МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ государственное БЮДЖЕТНОЕ

образовательное учреждение

высшего образования

«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кафедра защиты информации

**

**ОТЧЁТ**

**по лабораторной работе № 2**

**«Построение элементарных классов»**

**по дисциплине: «***Программирование***»**

Выполнил:Проверил:

Студент гр. «АБ-121», «АВТФ» *доцент кафедры ЗИ*

*Новиков Втюрин Александр Романович Архипова А. Б.*

«21» декабря 2022г«\_\_\_» \_\_\_\_\_\_ 2022 г.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись) (подпись)

Новосибирск 2022

**Задание к работе:**

**Задание 1.**

**Вариант 2.**

Создать функциональный класс “Время”. Класс должен иметь 3 конструктора: по умолчанию, параметр – число,  параметр – структура. В качестве переменных: секунды, минуты, часы. Методы: конвертация часов и минут в секунды, сложение, вычитание, сравнение (>, <, =). Реализовать перегрузки, как для работы методов, так и для отдельной работы в коде.

**Задание 2**

Построение арифметического выражения.

**С++**

**Main.cpp**

#include "Algebra.h"

#include "Comparison.h"

#include <fstream>

using namespace std;

int main()

{

TimeData Mos(12, 0, 14, 10);

Time NewCity;

Time London(12, 60, 14, 52);

Time Moscow(Mos);

while (true)

{

system("cls");

int n;

cout << "What do you want?\n";

cout << "1) Create city with time.\n";

cout << "2) Output city\n";

cout << "3) Addup times\n";

cout << "4) Subtract times\n";

cout << "5) Compare times\n";

cout << "6) Exit\n";

cout << "Your Choice: ";

cin >> n;

if (cin.fail()) {

cin.clear();

cout << "Inncorrect input!!!\n\n";

exit(1);

}

system("cls");

switch (n)

{

case 1:

cin >> NewCity;

break;

case 2:

cout << " 1) File\n 2) Console\n Your Choice: ";

cin >> n;

if (cin.fail()) {

cin.clear();

cout << "Inncorrect input!!!\n\n";

exit(1);

}

if (n == 1) {

ofstream fout;

fout.open("Out.txt");

if (fout.is\_open())

{

fout << "In city N now " << NewCity.hour << " hours " << NewCity.min << " minutes " << NewCity.sec << " seconds " << NewCity.msec << " milliseconds\n";

}

fout.close();

}

else if (n == 2) {

cout << NewCity << endl;

system("pause");

}

break;

case 3:

cout << London << endl << Moscow << endl;

NewCity = NewCity.CallingAdd(London, Moscow);

cout << "Result of addition: " << NewCity << endl;

system("pause");

break;

case 4:

cout << London << endl << Moscow << endl;

NewCity = NewCity.CallingSub(London, Moscow);

cout << "Result of subtract: " << NewCity << endl;

system("pause");

break;

case 5:

cout << "Result comparison: " << NewCity.CallingEquals(London, Moscow) << endl << endl;

cout << "Result comparison: " << NewCity.CallingGreater(London, Moscow) << endl << endl;

cout << "Result comparison: " << NewCity.CallingLess(London, Moscow) << endl << endl;

system("pause");

break;

case 6:

exit(0);

default:

cout << "Inncorrect input!!!\n\n";

main();

break;

}

}

}

**Algebra.h**

#pragma once

#include <iostream>

#include <string>

#include "Comparison.h"

#include "Time.h"

using namespace std;

class Arithmetic

{

public:

int add(const int time1, const int time2);

int addHour(const int time1, const int time2);

int sub(const int time1, const int time2);

int subHour(const int time1, const int time2);

private:

int unitTrans;

};

**Comparison.h**

#pragma once

#include <iostream>

#include <string>

#include "Algebra.h"

#include "Time.h"

using namespace std;

class Comparison

{

public:

bool equals(int city1, int city2);

bool greater(int city1, int city2);

bool less(int city1, int city2);

};

**Time.h**

#pragma once

#include <iostream>

#include <string>

#include "Algebra.h"

#include "Comparison.h"

using namespace std;

struct TimeData

{

public:

int hour, min, sec, msec;

TimeData(const int& h, const int& m, const int& s, const int& ms)

{

hour = h;

min = m;

sec = s;

msec = ms;

}

};

class Time

{

public:

Time()

{

hour = 0;

min = 0;

sec = 0;

msec = 0;

}

Time(const int& h, const int& m, const int& s, const int& ms)

{

hour = h;

min = m;

sec = s;

msec = ms;

}

Time(const struct TimeData& data)

{

hour = data.hour;

min = data.min;

sec = data.sec;

msec = data.msec;

}

friend ostream& operator <<(ostream& os, const Time& city) {

os << "In city N now " << city.hour << " hours " << city.min << " minutes " << city.sec << " seconds " << city.msec << " milliseconds\n";

return os;

}

friend istream& operator>> (istream& is, Time& dt)

{

is >> dt.hour >> dt.min >> dt.sec >> dt.msec;

return is;

}

int hour, min, sec, msec;

Time CallingAdd(Time city1, Time city2);

Time CallingSub(Time city1, Time city2);

string CallingEquals(Time day1, Time day2);

string CallingGreater(Time day1, Time day2);

string CallingLess(Time day1, Time day2);

};

Algebra.cpp

#include "Algebra.h"

#include "Time.h"

using namespace std;

int Arithmetic::add(const int time1, const int time2)

{

int time = time1 + time2;

if (unitTrans > 0) {

time += unitTrans;

unitTrans = 0;

}

if (time / 60 == 0) return time;

else {

unitTrans = time / 60;

return time % 60;

}

}

int Arithmetic::addHour(const int time1, const int time2)

{

int time = time1 + time2;

if (unitTrans > 0) {

time += unitTrans;

unitTrans = 0;

}

if (time / 24 == 0) return time;

else {

unitTrans = time / 24;

return time % 24;

}

}

int Arithmetic::sub(const int time1, const int time2)

{

int time = time1 - time2;

if (unitTrans > 0) {

time += 60;

time -= unitTrans;

unitTrans = 0;

}

if (time / 60 == 0) return time;

else {

unitTrans = time / 60;

return time % 60;

}

}

int Arithmetic::subHour(const int time1, const int time2)

{

int time = time1 - time2;

if (time / 24 == 0) return time;

else {

unitTrans = time / 24;

return time % 24;

}

}

Time Time::CallingAdd(Time city1, Time city2)

{

Time siteN;

Arithmetic arifm;

siteN.msec = arifm.add(city1.msec, city2.msec);

siteN.sec = arifm.add(city1.sec, city2.sec);

siteN.min = arifm.add(city1.min, city2.min);

siteN.hour = arifm.addHour(city1.hour, city2.hour);

return siteN;

}

Time Time::CallingSub(Time city1, Time city2)

{

Time siteN;

Arithmetic arifm;

siteN.msec = arifm.sub(city1.msec, city2.msec);

siteN.sec = arifm.sub(city1.sec, city2.sec);

siteN.min = arifm.sub(city1.min, city2.min);

siteN.hour = arifm.subHour(city1.hour, city2.hour);

return siteN;

}

**Comparison.cpp**

#include "Comparison.h"

#include "Time.h"

using namespace std;

bool Comparison::equals(int city1, int city2)

{

return city1 == city2;

}

bool Comparison::greater(int city1, int city2)

{

return city1 > city2;

}

bool Comparison::less(int city1, int city2)

{

return city1 < city2;

}

string Time::CallingEquals(Time day1, Time day2)

{

Comparison comparison;

string res = "";

if (comparison.equals(day1.hour, day2.hour) == true) {

res += "hours equal ";

}

else res += "hours not equal ";

if (comparison.equals(day1.min, day2.min) == true) {

res += " minutes equal ";

}

else res += " minutes not equal ";

if (comparison.equals(day1.sec, day2.sec) == true) {

res += " seconds equal ";

}

else res += " seconds not equal ";

if (comparison.equals(day1.msec, day2.msec) == true) {

res += " milliseconds equal ";

}

else res += " milliseconds not equal ";

return res;

}

string Time::CallingGreater(Time day1, Time day2)

{

Comparison comparison;

string res = "";

if (comparison.greater(day1.hour, day2.hour) == true) {

res += "hours greater ";

}

else res += "hours not greater ";

if (comparison.greater(day1.min, day2.min) == true) {

res += " minutes greater ";

}

else res += " minutes not greater ";

if (comparison.greater(day1.sec, day2.sec) == true) {

res += " seconds greater ";

}

else res += " seconds not greater ";

if (comparison.greater(day1.msec, day2.msec) == true) {

res += " milliseconds greater ";

}

else res += " milliseconds not greater ";

return res;

}

string Time::CallingLess(Time day1, Time day2)

{

Comparison comparison;

string res = "";

if (comparison.less(day1.hour, day2.hour) == true) {

res += "hours less ";

}

else res += "hours not less ";

if (comparison.less(day1.min, day2.min) == true) {

res += " minutes less ";

}

else res += " minutes less ";

if (comparison.less(day1.sec, day2.sec) == true) {

res += " seconds less ";

}

else res += " seconds not less ";

if (comparison.less(day1.msec, day2.msec) == true) {

res += " milliseconds less ";

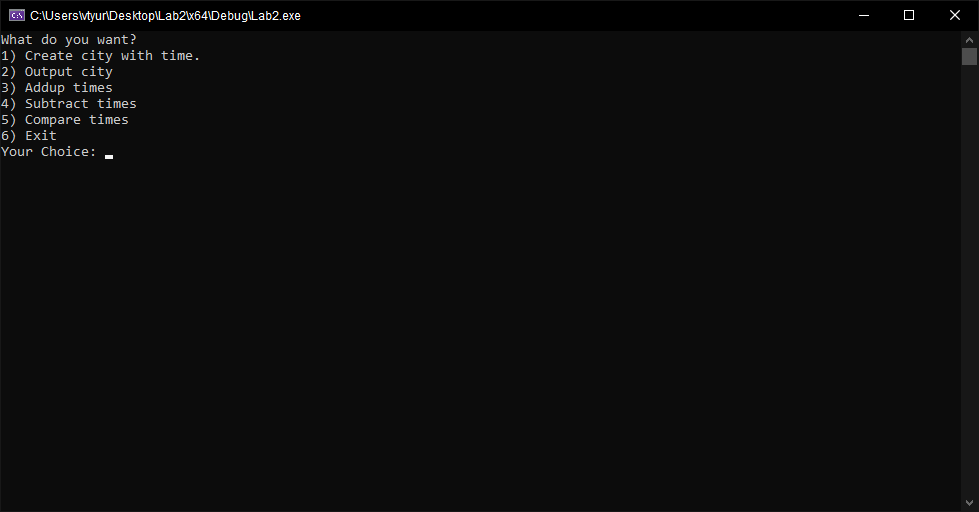
}

else res += " milliseconds not less ";

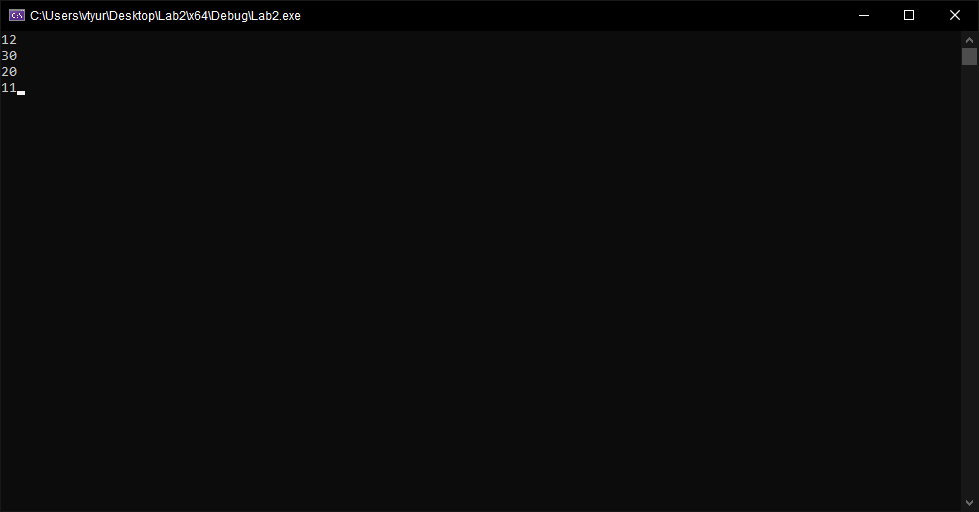
return res;

}

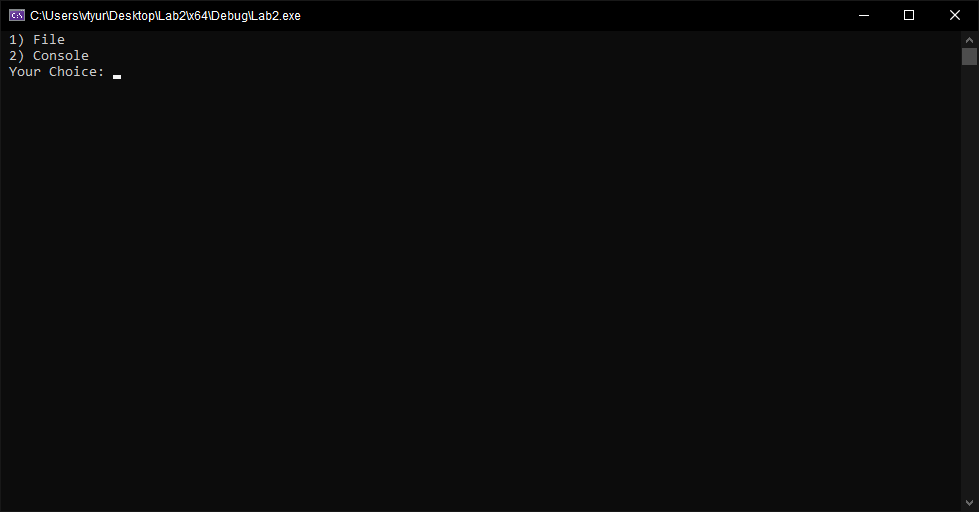
Главное меню



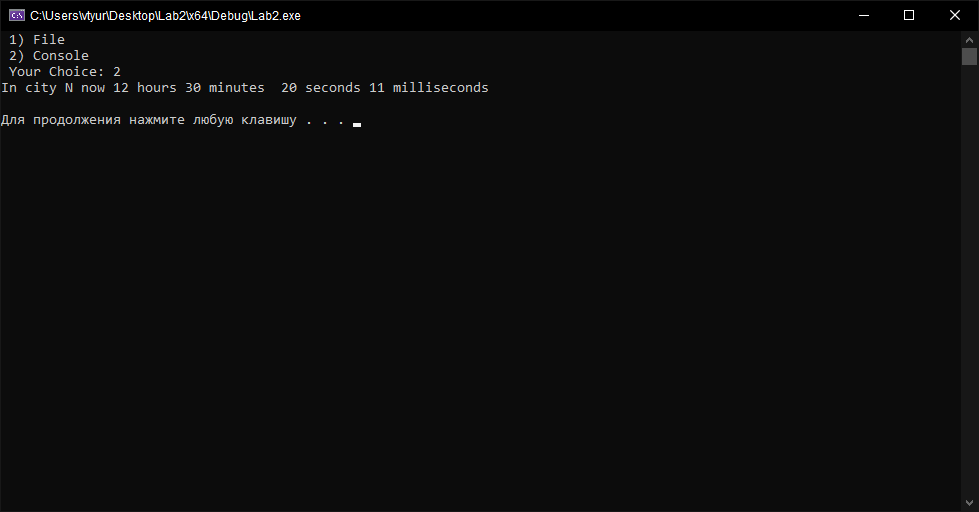
Create city with time



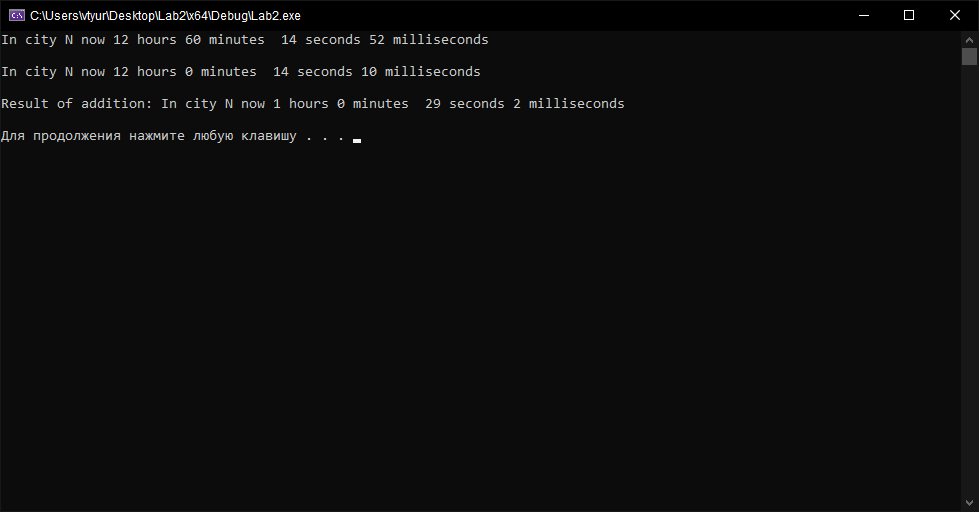
Output city



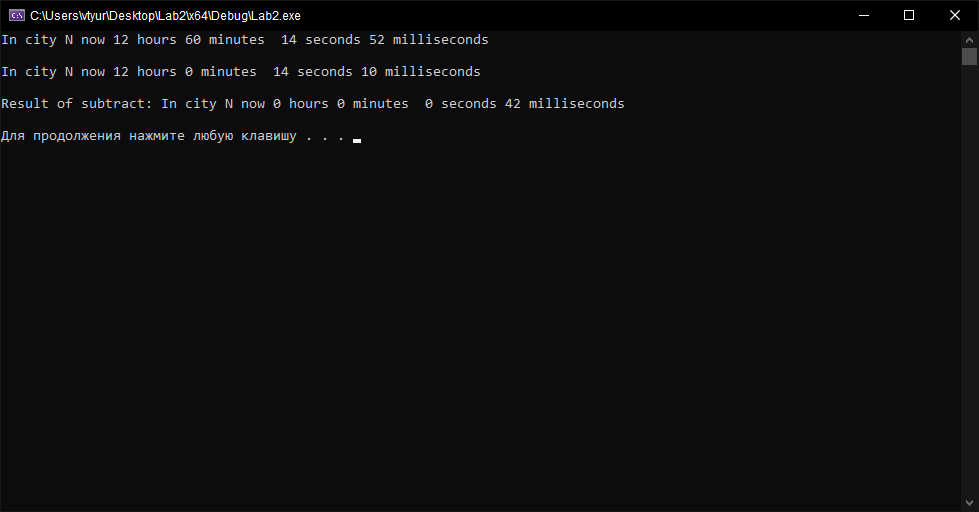
Output city, Console



Addup times



Subtract times



Compare times

