**ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ  
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ  
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»**

Факультет компьютерных наук  
Департамент программной инженерии

**ПРОГРАММА ОПРЕДЕЛЕНИЯ КВАДРАТА ПО ЗАДАННЫМ ТОЧКАМ**

**Пояснительная записка**

Дисциплина: «Архитектура вычислительных систем»

Исполнитель:  
студент группы БПИ198  
Гудзикевич М. С.

**Москва 2020**

СОДЕРЖАНИЕ

[1. ТЕКСТ ЗАДАНИЯ 3](#_Toc55146584)

[2. ПРИМЕНЯЕМЫЕ РАСЧЕТНЫЕ МЕТОДЫ 3](#_Toc55146585)

[3. ТЕСТИРОВАНИЕ ПРОГРАММЫ 4](#_Toc55146586)

[3.1 Корректные значения 4](#_Toc55146587)

[3.2 Некорректные значения 4](#_Toc55146588)

[ИСТОЧНИКИ 5](#_Toc55146589)

[ПРИЛОЖЕНИЕ 1 6](#_Toc55146590)

[КОД ПРОГРАММЫ 6](#_Toc55146591)

1. ТЕКСТ ЗАДАНИЯ

Разработать программу, которая по координатам четырёх точек (задаются целыми без знака) решает, образуют ли заданные точки квадрат.

1. ПРИМЕНЯЕМЫЕ РАСЧЕТНЫЕ МЕТОДЫ

Программа работает по следующему алгоритму: пользователь вводит четыре пары целых чисел – координаты точек (назовём их (x1, y1), (x2, y2), (x3, y3), (x4, y4)). Далее программа вычисляет шесть квадратов расстояний (расстояния между всеми возможными парами точек):

d1 = (x1 – x2)^2 + (y1 – y2)^2

d2 = (x1 – x3)^2 + (y1 – y3)^2

d3 = (x1 – x4)^2 + (y1 – y4)^2

d4 = (x2 – x3)^2 + (y2 – y3)^2

d5 = (x2 – x4)^2 + (y2 – y4)^2

d6 = (x3 – x4)^2 + (y3 – y4)^2

Далее программа работает по следующей логике: она рассматривает три ситуации, когда точки 1 и 2 диагональны, 1 и 3 диагональны, 1 и 4 диагональны. В каждой из этих ситуаций программа проверяет, что расстояния между парами диагональных точек равны, а оставшиеся четыре расстояния между точками равны между собой (предположение, что это стороны квадрата). Если условие не выполняется или все стороны оказались равны нулю – четыре точки не образуют квадрат, иначе они образуют квадрат.

Для реализации были написаны вспомогательные процедуры:

void checkFour(int a, int b, int c, int d) – метод, который проверяет четыре значения на попарное равенство и записывает при равенстве 1 в переменную tmpBool, и 0 в противном случае

void sqrDist(int x1, int y1, int x2, int y2) – метод, который высчитывает квадрат расстояния между двумя заданными точками. Результат записывается в переменную tmpDist.

Отдельное внимание уделено тому, что программа ограничивает ввод пользователя. Программа просит на вход пары целых чисел - координаты в пределах 1000 по модулю (предел расширяем при желании). Это необходимо для обработки некорректных ситуаций и переполнения при расчёте расстояний. Самый простой контрпример, если не ограничивать ввод, – координаты квадрата со стороной 2^16. При расчёте расстояний 2^16 возводится в квадрат и зануляется (из-за размера типа данных) – тогда программа скажет, что все расстояния равны 0 и это не квадрат, что неверно.

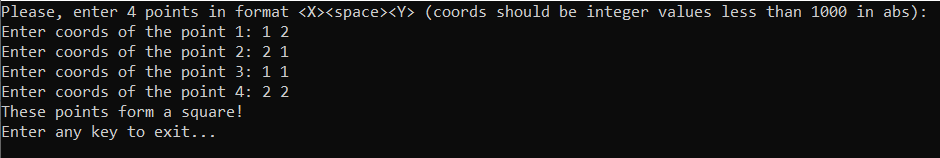
1. ТЕСТИРОВАНИЕ ПРОГРАММЫ
   1. **Корректные значения

Рисунок 1 – точки действительно образуют квадрат.

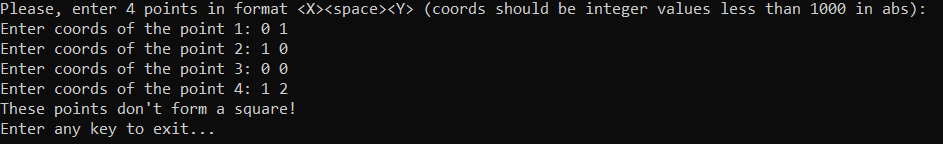
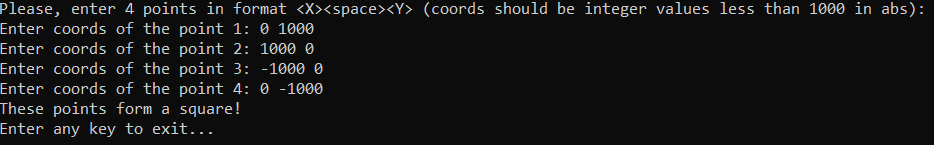


Рисунок 2 – точки не образуют квадрат.



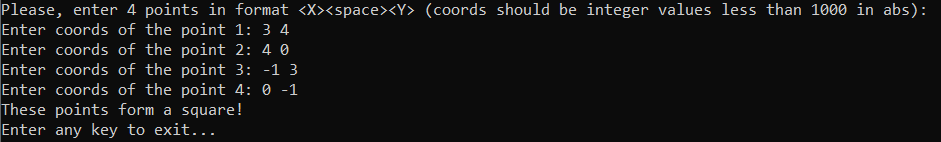
Рисунок 3 – пример на границе также работает корректно.

Рисунок 4 – менее очевидный пример, однако это действительно квадрат.

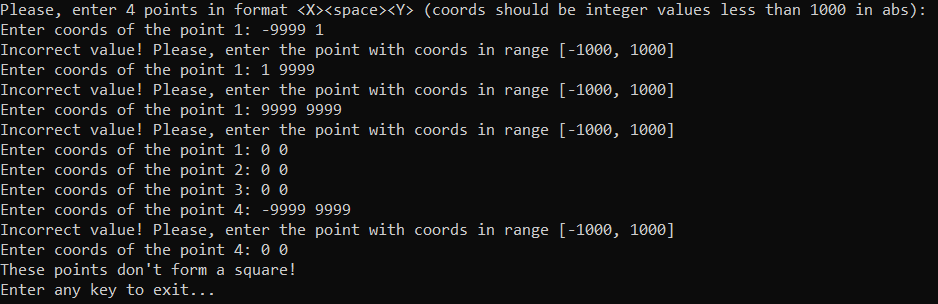
* 1. **Некорректные значения

Рисунок 5 – тут мы видим, что программа уведомляет пользователя о некорректном вводе, однако в конце концов после ввода корректных значений программа выдает верный ответ – 4 одинаковых точки не образуют невырожденный квадрат.

ИСТОЧНИКИ

1. SoftCraft, сайт по учебной дисциплине. [Электронный ресурс] http://softcraft.ru/ (дата обращения: 25.10.2020)
2. Ravesli, уроки по ассемблеру [Электронный ресурс] <https://ravesli.com/uroki-assemblera/> (дата обращения: 23.10.2020)

ПРИЛОЖЕНИЕ 1

КОД ПРОГРАММЫ

format PE console

entry start

include 'win32a.inc'

section '.data' data readable writable

formatNum db '%d %d', 0

formatN db '%d', 10, 0

formatDist db 'd[%d]^2 = %d', 10, 0

msgIsSquare db 'These points form a square!', 10, 0

msgNotSquare db 'These points don''t form a square!', 10, 0

msgIncorrect db 'Incorrect value! Please, enter the point with coords in range [-1000, 1000]', 10, 0

msgInput db 'Please, enter 4 points in format <X><space><Y> (coords should be integer values less than 1000 in abs):', 10, 0

msgEnd db 'Enter any key to exit...', 10, 0

empty db '', 10, 0

msgInputCoords db 'Enter coords of the point %d: ', 0

N dd 0 ; Iterator (from 1 to 4)

tmpX dd ? ; Temporary pointer to arrays

tmpY dd ?

arrX rd 4 ; Coords arrays

arrY rd 4

tmpx dd 0 ; Temporary vars to save input coords

tmpy dd 0

d1 dd ? ; All distances^2

d2 dd ?

d3 dd ?

d4 dd ?

d5 dd ?

d6 dd ?

tmpDist dd ? ; Temporary distance

tmpBool dd ? ; Bool var

;-------------------------------------------------------------------------------------------;

section '.code' code readable executable

start:

call readInput

call calculateDist

call defineSqr

finish:

invoke printf, msgEnd

add esp, 4

call [getch]

push 0

call [ExitProcess]

;-------------------------------------------------------------------------------------------;

proc readInput

invoke printf, msgInput

add esp, 4

mov eax, arrX

mov ebx, arrY

readLoop:

inc [N]

mov [tmpX], eax

mov [tmpY], ebx

invoke printf, msgInputCoords, [N]

add esp, 8

invoke scanf, formatNum, tmpx, tmpy

add esp, 12

cmp [tmpx], 1000 ; Make a comparsion

jg penalty

cmp [tmpx], -1000

jl penalty

cmp [tmpy], 1000

jg penalty

cmp [tmpy], -1000

jl penalty

mov eax, [tmpX]

mov ebx, [tmpY]

mov ecx, [tmpx]

mov edx, [tmpy]

mov [eax], ecx

mov [ebx], edx

add eax, 4

add ebx, 4

cmp [N], 4

jne readLoop

ret

penalty:

invoke printf, msgIncorrect

add esp, 4

dec [N]

mov eax, [tmpX]

mov ebx, [tmpY]

jmp readLoop

endp

;-------------------------------------------------------------------------------------------;

proc calculateDist ; Calculate all distances using own proc

stdcall sqrDist, [arrX], [arrY], [arrX + 4], [arrY + 4]

add esp, 16

mov eax, [tmpDist]

mov [d1], eax

stdcall sqrDist, [arrX], [arrY], [arrX + 8], [arrY + 8]

add esp, 16

mov eax, [tmpDist]

mov [d2], eax

stdcall sqrDist, [arrX], [arrY], [arrX + 12], [arrY + 12]

add esp, 16

mov eax, [tmpDist]

mov [d3], eax

stdcall sqrDist, [arrX + 4], [arrY + 4], [arrX + 8], [arrY + 8]

add esp, 16

mov eax, [tmpDist]

mov [d4], eax

stdcall sqrDist, [arrX + 4], [arrY + 4], [arrX + 12], [arrY + 12]

add esp, 16

mov eax, [tmpDist]

mov [d5], eax

stdcall sqrDist, [arrX + 12], [arrY + 12], [arrX + 8], [arrY + 8]

add esp, 16

mov eax, [tmpDist]

mov [d6], eax

ret

endp

;-------------------------------------------------------------------------------------------;

proc defineSqr ; Square define

mov eax, [d2]

cmp eax, 0

je notSqr

cmp eax, [d5]

je diag1case

jmp continue1

diag1case:

stdcall checkFour, [d1], [d4], [d6], [d3]

add esp, 16

cmp [tmpBool], 1

je isSqr

continue1:

mov eax, [d1]

cmp eax, [d6]

je diag2case

jmp continue2

diag2case:

stdcall checkFour, [d2], [d3], [d4], [d5]

add esp, 16

cmp [tmpBool], 1

je isSqr

continue2:

mov eax, [d3]

cmp eax, [d4]

je diag3case

jmp notSqr

diag3case:

stdcall checkFour, [d1], [d2], [d5], [d6]

add esp, 16

cmp [tmpBool], 1

je isSqr

jmp notSqr

isSqr:

invoke printf, msgIsSquare

add esp, 4

ret

notSqr:

invoke printf, msgNotSquare

add esp, 4

ret

endp

;-------------------------------------------------------------------------------------------;

;void checkFour(int a, int b, int c, int d), writes 1 in tmpBool if all values equal, 0 otherwise

proc checkFour

mov [tmpBool], 1

mov eax, [esp + 4]

cmp eax, [esp + 8]

jne false

mov eax, [esp + 8]

cmp eax, [esp + 12]

jne false

mov eax, [esp + 12]

cmp eax, [esp + 16]

jne false

ret

false:

mov [tmpBool], 0

ret

endp

;-------------------------------------------------------------------------------------------;

; void sqrDist(int x1, int y1, int x2, int y2), writes the sqr of distance between points (x1, y1) and (x2, y2) in tmpDist

proc sqrDist

mov eax, [esp + 4] ; x1

sub eax, [esp + 12] ; x1 - x2

imul eax ; (x1 - x2)^2

mov [tmpDist], eax ; tmpDist = (x1 - x2)^2

mov eax, [esp + 8] ; y1

sub eax, [esp + 16] ; y1 - y2

imul eax ; (y1 - y2)^2

add [tmpDist], eax ; tmpDist = (x1 - x2)^2 + (y1 - y2)^2

ret

endp

;-------------------------------------------------------------------------------------------;

section '.idata' import data readable

library kernel, 'kernel32.dll',\

msvcrt, 'msvcrt.dll'

import kernel,\

ExitProcess, 'ExitProcess'

;HeapAlloc, 'HeapAlloc',\

;GetProcessHeap, 'GetProcessHeap'

import msvcrt,\

printf, 'printf',\

scanf, 'scanf',\

getch, '\_getch'