminutespurchased, aparkedminutes;

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

cout << " ~Parking Ticket Simulator~" << endl;

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

cout << endl;

cout << "Vehicle Information:" << endl;

cout << "Make: ";

cin>>amake;

cout << "Model: ";

cin>>amodel;

cout << "Color: ";

cin>>acolor;

cout << "License Number: ";

cin>>alicensenumber;

cout << "Parked Minutes: ";

cin>> aparkedminutes;

car.set(amake, amodel, acolor, alicensenumber, aparkedminutes);

cout << "Minutes Purchased: ";

cin >> aminutespurchased;

meter.setparkingmeter(aminutespurchased);

cout << endl;

cout << "Officer Information:" << endl;

cout << "Name: ";

cin>>aname;

cout << "Badge ID: ";

cin>>abadge;

PoliceOfficer officer(aname, abadge,amake,amodel,acolor,alicensenumber,aparkedminutes,aminutespurchased);

system("cls");

officer.patrol();

cout << endl;

return 0;

}

**ParkedCar.h**

#ifndef PARKEDCAR\_H

#define PARKEDCAR\_H

#include<iostream>

#include<string>

using namespace std;

class ParkedCar

{

private:

string make;

string model;

string color;

string licensenumber;

int parkedminutes;

public:

ParkedCar();

ParkedCar(string, string, string, string, int);

~ParkedCar();

void setmake(string);

void setmodel(string);

void setcolor(string);

void setlicensenumber(string);

void setparkedminutes(int);

void set(string, string, string, string, int);

string getmake() const;

string getmodel()const;

string getcolor()const;

string getlicensenumber()const;

int getparkedminutes()const;

void parkedcardisplay() const;

};

#endif

**ParkedCar.cpp**

#include"ParkedCar.h"

ParkedCar::ParkedCar()

{

make=" ";

model = " ";

color = " ";

licensenumber = " ";

parkedminutes = 0;

}

ParkedCar::ParkedCar(string amake, string amodel, string acolor, string alicensenumber, int aparkedminutes)

{

make = amake;

model = amodel ;

color = acolor;

licensenumber = alicensenumber;

parkedminutes = aparkedminutes;

}

ParkedCar::~ParkedCar() {}

void ParkedCar::setmake(string amake)

{

make = amake;

}

void ParkedCar::setmodel(string amodel)

{

model = amodel;

}

void ParkedCar::setcolor(string acolor)

{

color = acolor;

}

void ParkedCar::setlicensenumber(string alicensenumber)

{

licensenumber = alicensenumber;

}

void ParkedCar::setparkedminutes(int aparkedminutes)

{

parkedminutes = aparkedminutes;

}

void ParkedCar::set(string amake, string amodel, string acolor, string alicensenumber, int aparkedminutes)

{

make = amake;

model = amodel;

color = acolor;

licensenumber = alicensenumber;

parkedminutes = aparkedminutes;

}

string ParkedCar::getmake() const

{

return make;

}

string ParkedCar::getmodel()const

{

return model;

}

string ParkedCar::getcolor()const

{

return color;

}

string ParkedCar::getlicensenumber()const

{

return licensenumber;

}

int ParkedCar::getparkedminutes()const

{

return parkedminutes;

}

void ParkedCar::parkedcardisplay() const

{

cout << "Make: " << make << endl;

cout << "Model: " << model << endl;

cout << "Color: " << color << endl;

cout << "License Number: " << licensenumber << endl;

}

**ParkingMeter.h**

#ifndef PARKINGMETER\_H

#define PARKINGMETER\_H

#include<iostream>

using namespace std;

class ParkingMeter {

private:

int minutespurchased;

public:

ParkingMeter();

ParkingMeter(int);

~ParkingMeter();

void setparkingmeter(int);

int getparkingmeter() const;

};

#endif

**ParkingMeter.cpp**

#include"ParkingMeter.h"

ParkingMeter::ParkingMeter()

{

minutespurchased = 0;

}

ParkingMeter::ParkingMeter(int aminutespurchased)

{

minutespurchased = aminutespurchased;

}

ParkingMeter::~ParkingMeter() {}

void ParkingMeter::setparkingmeter(int aminutespurchased)

{

minutespurchased = aminutespurchased;

}

int ParkingMeter::getparkingmeter() const

{

return minutespurchased;

}

**ParkingTicket.h**

#ifndef PARKINGTICKET\_H

#define PARKINGTICKET\_H

#include<iostream>

#include"PoliceOfficer.h"

#include<string>

using namespace std;

class ParkingTicket {

private:

int fine;

ParkedCar PC;

ParkingMeter PM;

PoliceOfficer PO;

public:

ParkingTicket();

ParkingTicket(PoliceOfficer PO);

ParkingTicket(string aname, string abadge, string amake, string amodel, string acolor, string alicensenumber, int aparkedminutes, int aminutespurchased);

~ParkingTicket();

int getfine() const;

void generateticket();

void reportCarInfo();

void reportfine();

void reportOfficer();

};

#endif

**ParkingTicket.cpp**

#include"ParkingTicket.h"

#include"PoliceOfficer.h"

#include <cmath>

ParkingTicket::ParkingTicket()

{

fine = 0;

}

ParkingTicket::ParkingTicket(string aname, string abadge, string amake, string amodel, string acolor, string alicensenumber, int aparkedminutes, int aminutespurchased)

{

PC.set(amake, amodel, acolor, alicensenumber, aparkedminutes);

PM.setparkingmeter(aminutespurchased);

//PO.setOfficer(aname, abadge);

}

ParkingTicket::ParkingTicket(PoliceOfficer PO)

{

PC.set(PO.getCar().getmake(),PO.getCar().getmodel(),PO.getCar().getcolor(),PO.getCar().getlicensenumber(),PO.getCar().getparkedminutes());

PM.setparkingmeter(PO.getMeter().getparkingmeter());

//PO.setOfficer(PO.getname(),PO.getbadge());

}

ParkingTicket::~ParkingTicket()

{

}

void ParkingTicket::reportCarInfo()

{

PC.parkedcardisplay();

}

void ParkingTicket::reportfine()

{

int difference = abs(PC.getparkedminutes() - PM.getparkingmeter());

int hour = 60; //60 mins = 1 hour

if (PC.getparkedminutes() > PM.getparkingmeter()) {

double extra\_hours = (double) difference / (double) hour;

if(extra\_hours <= 1) {

fine = 25;

} else if(extra\_hours > 1 ) {

int h = (int) round(extra\_hours);

fine = 25 + (h-1) \* 10;

}

}

cout << "Fine: $" << getfine() << endl;

}

void ParkingTicket::reportOfficer()

{

PO.displayofficer();

}

int ParkingTicket::getfine() const

{

return fine;

}

void ParkingTicket::generateticket()

{

cout << endl;

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

cout << " ~ILLEGAL PARKING TICKET~" << endl;

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

reportCarInfo();

reportfine();

//reportOfficer();

cout << endl;

}

**PoliceOficcer.h**

#ifndef POLICEOFFICER\_H

#define POLICEOFFICER\_H

#include<iostream>

#include"ParkedCar.h"

#include"ParkingMeter.h"

#include<string>

using namespace std;

class PoliceOfficer {

private:

string name;

string badge;

ParkedCar PC;

ParkingMeter PM;

public:

ParkedCar getCar();

ParkingMeter getMeter();

PoliceOfficer();

PoliceOfficer(string aname, string abadge, string amake, string amodel, string acolor, string alicensenumber, int aparkedminutes, int aminutespurchased);

~PoliceOfficer();

void setname(string);

void setbadge(string);

void setOfficer(string, string);

string getname() const;

string getbadge() const;

void patrol();

void displayofficer() const;

};

#endif

**PoliceOfficer.cpp**

#include"PoliceOfficer.h"

#include"ParkingTicket.h"

#include<iostream>

#include<string>

using namespace std;

PoliceOfficer::PoliceOfficer()

{

badge = " ";

name = " ";

}

PoliceOfficer::PoliceOfficer(string aname, string abadge, string amake, string amodel, string acolor, string alicensenumber, int aparkedminutes, int aminutespurchased)

{

PC.set(amake, amodel, acolor, alicensenumber, aparkedminutes);

PM.setparkingmeter(aminutespurchased);

setOfficer(aname, abadge);

}

PoliceOfficer::~PoliceOfficer()

{

}

void PoliceOfficer::setname(string aname)

{

name = aname;

}

void PoliceOfficer::setbadge(string abadge)

{

badge = abadge;

}

void PoliceOfficer::setOfficer(string aname, string abadge)

{

badge = abadge;

name = aname;

}

string PoliceOfficer::getname() const

{

return name;

}

ParkedCar PoliceOfficer::getCar()

{

return PC;

}

ParkingMeter PoliceOfficer::getMeter()

{

return PM;

}

string PoliceOfficer::getbadge() const

{

return badge;

}

void PoliceOfficer::displayofficer() const

{

cout << "Name: " << getname() << endl;

cout << "Badge ID: " << getbadge() << endl;

}

void PoliceOfficer::patrol()

{

if (PC.getparkedminutes() > PM.getparkingmeter()) /\*parked minutes >= minutes purchased\*/

{

ParkingTicket PT(\*this);

PT.generateticket();

displayofficer();

}

else

{

cout << "No crime has been commited." << endl;

}

}

**Mortage Payment** (without operators)

**main.cpp**

//

// main.cpp

// Clase 9 - Talavera - Mortgage Payment Class

//

// Created by Rafael Pagan on 4/4/17.

// Copyright © 2017 Rafael Pagan. All rights reserved.

//

#include <iostream>

#include "MortgagePayment.h"

#include "Loan.h"

using namespace std;

int main() {

Loan L;

L.setContact();

L.setLoanNo();

L.setFinances();

L.displayLoan();

return 0;

}

**Loan.h**

#ifndef LOAN\_H

#define LOAN\_H

#include <stdio.h>

#include <string>

#include "MortgagePayment.h"

#include "ContactInfo.h"

using namespace std;

class Loan {

private:

string loan\_no;

MortgagePayment finances;

ContactInfo contact;

public:

//Constructors & Destructor

Loan() ;

Loan(string loan\_no, MortgagePayment finances, ContactInfo contact);

~Loan();

//Accessors

string getLoanNo() const;

MortgagePayment getFinances() const;

ContactInfo getContact() const;

//Mutators

void setLoanNo();

void setFinances();

void setContact();

//Display info

void displayLoan();

};

#endif /\* Loan\_h \*/

**Loan.cpp**

#include "Loan.h"

#include "ContactInfo.h"

#include "MortgagePayment.h"

#include <iostream>

//Constructors & Destructor

Loan::Loan() : contact() , finances() {

loan\_no = "";

}

Loan::Loan(string loan\_no, MortgagePayment finances, ContactInfo contact): loan\_no(loan\_no), finances(finances), contact(contact) {}

Loan::~Loan() {}

//Accessors

string Loan::getLoanNo() const {

return loan\_no;

}

MortgagePayment Loan::getFinances() const {

return finances;

}

ContactInfo Loan::getContact() const {

return contact;

}

//Mutators

void Loan::setLoanNo() {

cout << "Enter the loan number: " << endl;

cin >> this->loan\_no;

}

void Loan::setFinances() {

this->finances.setLoan();

this->finances.setYears();

this->finances.setRate();

this->finances.setTerm();

}

void Loan::setContact() {

cout << "Enter contact info." << endl;

cout << "Name: " << endl;

cin >> this->contact.getName();

cout << "Phone: " << endl;

cin >> this->contact.getPhone();

}

void Loan::displayLoan() {

this->finances.displayPayment();

}

**MortgagePayment.h**

//

// MortgagePayment.hpp

// Clase 9 - Talavera - Mortgage Payment Class

//

// Created by Rafael Pagan on 4/4/17.

// Copyright © 2017 Rafael Pagan. All rights reserved.

//

#ifndef MortgagePayment\_h

#define MortgagePayment\_h

#include <stdio.h>

class MortgagePayment {

private:

double payment; //monthly payment

double loan; //dollar amount of the loan Rate = the annual interest rate

double years; //number of years of the loan

double rate;

double term;

public:

//Constructor & Desctructor

MortgagePayment( double payment, double loan, double years, double rate, double term);

MortgagePayment();

~MortgagePayment();

//Accessors

double getPayment() const;

double getLoan() const;

double getYears() const;

double getRate() const;

double getTerm() const;

//Mutators

void setPayment();

void setLoan();

void setYears();

void setRate();

void setTerm();

//Display

void displayPayment();

};

#endif /\* MortgagePayment\_h \*/

**MortgagePayment.cpp**

#include "MortgagePayment.h"

#include <cmath>

#include <iostream>

using namespace std;

//Constructor & Desctructor

MortgagePayment::MortgagePayment( double payment, double loan, double years, double rate, double term) : payment(payment), loan(loan), years(years), rate(rate), term(term) {}

MortgagePayment::MortgagePayment() {

this->payment=1;

this->loan=1;

this->years=1;

}

MortgagePayment::~MortgagePayment() {}

//Accessors

double MortgagePayment::getPayment() const {

return payment;

}

double MortgagePayment::getLoan() const {

return loan;

}

double MortgagePayment::getYears() const {

return years;

}

double MortgagePayment::getRate() const {

return rate;

}

double MortgagePayment::getTerm() const {

return term;

}

//Mutators

void MortgagePayment::setPayment() {

payment = (loan \* (rate/12) \* term) / (term - 1);

}

void MortgagePayment::setLoan() {

cout << "Enter the loan amount in dollars:" << endl;

cin >> this->loan;

}

void MortgagePayment::setYears() {

cout << "Enter of years of the loan:" << endl;

cin >> this->years;

}

void MortgagePayment::setRate() {

cout << "Enter annual interest rate of the loan:" << endl;

cin >> this->rate;

}

void MortgagePayment::setTerm() {

term = pow((1 + (rate/12)), (12 \* years));

}

void MortgagePayment::displayPayment() {

this->setTerm();

this->setPayment();

cout << "--Loan Summary--" << endl;

cout << "Loan amount: $" << this->loan << endl;

cout << "Years of the loan: " << this->years << endl;

cout << "Annual interest rate: " << this->rate << endl;

cout << "Term: " << this->term << endl;

cout << "Total Monthly Payment: $" << this->payment << endl;

}

**ContactInfo.h**

#ifndef INFO\_H\_

#define INFO\_H\_

#include <stdio.h>

#include <string> // Needed for strlen and strcpy

// ContactInfo class declaration.

class ContactInfo {

private:

char \*name; // The name

char \*phone; // The phone number

public:

// Constructor

ContactInfo();

ContactInfo(char \*n, char \*p);

// Destructor

~ContactInfo();

// Mutators

char \*getName() const;

char \*getPhone() const;

void setName(std::string n) const;

void setPhone(std::string p) const;

void Display() const;

// Copy Constructor

ContactInfo(const ContactInfo &aContactInfo);

};

#endif /\* Info\_h \*/

**ContactInfo.cpp**

#include <iostream>

#include "ContactInfo.h"

ContactInfo::ContactInfo(){ // Allocate just enough memory for the name and phone number.

name = new char[20];

phone = new char[20];

// Copy the name and phone number to the allocated memory.

strcpy(name, "");

strcpy(phone, "");

}

ContactInfo::ContactInfo(char \*n, char \*p){ // Allocate just enough memory for the name and phone number.

name = new char[strlen(n) + 1];

phone = new char[strlen(p) + 1];

// Copy the name and phone number to the allocated memory.

strcpy(name, n);

strcpy(phone, p);

}

//Rect::Rect(float base, float altura, float area): base(base), altura(altura), area(area)

//{}

ContactInfo::~ContactInfo() {

delete [] name;

delete [] phone;

}

char\* ContactInfo::getName() const {

return name;

}

char\* ContactInfo::getPhone() const {

return phone;

}

void ContactInfo::setName(std::string n) const {

n = \*name;

}

void ContactInfo::setPhone(std::string p) const {

p = \*phone;

}

void ContactInfo::Display() const {

std::cout << "\n--Contact Info--";

std::cout << "Name: " << getName() << "\n";

std::cout << "Phone: " << getPhone() << "\n";

}

ContactInfo::ContactInfo( const ContactInfo &aContactInfo) {

setName( aContactInfo.getName() );

setPhone( aContactInfo.getPhone() );

std::cout << "Copied rectangle using the copy constructor.\n";

}