**НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ**

**«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»**

Факультет компьютерных наук

Департамент программной инженерии

Дисциплина: «Архитектура вычислительных систем»

**Консольное приложение, которое по параметрам N=4 отрезков решает, могут ли эти отрезки являться сторонами многоугольника.**

Пояснительная записка

**Выполнил:**  
Темирханов Михаил,   
*студент гр. БПИ199.*

*01.11.2020г.*

**Москва**2020

Содержание

[1. Текст задания 2](#_Toc55160397)

[2. Описание решения 3](#_Toc55160398)

[3. Тестирование программы 4](#_Toc55160399)

[ПРИЛОЖЕНИЕ 1 7](#_Toc55160400)

[Список литературы 7](#_Toc55160401)

[ПРИЛОЖЕНИЕ 2 8](#_Toc55160402)

[Код программы 8](#_Toc55160403)

1. Текст задания

Разработать программу, которая по параметрам N=4 отрезков (задаются декартовыми координатами концов отрезков в виде машинного слов) решает, могут ли эти отрезки являться сторонами многоугольника.

1. Описание решения

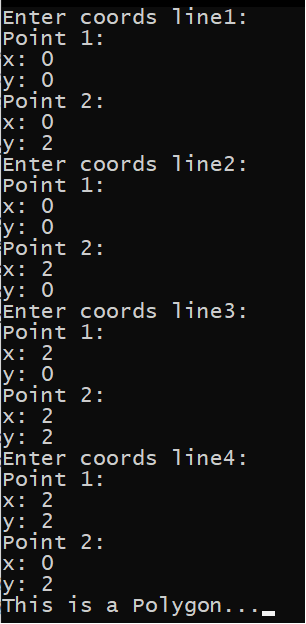
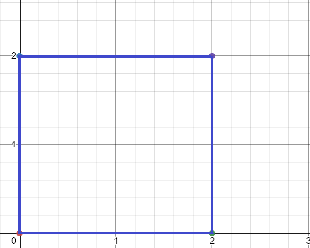
Декартова система координат — прямолинейная система координат с взаимно перпендикулярными осями на плоскости или в пространстве.

Для решения поставленной задачи было решено использовать следующий алгоритм:

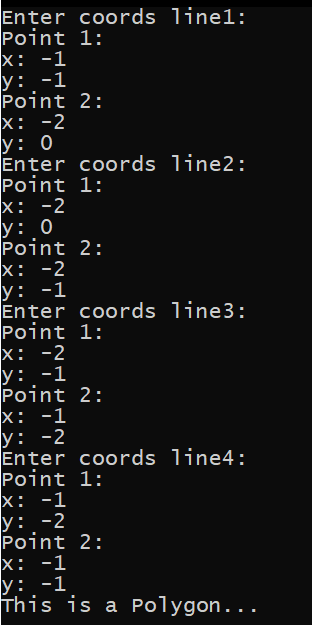
Записываем координаты точек начала и конца каждого из четырех отрезков в соответствующие переменные. Далее, проходимся по этим данным следующим образом: берём первую точку, определяющую N-ый отрезок, и сравниваем её со всеми точками других отрезков, если среди них нашлась ровно 1 точка, которая совпадает с изначальной, то запускаем идентичную проверку для второй точки, определяющий N-ый отрезок, иначе, если точка не пересекается ни с какой точкой, или пересекается с 2 и более точками, многоугольник построен быть не может, а значит выводим соответствующее сообщение.

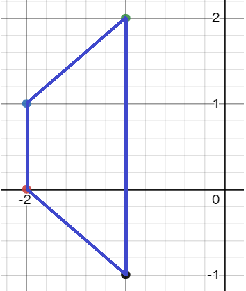
Также, определяем случаи, когда какие-либо отрезки сливаются. Для этого берём пару точек N-го отрезка и сверяем их с парами точек каждого другого отрезка. Если находится такой случай, что обе точки N-го отрезка пересекаются с точками из одного другого отрезка, то выводим сообщение о невозможности построить многоугольник.

1. Тестирование программы

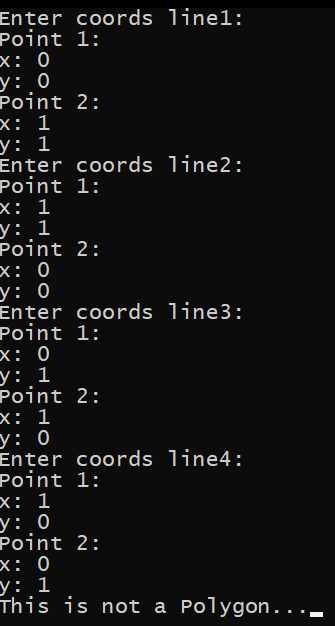
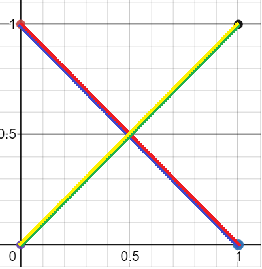
Тест 1:

*Рисунки 1, 2*

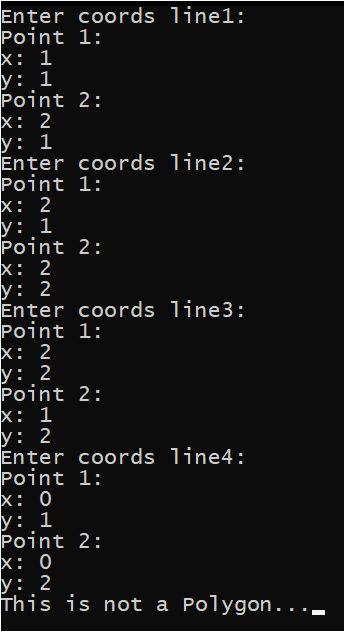
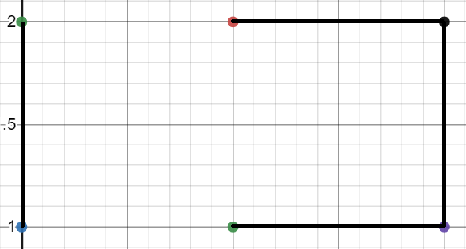


Тест 2:

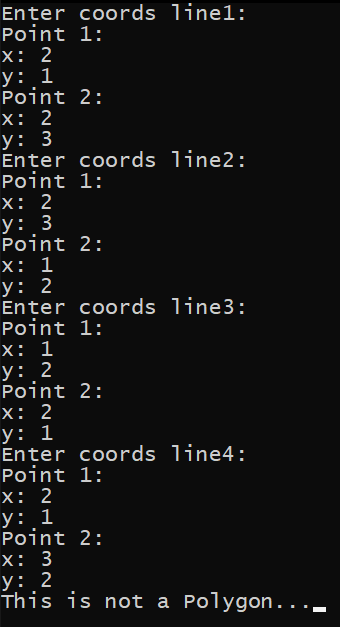
*Рисунки 3, 4*

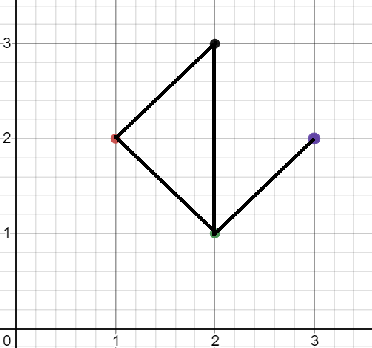
Тест 3:

*Рисунки 5, 6*

Тест 4:

*Рисунки 7, 8*

Тест 5:



*Рисунки 9, 10*

ПРИЛОЖЕНИЕ 1

Список литературы

1. [http://softcraft.ru/] (30.10.2020)
2. [https://fasmworld.ru/spravochnik-komand/] (31.10.2020)
3. [https://flatassembler.net/docs.php] (01.11.2020)
4. [https://en.wikipedia.org/wiki/Assembler] (01.11.2020)
5. [https://ru.wikipedia.org/wiki/Прямоугольная\_система\_координат] (01.11.2020)

ПРИЛОЖЕНИЕ 2

Код программы

format PE console

include 'win32a.inc'

entry start

section '.data' data readable writable

;Темирханов Михаил БПИ199 Вариант 22

;Условие:

;Разработать программу, которая по параметрам

;N=4 отрезков (задаются декартовыми координатами концов отрезков в виде машинного слоя)

;решает, могут ли эти отрезки являться сторонами многоугольника

strElemInArray db '[%d] = ', 0

strIntSc db '%d', 0

strNmb db '%d, ', 0

strBeginBracketArr db '[', 0

strEndBracketArr db ']', 10, 0

strLineEnter db 'Enter coords line%d:', 10, 0

strXCoordsArray db 'X coords: ', 0

strYCoordsArray db 'Y coords: ', 0

strPoint1 db 'Point 1:', 10, 0

strPoint2 db 'Point 2:', 10, 0

xStr dd 'x: ', 0

yStr dd 'y: ', 0

strTypeInXCoords db 'Type in X coords for all the line segments points: ',10, 0

strTypeInYCoords db 'Type in Y coords for all the line segments points: ',10, 0

strBreakMsg db 'This is not a Polygon...', 0

strMsg db 'This is a Polygon...', 0

pointStack dd ?

xCoordsArray rd 8

yCoordsArray rd 8

arrayXSize dd 8

arrayYSize dd 8

tmp dd ?

tmpArrB dd ?

aFirst dd ?

aLast dd ?

stackPointer dd ?

x dd ?

y dd ?

NULL = 0

section '.code' code readable executable

;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Main()\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

start:

mov [stackPointer], esp

push xCoordsArray

push yCoordsArray

push 1

call readLine

push xCoordsArray

push yCoordsArray

push 2

call readLine

push xCoordsArray

push yCoordsArray

push 3

call readLine

push xCoordsArray

push yCoordsArray

push 4

call readLine

push xCoordsArray

push yCoordsArray

call checkDots

add esp, 8

jmp finish

finish:

call [getch]

mov esp, [stackPointer]

stdcall [ExitProcess], 0

;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Main()\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

checkDots:

push ecx

push ebp

mov ebp, esp

sub esp, 8

;Локальные перменные

line1 equ ebp-4

lineI equ ebp-8

;Аргументы

yArrPtr3 equ ebp+12

xArrPtr3 equ ebp+16

mov [lineI], dword 0

checkDotsLoop:

cmp [lineI], dword 4

jge endCheckDotsLoop

mov ecx, [lineI]

imul ecx, 2

push dword [xArrPtr3]

push dword [yArrPtr3]

push dword [lineI]

push ecx

call findPair

add esp, 16

mov [line1], eax

mov ecx, [lineI]

imul ecx, 2

inc ecx

push dword [xArrPtr3]

push dword [yArrPtr3]

push dword [lineI]

push ecx

call findPair

add esp, 16

cmp eax, [line1]

je myBreak

inc dword [lineI]

jmp checkDotsLoop

endCheckDotsLoop:

push strMsg

call [printf]

endCheckDots:

mov esp, ebp

pop ebp

pop ecx

ret

;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

;Сравнивает две точки получая их индексы и массивы координат

equals:

push ebx

push ecx

push edx

push ebp

mov ebp, esp

sub esp, 16

;Локальные переменные

pointPtr1 equ ebp-4

pointPtr2 equ ebp-8

;Аргументы:

pointI2 equ ebp+20

pointI1 equ ebp+24

yPtr1 equ ebp+28

xPtr1 equ ebp+32

mov ecx, [pointI1] ;получаем индекс первой точки

imul ecx, 4

mov edx, [xPtr1] ;edx = указатель на массив x

mov [pointPtr1], edx ;pointPtr1 = xPtr1

add [pointPtr1], ecx ;сохраняем ссылку на координату x первой точки

mov ecx, [pointI2] ;получаем индекс второй точки

imul ecx, 4

mov edx, [xPtr1] ;edx = указатель на массив x

mov [pointPtr2], edx ;pointPtr2 = xPtr1

add [pointPtr2], ecx ;сохраняем ссылку на координату x второй точки

mov edx, [pointPtr2] ;записываем в edx указатель на координату x второй точки

mov ecx, [pointPtr1] ;записываем в edx указатель на координату x первой точки

mov ebx, [edx] ;получаем значение координаты x второй точки

cmp ebx, [ecx] ;сравниваем x координаты точек

jne notEqu

mov ecx, [pointI1] ;получаем индекс первой точки

imul ecx, 4

mov edx, [yPtr1] ;edx = указатель на массив y

mov [pointPtr1], edx ;pointPtr1 = yPtr1

add [pointPtr1], ecx ;сохраняем ссылку на координату y первой точки

mov ecx, [pointI2] ;получаем индекс второй точки

imul ecx, 4

mov edx, [yPtr1] ;edx = указатель на массив y

mov [pointPtr2], edx ;pointPtr2 = yPtr1

add [pointPtr2], ecx ;сохраняем ссылку на координату y второй точки

mov edx, [pointPtr2] ;записываем в edx указатель на координату y второй точки

mov ecx, [pointPtr1] ;записываем в edx указатель на координату y первой точки

mov ebx, [edx] ;получаем значение координаты x второй точки

cmp ebx, [ecx] ;сравниваем y координаты точек

jne notEqu

mov eax, 1

jmp endEqu

notEqu:

mov eax, 0

jmp endEqu

endEqu:

mov esp, ebp

pop ebp

pop edx

pop ecx

pop ebx

ret

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

findPair:

push ecx

push edx

push ebp

mov ebp, esp

sub esp, 12

;Аргументы

pointInd equ ebp+16

lineNum equ ebp+20

yArrPtr2 equ ebp+24

xArrPtr2 equ ebp+28

;Локальные переменные

pairLine equ ebp-4

pairI equ ebp-8

lineOther equ ebp-12

mov [lineOther], dword 4

mov [pairI], dword 0

findPairLoop:

cmp [pairI], dword 8

jge endFindPairLoop

mov ecx, [pointInd]

cmp ecx, [pairI]

je incPairI ;pairI != pointInd, то продолжаем иначе переходим к следубщей итерации цикла

push dword [xArrPtr2]

push dword [yArrPtr2]

push dword [pointInd]

push dword [pairI]

call equals

add esp, 16

cmp eax, 1 ;point[pointInd] == point[pairI]

jne incPairI

cmp [lineOther], dword 4

jne myBreak

mov ecx, [pairI]

mov eax, ecx

mov edx, 0

mov ecx, 2

div ecx

mov [lineOther], eax ;lineNum = pairI / 2

incPairI:

inc dword [pairI] ;pairI++

jmp findPairLoop

endFindPairLoop:

mov ecx, [lineOther]

cmp ecx, [lineNum]

je myBreak

cmp ecx, 4

je myBreak

mov eax, [lineOther] ;выводим номер строки подходяшей точки в качестве результата

endFindPair:

mov esp, ebp

pop ebp

pop edx

pop ecx

ret

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

readPoint:

push ecx

push edx

push ebp

mov ebp, esp

sub esp, 8

;Аргументы

xArrPtr4 equ ebp+16

yArrPtr4 equ ebp+20

index equ ebp+24

;Локальные переменные

xPtr equ ebp-4

yPtr equ ebp-8

push xStr

call [printf]

push x

push strIntSc

call [scanf]

add esp, 8

push yStr

call [printf]

push y

push strIntSc

call [scanf]

add esp, 8

mov ecx, [xArrPtr4]

mov [xPtr], ecx

mov ecx, [index]

imul ecx, 4

add [xPtr], ecx

mov ecx, [xPtr]

mov edx, [x]

mov [ecx], edx

mov ecx, [yArrPtr4]

mov [yPtr], ecx

mov ecx, [index]

imul ecx, 4

add [yPtr], ecx

mov ecx, [yPtr]

mov edx, [y]

mov [ecx], edx

endReadPoint:

mov esp, ebp

pop ebp

pop edx

pop ecx

ret

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

readLine:

push ecx

push ebp

mov ebp, esp

sub esp, 4

;Локальные переменные

pIndex equ ebp-4

;Аргументы

lineNum equ ebp+12

yArrPtr4 equ ebp+16

xArrPtr4 equ ebp+20

push dword [lineNum]

push strLineEnter

call [printf]

add esp, 8

mov ecx, [lineNum]

dec ecx

imul ecx, 2

mov [pIndex], ecx

push strPoint1

call [printf]

add esp, 4

push dword [pIndex]

push dword [yArrPtr4]

push dword [xArrPtr4]

call readPoint

add esp, 12

push strPoint2

call [printf]

add esp, 4

inc dword [pIndex]

push dword [pIndex]

push dword [yArrPtr4]

push dword [xArrPtr4]

call readPoint

add esp, 12

endReadLine:

mov esp, ebp

pop ebp

pop ecx

ret

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

myBreak:

push strBreakMsg

call [printf]

jmp finish

section '.idata' data readable import

library kernel, 'kernel32.dll',\

msvcrt, 'msvcrt.dll'

import kernel,\

ExitProcess, 'ExitProcess'

import msvcrt,\

printf, 'printf',\

scanf, 'scanf',\

getch, '\_getch'