САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ

УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ

Практическая работа №2

«Реализация управляющих операторов»

Выполнил: Мороз И.О.

Группа К3120

Проверил: Осипов Н.А.

Санкт-Петербург

2018 г.

# Упражнение 1.

Задание 1.

1. Были составлены алгоритм и программа для определения принадлежности точки к фигуре

#include "pch.h"

#include <iostream>

#include <math.h>

using namespace std;

int main()

{

setlocale(LC\_ALL, "Russian");

double x, y;

cout << "Введите координаты точки\n";

cout << "x = ";

cin >> x;

cout << "y = ";

cin >> y;

if (pow(x, 2) + pow(y, 2) < 9 && y > 0) {

cout << "Точка находится внутри фигуры";

}

else if (pow(x, 2) + pow(y, 2) > 9 || y < 0) {

cout << "Точка находится за пределами фигуры";

}

else {

cout << "Точка находится на границе фигуры";

}

return 0;

}

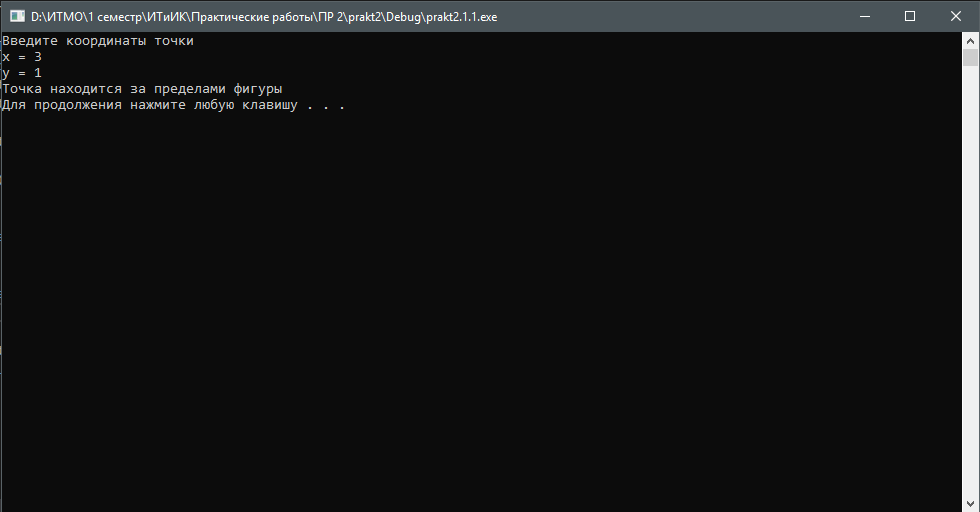


Рисунок 1 – пример работы алгоритма

Задание 2.

1. Были составлены алгоритм и программа для определения, является ли год под номером, введённым с клавиатуры, високосным

#include "pch.h"

#include <iostream>

using namespace std;

int main()

{

setlocale(LC\_ALL, "Russian");

int year;

cout << "Введите год: ";

cin >> year;

if ((year % 4 == 0 && year % 100 != 0) || (year % 400 == 0)) {

cout << "YES";

}

else {

cout << "NO";

}

return 0;

}

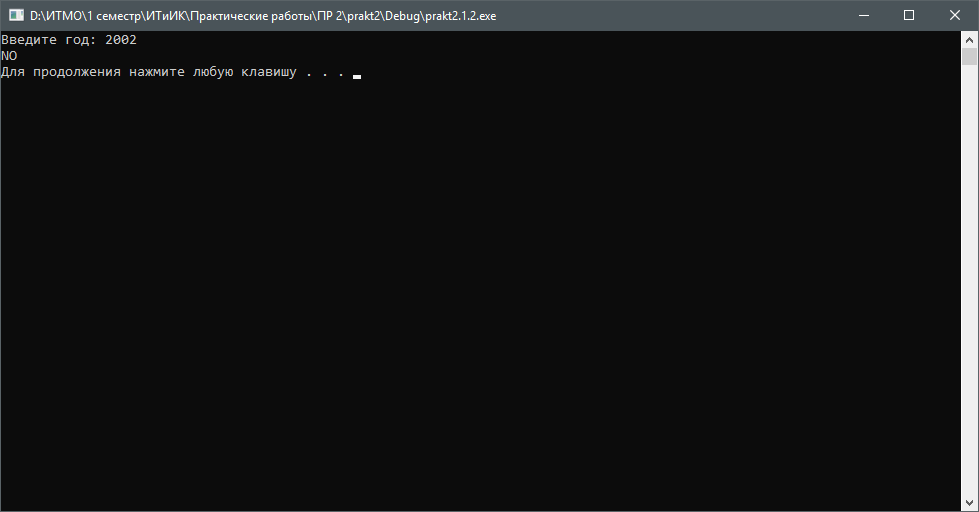


Рисунок 2 – пример работы алгоритма

Задание 3.

1. Был создан новый проект
2. Был реализован запрос от пользователя для выбора

char op;

cout << "Сделай свой выбор, собери авто своей мечты: ";

cin >> op;

1. Был реализован выбор альтернативного варианта с использованием оператора switch

switch (op) {

case 'S':

cout << "Радио играть должно\n";

cout << "Колеса круглые\n";

cout << "Мощный двигатель\n";

break;

case 'V':

cout << "Кондиционер хочу\n";

cout << "Радио играть должно\n";

cout << "Колеса круглые\n";

cout << "Мощный двигатель\n";

break;

default:

cout << "Колеса круглые\n";

cout << "Мощный двигатель\n";

}

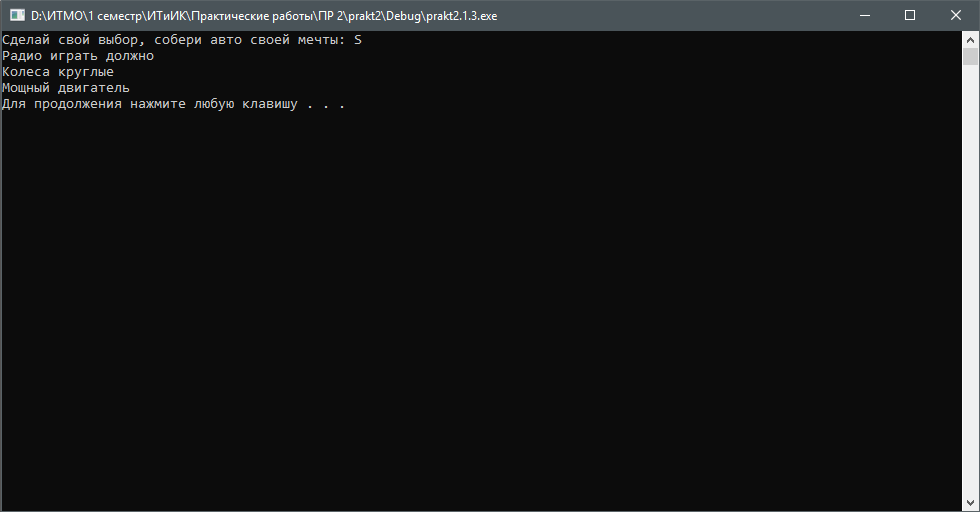


Рисунок 3 – реазультат работы алгоритма

1. В алгоритм были внесены изменения, чтобы избежать повторений в операторе switch

switch (op) {

case 'V':

cout << "Кондиционер хочу\n";

case 'S':

cout << "Радио играть должно\n";

default:

cout << "Колеса круглые\n";

cout << "Мощный двигатель\n";

}

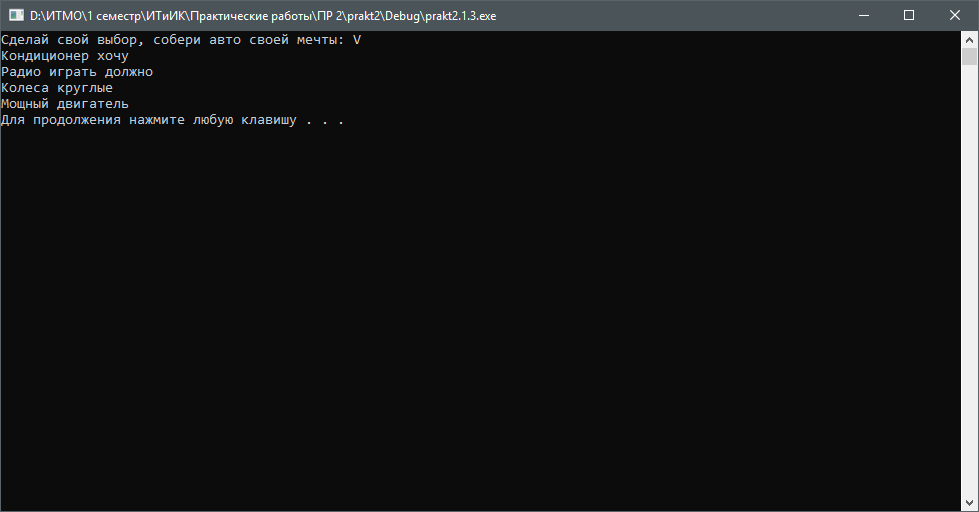


Рисунок 4 – результат работы алгоритма после внесённых изменений

1. Как можно заметить по рисунку 4, изменения не повлияли на работу алгоритма

# Упражнение 2.

Задание 1.

1. Был создан новый проект
2. Были объявлены четыре переменные вещественного типа: x – аргумент функции, x1, x2 – границы интервала, y – выходной параметр функции. Для границ интервала был реализован ввод значений с клавиатуры

double x, x1, x2, y;

cout << "x1 = ";

cin >> x1;

cout << "x2 = ";

cin >> x2;

1. Был реализован вывод заголовка таблицы, а также расчёт значений функции sin(x) для аргумента, соответствующего границам интервала и изменяющегося с шагом в 0.01. Для изменения аргуммента использовался цикл с постусловием

cout << "\tx\tsin(x)\n";

x = x1;

do {

y = sin(x);

cout << "\t" << x << "\t" << y << endl;

x = x + 0.01;

}

while (x <= x2);

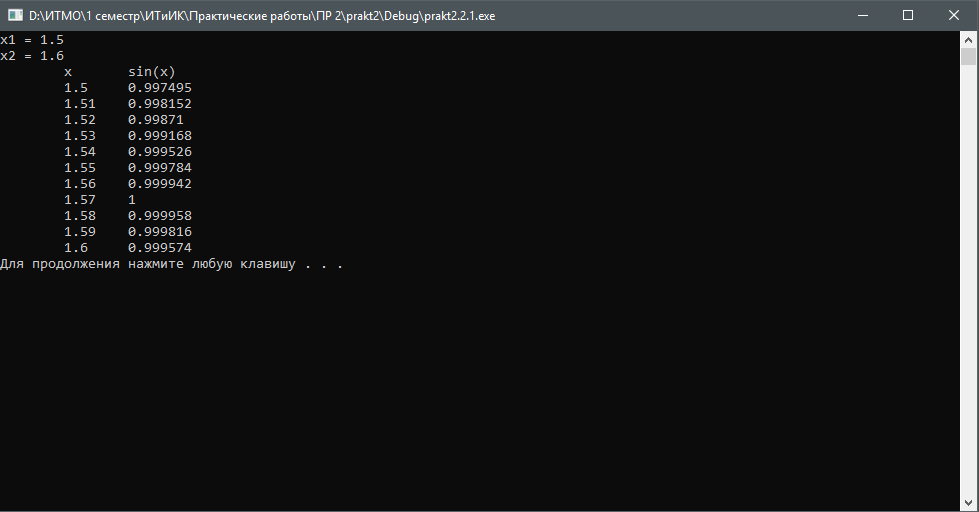


Рисунок 5 – вывод значений функции sin(x) в консоль

Задание 2.

1. Был создан новый проект
2. Были объявлены три переменные: a и b – исходные данные, temp – временная переменная для реализации алгоритма. Также для переменных a и b был реализован ввод значений

int a, b, temp;

cout << "a = ";

cin >> a;

cout << "b = ";

cin >> b;

1. Далее был реализован алгоритм Евклида для поиска НОД двух чисел и вывод результата в консоль. В алгоритме Евклида используется цикл с предусловием

temp = a;

while (temp != b) {

a = temp;

if (a < b) {

temp = a;

a = b;

b = temp;

}

temp = a - b;

a = b;

}

cout << "НОД = " << b << endl;

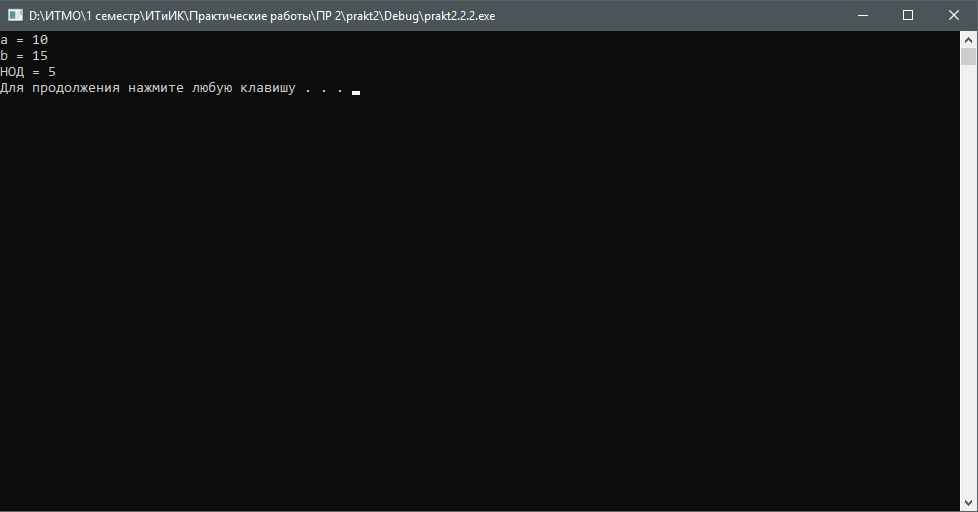


Рисунок 6 – вывод НОД двух чисел в консоль

Задание 3.

1. Задача первого задания была реализована с использованием цикла с предусловием. Изменённый цикл:

while (x <= x2) {

y = sin(x);

cout << "\t" << x << "\t" << y << endl;

x = x + 0.01;

}

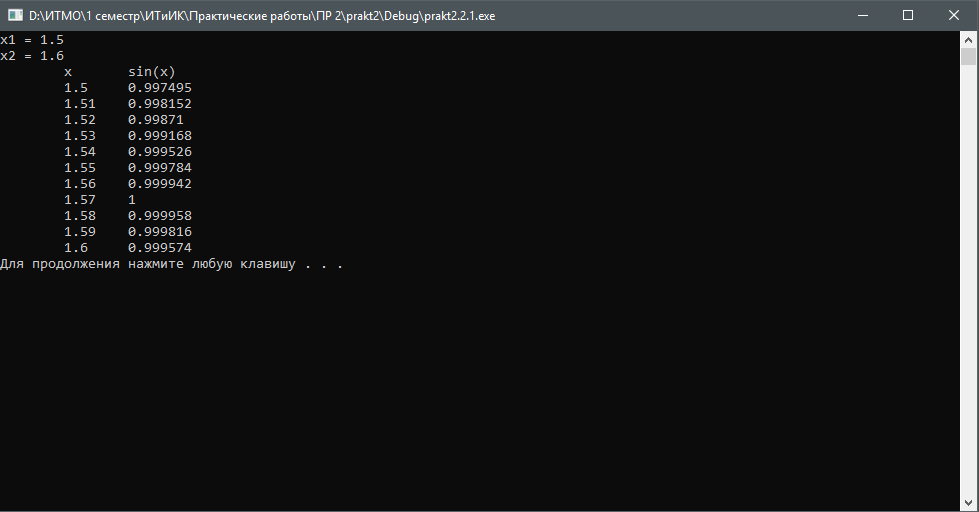


Рисунок 7 – изменение цикла не повлияло на работу алгоритма

1. Задача второго задания была реализована с использованием цикла с постусловием. Изменённый цикл:

do {

a = temp;

if (a < b) {

temp = a;

a = b;

b = temp;

}

temp = a - b;

a = b;

} while (temp != b);

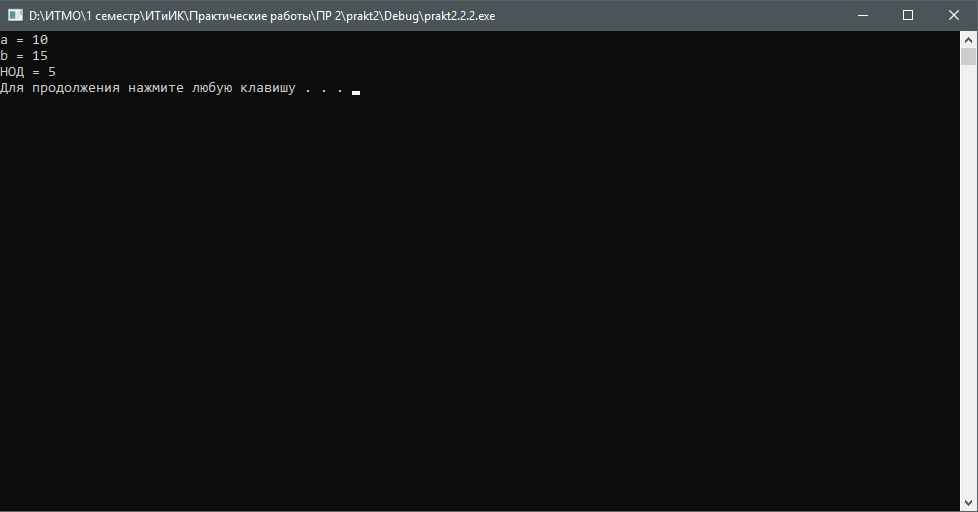


Рисунок 8 – изменение цикла не повлияло на работу алгоритма

# Упражнение 3.

1. Был реализован ввод значений переменных k и m с клавиатуры. Объявлена переменная sum для хранения суммы элементов, удовлетворяющих условию. Для корректной работы программы необходимо присвоить переменной sum значение 0

int k, m;

int sum = 0;

cout << "k = ";

cin >> k;

cout << "m = ";

cin >> m;

1. Для нахождения суммы элементов, находящихся на интервале [1; k] ∧ [m; 100] был использован цикл for. Его реализация, а также реализация вывода суммы элементов в консоль:

for (int i = 1; i <= 100; i++) {

if (i > k && i < m) {

continue;

}

sum += i;

}

cout << "sum = " << sum << endl;

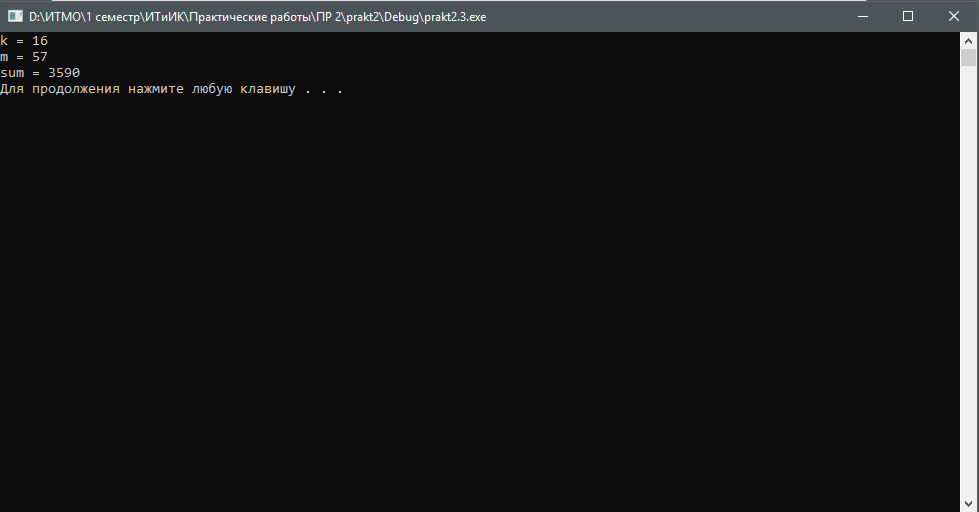


Рисунок 9 – вывод суммы элементов в консоль

# Упражнение 4.

1. Был выбран вариант мишени под номером 2



Рисунок 10 – выбранная мишень

1. Были объявлены переменные shotCount – желаемое кол-во выстрелов, x и y – координаты совершённого выстрела, score – кол-во набранных стрелком очков. shotCount и score имеют тип int. x и y имеют тип double. Был реализован ввод кол-ва желаемых выстрелов с клавиатуры

int shotCount;

double x, y;

int score = 0;

cout << "Введите желаемое кол-во выстрелов:\n";

cin >> shotCount;

1. Ввод координат выстрелов был реализован в цикле for. Конструкция if – else позволила реализовать подсчёт кол-ва заработанных очков

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

cout << "Введите координаты " << i + 1 << "-го выстрела\n";

cout << "x = ";

cin >> x;

cout << "y = ";

cin >> y;

if (pow(x, 2) + pow(y, 2) <= 1) {

score += 10;

}

else if (pow(x, 2) + pow(y, 2) <= 4) {

score += 5;

}

else if (pow(x, 2) + pow(y, 2) <= 9) {

score += 1;

}

}

1. После всех выстрелов в консоль выводится кол-во заработанных очков, а также характеристика стрелка

cout << "Ты настрелял " << score << " очков\n";

if (score == shotCount \* 10) {

cout << "Ты настоящий снайпер!" << endl;

}

else if (score > shotCount \* 1) {

cout << "Ты неплохо стреляешь, но тебе стоит потренироваться больше" << endl;

}

else {

cout << "Да ты мазила! В следующий раз будь более сосредоточенным" << endl;

}

1. Программа была собрана и протестирована

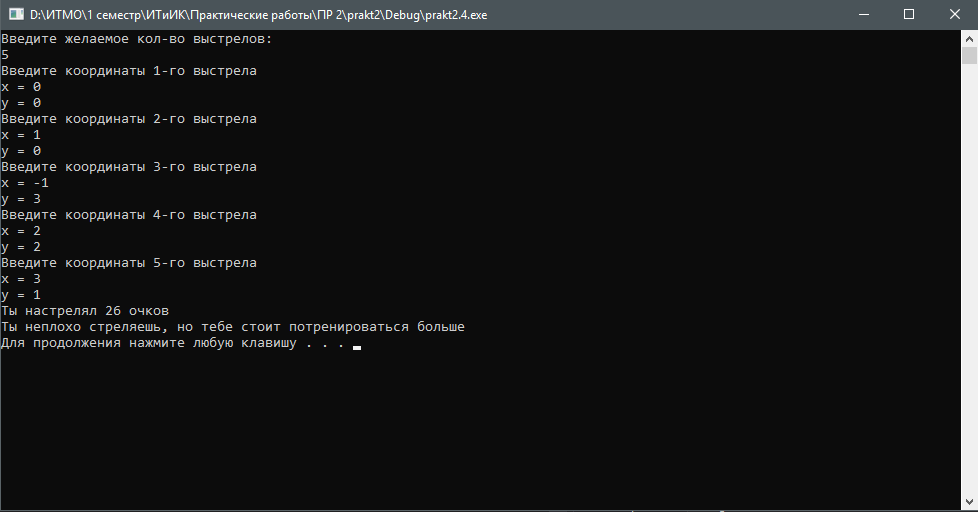


Рисунок 11 – результат работы программы

1. Были выполнены дополнительные задания
2. «Стрельба вслепую» была реализована случайным смещением центра мишени на 1 ед. влево, вправо, вниз или вверх. Центр мишени может не смещаться, что предусмотрено в программе. Случайное число генерируется функцией rand(), интервал масштабируется операцией %. Чтобы центр мог сдвигаться влево или вниз, от полученного случайного числа отнимается 1.

int centerX, centerY;

centerX = rand() % 3 - 1;

centerY = rand() % 3 - 1;

1. Также изменения были внесены в конструкциях if – else, определяющих, сколько очков зарабатывает стрелок

if (pow(x - centerX, 2) + pow(y - centerY, 2) <= 1) {

score += 10;

}

else if (pow(x - centerX, 2) + pow(y - centerY, 2) <= 4) {

score += 5;

}

else if (pow(x - centerX, 2) + pow(y - centerY, 2) <= 9) {

score += 1;

}

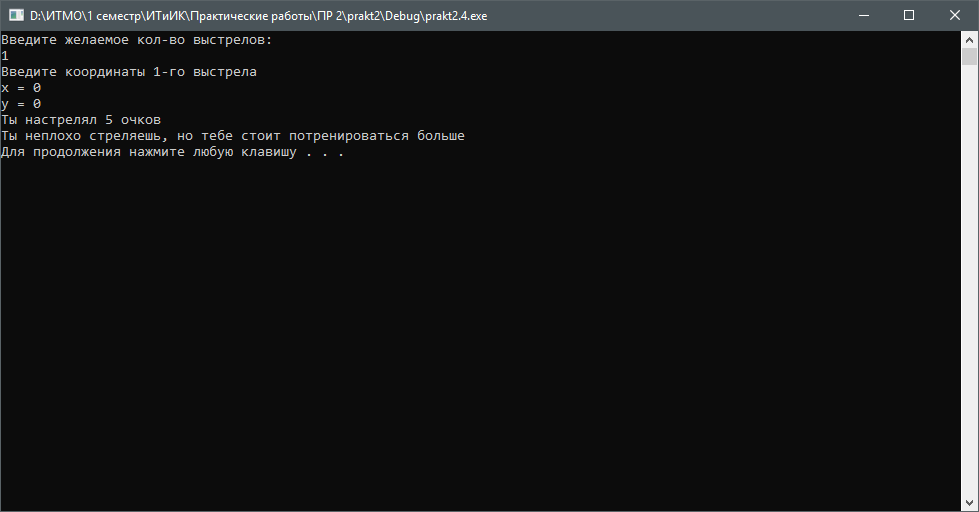


Рисунок 12 – попадание в центр координат не гарантирует попадание в центр мишени

1. Была реализована случайная помеха при стрельбе. С вероятностью 20% во время выстрела может произойти резкий порыв ветра, из-за которого стрелок не попадёт в мишень. Для этого была объявлена переменная isWindy типа bool

bool isWindy;

1. Переменная isWindy получает значение при каждом выстреле. Условие для получения значения задаётся в цикле for

isWindy = (rand() % 5 == 2) ? true : false;

1. Также были внесены изменения в конструкции if – else. Теперь попасть в мишень можно только при условии, что isWindy имеет значение false

if ((pow(x - centerX, 2) + pow(y - centerY, 2) <= 1) && !isWindy) {

score += 10;

}

else if ((pow(x - centerX, 2) + pow(y - centerY, 2) <= 4) && !isWindy) {

score += 5;

}

else if ((pow(x - centerX, 2) + pow(y - centerY, 2) <= 9) && !isWindy) {

score += 1;

}

1. Для упрощения отладки программы в консоль выводится сообщение, если произошол порыв ветра

if (isWindy) {

cout << "Подул сильный ветер\n";

}

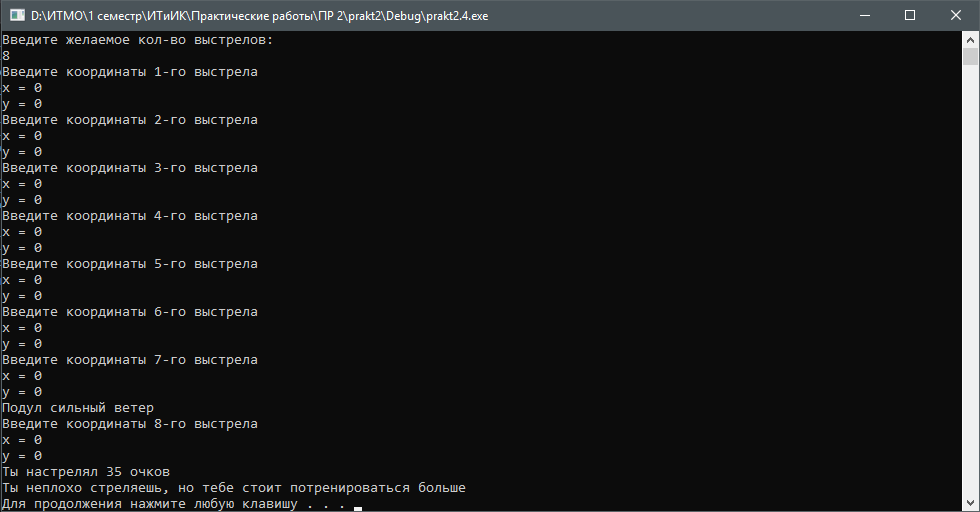


Рисунок 13 – во время 7-го выстрела произошёл порыв ветра, из-за которого случился промах

1. Листинг готовой программы:

#include "pch.h"

#include <iostream>

#include <math.h>

using namespace std;

int main()

{

setlocale(LC\_ALL, "Russian");

int shotCount;

double x, y;

int centerX, centerY;

bool isWindy;

centerX = rand() % 3 - 1;

centerY = rand() % 3 - 1;

int score = 0;

cout << "Введите желаемое кол-во выстрелов:\n";

cin >> shotCount;

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

cout << "Введите координаты " << i + 1 << "-го выстрела\n";

cout << "x = ";

cin >> x;

cout << "y = ";

cin >> y;

isWindy = (rand() % 5 == 2) ? true : false;

if ((pow(x - centerX, 2) + pow(y - centerY, 2) <= 1) && !isWindy) {

score += 10;

}

else if ((pow(x - centerX, 2) + pow(y - centerY, 2) <= 4) && !isWindy) {

score += 5;

}

else if ((pow(x - centerX, 2) + pow(y - centerY, 2) <= 9) && !isWindy) {

score += 1;

}

if (isWindy) {

cout << "Подул сильный ветер\n";

}

}

cout << "Ты настрелял " << score << " очков\n";

if (score == shotCount \* 10) {

cout << "Ты настоящий снайпер!" << endl;

}

else if (score > shotCount \* 1) {

cout << "Ты неплохо стреляешь, но тебе стоит потренироваться больше" << endl;

}

else {

cout << "Да ты мазила! В следующий раз будь более сосредоточенным" << endl;

}

system("pause");

return 0;

}

# Вывод

Были получены новые навыки работы с ЯП C++. Были использованы условные операторы, рассмотрены циклы for и while, их различия.Для закрепления новых навыков были решены некоторые задачи.