**main.cpp**

#include <iostream>

#include "LoanList.h"

using namespace std;

/\*\*

15. Mortgage Payment - p.807, Gaddis

Design a class that will determine the monthly payment on a home mortgage. The monthly payment with

interest compounded monthly can be calculated as follows:

Payment = (Loan \* Rate \* Term) / (Term - 1)

where

Payment = the monthly payment

Loan = the dollar amount of the loan Rate = the annual interest rate

Years = the number of years of the loan

The class should have:

1) Member functions for setting the loan amount, interest rate, and number of years

of the loan.

2) It should also have member functions for returning the monthly payment amount and the

total amount paid to the bank at the end of the loan period.

3) Implement the class in a complete program.

Input Validation: Do not accept negative numbers for any of the loan values.

\*\*/

int main() {

LoanList LL;

Loan L, M;

//ContactInfo C("Rafael", "787-600-3684"), D;

ContactInfo C("1", "2"), D;

cout << "Contact Loan List" << endl;

/\*

cin >> L;

cout << L;

LL += L;

cin >> M;

cout << M;

LL += M;

\*/

cout << "Contact Info Debug"<< endl;

cout << "Contact C: " << C << endl;

cout << "C.getName(): " << C.getName()<< endl;

D = C;

cout << "Contact D: " << D << endl;

cout << "End." << endl;

cout << "C.getName(): " << C.getName()<< endl;

return 0;

}

**LoanList.h**

#ifndef LOANLIST\_H

#define LOANLIST\_H

#include <stdio.h>

#include "Loan.h"

const int MAX\_LN = 2;

class LoanList {

private:

Loan ln[MAX\_LN];

int size;

friend ostream &operator<<(ostream &, const LoanList &);

friend istream &operator>>(istream &, LoanList &);

public:

LoanList();

LoanList(const LoanList &);

~LoanList();

bool isFull() const;

bool isEmpty() const;

int getSize() const;

int subscriptLoan(const Loan &tempLoan) const;

//Overloaded Operators

void operator+=(const Loan&); //add loan

void operator-=(const Loan&); //delete loan

};

#endif /\* LoanList\_h \*/

**LoanList.cpp**

#include <iostream>

#include "LoanList.h"

using namespace std;

LoanList::LoanList() {

this->size = 0;

}

LoanList::LoanList(const LoanList &temp) : size(temp.size) {

for(int i = 0; i < getSize(); i++){

\*(ln + i) = \*(temp.ln + i);

}

}

LoanList::~LoanList() {}

bool LoanList::isFull() const {

return(this->size == MAX\_LN);

}

bool LoanList::isEmpty() const {

return(this->size == 0);

}

int LoanList::getSize() const {

return (this->size);

}

int LoanList::subscriptLoan(const Loan &tempLoan) const{

int subscript = -1;

for(int i = 0; i < this->getSize(); i++) {

if(ln[i].getLoanNo() == tempLoan.getLoanNo()) {

subscript = i;

i = size;

}

}

return subscript;

}

void LoanList::operator+=(const Loan &temp) {

if(!isFull()) {

cout << "Adding loan to list." << endl;

\*(ln + size) = temp;

this->size++;

}

else{

cout << "Array is full or repeated loan number." << endl;

}

}

void LoanList::operator-=(const Loan &temp){

int subscript;

if(!isEmpty()){

cout << "Removing loan from list." << endl;

subscript = subscriptLoan(temp);

for(int i = subscript; i < getSize() - 1; i++) {

\*(ln + i) = \*(ln + (i + 1));

}

size--;

}

else {

cout << "Invalid loan number" << endl;

}

}

**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() const;

//Overloaded operators

Loan &operator = (const Loan &);

friend ostream &operator<<(ostream &, const Loan &);

friend istream &operator>>(istream &, Loan &);

};

#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() const {

this->finances.displayPayment();

}

Loan & Loan::operator=(const Loan &temp) {

this->loan\_no = temp.getLoanNo();

this->finances = temp.getFinances();

this-> contact = temp.getContact();

return \*this;

}

ostream &operator << (ostream &output, const Loan &temp) {

output << "Loan Number: " << temp.getLoanNo() << endl;

temp.displayLoan();

return output;

}

istream &operator >> (istream &input, Loan &temp) {

cout << "Please enter the loan number." << endl;

input >> temp.loan\_no;

cout << "Please enter the contact info." << endl;

input >> temp.contact;

cout << "Enter the loan information." << endl;

input >> temp.finances;

return input;

}

**MortgagePayment.h**

#ifndef MortgagePayment\_h

#define MortgagePayment\_h

#include <stdio.h>

#include <iostream>

using namespace std;

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() const;

//Overloaded Operators

MortgagePayment &operator = (const MortgagePayment &);

friend ostream &operator<<(ostream &, const MortgagePayment &);

friend istream &operator>>(istream &, MortgagePayment &);

};

#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() const{

//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;

}

//Overloaded Operators

//Assign Operator

MortgagePayment & MortgagePayment::operator=(const MortgagePayment &temp) {

this->loan = temp.getLoan();

this->years = temp.getYears();

this->rate = temp.getRate();

this->term = temp.getRate();

return \*this;

}

//Stream Operators

ostream &operator << (ostream &output, const MortgagePayment &temp) {

output << "--Loan Summary--" << endl;

output << "Loan amount: $" << temp.loan << endl;

output << "Years of the loan: " << temp.years << endl;

output << "Annual interest rate: " << temp.rate << endl;

output << "Term: " << temp.term << endl;

output << "Total Monthly Payment: $" << temp.payment << endl;

return output;

}

istream &operator >> (istream &input, MortgagePayment &temp) {

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

input >> temp.loan;

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

input >> temp.years;

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

input >> temp.rate;

temp.setTerm();

temp.setPayment();

return input;

}

**Info.h**

#ifndef INFO\_H\_

#define INFO\_H\_

#include <stdio.h>

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

#include <iostream>

using namespace std;

// 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);

//ContactInfo &operator = (const ContactInfo &);

friend ostream &operator<<(ostream &, const ContactInfo &);

friend istream &operator>>(istream &, ContactInfo &);

};

#endif /\* Info\_h \*/

**Info.cpp**

#include <iostream>

#include "ContactInfo.h"

using namespace std;

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() {

cout << "\*this = " << \*this << endl;

delete [] name;

delete [] phone;

}

char\* ContactInfo::getName() const {

return name;

}

char\* ContactInfo::getPhone() const {

return phone;

}

void ContactInfo::setName(string n) const {

n = \*name;

}

void ContactInfo::setPhone(string p) const {

p = \*phone;

}

void ContactInfo::Display() const {

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

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

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

}

ContactInfo::ContactInfo( const ContactInfo &aContactInfo) {

//Allocate memory.

name = new char[strlen( aContactInfo.getName() ) + 1];

phone = new char[strlen(aContactInfo.getPhone()) + 1];

//Store values in temporary variables to convert from string to \*char.

string temp\_name = aContactInfo.getName();

string temp\_phone = aContactInfo.getPhone();

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

strcpy(name, temp\_name.c\_str());

//cout << "HELLO" << endl;

//cout << "name: " << name << endl;

strcpy(phone, temp\_phone.c\_str());

}

//Overloaded operators

//Assign Operator

/\*

ContactInfo & ContactInfo::operator=(const ContactInfo &temp) {

this->name = temp.getName();

this->phone = temp.getPhone();

//Copy Constructor

ContactInfo \*this(temp);

return \*this;

}

\*/

//Stream Operators

ostream &operator << (ostream &output, const ContactInfo &temp) {

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

cout << "Name: " << endl;

output << temp.name << endl;

cout << "Phone: " << endl;

output << temp.phone << endl;

return output;

}

istream &operator >> (istream &input, ContactInfo &temp) {

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

input >> temp.name;

cout << "Enter contact phone number." << endl;

input >> temp.phone;

return input;

}