ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
“ВЫСШАЯ ШКОЛА ЭКОНОМИКИ”

Факультет компьютерных наук

Образовательная программа бакалавриата “Программная инженерия”

Микропроект 1 по дисциплине “Архитектура вычислительных систем”

Пояснительная записка

Выполнил

Студент группы БПИ191

Дадугин Егор Артемович

Москва 2020

Оглавление

1. Введение

1.1 Текст задания

Разработать программу, которая по параметрам N>3 отрезков (задаются как декартовы координаты концов отрезков в виде целых чисел) решает, могут ли эти отрезки являться сторонами многоугольника.

1.2 Алгоритм программы

Программа состоит из трех сегментов. Сначала пользователь осуществляет ввод количества отрезков и координаты их начал и концов, причем сначала по оси абсцисс для всех отрезков, а затем по оси ординат. После этого программа сопоставляет координаты точек, ищет совпадения и определяет могут ли отрезки быть сторонами многоугольника.

2. Входные и выходные данные

2.1 Входные данные

На вход программа получает целое число больше 3 – количество отрезков, а затем для каждого отрезка координаты его начала и конца, причем сначала по оси абсцисс для всех отрезков, а затем по оси ординат.

2.2 Выходные данные

На выходе программа выводит сообщение о том могут ли заданные отрезки являться сторонами многоугольника или нет.

3. Переменные

vec_size - размер вектора
x1 - абсцисса
y1 - ордината
i - индекс элемента
matches - количество совпадений
prevMatches – количество совпадений после прошлой итерации
tmp1 - переменная для восстановления
tmp2 - переменная для восстановления
vecX - вектор абсцисс
vecY - вектор ординат

4. Тестирование программы

4.1 Ввод 4 отрезков с концами (0,0) (1,0), (1,0) (1,1), (1,1) (0,1), (0,1) (0,0) образующих квадрат

Программа выводит сообщение о том что, из заданных отрезков можно составить многоугольник.

4.2 Ввод числа отрезков меньше 4

Программа выводит сообщение о некорректном вводе.

4.3 Ввод 5 отрезков с концами (0,1) (1,0), (1,0) (1,1), (5,4) (6,5), (1,6) (1,7), (2,1) (3,3) не образующих многоугольник

```
Enter number of sections: 5
New section input
Enter x1: 0
Enter x2: 1
New section input
Enter x1: 0
Enter x2: 1
New section input
Enter x1: 1
Enter x2: 2
New section input
Enter x1: 2
Enter x2: 5
New section input
Enter x1: 0
Enter x2: 6
New section input
Enter y1: 1
Enter y2: 0
New section input
Enter y1: 1
Enter y2: 1
New section input
Enter y1: 1
Enter y2: 1
New section input
Enter y1: 1
Enter y2: 1
New section input
Enter y1: 1
Enter y2: 1
Sections do not create polygon
```

Программа выводит сообщение о том, что из заданных отрезков нельзя составить многоугольник.

4.4 Ввод 5 отрезков с концами (0,1) (1,0), (0,1) (1,1), (1,1) (2,1), (2,1) (5,1), (0,1) (6,1) не образующих многоугольник

```
Enter number of sections: 5
New section input
Enter x1: 0
Enter x2: 1
New section input
Enter x1: 1
Enter x2: 1
New section input
Enter x1: 5
Enter x2: 6
New section input
Enter x1: 1
Enter x2: 1
New section input
Enter x1: 2
Enter x2: 3
New section input
Enter y1: 1
Enter y2: 0
New section input
Enter y1: 0
Enter y2: 1
New section input
Enter y1: 4
Enter y2: 5
New section input
Enter y1: 6
Enter y2: 7
New section input
Enter y1: 1
Enter y2: 3
Sections do not create polygon
```

В данном случае хитрость состоит в том, что точка (0,1) встречается три раза. Программа выводит сообщение о том, что из заданных отрезков нельзя составить многоугольник.

4.5 Ввод 4 отрезков с концами (0,0) (0,-1), (0,-1) (-1,-1), (-1,-1) (-1,0), (-1,0) (0,0) образующих квадрат

Данный тест демонстрирует корректность работы программы при вводе отрицательных координат. Программа выводит сообщение о том, что из заданных отрезков можно составить многоугольник.

5. Список использованных источников

1) Flat Assembler 1.64 – мануал программера [Электронный ресурс] URL: http://flatassembler.narod.ru/fasm.htm (Режим доступа: свободный)

ПРИЛОЖЕНИЕ 1

ТЕКСТ ПРОГРАММЫ

```
;Дадугин Егор Артемович
;БПИ191
;Вариант 12
;Разработать программу, которая по параметрам N>3 отрезков
;(задаются как декартовы координаты концов отрезков в виде целых чисел)
;решает, могут ли эти отрезки являться сторонами многоугольника.

format PE console
entry start

include 'win32a.inc'

;--------------------------------------------------------------------------
section '.data' data readable writable

      strNumLine   db 'Enter number of sections: ', 0
      strIncorNum db 'Incorrect number of sections!', 10, 0
      strIsPolygon db 'Sections create polygon', 10, 0
      strIsNotPolygon db 'Sections do not create polygon', 10, 0
      strNewSection  db 'New section input' ,10, 0
      strScanX1   db 'Enter x1: ', 0
      strScanY1   db 'Enter y1: ', 0
      strScanX2   db 'Enter x2: ', 0
      strScanY2   db 'Enter y2: ', 0
      strScanInt   db '%d', 0

      vec_size     dd 0      ;размер вектора
      x1          dd ?      ;абсцисса
      y1          dd ?      ;ордината
      i          dd ?
      matches      dd 0      ;количество совпадений
      prevMatches  dd 0      ;количество совпадений после прошлой итерации
      tmp1         dd ?
      tmp2         dd ?
      vecX         rd 100    ;вектор абсцисс
      vecY         rd 100    ;вектор ординат

;--------------------------------------------------------------------------
section '.code' code readable executable
start:
     call VectorInput        ;ввод данных
     call DoubleSize         ;удвоение значения размера вектора
     call CheckPolygon       ;проверка на возиожность составления многоугольника
     mov eax, [matches]      ;удвоение количества совпадений
     mov ecx, 2
     mul ecx
     cmp eax, [vec_size]     ;сравнение совпадений и необходимого для составления
многоугольника числа
     je IsPolygon            ;вывод о возможности составления многоугольника
     jmp IsNotPolygon        ;вывод о невозможности составления многоугольника
```

```
finish:
	call [getch]

	push 0
	call [ExitProcess]

;-----------------------------------------------------------------------
VectorInput:
	push strNumLine
	call [printf]
	add esp, 4

	push vec_size
	push strScanInt
	call [scanf]		;ввод количества отрезков
	add esp, 8

	mov eax, [vec_size]
	cmp eax, 3		;количество отрезков должно быть больше 3
	jg  getVector

	push strIncorNum
	call [printf]
	call [getch]
	push 0
	call [ExitProcess]

getVector:
	xor ecx,ecx
	mov ebx, vecX
coordsXLoop:			;цикл для ввода абсцисс
	mov [i], ecx
	cmp ecx, [vec_size]
	jge coodsYprep

	push strNewSection
	call [printf]
	add esp,4

	push strScanX1
	call [printf]
	add esp,4
	push ebx
	push strScanInt
	call [scanf]
	add esp, 8
	add ebx, 4

	push strScanX2
	call [printf]
	add esp,4
	push ebx
	push strScanInt
```

```
        call [scanf]
        add esp, 8
        add ebx, 4

        mov ecx, [i]
        inc ecx
        jmp coordsXLoop

coodsYprep:
        xor ecx, ecx
        mov ebx, vecY
coordsYLoop:                ;цикл для ввода ординат
        mov [i], ecx
        cmp ecx, [vec_size]
        jge endCoordsInput

        push strNewSection
        call [printf]
        add esp,4

        push strScanY1
        call [printf]
        add esp,4
        push ebx
        push strScanInt
        call [scanf]
        add esp, 8
        add ebx, 4

        push strScanY2
        call [printf]
        add esp,4
        push ebx
        push strScanInt
        call [scanf]
        add esp, 8
        add ebx, 4

        mov ecx, [i]
        inc ecx
        jmp coordsYLoop
endCoordsInput:
        ret
;-------------------------------------------------------------------------
CheckPolygon:
        xor ecx,ecx
        mov ebx, vecX
        mov edx, vecY
checkLoop:                  ;внешний цикл, в нем запоминаются координаты точки
        mov eax, [matches]
        mov [prevMatches], eax
        mov [i], ecx
        mov eax ,[ebx]
```

```
        mov [x1], eax
        mov eax ,[edx]
        mov [y1], eax
        mov [tmp1], ebx
        mov [tmp2], edx
        cmp ecx, [vec_size]
        jge endCheckPolygon
compareLoop:                ;внутренний цикл, в нем координаты точки сравниваются с координатами
других точек
        inc ecx
        cmp ecx, [vec_size]
        jge nextLoop

        add ebx, 4
        add edx, 4

        mov eax ,[ebx]
        cmp eax, [x1]
        jne compareLoop

        mov eax ,[edx]
        cmp eax, [y1]
        jne compareLoop

        xor eax, eax
        mov eax, [matches]      ;если нашлась эквивалентная точка прибавляем к совпадениям 2
        add eax, 2
        mov [matches], eax
        jmp compareLoop
nextLoop:
        mov ecx, [i]
        inc ecx
        mov ebx, [tmp1]
        add ebx, 4
        mov edx, [tmp2]
        add edx, 4
        mov eax, [matches]
        cmp eax, [prevMatches]  ;если у точки не нашлось эквивалентной, вычитаем из совпадений 1
        je subMatches
        jmp checkLoop
subMatches:
    sub [matches], 1
    jmp checkLoop
endCheckPolygon:
        ret

;-------------------------------------------------------------------------
IsPolygon:
        push strIsPolygon
        call [printf]           ;вывод результата
endIsPolygon:
        jmp finish
;-------------------------------------------------------------------------
```

```
IsNotPolygon:
      push strIsNotPolygon
      call [printf]         ;вывод результата
endIsNotPolygon:
      jmp finish
;------------------------------------------------------------------------
DoubleSize:
      xor ecx, ecx
      mov eax, [vec_size]
      mov ecx, 2
      mul ecx               ;удваиваем размер вектора
      mov [vec_size], eax
      xor ecx, ecx
endDoubleSize:
      ret
;-------------------------------third act - including HeapApi--------------------------

section '.idata' import data readable
   library kernel, 'kernel32.dll',\
          msvcrt, 'msvcrt.dll',\
          user32,'USER32.DLL'

include 'api\user32.inc'
include 'api\kernel32.inc'
   import kernel,\
         ExitProcess, 'ExitProcess',\
         HeapCreate,'HeapCreate',\
         HeapAlloc,'HeapAlloc'
  include 'api\kernel32.inc'
   import msvcrt,\
         printf, 'printf',\
         scanf, 'scanf',\
         getch, '_getch'
```