# **Звіт про виконання лабораторної роботи № 2.7**

## «Конструктори та

## перевантаження операцій для класів з композицією –

## складніші завдання» з дисципліни «Об’єктно-орієнтоване програмування»

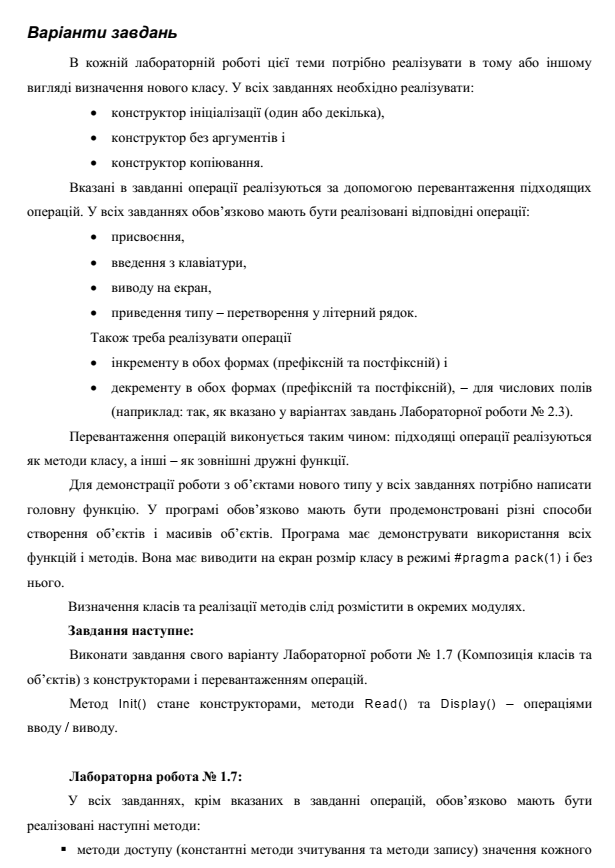
## Студента групи «**ІТ-12**» - **Степанчука Сергія**

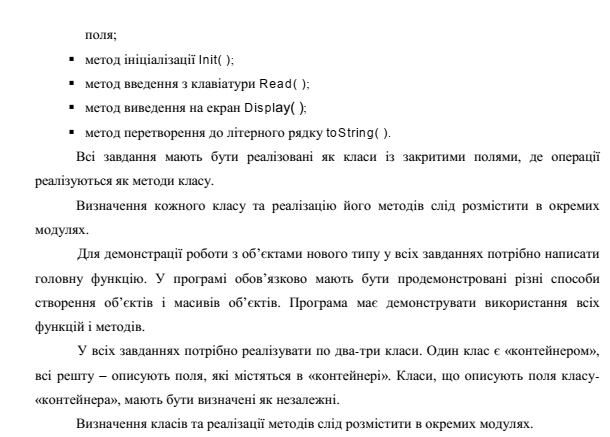
## Мета роботи

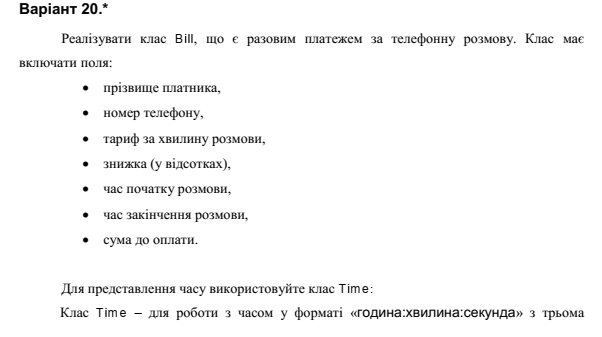
Освоїти використання конструкторів та перевантаження операцій для класів з

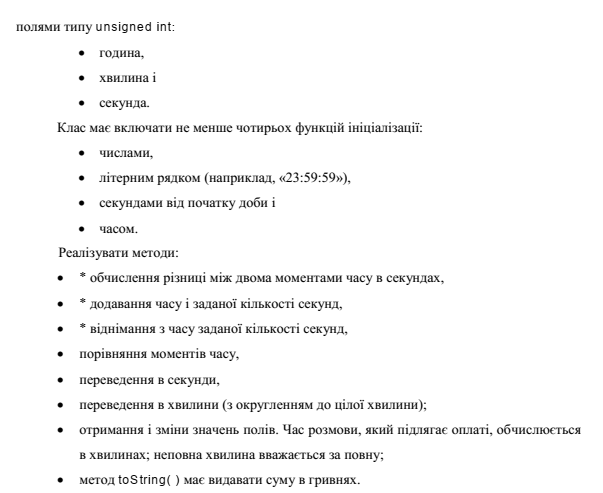
композицією.

## Умова завдання





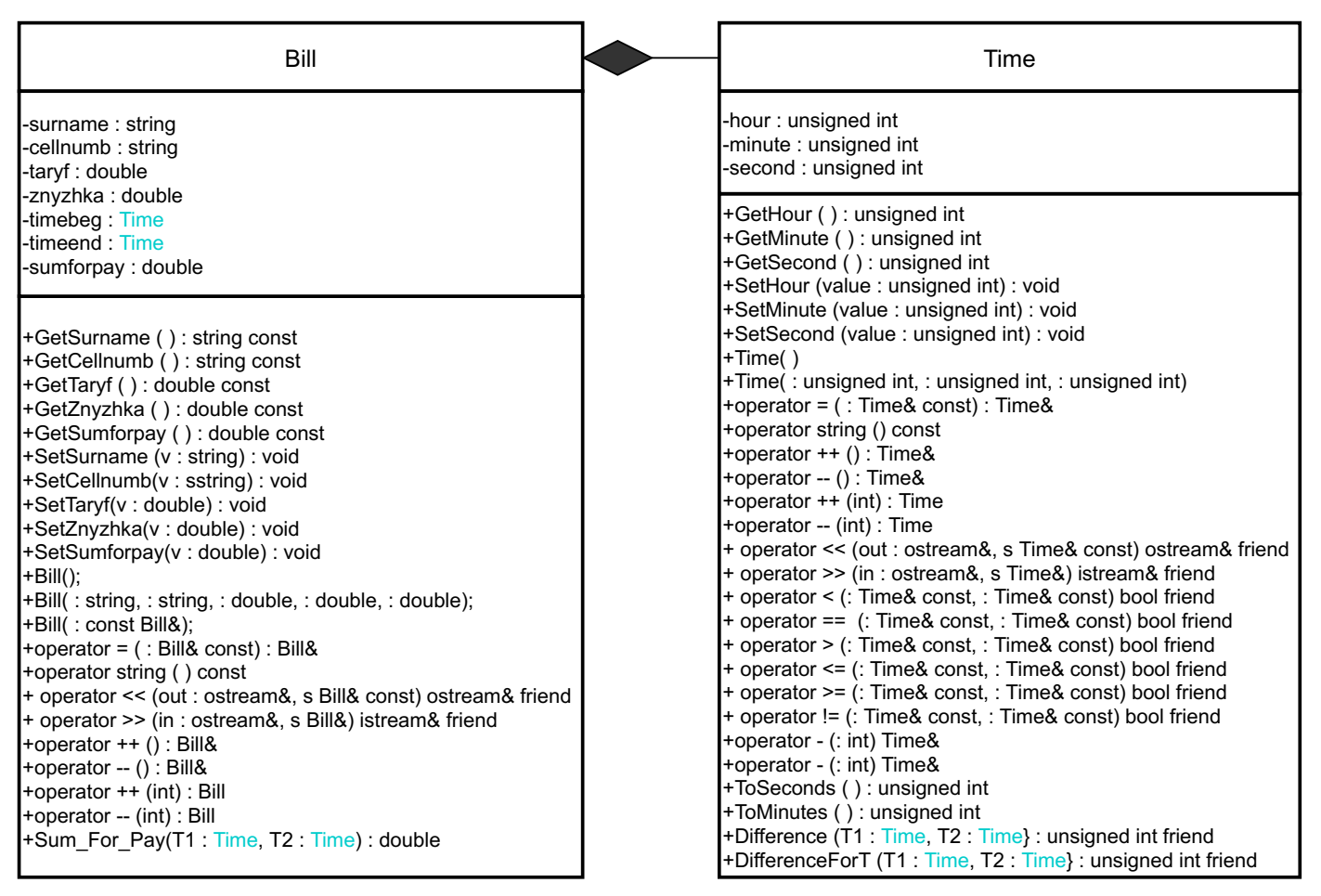




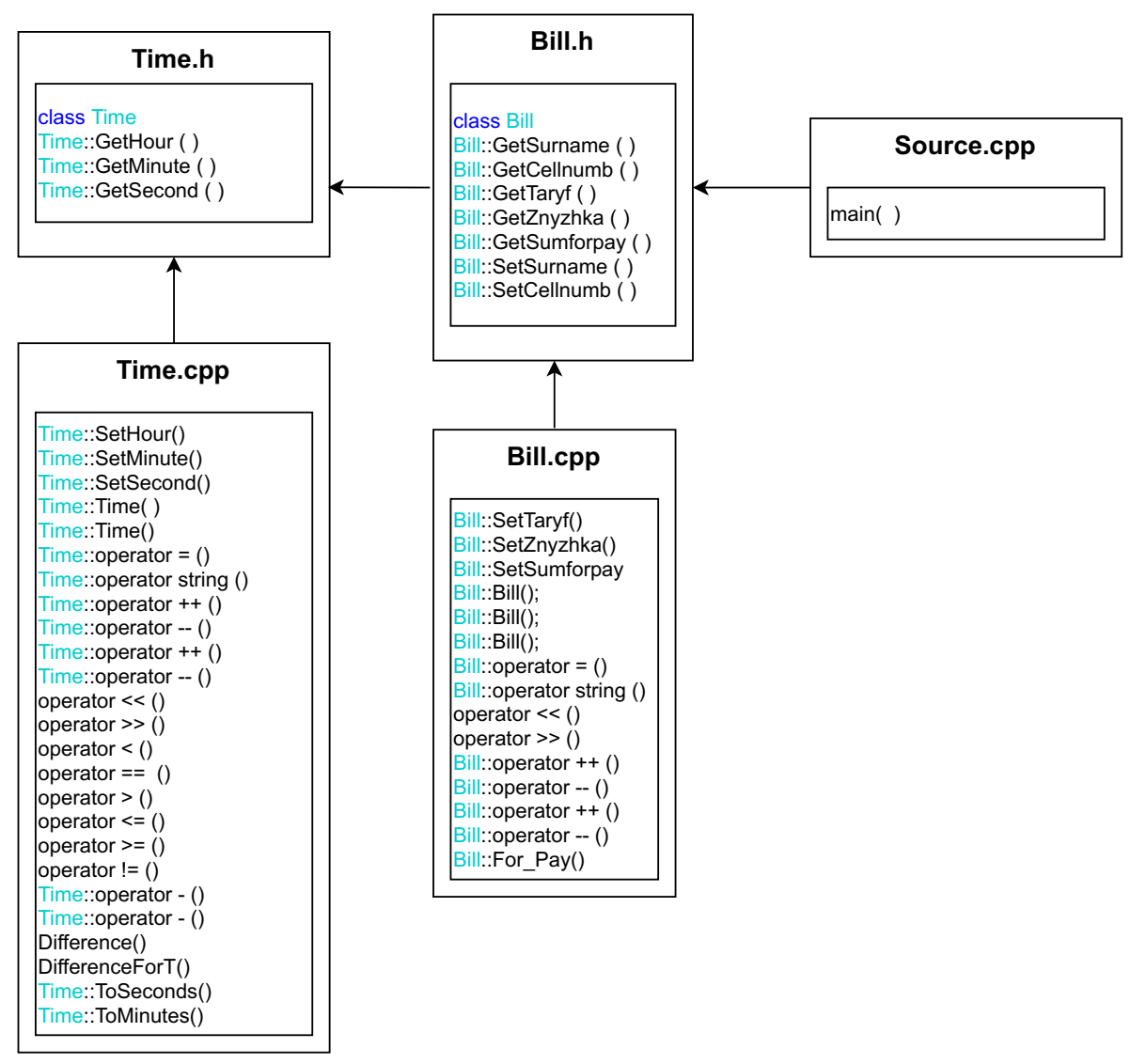
## Посилання на git-репозиторій з проектом:

<https://github.com/SergiyStepanchuk/OOP_Lab_2.7>

## UML-діаграма класів



## Структурна схема



## Текст програми

### // Bill.h

#pragma once

#pragma pack(1)//

#include <iostream>

#include <sstream>

#include <string>

#include "Time.h"

using namespace std;

class Bill

{

private:

string surname;

string cellnumb;

double taryf;

double znyzhka;

Time timebeg;

Time timeend;

double sumforpay;

public:

string GetSurname() const { return surname; };

void SetSurname(string v) { this->surname = v; };

string GetCellnumb() const { return cellnumb; };

void SetCellnumb(string v) { this->cellnumb = v; }

double GetTaryf() const { return taryf; };

void SetTaryf(double);

double GetZnyzhka() const { return znyzhka; };

void SetZnyzhka(double);

double GetSumforpay() const { return sumforpay; };

void SetSumforpay(double);

Bill();

Bill(string, string, double, double, double);

Bill(const Bill&);

Bill& operator = (const Bill&);

operator string () const;

friend ostream& operator << (ostream& out, const Bill& s);

friend istream& operator >> (istream& in, Bill& s);

Bill& operator ++ ();

Bill& operator -- ();

Bill operator ++ (int);

Bill operator -- (int);

double For\_Pay(Time T1, Time T2);

};

### // Bill.cpp

#include "Bill.h"

void Bill::SetTaryf(double v)

{

this->taryf = v;

}

void Bill::SetZnyzhka(double v)

{

if (v > 100 || v < 0)

{

v = 0;

}

this->znyzhka = v;

}

void Bill::SetSumforpay(double v)

{

this->sumforpay = v;

}

Bill::Bill()

:surname("a"), cellnumb("0"), taryf(0), znyzhka(0), sumforpay(0)

{}

Bill::Bill(string surname, string cellnumb, double taryf, double znyzhka, double sumforpay)

{

SetSurname(surname);

SetCellnumb(cellnumb);

SetTaryf(taryf);

SetZnyzhka(znyzhka);

SetSumforpay(sumforpay);

}

Bill::Bill(const Bill& v)

{

\*this = v;

}

Bill& Bill::operator = (const Bill& n)

{

surname = n.surname;

cellnumb = n.cellnumb;

taryf = n.taryf;

znyzhka = n.znyzhka;

timebeg = n.timebeg;

timeend = n.timeend;

sumforpay = n.sumforpay;

return \*this;

}

Bill::operator string () const

{

stringstream sout;

sout << endl << " Surname: " << surname << endl;

sout << " Phone number: " << cellnumb << endl;

sout << " Taryf per minute: " << taryf << "UAH" << endl;

sout << " Discount: " << znyzhka << "%" << endl;

return sout.str();

}

ostream& operator <<(ostream& out, const Bill& s)

{

out << string(s);

return out;

}

istream& operator >>(istream& in, Bill& s)

{

string surname;

cout << " Enter surname: "; in >> surname;

s.SetSurname(surname);

string cellnumb;

cout << " Enter phone number: "; in >> cellnumb;

s.SetCellnumb(cellnumb);

double taryf, znyzhka;

cout << " Enter taryf per minute: "; in >> taryf;

do {cout << " Enter discount, %: "; in >> znyzhka;

} while (znyzhka > 100 || znyzhka < 0);

s.SetTaryf(taryf);

s.SetZnyzhka(znyzhka);

return in;

}

double Bill::For\_Pay(Time T1, Time T2)

{

return DifferenceForT(T1, T2) \* taryf - (DifferenceForT(T1, T2) \* taryf) \* (znyzhka / 100);

}

Bill& Bill::operator ++()

{

++taryf;

return \*this;

}

Bill& Bill::operator --()

{

--taryf;

return \*this;

}

Bill Bill::operator ++(int)

{

Bill s(\*this);

taryf++;

return s;

}

Bill Bill::operator --(int)

{

Bill s(\*this);

taryf--;

return s;

}

### // Time.h

#pragma once

#include <iostream>

#include <sstream>

#include <string>

using namespace std;

class Time

{

private:

unsigned int hour, minute, second;

public:

unsigned int GetHour() const { return hour; };

unsigned int GetMinute() const { return minute; };

unsigned int GetSecond() const { return second; };

void SetHour(unsigned int);

void SetMinute(unsigned int);

void SetSecond(unsigned int);

Time();

Time(unsigned int, unsigned int, unsigned int);

Time& operator = (const Time&);

operator string () const;

Time& operator ++ ();

Time& operator -- ();

Time operator ++ (int);

Time operator -- (int);

friend ostream& operator << (ostream& out, const Time& p);

friend istream& operator >> (istream& in, Time& p);

//////////////////////////////////////////////////////////////////////

// Порівняння моментів часу

friend bool operator <(const Time&, const Time&);

friend bool operator == (const Time&, const Time&);

friend bool operator > (const Time&, const Time&);

friend bool operator <=(const Time&, const Time&);

friend bool operator >= (const Time&, const Time&);

friend bool operator != (const Time&, const Time&);

Time& operator - (int);//відніманняння з часу заданої кількості секунд

Time& operator + (int);//додавання до часу задану кількість секунд

friend unsigned int Difference(Time T1, Time T2);//seconds

friend unsigned int DifferenceForT(Time T1, Time T2);//minutes

unsigned int ToSeconds();// Переведення в секунди

unsigned int ToMinutes();// Переведення у хвилини (з округленням до цілої хвилини)

};

### // Time.cpp

#include "Time.h"

#include <iostream>

void Time::SetHour(unsigned int value)

{

if (value > 23 || value < 0)

value = 0;

this->hour = value;

}

void Time::SetMinute(unsigned int value)

{

if (value > 59 || value < 0)

value = 0;

this->minute = value;

}

void Time::SetSecond(unsigned int value)

{

if (value > 59 || value < 0)

value = 0;

this->second = value;

}

Time::Time()

:hour(0), minute(0), second(0)

{}

Time::Time(unsigned int y, unsigned int m, unsigned int d)

{

SetHour(y);

SetMinute(m);

SetSecond(d);

}

Time& Time::operator = (const Time& n)

{

hour = n.hour;

minute = n.minute;

second = n.second;

return \*this;

}

Time::operator string () const

{

stringstream sout;

sout << " " << hour << ":" << minute << ":" << second;

return sout.str();

}

ostream& operator <<(ostream& out, const Time& p)

{

out << string(p);

return out;

}

istream& operator >>(istream& in, Time& p)

{

unsigned int h, m, s;

do {

cout << " Enter hour: "; in >> h;

} while (h > 23 || h < 0);

do {

cout << " Enter minute: "; in >> m;

} while (m > 59 || m < 0);

do {

cout << " Enter second: "; in >> s;

} while (s > 59 || s < 0);

p.SetHour(h);

p.SetMinute(m);

p.SetSecond(s);

return in;

}

Time& Time::operator --()

{

--hour;

return \*this;

}

Time& Time::operator ++()

{

++hour;

return \*this;

}

Time Time::operator --(int)

{

Time a = \*this;

second--;

//--second;

return a;

}

Time Time::operator ++(int)

{

Time a = \*this;

second++;

//++second;

return a;

}

bool operator <(const Time& T1, const Time& T2)

{

return T1.hour < T2.hour ||

T1.hour == T2.hour && T1.minute < T2.minute ||

T1.hour == T2.hour && T1.minute == T2.minute && T1.second < T2.second;

}

bool operator == (const Time& T1, const Time& T2)

{

return T1.hour == T2.hour && T1.minute == T2.minute && T1.second == T2.second;

}

bool operator > (const Time& T1, const Time& T2)

{

return T1.hour > T2.hour ||

T1.hour == T2.hour && T1.minute > T2.minute ||

T1.hour == T2.hour && T1.minute == T2.minute && T1.second > T2.second;

}

bool operator <=(const Time& T1, const Time& T2)

{

return !(T1 > T2);

}

bool operator >= (const Time& T1, const Time& T2)

{

return !(T1 < T2);

}

bool operator != (const Time& T1, const Time& T2)

{

return !(T1 == T2);

}

Time& Time::operator - (int substr)

{

if (substr < 1)

substr = 1;

do {

if (substr < second)

{

second = second - substr;

substr = 0;

}

else {

substr = substr - second;

if (minute == 0)//1//60

{

minute = 60;//12//0//59

hour--;

}

else {

minute--;

second = 60;

}

}

} while (substr != 0);

return \*this;

}

Time& Time::operator + (int seconds)

{

short SecondsLeft;

do {

SecondsLeft = 60 - second;

if (seconds < SecondsLeft)

{

second = second + seconds;

seconds = 0;

}

else

{

second = second + SecondsLeft;

seconds = seconds - SecondsLeft;

if (minute == 60)

{

minute = 0;//

hour++;

}

else

{

second = 0;//

minute++;

}

}

} while (seconds != 0);

return \*this;

}

unsigned int Time::ToSeconds()

{

unsigned int t;

t = hour \* 3600 + minute \* 60 + second;

return t;

}

unsigned int Time::ToMinutes()

{

unsigned int t;

if (second < 1)

{

t = hour \* 60 + minute;

}

if (second > 0)

{

t = hour \* 60 + minute + 1;

}

return t;

}

unsigned int Difference(Time T1, Time T2)

{

return T2.ToSeconds() - T1.ToSeconds();

}

unsigned int DifferenceForT(Time T1, Time T2)

{

int k;

k = (T2.ToSeconds() - T1.ToSeconds()) / 60.;

return k;

}

### // Source.cpp

#include <iostream>

#include "Time.h"

#include "Bill.h"

using namespace std;

int main()

{

Bill B1;

Time T1, T2; //T1-timebeg, T2 - timeend

cin >> B1;

cout << endl;

cout << " Enter talk start time:" << endl;

cin >> T1;

cout << " Enter talk end time:" << endl;

cin >> T2;

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

cout << B1;

cout << " Talk start time: ";

cout << T1 << endl;

cout << " Talk end time: ";

cout << T2;

cout << endl;

cout << " Sum to pay: " << B1.For\_Pay(T1, T2) << "UAH" << endl;

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

cout << endl;

cout << "T1 (Talk start time): " << T1 << endl;

cout << "++T1 (hour): " << ++T1 << endl;

cout << "--T1 (hour): " << --T1 << endl;

cout << "T1++ (second): " << T1++ << endl;

cout << "T1-- (second): " << T1-- << endl << endl;

cout << "B1 (taryf): " << B1 << endl;

cout << "++B1 (taryf): " << ++B1 << endl;

cout << "--B1 (taryf): " << --B1 << endl;

cout << "B1++ (taryf): " << B1++ << endl;

cout << "B1-- (taryf): " << B1-- << endl << endl;

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

cout << " MANIPULATIONS WITH TIME " << endl << endl;

cout << "Talk start time is earlier than talk end time: ";

if (T1 < T2)

cout << " TRUE" << endl << endl;

else

cout << " FALSE" << endl << endl;

cout << "Talk end time minus talk start time (in seconds): ";

unsigned int q = Difference(T1, T2);

cout << q << endl;

cout << "Talk end time minus talk start time (in minutes): ";

unsigned int k = DifferenceForT(T1, T2);

cout << k << endl << endl;

cout << "Talk start time in seconds: ";

unsigned int s = T1.ToSeconds();

cout << s << endl;

cout << "Talk start time in minutes: ";

unsigned int m = T1.ToMinutes();

cout << m << endl << endl;

cout << "Talk end time in seconds: ";

unsigned int o = T2.ToSeconds();

cout << o << endl;

cout << "Talk end time in minutes: ";

unsigned int i = T2.ToMinutes();

cout << i << endl << endl;

cout << "Subtracting from time a specified number of seconds (talk start time): " << endl;

unsigned int substr;

do { cout << "Enter how much seconds to substract: "; cin >> substr;

} while (substr < 1);

Time p = T1 - substr;

cout << p << endl;

unsigned int seconds;

cout << "Adding to time a specified number of seconds (talk start time which was substracted): " << endl;

do { cout << "Enter how much seconds to add: "; cin >> seconds;

} while (seconds < 1);

Time pp = T1 + seconds;

cout << pp << endl;

return 0;

}

## UnitTest

### Код

#include "pch.h"

#include "CppUnitTest.h"

#include "../Program/Bill.h"

#include "../Program/Bill.cpp"

#include "../Program/Time.h"

#include "../Program/Time.cpp"

using namespace Microsoft::VisualStudio::CppUnitTestFramework;

namespace UnitTest

{

TEST\_CLASS(UnitTest17)

{

public:

TEST\_METHOD(TestMethod1)

{

Bill B1;

B1.SetSurname("Stepanchuk");

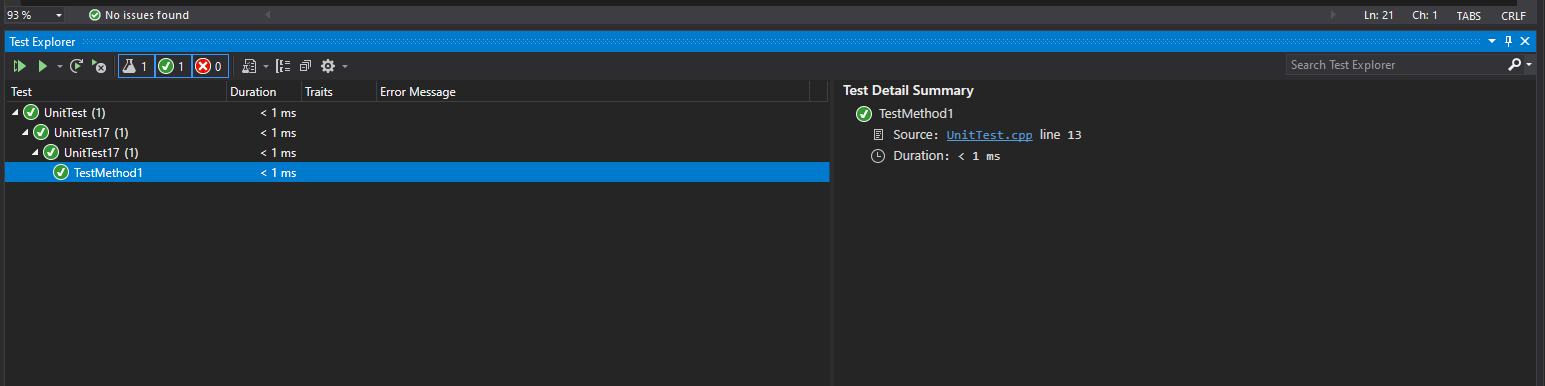
Assert::AreEqual("Stepanchuk", B1.GetSurname().c\_str());

}

};

}

### Результат



## Висновок

Освоїв використання конструкторів та перевантаження операцій для класів з композицією.