Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования

«Санкт-Петербургский государственный морской технический университет»

Кафедра Систем Автоматического Управления и Бортовой Вычислительной Техники

Отчет по практике №1

Выполнила: Комиссарова М.В

Проверил: Сакович С.Ю

Санкт-Петербург 2019г.

Оглавление

[Практика. 3](#_Toc26553078)

[№1 3](#_Toc26553079)

[№2 4](#_Toc26553080)

[№3 6](#_Toc26553081)

[№4 7](#_Toc26553082)

[№5 9](#_Toc26553083)

[№6 10](#_Toc26553084)

[№7 Виртуальная функция. 13](#_Toc26553085)

[№8 14](#_Toc26553086)

[№9 17](#_Toc26553087)

[Домашние задания 19](#_Toc26553088)

[Домашнее задание №1 19](#_Toc26553089)

[Домашнее задание №2 22](#_Toc26553090)

[Домашнее задание №3 28](#_Toc26553091)

# Практика.

## №1

Составить описание иерархии классов решающую задачу табулирования функции на заданном интервале с заданным шагом.

Базовый класс: данные (границы интервала, шаг, переменная табулирования, значение функции), методы (ввод данных, табулирование, вывод и решение задачи, вычисление функции).

#include "pch.h"

#include <iostream>

using namespace std;

class Tab {

private:

double xs, xk, dx, i, y;

public:

void input();

void calc();

void output();

void task();

virtual double func(double);

};

class Solve : public Tab

{

double func(double x)

{

return x \* x + 1;

}

};

void Tab::input() {

cout << "Input data \n";

cout << "xs="; cin >> xs;

cout << "xk="; cin >> xk;

cout << "dx="; cin >> dx;

}

void Tab::output() {

cout << "x=" << i << '\t' << "y=" << y << endl;

}

void Tab::calc() {

for (i = xs; i <= xk; i += dx)

{

y = func(i); output();

}

}

void Tab::task() {

input();

cout << "\n Output data \n";

calc();

}

double Tab::func(double x)

{

return x;

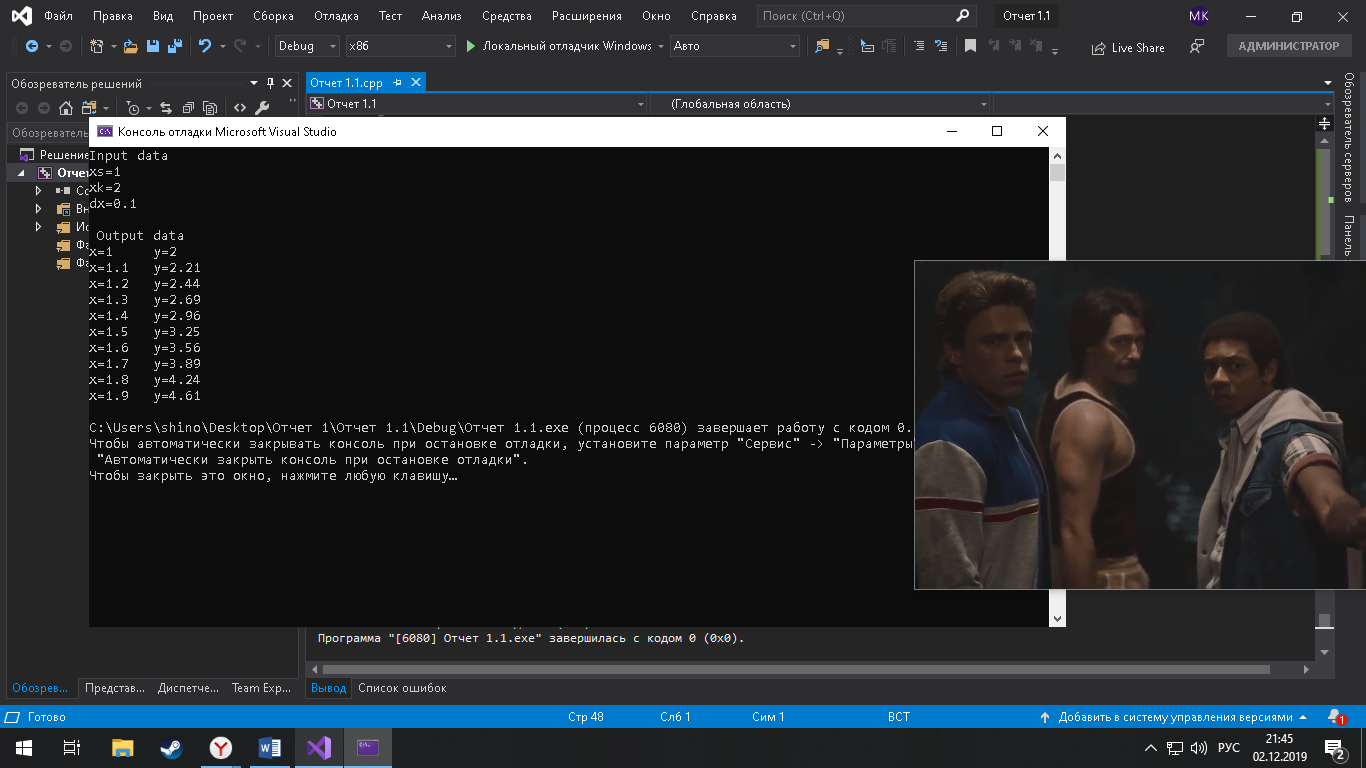
}

void main() {

Solve A;

A.task();

}



## №2

Составить описание класса, который решает задачу сложения двух чисел. В качестве данных класса используются слагаемые, в качестве методов: ввод первого слагаемого, ввод второго слагаемого, вычисление суммы, вывод первого слагаемого, вывод второго слагаемого.

#include "pch.h"

#include <iostream>

using namespace std;

class Sum {

private:

double x, y;

public:

void input1();

void input2();

double output1() {

return x;

}

double output2() {

return y;

}

double summa() {

return x + y;

}

};

void Sum::input1() {

cout << "Input data in";

cout << "x= ";

cin >> x;

}

void Sum::input2() {

cout << "Input data in";

cout << "y= ";

cin >> y;

}

void main()

{

Sum A;

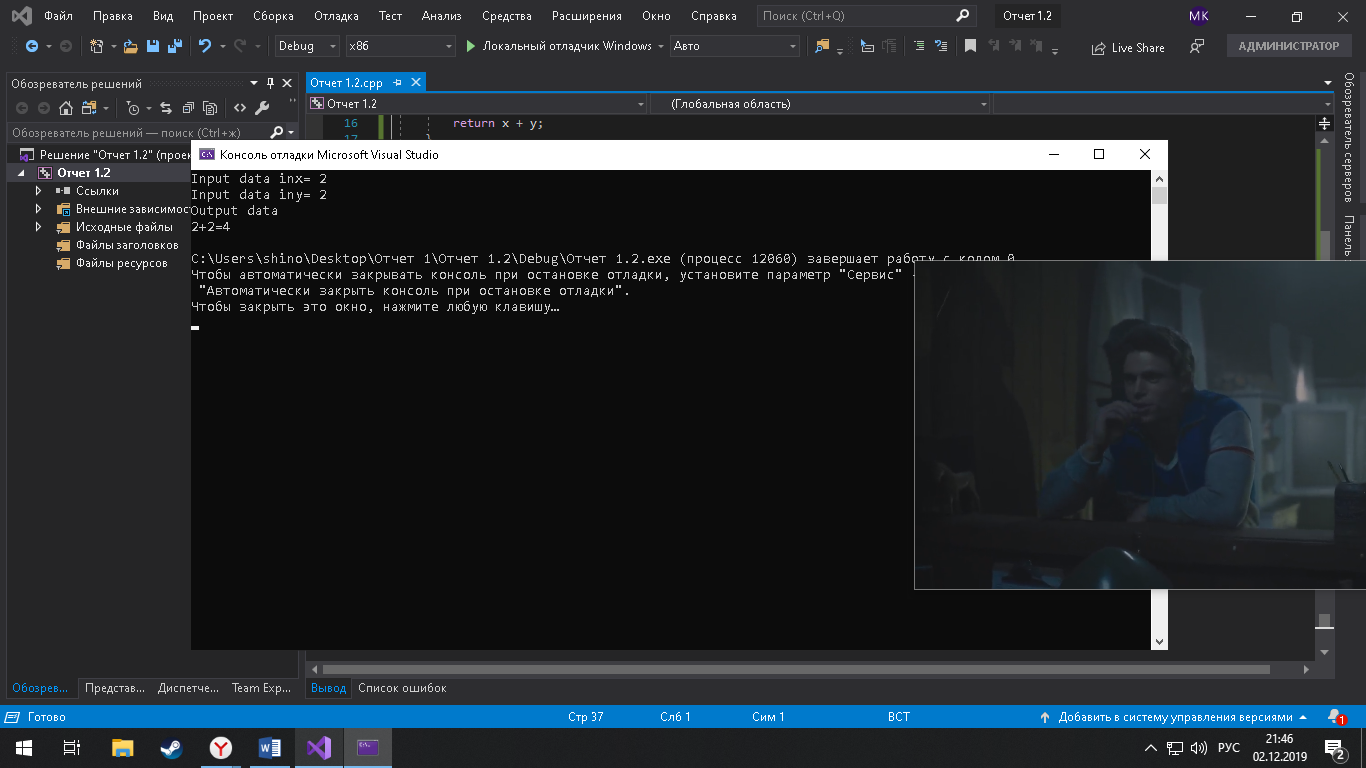
A.input1();

A.input2();

cout << "Output data" << endl;

cout << A.output1() << "+" << A.output2() << "=" << A.summa() << endl;

}



## №3

Составить иерархию классов, решающих задачу выделения памяти, возвращению выделенной памяти. Класс должен включать методы ввода и вывода массива.

#include "pch.h"

#include <iostream>

using namespace std;

class mass {

int n, \*x;

public:

mass();

mass(int k);

~mass();

void inmass();

void outmass();

};

mass::mass(int k):n(k) {

x = new int[n];

}

mass::~mass() {

delete[]x;

}

void mass::inmass() {

cout << "Input" << endl;

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

cout << "x[" << i << "]";

cin >> x[i];

cout << endl;

}

}

void mass::outmass() {

cout << "Output" << endl;

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

cout << "x[" << i << "]= " << x[i] << endl;

}

}

void main()

{

int n;

cin >> n;

mass \*M;

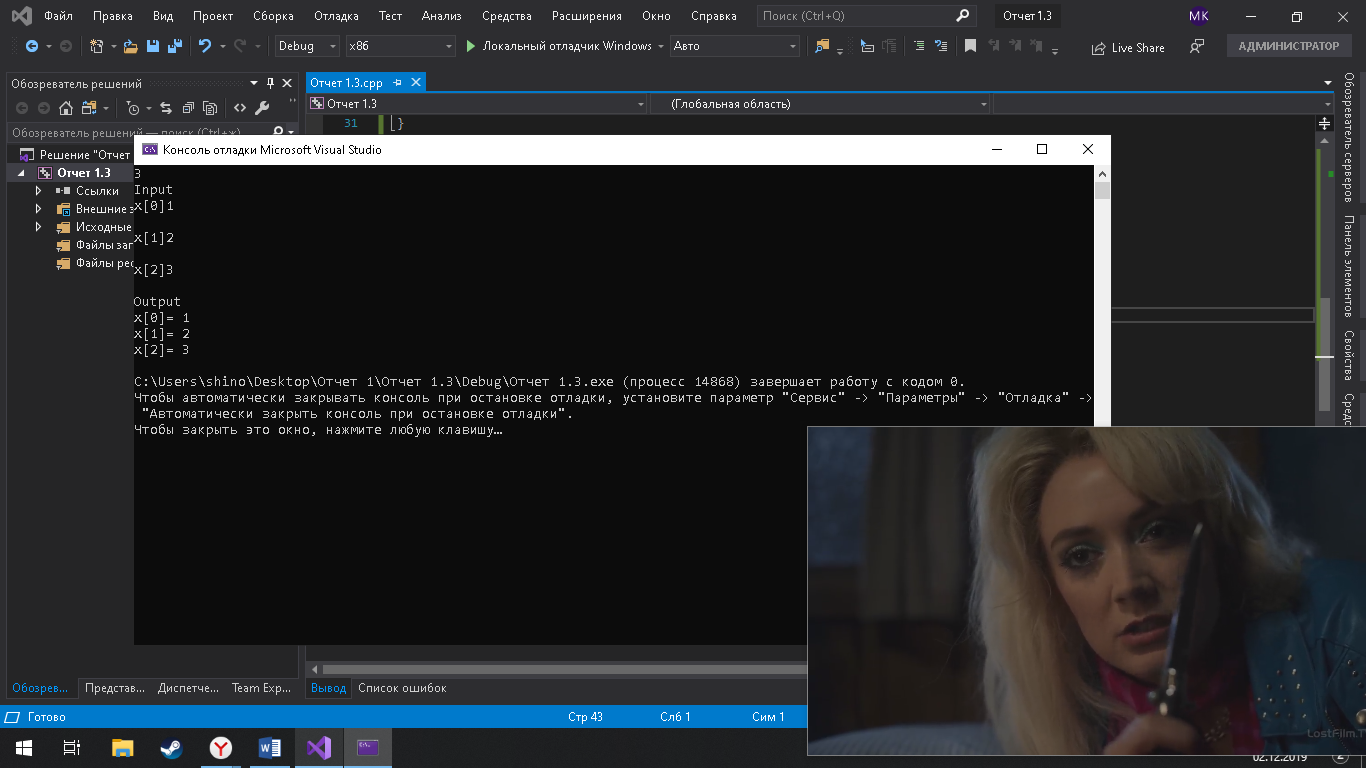
M = new mass(n);

M->inmass();

M->outmass();

delete M;

}



№4

Составить описание класса, решающую задачу формирование вектора из максимальных значений в строках матрицы целых чисел, значения которых задаются случайным образом от 1 до 100; Класс должен содержать указание на матрицу, указание на вектор и параметры строки/столбца. Методы формирования матрицы случайными числами, вывод матрицы, определение максимального элемента, формирование вектора, доступ к вектору.

#include "pch.h"

#include <iostream>

using namespace std;

class mtv {

int n, m, \*\*x, \*v;

public:

mtv(int m, int n);

~mtv();

void inmass();

void outmass();

void vector();

int max(int \*);

int \*getvector();

};

mtv::mtv(int m = 1, int n = 1) : n(n), m(m) {

x = new int \*[m];

for (int i = 0; i < m; i++)

x[i] = new int[n];

v = new int[m];

}

mtv::~mtv(){

for (int i = 0; i < m; i++)

delete[]x[i];

delete[]x;

delete[]v;

}

void mtv::inmass() {

for (int i = 0; i < m; i++)

for (int j = 0; j < n; j++)

x[i][j] = rand() % 100+1;

}

void mtv::outmass() {

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

for (int j = 0; j < n; j++)

cout << "x[" << i << "]" << "[" << j << "]= " << x[i][j] << "\t";

cout << endl;

}

}

int mtv::max(int \*S) {

int maximum = S[0];

for (int j = 0; j < n; j++)

maximum = (S[j] > maximum) ? S[j] : maximum;

return maximum;

}

void mtv::vector() {

for (int i = 0; i < m; i++)

v[i] = max(x[i]);

}

int \*mtv::getvector() {

return v;

}

int main()

{

int m, n, \*v;

cout << "m" << endl;

cin >> m;

cout << "n" << endl;

cin >> n;

mtv \*M;

M = new mtv(m, n);

M->inmass();

M->vector();

M->outmass();

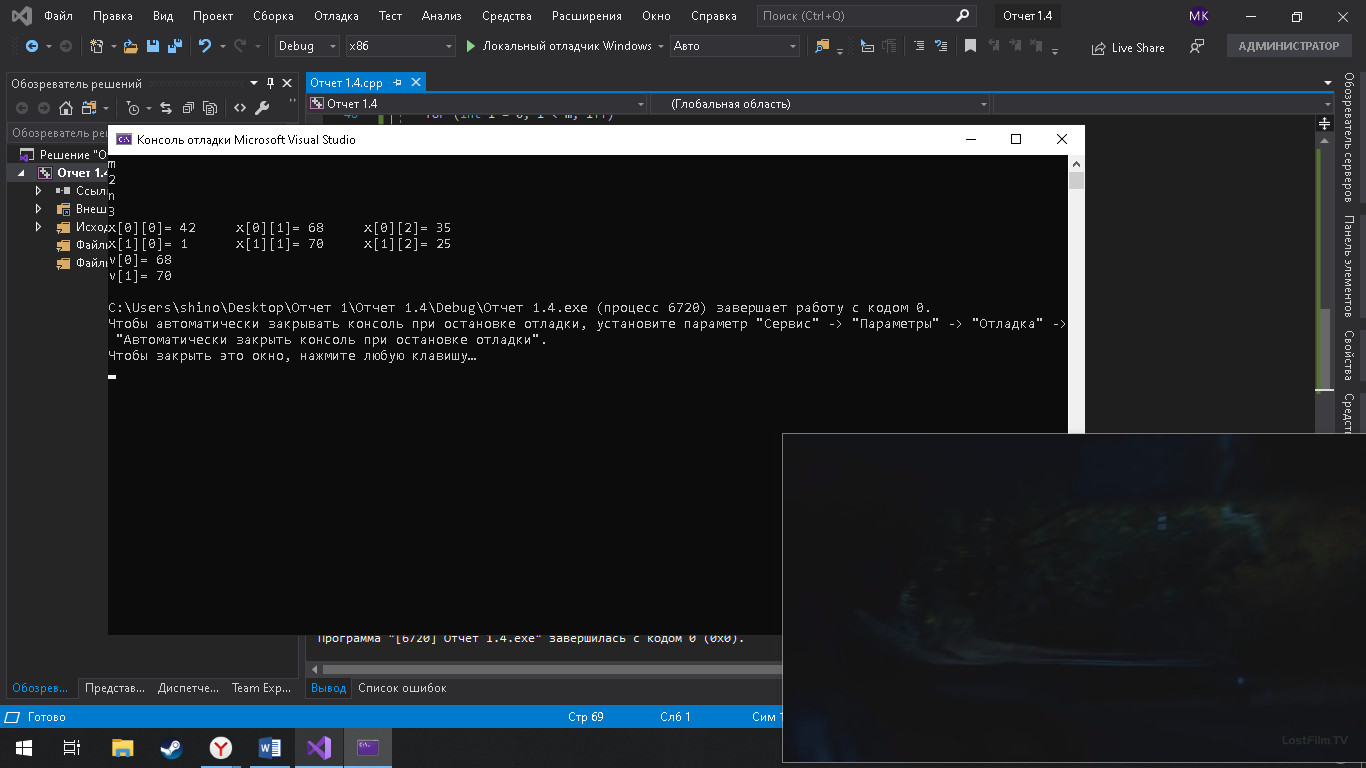
v = M->getvector();

for (int i = 0; i < m; i++)

cout << "v[" << i << "]= " << v[i] << endl;

delete M;

}



## №5

Составить иерархию классов, решающих задачу вычисления суммы двух чисел. Базовый класс должен содержать: данные (первое значение), методы(ввод, доступ к значению); производный класс: методы(ввод значения, вычисление суммы), данные(второе значение). Конструктор и деструктор по умолчанию.

#include "pch.h"

#include <iostream>

using namespace std;

class first {

private:

int a;

public:

void ina() { cout << "Input number a "; cin >> a; }

int outa() { return a; }

};

class second :public first {

private: int b;

public:

void inb() { cout << "Input number b "; cin >> b; }

int sum() { return outa() + b; }

int outb() { return b; }

};

void main()

{

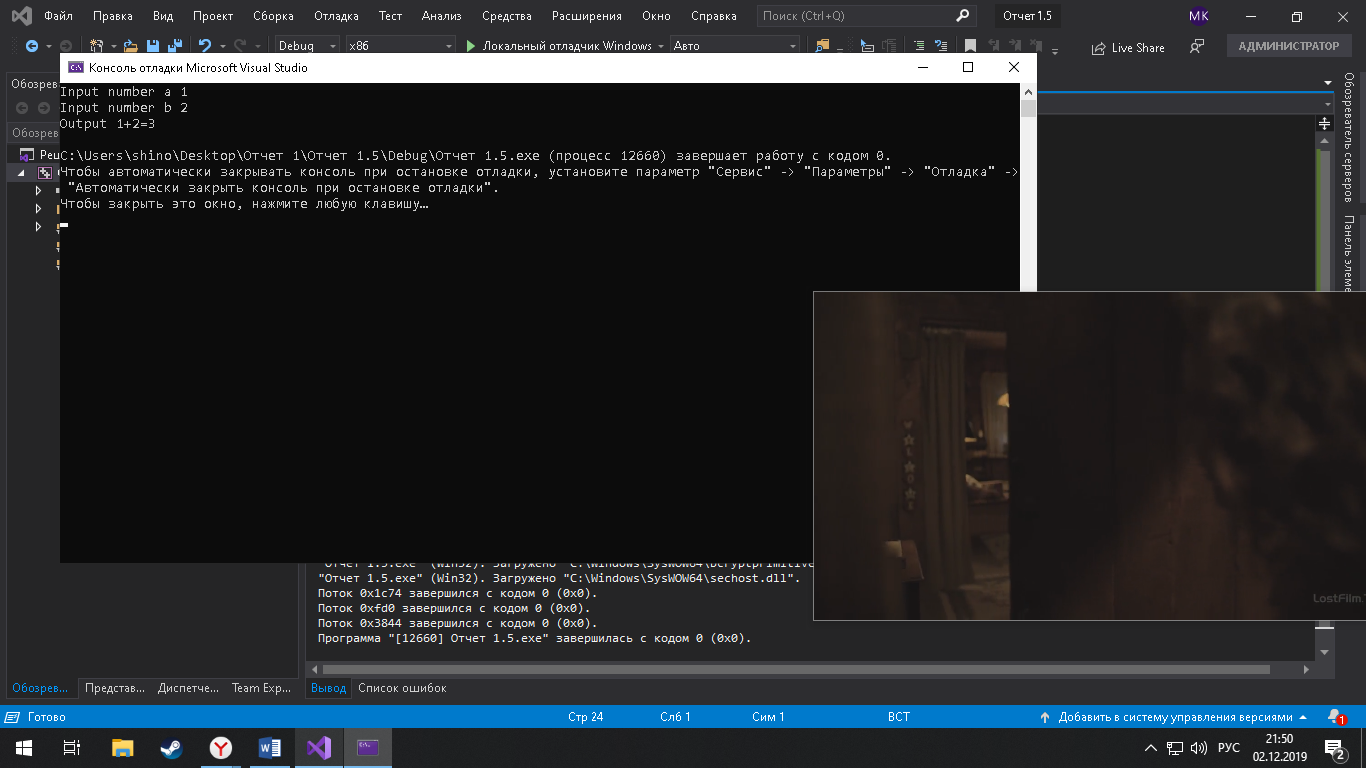
second B;

B.ina();

B.inb();

cout << "Output " << B.outa() << "+" << B.outb() << "=" << B.sum() << endl;

}



## №6

Составить программу сортировки по убыванию массива целых чисел. Базовый класс содержит: указатель на массив, длину массива, конструктор, деструктор, доступ к массиву, ввод и вывод; производный: метод сортировки, конструктор.

#include "pch.h"

#include <iostream>

using namespace std;

class mas {

private: int \*n, len;

public:

mas(int);

~mas();

int \*ret\_n(){ return n; };

int ret\_len(){ return len; };

void in\_n();

void out\_n();

};

class sort :public mas {

public:

sort(int k = 1):mas(k) {}

void sort\_n();

};

mas::mas(int k = 1) :len(k) {

n = new int[len];

}

void mas::in\_n() {

cout << "Input len" << endl; cin >> len;

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

cout << "n[" << i << "]= ";

cin >> n[i];

cout << endl;

}

}

mas::~mas() {

delete[]n;

}

void mas::out\_n() {

cout << "Output massive" << endl;

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

cout << "n[" << i << "]= " << n[i] << endl;

}

}

void sort::sort\_n() {

int temp, len, \*n,a;

n = ret\_n();

len = ret\_len();

for (int i = 0; i < len - 1; i++)

for (int j = i + 1; j < len; j++)

if (n[i] > n[j])

{

a = n[i];

n[i] = n[j];

n[j] = a;

};

}

int main()

{

int n, len;

cout << "Input number of elements"; cin >> len;

sort \*O;

O = new sort(len);

O->in\_n();

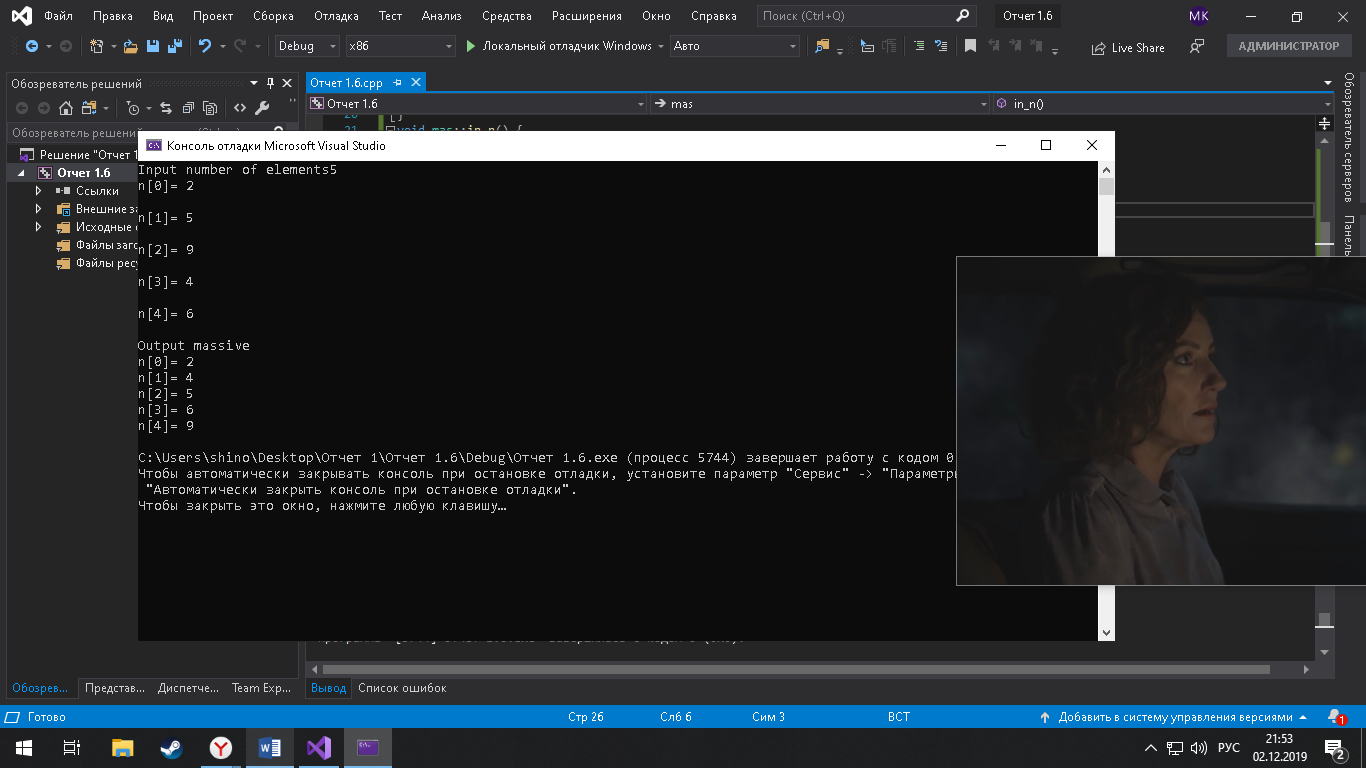
O->sort\_n();

O->out\_n();

delete O;

return 0;

}



## №7 Виртуальная функция.

#include "pch.h"

#include <iostream>

using namespace std;

class Base {

public:

static int a;

void outF() { cout << "Base"; }

virtual void OutS() { cout << "\n Virtual Base"; }

};

int Base::a = 5;

class Der : public Base

{

public:

static int a;

void outF() { cout << "\n Der"; }

};

int Der::a = 10;

void main() {

Base \*A = new Base;

A->outF();

A->OutS();

cout << "\n Test";

delete A;

A = new Der;

A->outF();

A->OutS();

cout << "\n Test N2";

delete A;

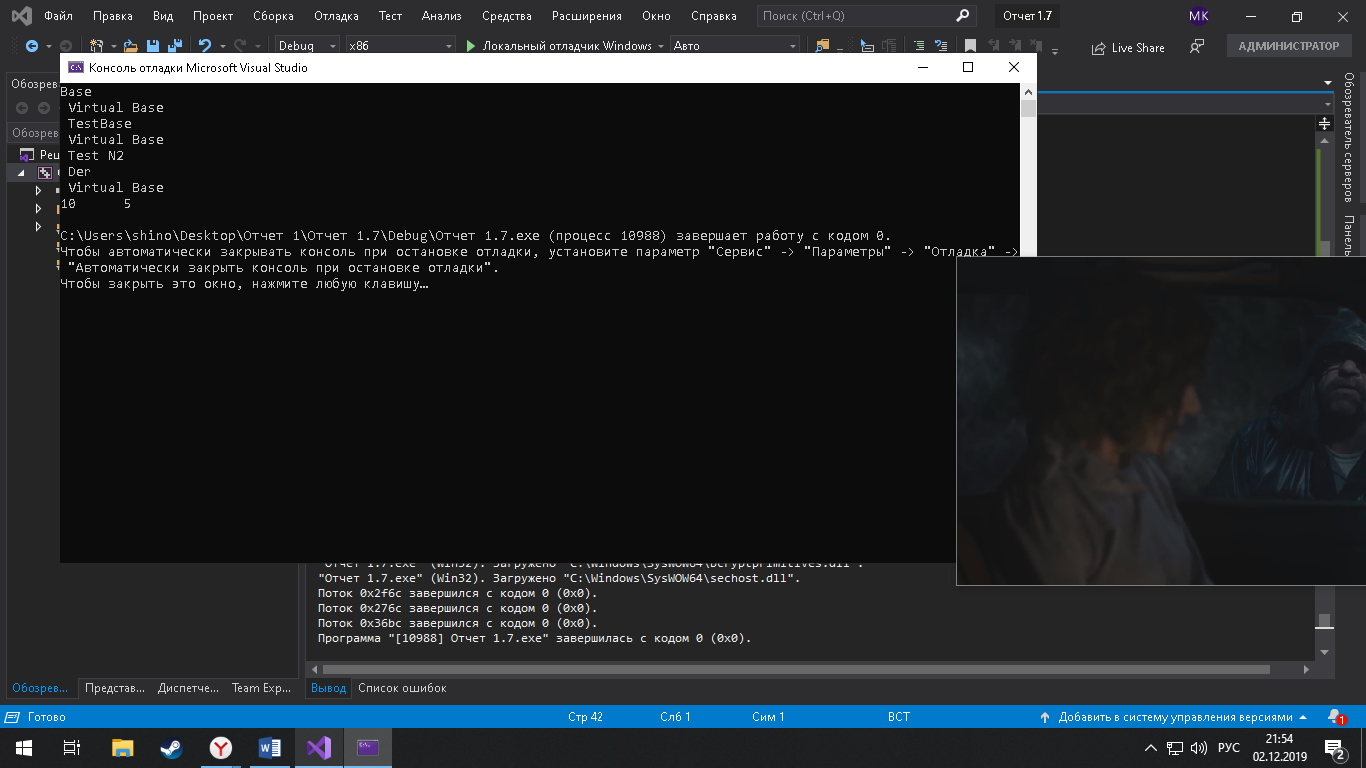
Der B;

B.outF();

B.OutS();

cout << "\n" << B.a << '\t' << B.Base::a << "\n";

}



## 

## №8

Составить иерархию классов, элементы которой решают задачу вычисления площадей фигур. Базовый класс: виртуальная функция вычисления площади. Производный класс: содержат описание для круга, треугольника, прямоугольника.

#include <iostream>

#include <math.h>

using namespace std;

class Base {

public: double S;

void OutS() { S = Sq(); cout << "Square = " << S << endl; }

virtual double Sq() = 0;

virtual void input() = 0;

};

class Tri :public Base {

public: int a, b, c;

double p;

void input() {

cout << "Input a,b,c a= "; cin >> a;

cout << "\n b= "; cin >> b;

cout << "\n c= "; cin >> c;

}

double Sq();

};

double Tri::Sq() {

p = (a + b + c) / 2;

return sqrt(p \* (p - a) \* (p - b) \* (p - c));

}

class rect :public Base {

public: int a, b;

double p;

void input() {

cout << "Input a,b a= "; cin >> a;

cout << "\n b= "; cin >> b;

}

double Sq();

};

double rect::Sq() {

return a\*b;

};

class circle :public Base {

public:

int r;

void input() {

cout << "Input r "; cin >> r;

}

double Sq();

};

double circle::Sq() {

return 3.14 \* r \* r;

}

int main()

{

Base\* A = 0;

cout << "Input figure" << endl;

int type;

cin >> type;

switch (type) {

case 1: A = new Tri; break;

case 2:A = new rect; break;

case 3: A = new circle; break;

default:cout << "error" << endl;

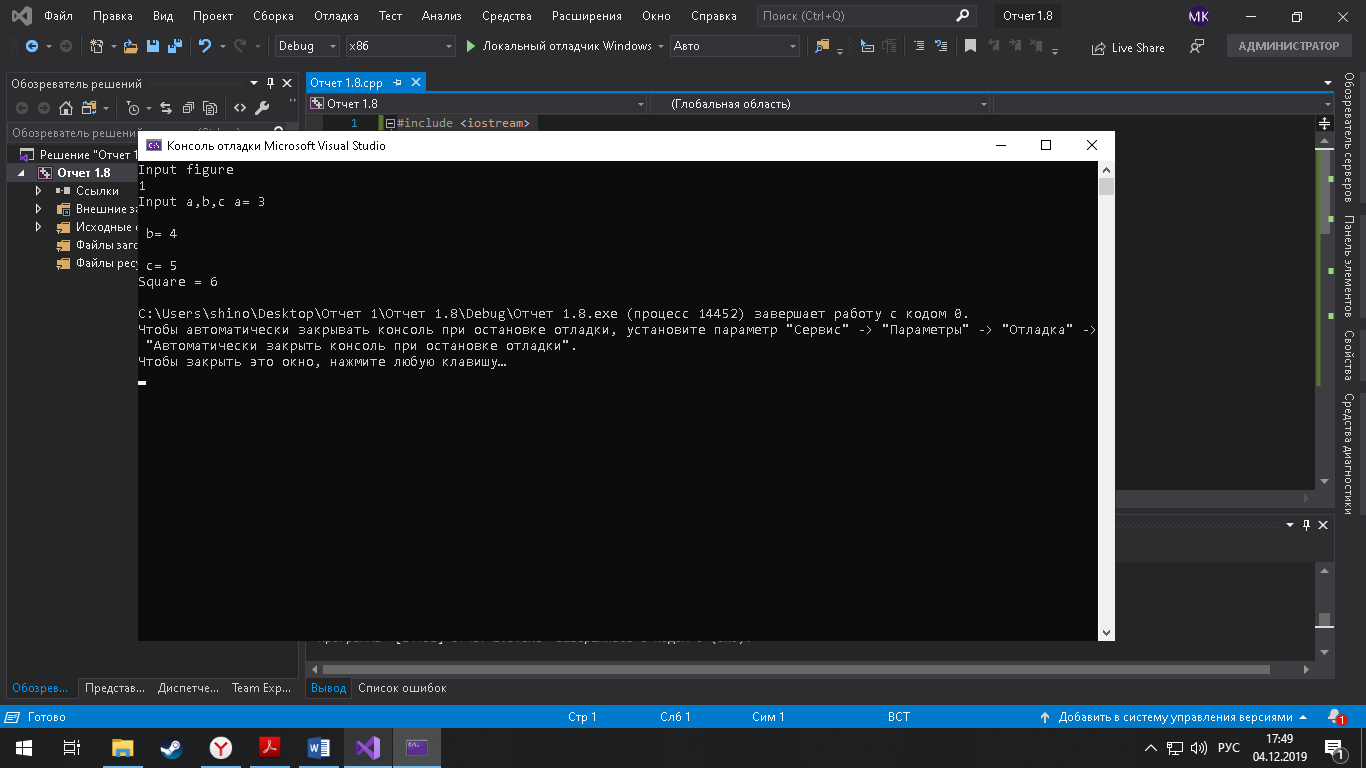
}

A->input();

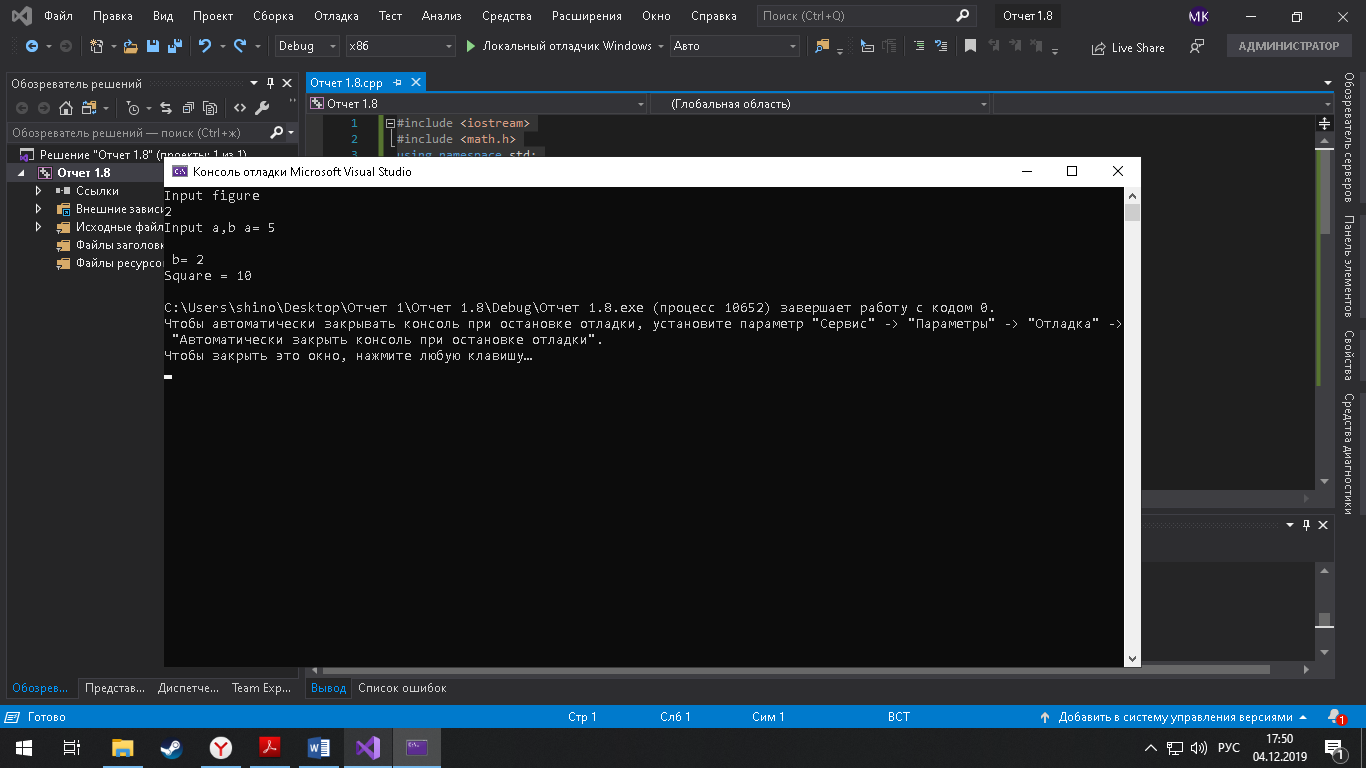
A->OutS();

}

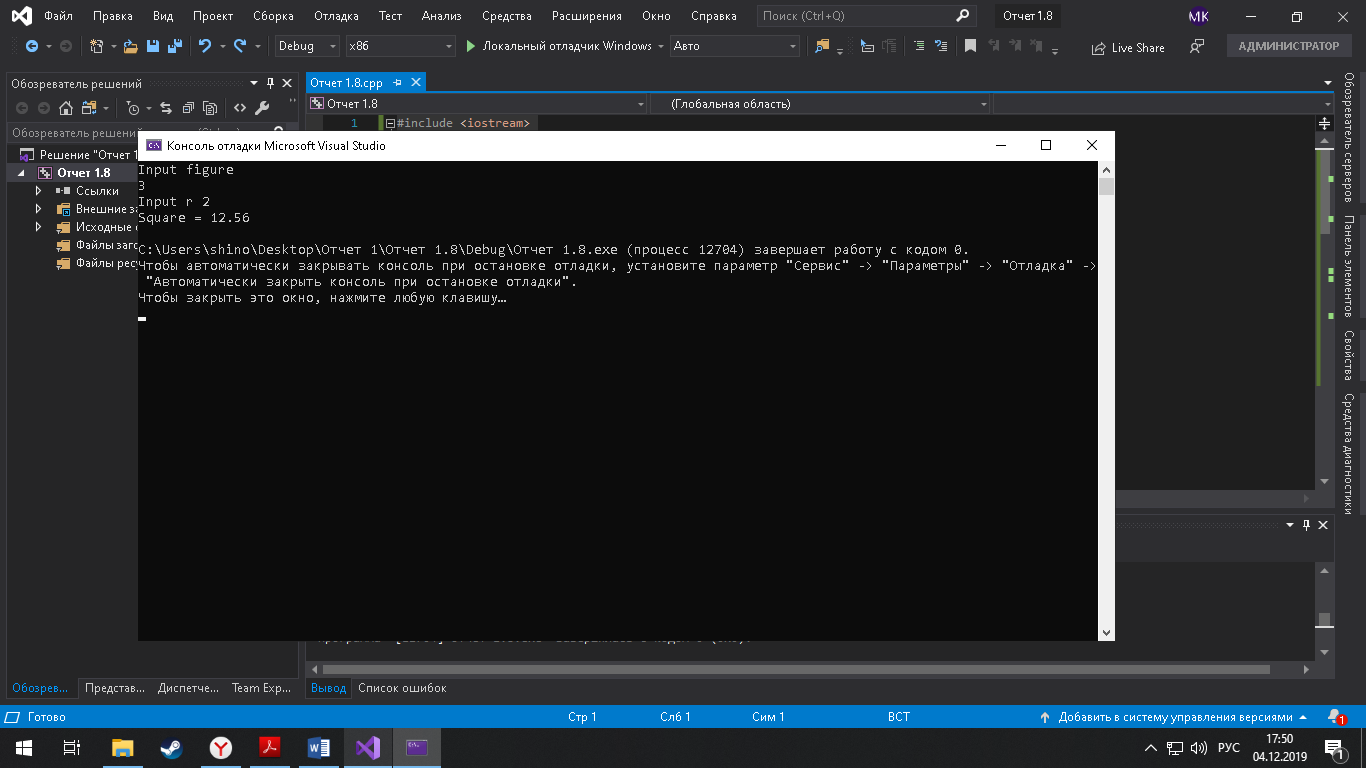
Треугольник



Прямоугольник



Круг



## №9

Составить иерархию классов, которые решают задачу дифференцирования функции.

Базовый: чистая виртуальная функция и метод численного дифференцирования.

Производный: определение функции численного дифференцирования.

#include <iostream>

#include <math.h>

using namespace std;

class Base {

public:

virtual double f(double) = 0;

double diff(double, double);

};

double Base::diff(double x, double dx) {

double dy, dzy, u, v;

dy = (u = f(x + dx) - (v = f(x)));

dzy = f(x + 2 \* dx) - 2 \* u + v;

return(dy - 0.5 \* dzy) / dx;

}

class Azt :public Base {

public:

double f(double x) { return sin(x); };

};

void main() {

double x;

Azt\* A = new Azt;

cout << "Output data" << endl;

for (x = 0; x <= 6.28; x += 0.1)

cout << A->diff(x, 0.1) << "\t" << cos(x) << endl;

delete A;

}

# Домашние задания

## Домашнее задание №1

Разработать класс, который выводит десять простых чисел подряд от заданного значения.

Консоль:

#pragma once

#include <iostream>

using namespace std;

class Base {

public: int n, y = 0, i, j, a;

void nahod(int);

};

void Base::nahod(int n) {

i = 2; j = 0;

while (y < 10) {

while ((j != 1) && (i \* i <= n)) {

if (n % i == 0) {

j = 1;

i = i++;

}

else {

i = i++;

}

}

if (j != 1) {

a = n;

cout << a << endl;

n++;

y++;

}

else {

n++;

}

i = 2; j = 0;

}

}

#include "Header.h"

int main() {

cout << "Vvedite pervoe chislo" << endl;

int n;

cin >> n;

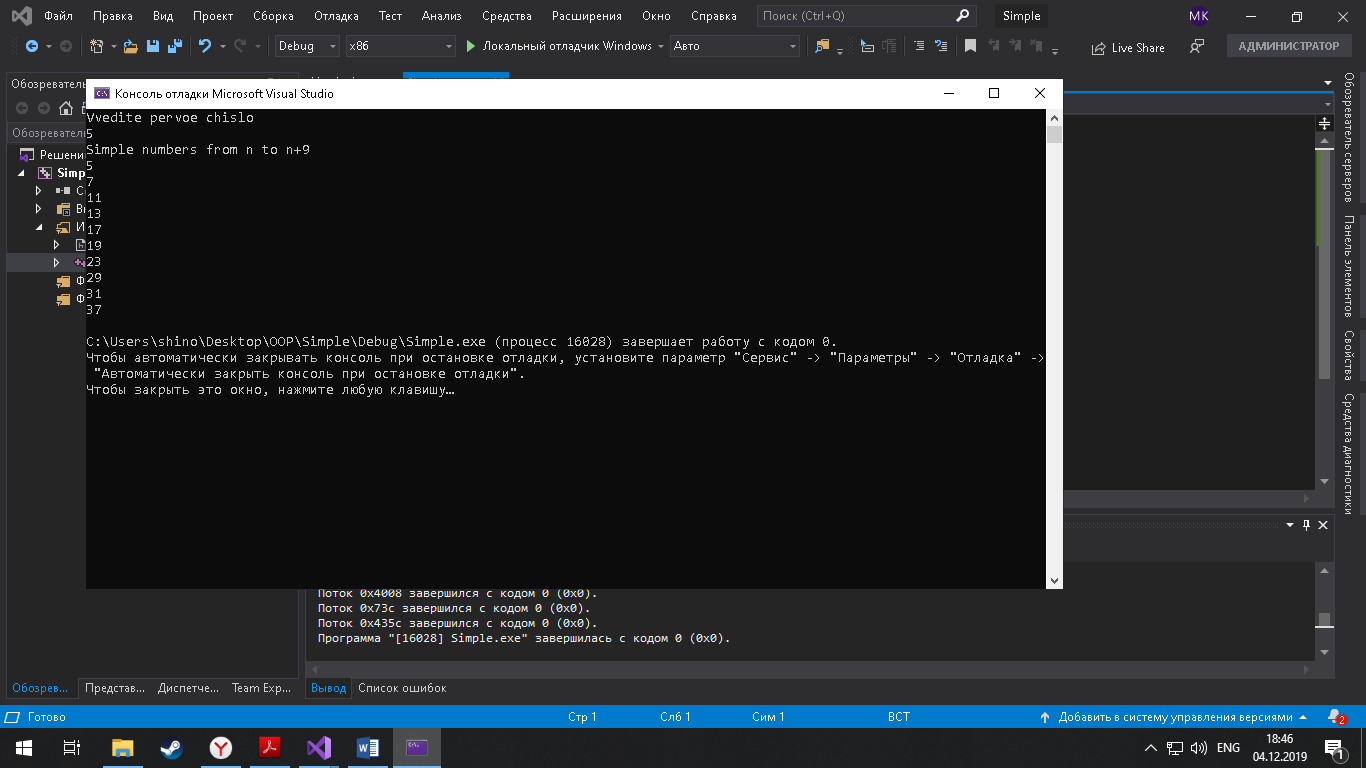
Base\* A=new Base;

cout << "Simple numbers from n to n+9" << endl;

A->nahod(n);

delete A;

}



Windows forms:

#pragma endregion

private: System::Void Button1\_Click(System::Object^ sender, System::EventArgs^ e) {

int n, y = 0, i, j;

String^ a;

n = n.Parse(this->maskedTextBox1->Text);

i = 2; j = 0;

while (y < 10) {

while ((j != 1) && (i \* i <= n)) {

if (n % i == 0) {

j = 1;

i = i++;

}

else {

i = i++;

}

}

if (j != 1) {

a = System::Convert::ToString(n);

a += " ";

label4->Text += a;

n++;

y++;

}

else {

n++;

}

i = 2; j = 0;

}

maskedTextBox1->Clear();

};

private: System::Void TextBox1\_TextChanged(System::Object^ sender, System::EventArgs^ e) {

}

private: System::Void TextBox2\_TextChanged(System::Object^ sender, System::EventArgs^ e) {

}

private: System::Void MyForm\_Load(System::Object^ sender, System::EventArgs^ e) {

}

private: System::Void Label2\_Click(System::Object^ sender, System::EventArgs^ e) {

}

private: System::Void MaskedTextBox1\_MaskInputRejected(System::Object^ sender, System::Windows::Forms::MaskInputRejectedEventArgs^ e) {

this->ActiveControl = maskedTextBox1;

}

private: System::Void Label4\_Click(System::Object^ sender, System::EventArgs^ e) {

}

private: System::Void Label1\_Click(System::Object^ sender, System::EventArgs^ e) {

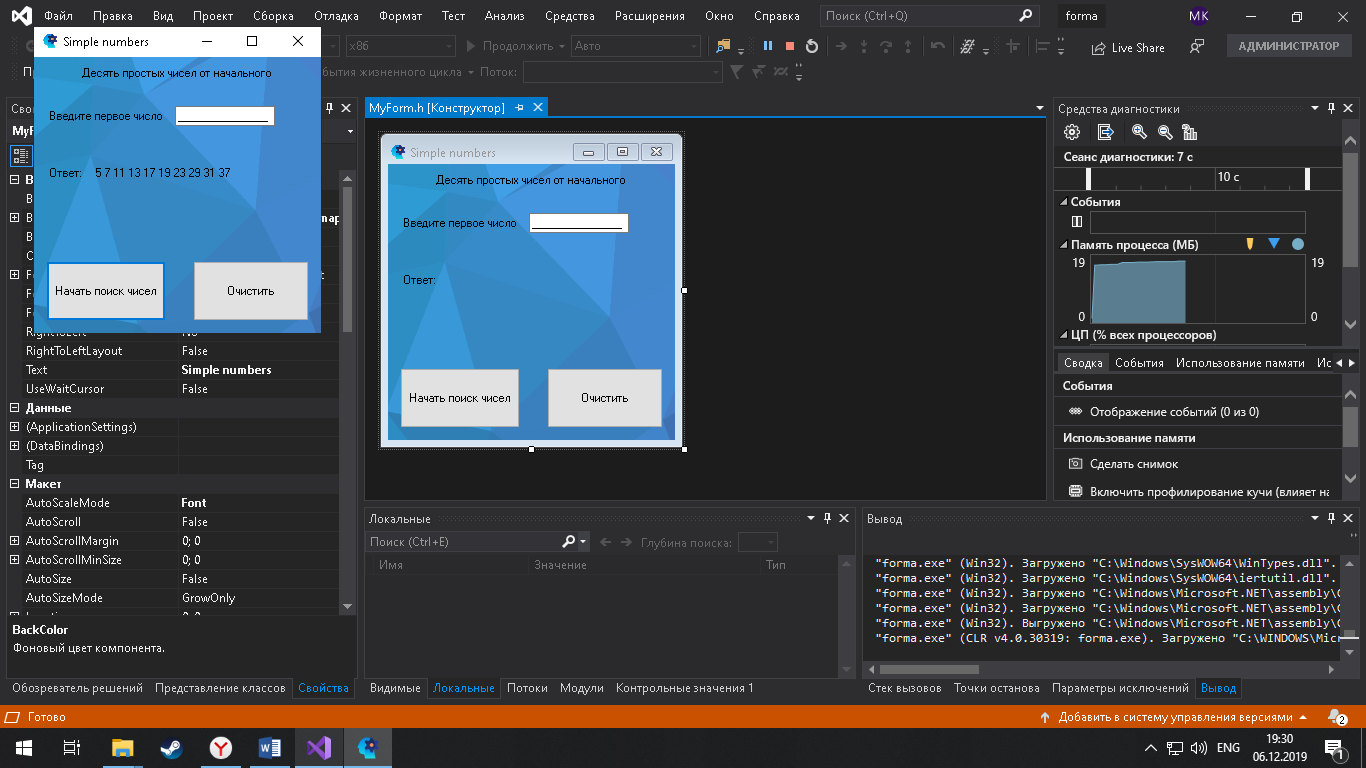
}

private: System::Void Button2\_Click(System::Object^ sender, System::EventArgs^ e) {

label4->Text = " ";

}

};



## Домашнее задание №2

Составить программу решающую задачу перемножения двух матриц, задаваемых базовым классом и производным классом.

Файл Header.h

#pragma once

#include <iostream>

#include <ctime>

using namespace std;

class Base {

public:

Base();

~Base();

void in\_1();

void out\_1();

int\*\* mas, m, n;

};

class second :public Base {

private: int\*\* mas2, x, y, \*\* mas3;

public:

second();

~second();

void in\_2();

void out\_2();

void multi();

};

Файл multi.cpp

#include "Header.h"

second::second() {

int NSIZE1, MSIZE1;

cout << "Введите количество строк второй матрицы = " << endl; cin >> MSIZE1;

cout << "Введите количество столбцов второй матрицы = " << endl; cin >> NSIZE1;

cout << endl;

x = MSIZE1;

y = NSIZE1;

mas2 = new int\* [x];

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

mas2[i] = new int[y];

}

}

second::~second() {

for (int i = 0; i < x; i++)

delete[]mas2[i];

delete[]mas2;

}

Base::Base() {

int NSIZE, MSIZE;

cout << "Введите количество строк первой матрицы = " << endl; cin >> MSIZE;

cout << "Введите количество столбцов второй матрицы = " << endl; cin >> NSIZE;

cout << endl;

m = MSIZE;

n = NSIZE;

mas = new int\* [m];

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

mas[i] = new int[n];

}

}

Base::~Base() {

for (int i = 0; i < m; i++)

delete[]mas[i];

delete[]mas;

}

void Base::in\_1() {

srand(time(0));

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

for (int j = 0; j < n; j++) {

mas[i][j] =rand()%20;

}

}

}

void Base::out\_1() {

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

for (int j = 0; j < n; j++) {

cout << mas[i][j] <<"\t";

}

cout << endl;

}

}

void second::multi() {

mas3 = new int\* [m];

if (n == x && m==y) {

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

mas3[i] = new int[y]; {

for (int j = 0; j < y; j++) {

mas3[i][j] = 0;

for (int k = 0; k < n; k++) {

mas3[i][j] += mas[i][k] \* mas2[k][j];

}

}

}

}

cout << "Перемножение первой матрицы со второй" << endl;

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

for (int j = 0; j < y; j++) {

cout << mas3[i][j] << "\t";

}

cout << endl;

}

for (int i = 0; i < m; i++)

delete[]mas3[i];

delete[]mas3;

}

else {

cout << "\nОшибка. Количество строк первой матрицы не совпадает с количеством стобцов второй." << endl;

}

}

void second::in\_2(){

srand(time(0));

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

for (int j = 0; j < y; j++) {

mas2[i][j] = 10+ rand() % 20;

}

}

}

void second::out\_2() {

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

for (int j = 0; j < y; j++) {

cout << mas2[i][j] << "\t";

}

cout << endl;

}

}

int main()

{

setlocale(LC\_ALL, "Russian");

cout << "Для перемножения матриц необходимо, чтобы выполнялось условие:\n Количество строк первой матрицы = количество столбцов второй" << endl;

second\* B;

B = new second;

B->in\_1();

B->out\_1();

cout << "\n" << endl;

B->in\_2();

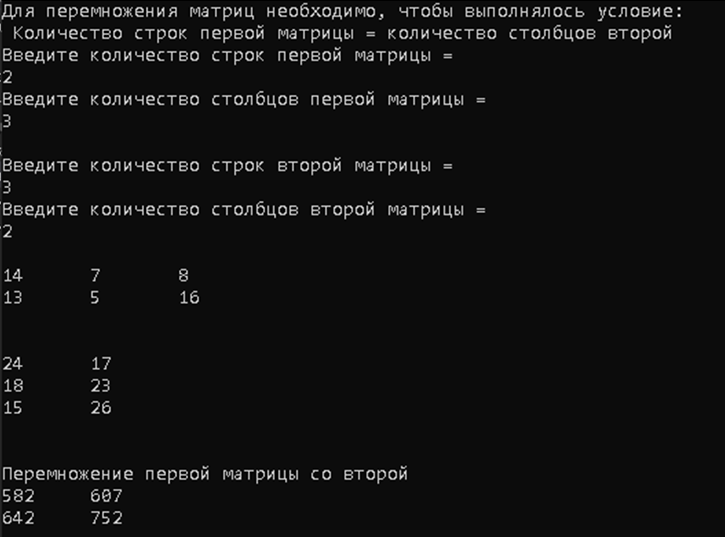
B->out\_2();

cout << "\n" << endl;

B->multi();

return 0;

}



namespace WindowsFormsApp1

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

private void button1\_Click(object sender, EventArgs e)

{

dataGridView1.Rows.Clear();

dataGridView1.Columns.Clear();

dataGridView2.Rows.Clear();

dataGridView2.Columns.Clear();

dataGridView3.Rows.Clear();

dataGridView3.Columns.Clear();

int m = Convert.ToInt32(textBox1.Text);

int n = Convert.ToInt32(textBox2.Text);

int m1 = Convert.ToInt32(textBox3.Text);

int n1 = Convert.ToInt32(textBox4.Text);

dataGridView1.ColumnCount=n;

dataGridView1.RowCount=m;

dataGridView2.ColumnCount=n1;

dataGridView2.RowCount=m1;

int[,] matrix1 = new int[m,n];

int[,] matrix2 = new int[m1,n1];

Random random = new Random();

for(int i=0; i<m; i++)

{

for (int j = 0; j <n; j++)

{

matrix1[i, j] = random.Next(1, 20);

dataGridView1.Rows[i].Cells[j].Value = matrix1[i, j].ToString();

}

}

for (int i = 0; i < m1; i++)

{

for (int j = 0; j < n1; j++)

{

matrix2[i, j] = random.Next(1, 20);

dataGridView2.Rows[i].Cells[j].Value = matrix2[i, j].ToString();

}

}

int[,] matrix3 = new int[m,n1];

if ((m == n1) && (m1 == n))

{

for (int i = 0; i < dataGridView2.ColumnCount; i++)

{

dataGridView3.Columns.Add("", "");

}

for (int j = 0; j < dataGridView1.RowCount; j++)

{

dataGridView3.Rows.Add();

}

for (int i = 0; i < dataGridView1.RowCount; i++)

{

for (int j = 0; j < dataGridView2.ColumnCount; j++)

{

int s = 0;

for (int k = 0; k < dataGridView2.RowCount; k++)

s += matrix1[i, k] \* matrix2[k, j];

dataGridView3[j, i].Value = s;

}

}

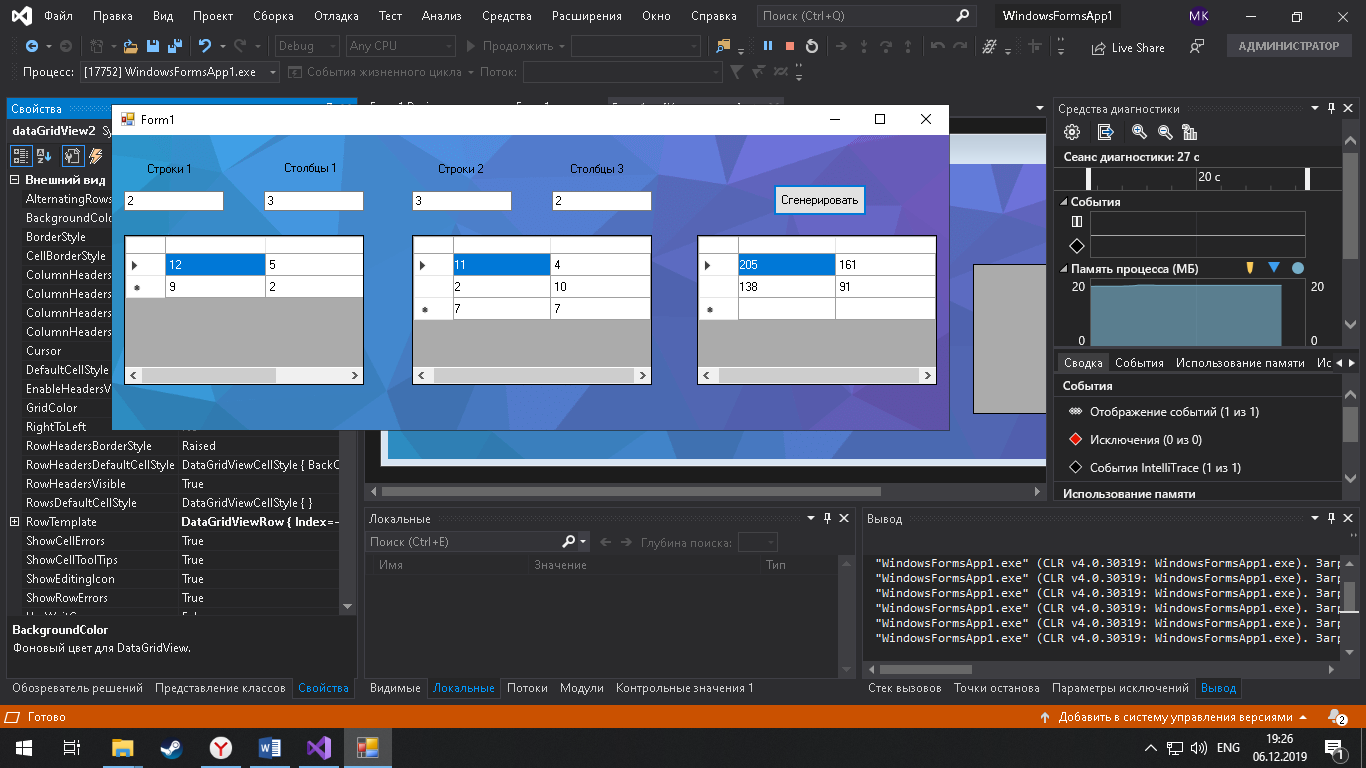
}

else MessageBox.Show("Неправильный ввод данных");

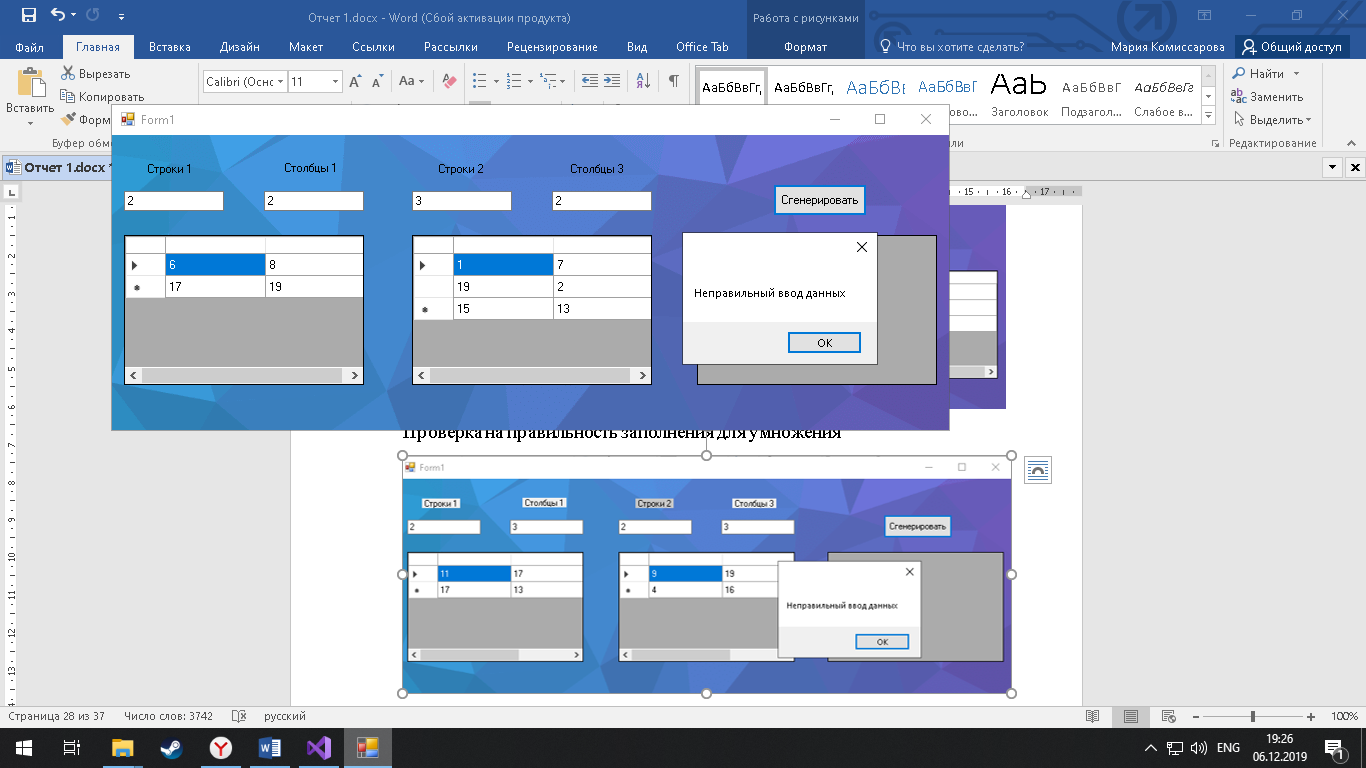
}

}

}



Проверка на правильность заполнения для умножения



## Домашнее задание №3

Сортировка массива

#include "Header.h"

void main()

{

Base\* A;

cout << "Input type of array" << endl;

cout << "1 - integer, 2 - double, 3 - symbols" << endl;

int type;

cin >> type;

switch (type){

case 1: {int len;

cout << "Vvedite colvo elementov massiva " << endl;

cin >> len;

integer\* B;

B = new integer(len);

B->inmass();

B->sort();

B->outmass();

delete B;

}

case 2: {

int len;

cout << "Vvedite colvo elementov massiva " << endl;

cin >> len;

doub\* B;

B = new doub(len);

B->in\_a();

B->sort();

B->out\_a();

delete B;

}

case 3:

int len;

cout << "Vvedite colvo elementov massiva " << endl;

cin >> len;

str\* B;

B = new str(len);

B->in\_b();

B->sort();

B->out\_b();

delete B;

}

}

#pragma once

#include <iostream>

#include <locale.h>

using namespace std;

class Base {

private:

int len, \* x;

public:

virtual void sort() = 0;

};

class integer :public Base {

int len, \* x;

public:

integer(int k) :len(k) {

x = new int[len];

}

~integer() {

delete[]x;

}

void inmass() {

cout << "Vvod elementov massiva " << endl;

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

cout << "x[" << i << "]= " << endl;

cin >> x[i];

}

}

void outmass() {

cout << "Vivod elementov massiva " << endl;

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

cout << "x[" << i << "]= " << x[i] << endl;

}

}

void sort() override {

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

for (int j = i + 1; j < len; j++) {

if (x[i] < x[j])

{

int buf = x[i];

x[i] = x[j];

x[j] = buf;

}

}

}

}

};

class doub : public Base {

int len;

double\* x;

public:

doub(int k) :len(k) {

x = new double[len];

}

~doub() {

delete[]x;

}

void in\_a() {

cout << "Vvod elementov massiva " << endl;

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

cout << "x[" << i << "]= " << endl;

cin >> x[i];

}

}

void out\_a() {

cout << "Vivod elementov massiva " << endl;

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

cout << "x[" << i << "]= " << x[i] << endl;

}

}

void sort() override {

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

for (int j = i + 1; j < len; j++) {

if (x[i] < x[j])

{

double buf = x[i];

x[i] = x[j];

x[j] = buf;

}

}

}

}

};

class str :public Base {

int len;

string\* x;

public:

str(int k) :len(k) {

x = new string[len];

}

~str() {

delete[]x;

}

void in\_b() {

cout << "Vvod elementov massiva " << endl;

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

cout << "x[" << i << "]= " << endl;

cin >> x[i];

}

}

void out\_b() {

cout << "Vivod elementov massiva " << endl;

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

cout << "x[" << i << "]= " << x[i] << endl;

}

}

void sort() override {

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

for (int j = i + 1; j < len; j++) {

if (x[i] < x[j])

{

string buf = x[i];

x[i] = x[j];

x[j] = buf;

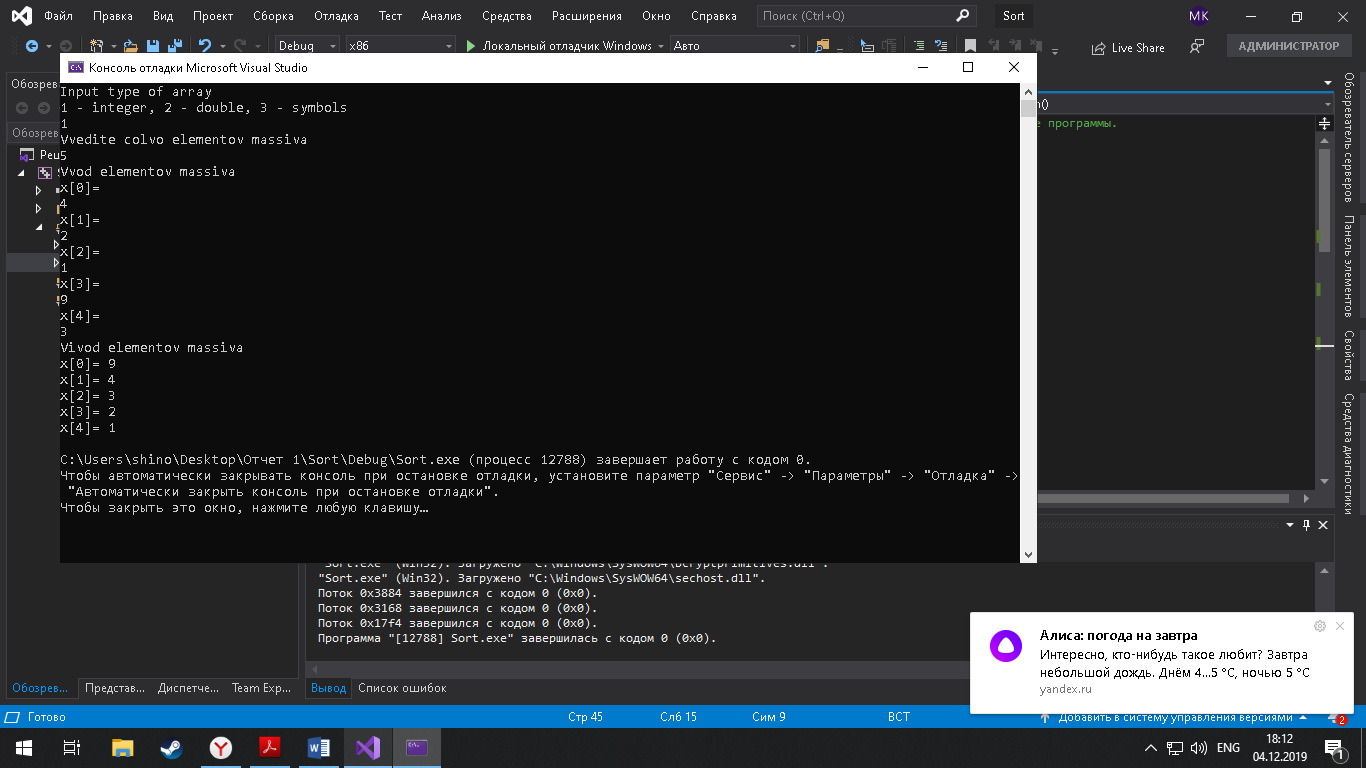
}

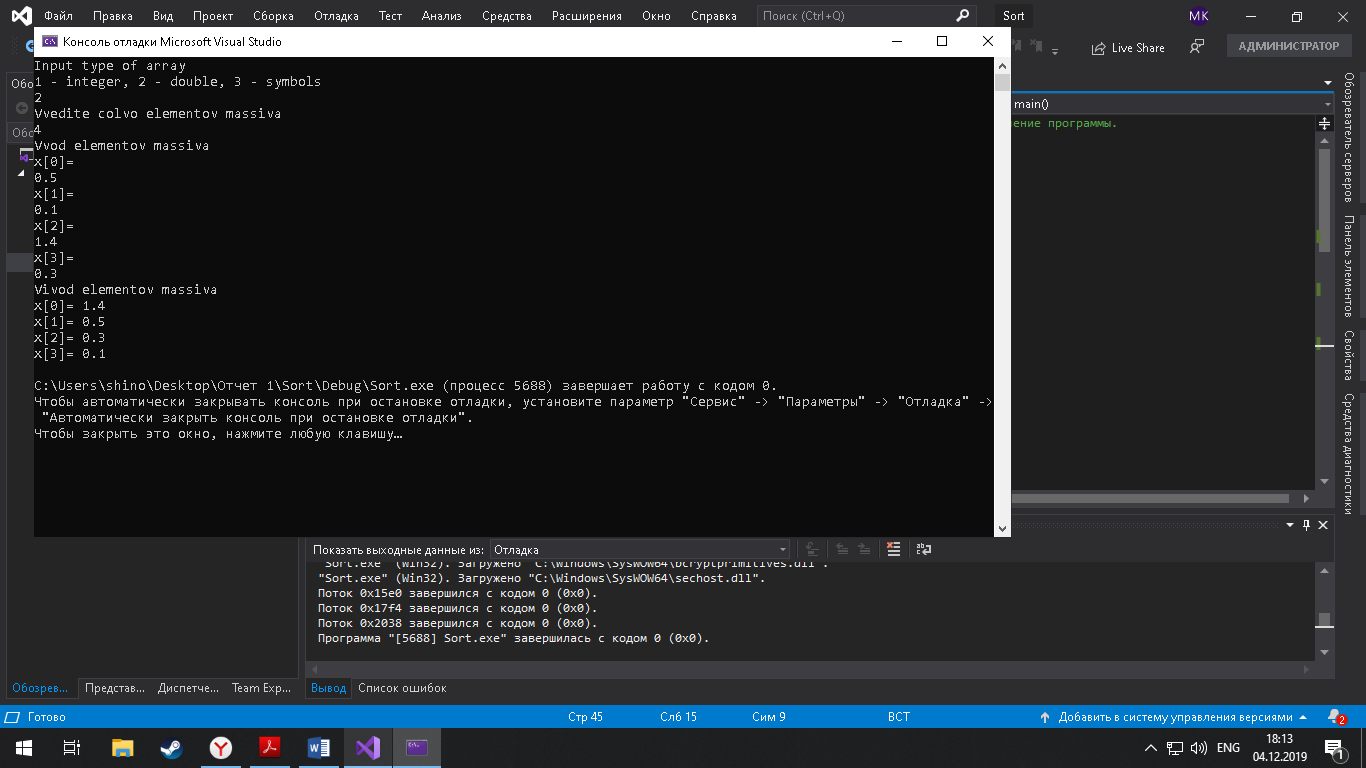
}

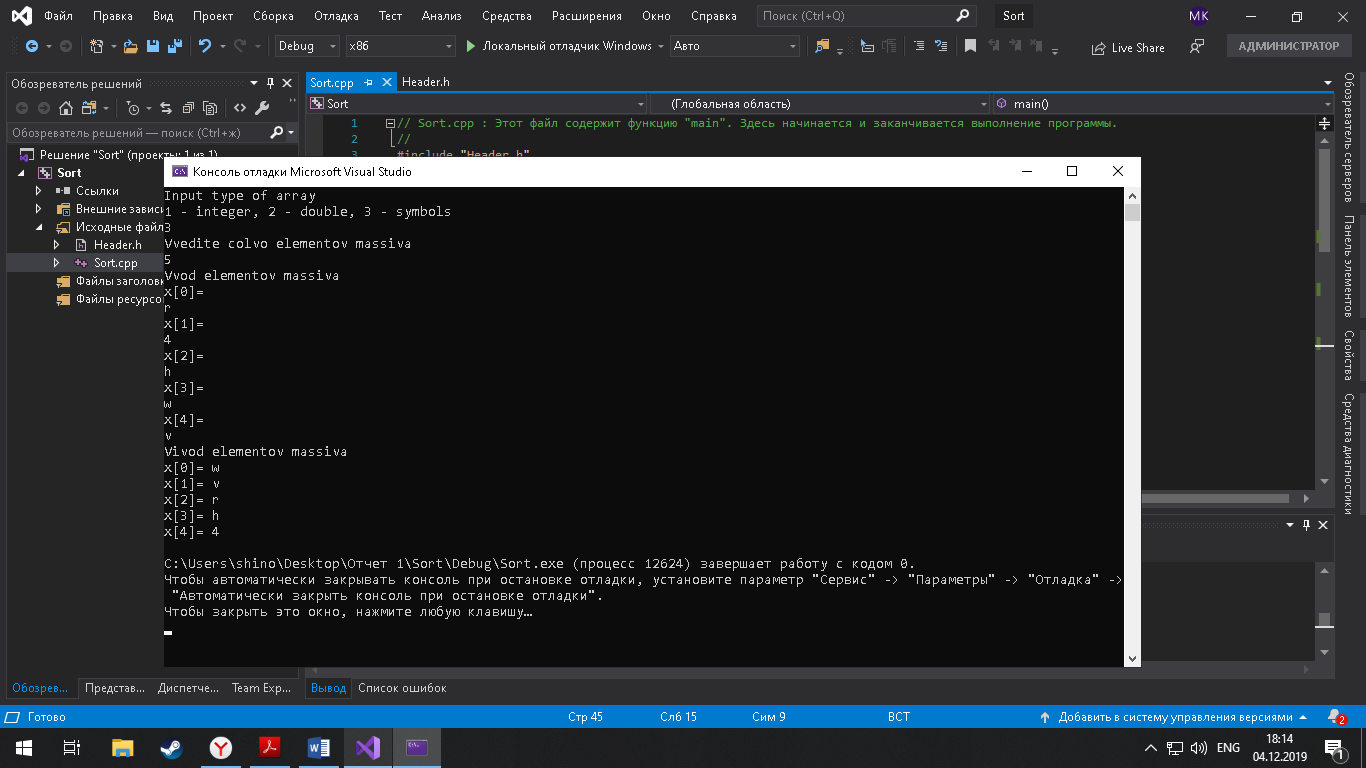
}

}

};







private void button1\_Click(object sender, EventArgs e)

{

richTextBox1.Clear();

listBox1.Items.Clear();

listBox2.Items.Clear();

}

private void button2\_Click(object sender, EventArgs e)

{

if (radioButton1.Checked == true)

{

listBox2.Items.Clear();

int n = richTextBox1.Lines.Length;

int[] a = new int[n];

for (int i = 0; i < n - 1; i++)

{

a[i] = Convert.ToInt32(richTextBox1.Lines[i]);

}

for (int i = 0; i < n-1; i++)

{

for (int j = i + 1; j < n-1; j++)

{

if (a[i] < a[j])

{

int buf = a[i];

a[i] = a[j];

a[j] = buf;

}

}

}

for (int i = 0; i < n - 1; i++)

{

listBox2.Items.Add(a[i]);

}

}

if (radioButton2.Checked == true)

{

listBox2.Items.Clear();

int n = richTextBox1.Lines.Length;

double[] a = new double[n];

for (int i = 0; i < n - 1; i++)

{

a[i] = Convert.ToDouble(richTextBox1.Lines[i]);

}

for (int i = 0; i < n - 1; i++)

{

for (int j = i + 1; j < n - 1; j++)

{

if (a[i] < a[j])

{

double buf = a[i];

a[i] = a[j];

a[j] = buf;

}

}

}

for (int i = 0; i < n - 1; i++)

{

listBox2.Items.Add(a[i]);

}

}

if (radioButton3.Checked == true)

{

listBox2.Items.Clear();

int n = richTextBox1.Lines.Length;

char[] a = new char[n];

for (int i = 0; i < n - 1; i++)

{

a[i] = Convert.ToChar(richTextBox1.Lines[i]);

}

for (int i = 0; i < n - 1; i++)

{

for (int j = i + 1; j < n - 1; j++)

{

if (a[i] < a[j])

{

char buf = a[i];

a[i] = a[j];

a[j] = buf;

}

}

}

for (int i = 0; i < n - 1; i++)

{

listBox2.Items.Add(a[i]);

}

}

}

private void richTextBox1\_TextChanged(object sender, EventArgs e)

{

if (radioButton1.Checked == true)

{

listBox1.Items.Clear();

int n = richTextBox1.Lines.Length;

int[] a = new int[n];

for (int i = 0; i < n - 1; i++)

{

a[i] = Convert.ToInt32(richTextBox1.Lines[i]);

listBox1.Items.Add(a[i]);

}

}

if (radioButton2.Checked == true)

{

listBox1.Items.Clear();

int n = richTextBox1.Lines.Length;

double[] a = new double[n];

for (int i = 0; i < n - 1; i++)

{

a[i] = Convert.ToDouble(richTextBox1.Lines[i]);

listBox1.Items.Add(a[i]);

}

}

if (radioButton3.Checked == true)

{

listBox1.Items.Clear();

int n = richTextBox1.Lines.Length;

char[] a = new char[n];

for (int i = 0; i < n - 1; i++)

{

a[i] = Convert.ToChar(richTextBox1.Lines[i]);

listBox1.Items.Add(a[i]);

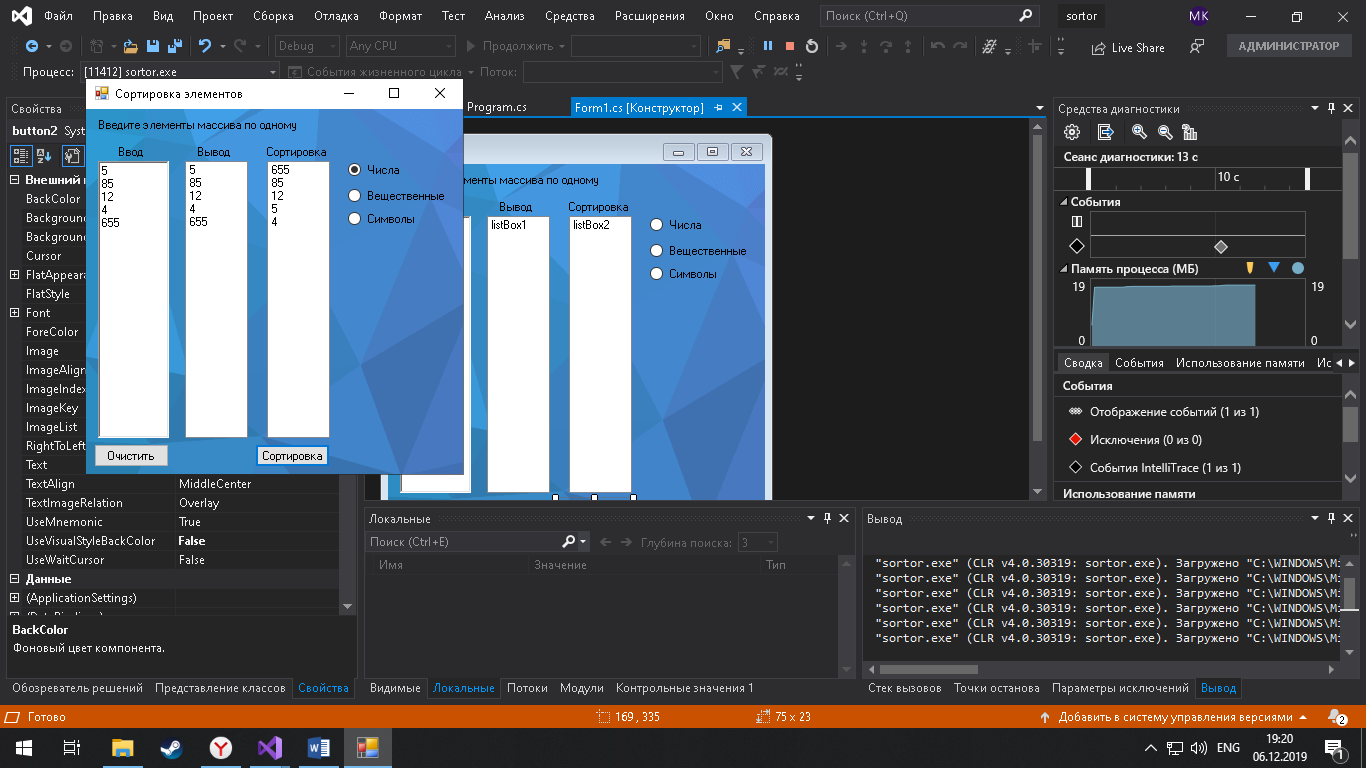
}

}

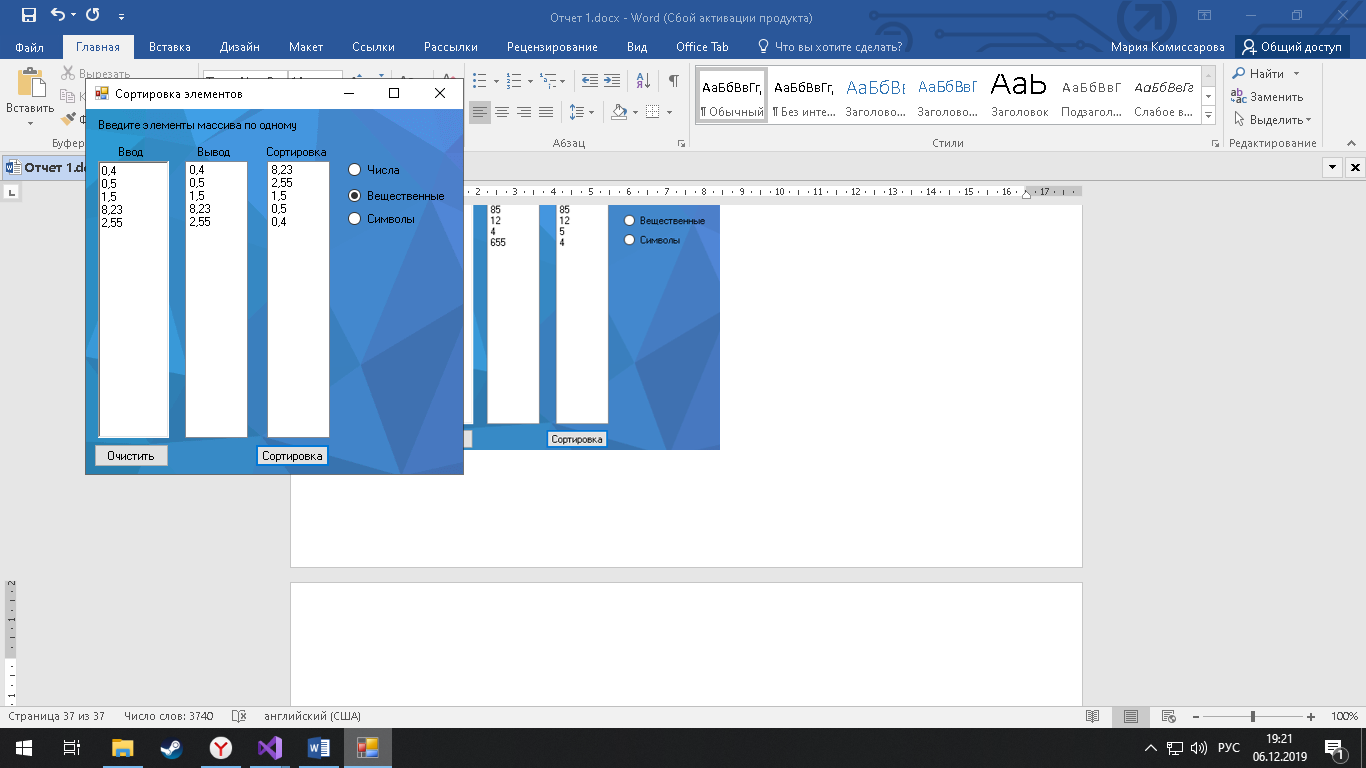
}

}

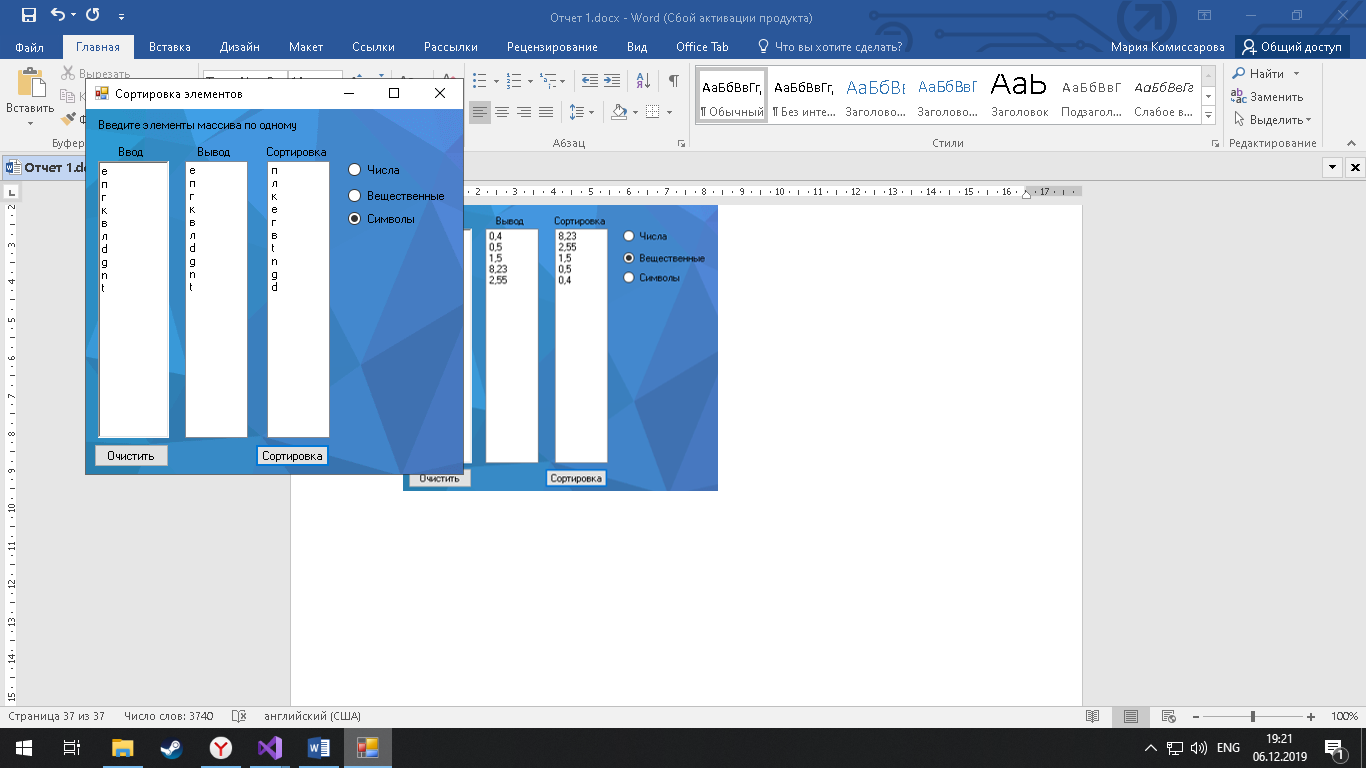
Числа



Double



Char



## Домашнее задание №4

#pragma once

#include <iostream>

#include <ctime>

#include <cmath>

using namespace std;

class Rabbit

{

public:

int x, y, count;

Rabbit(int, int);

Rabbit();

void jump(int, int);

void increase(Rabbit\*&, int, int);

void death(Rabbit\*&, int, int);

void life(int amount, int ticks, int chance, Rabbit\*&, int width, int height);

};

Rabbit::Rabbit(int width, int height) { //конструктор новых кроликов

x = rand() % width + 1;

y = rand() % height + 1;

count = 0;

}

Rabbit::Rabbit() {

x = 0;

y = 0;

count = 0;

}

void Rabbit::death(Rabbit\*& Old, int n, int index) {

int j = index;

Rabbit\* New = new Rabbit[n]; //удаление како-либо кролика и создание нового массива

for (int i = 0; i < index; i++)

{

New[i] = Old[i];

}

for (int i = index; i < n; i++)

{

j++;

New[i] = Old[j];

}

delete[] Old;

Old = New; //заменяем имя нового на имя старого, чтобы в остальном коде не было изменений

}

void Rabbit::jump(int width, int height) {

int chance;

chance = rand() % 9 + 1;

switch (chance) {

case 1: if (x < width) {

x++;

}

count++;

break;

case 2: if ((x > 0) || (x = width)) {

x--;

}

count++;

break;

case 3: if ((y < height) || (y == 0)) {

y++;

}

count++;

break;

case 4: if ((y > 0) || (y = height)) {

y--;

}

count++;

break;

case 5: if ((x < width) && (y < height)) {

x++;

y++;

}

count++;

break;

case 6: if ((x > 0) && (y > 0)) {

x--;

y--;

}

count++;

break;

case 7: if ((x > 0) && (y < height)) {

x--;

y++;

}

count++;

break;

case 8:if ((x < width) && (y > 0)) {

x++;

y--;

}

count++;

break;

case 9: {

count++;

}

break;

}

}

void Rabbit::increase(Rabbit\*& Old, int n, int index) {

Rabbit\* New = new Rabbit[n]; //создаем новый массив класса кролик (рождение нового кролика) и переписываем массив

for (int i = 0; i < n - 1; i++)

{

New[i] = Old[i];

}

New[n - 1].x = Old[index].x;

New[n - 1].y = Old[index].y;

New[n - 1].count = 0;

delete[] Old;

Old = New; //заменяем

}

void Rabbit::life(int amount, int ticks, int chance, Rabbit\*& A, int width, int height) {

int t, i, j, v;

int number, child = 0;

for (t = 0; t < ticks; t++)

{

cout << "Number of ticks: " << t << endl;

number = amount;

for (i = 0; i < amount; i++)

{

for (j = i + 1; j < amount; j++)

{

if (((A[i].x == A[j].x) && (A[i].y == A[j].y))&&(child<(width\*height))) //рожденных кроликов не больше, чем площадь поля

{

v = rand() % 100 + 1;

if (v <= chance)

{

number++;

child++;

A->increase(A, number, i);

}

}

}

}

amount = number;

for (i = 0; i < amount; i++)

{

A[i].jump(width, height); //прыжки кроликов по индексам

}

for (i = 0; i < amount; i++)

{

if (A[i].count == 20) { //если количество прыжков = 20, то кролик умирает

cout << "Кролик умер" << endl;

amount--;

A->death(A, amount, i);

i--;

}

}

if (amount != 0)

{

for (i = 0; i < amount; i++)

{

cout << "Номер зайца " << i + 1 << endl;

cout << "x= " << A[i].x << endl;

cout << "y= " << A[i].y << endl;

cout << "Количество прыжков " << A[i].count << endl;

cout << "\n" << endl;

}

cout << "Количество зайцев " << amount << endl;;

cout << "Количество рожденных " << child << endl;

}

else

{

cout << "Никого нет в живых" << endl;;

}

}

}

#include "Header.h"

int main() {

srand(time(NULL));

setlocale(LC\_ALL, "Russian");

int width, height, chance, ticks, i,j,t,v;

int amount;

cout << "Input\n";

cout << "Ширина поля "; cin >> width;

cout << "Длина поля "; cin >> height;

cout << "Шанс рождения "; cin >> chance;

cout << "Количество тиков "; cin >> ticks;

cout << "Количество зайцев в начале "; cin >> amount;

Rabbit\* A = new Rabbit[amount];

for (i = 0; i < amount; i++) {

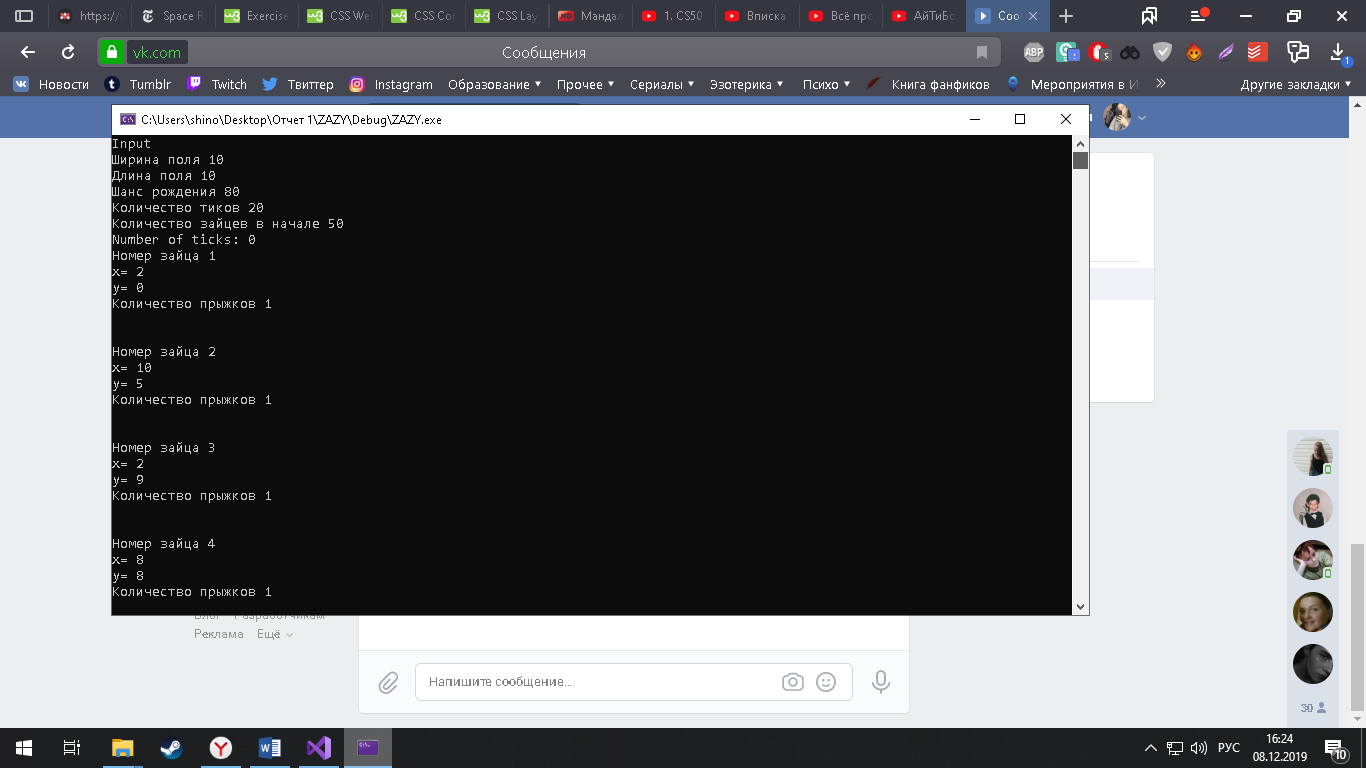
A[i] = Rabbit(width, height);

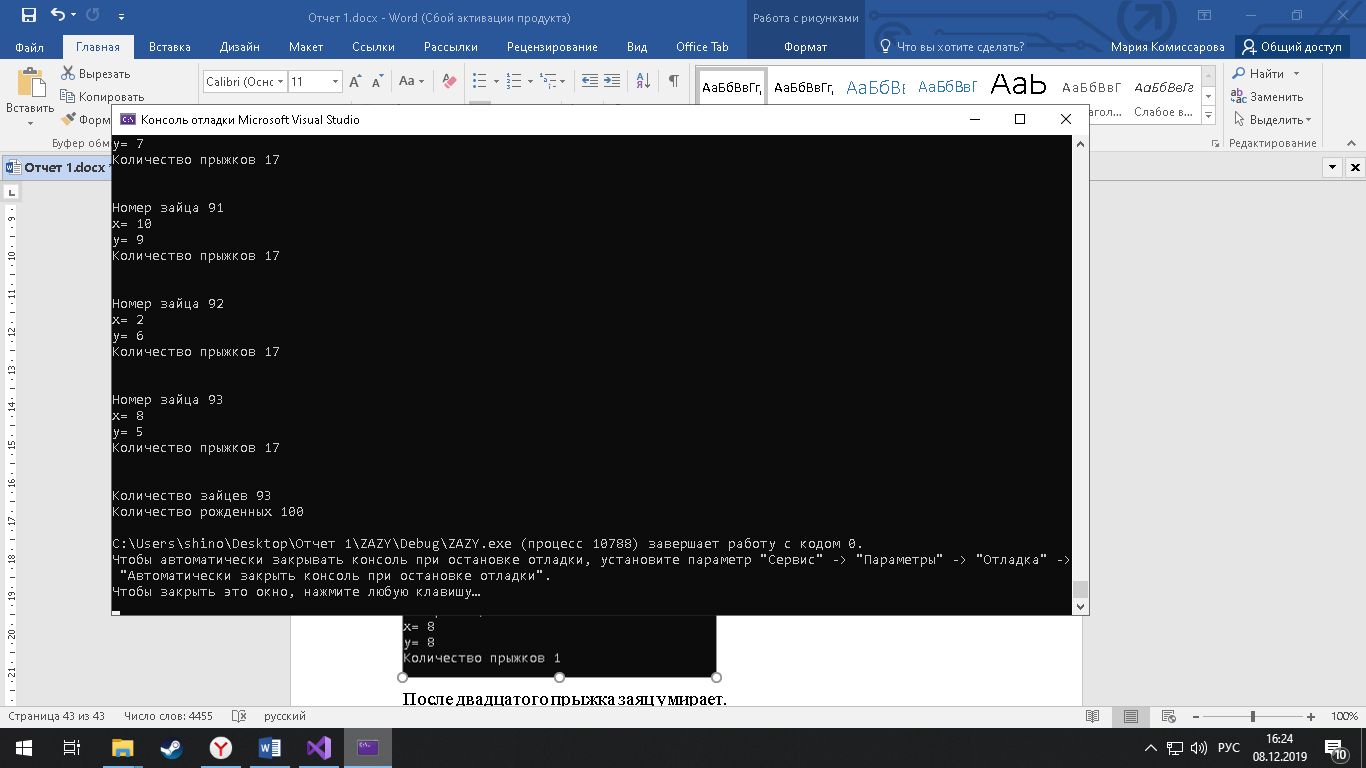
}

A->life(amount, ticks, chance, A, width, height);

delete[] A;

}





После двадцатого прыжка заяц умирает.

