

ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

Факультет компьютерных наук
Департамент программной инженерии

ПРОГРАММА, КОТОРАЯ РЕШАЕТ МОГУТ ЛИ ОТРЕЗКИ
ЯВЛЯТЬСЯ СТОРОНАМИ МНОГОУГОЛЬНИКА

Пояснительная записка

Исполнитель
Студент группы БПИ 196
Сахаров Никита Денисович

Москва, 2020

Оглавление

1. Постановка задачи	2
2. Описание Алгоритма.....	3
2.1. Описание методов	3
2.1.1. InputLine	3
2.1.2. LinesInput.....	3
2.1.3. CanConstuctPolygon	3
2.1.4. LenLinesOutput.....	3
3. Приложение 1. Код программы.....	4
4. Приложение 2 Тестирование программы.....	9

1. Постановка задачи

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.

2. Описание Алгоритма

Для решения этой задачи был выбран подход проверки математического критерия составления многоугольника. Критерий говорит о том, что можно составить из данных отрезков многоугольник тогда и только тогда, когда большая его сторона меньше суммы остальных. Используя средства FPU, я подсчитал для каждого отрезка его длину, нашел максимальный отрезок и сумму всех длин. После чего вычел из суммы максимальную длину и сравнил разность с ней же.

2.1. Описание методов

2.1.1. InputLine

Описание: Метод считывает у пользователя концы отрезка, проверяет валидность ввода и подсчитывает квадрат длины этого отрезка.

Возвращаемое значение: Метод записывает концы отрезка в глобальные переменные `x1`, `y1`, `x2`, `y2` и пишет в глобальную переменную `lenLine` квадрат его длины.

2.1.2. LinesInput

Описание: Метод вызывает считывание каждого отрезка, извлекает корень и каждой длины, попутно ищет максимум и считает сумму всех длин.

Возвращаемое значение: Метод записывает максимум и сумму в глобальные переменные `max` и `sumOfLines`, а также формирует массив длин для вывода справки в конце программы.

2.1.3. CanConstuctPolygon

Описание: Метод определяет по сумме и максимуму можно ли сложить многоугольник и выводит соответствующее сообщение.

Возвращаемое значение: Строка вердикта выводится в консоль.

2.1.4. LenLinesOutput

Описание: Метод выводит максимальную длину и сумму всех длин. А так же весь массив подсчитанных длин отрезков.

Возвращаемое значение: Вывод информации об отрезках в консоль.

3. Приложение 1. Код программы

```
format PE console
entry start

include 'win32a.inc'

;Условие
;Разработать программу, которая попараметрам N>3
;отрезков (задаются как декартовы координаты концов отрезков в виде целых
чисел) решает,
;могут ли эти отрезки являться сторонами многоугольника
;(Я подходил к вам после семинара и мы с Вами решили, что проверить
математический критерий достаточно.
;Для этого я включил в свою работу взаимодействие с FPU)
;Вариант: 20
;Студент: Сахаров Никита
;Группа: БПИ196
;-----
-----

section '.data' data readable writable

    strArrSize      db 'Input count of line segments (400 >= x > 3): ', 0
    strLineFormat   db 'Format of input: ', 10, 13, ' Line[i]: x1 y1 x2
y2', 10, 0
    strIncorSize     db 'Incorrect count of lines = %d', 10, 0
    strScanInt       db '%d', 0
    strLineElemI     db 'Line [%d]: ', 0
    strScanLine      db '%d %d %d %d', 0
    strNewLine       db 10, 0
    strdouble        db ' %f', 10, 0
    strNANSize       db 'The size of the line count must be a positive
integer', 0
    strNANElem       db 'The ends of line segment must be integer between -
1000 and 1000', 10, 0
    strANS           db 'MAX = %f Summ of all segments length = %f', 10, 0
    strSuccess       db 'You can create a polygon from these line segments',
10, 0
    strFail          db 'You can not create a polygon from these line
segments, ', \
' because max legnth of line segments less than sum of others', 10, 0
    strOutLen        db 'Line[%d] legnth: %f', 10, 0
    strSeparator     db 10, '-----'
    strInfo          db 'Lines legnth info:', 10, 10, 0
    strLines         db 'Your lines:', 10, 0

    line_count       dd 0
    arrOfLen          rq 10000
    i                dd ?
    tmpb             dd ?
    x1               dd ?
    x2               dd ?
    y1               dd ?
    y2               dd ?
    sqx              dd ?
    sqy              dd ?
    lenLine          dd ?
    tmpStack         dd ?
    summOfLines      dq ?
```

```

    max            dq ?
    currLen        dq ?
;-----
-----

section '.code' code readable executable
start:
    FINIT ;intialization of FPU

    FLDZ
    FSTP [summOfLines]      ; set sum 0
    FLDZ
    FSTP [max]              ; set max 0

    call LinesInput          ; array input

    call CanConstuctPolygon ; get verdict

    call LenLinesOutput      ; print info about lines

finish:
    call [getch]

    push 0
    call [ExitProcess]

;-----
-----
LinesInput:
    mov [tmpStack], esp
    cinvoke printf, strArrSize      ; input size

    push line_count
    push strScanInt
    call [scanf]
    cmp  eax, 0                      ; check scan success
    je failGetLineToSize

    mov eax, [line_count]

    cmp eax, 3
    jle failSize

    cmp eax, 400
    jle getLines

failSize:                                ; branch for incorrect size of
array
    push [line_count]
    push strIncorSize
    call [printf]

    call [getch]

    push 0
    call [ExitProcess]
failGetLineToSize:                        ; branch for incorrect size of
array
    push strNANSize

```

```

    call [printf]

    call [getch]

    push 0
    call [ExitProcess]

getLines:
    cinvoke printf, strSeparrator
    cinvoke printf, strLineFormat ; print info about format input
    cinvoke printf, strSeparrator
    cinvoke printf, strLines
    xor ecx, ecx
    mov ebx, arrOfLen
    mov ecx, [line_count]
getLinesLoop:
    push ecx
    mov [tmpb], ebx

    mov eax, [line_count]
    sub eax, ecx
    mov [i], eax
    cinvoke printf, strLineElemI, [i] ;
    call InputLine

    FILD [lenLine]
    FSQRT ; get srtr from length of line segmet
    FST [currLen]

    FCOM [max] ; compare to max
    fstsw AX
    sahf
    jb Skip ; skip update max
    FST [max] ; update max

Skip:
    FSTP [currLen]
    FLD [summOfLines]
    FADD [currLen] ; add current length to sum
    FSTP [summOfLines] ; update sum

    mov ebx, [tmpb]

    mov eax, dword[currLen] ; send value of length to array
    mov edx, dword[currLen + 4]
    mov [ebx], eax
    mov [ebx + 4], edx

    add ebx, 8
    pop ecx
    loop getLinesLoop

    jmp endInputLines
failGetLineToArr: ; branch for fail get line
    push strNANElem
    call [printf]

    call [getch]

```

```

    push 0
    call [ExitProcess]

endInputLines:
    mov esp, [tmpStack]
    ret

;-----
-----
InputLine:

    cinvoke scanf, strScanLine, x1, y1, x2, y2    ; input ends of lint
    segment
    cmp eax, 4
    jl failGetLineToSize
    mov edx, [x1]
    cmp edx, 1000
    jg failGetLineToArr
    cmp edx, -1000                                ; check bounds
    jl failGetLineToArr

    mov edx, [y1]
    cmp edx, 1000
    jg failGetLineToArr
    cmp edx, -1000                                ; check bounds
    jl failGetLineToArr

    mov edx, [x2]
    cmp edx, 1000
    jg failGetLineToArr
    cmp edx, -1000                                ; check bounds
    jl failGetLineToArr

    mov edx, [y2]
    cmp edx, 1000
    jg failGetLineToArr
    cmp edx, -1000                                ; check bounds
    jl failGetLineToArr

    mov eax, [x1]
    sub eax, [x2]
    imul eax
    mov [sqx], eax                                ; get squire for OX axis

    mov eax, [y1]
    sub eax, [y2]
    imul eax
    mov [sqy], eax                                ; get squire for OY axis

    add eax, [sqx]                                ; get squire of length

    mov [lenLine], eax

    ret

;-----
-----
CanConstuctPolygon:
    cinvoke printf, strSeparrator
    FLD [summOfLines]                            ;
    FSUB [max]                                    ; get su, without max
    FST [currLen]

```



```

        FCOMP [max]                ; check criterion
        fstsw AX
        sahf
        jnb SuccessConsturt
FailConstruct:
        cinvoke printf, strFail
        jmp endCanConstruct        ; print fail verdict
SuccessConsturt:
        cinvoke printf, strSuccess ; print success verdict
endCanConstruct:
        ret
;-----
-----
LenLinesOutput:
        mov [tmpStack], esp

        cinvoke printf, strSeparrator
        cinvoke printf, strInfo
        cinvoke printf, strANS, dword[max], dword[max+4], dword[summOfLines],
        dword[summOfLines + 4]
        cinvoke printf, strNewLine

        xor ecx, ecx
        mov ebx, arrOfLen
OutputLoop:
        mov [tmpb], ebx
        cmp ecx, [line_count]
        je endOutputArrays
        mov [i], ecx

        cinvoke printf, strOutLen, [i], dword[ebx], dword[ebx + 4]
                                ; print current line segmet length
        mov ecx, [i]
        inc ecx

        mov ebx, [tmpb]
        add ebx, 8
        jmp OutputLoop
endOutputArrays:
        mov esp, [tmpStack]
        ret
;-----
-----

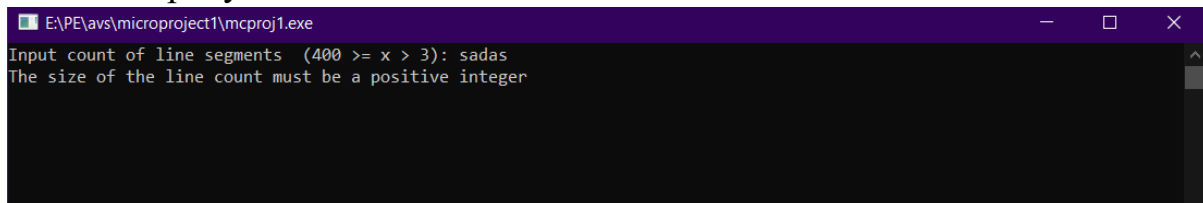
section '.idata' import data readable
library kernel, 'kernel32.dll',\
        msvcrt, 'msvcrt.dll',\
        user32, 'USER32.DLL'

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'

```

4. Приложение 2 Тестирование программы

Введем строку вместо числа:



```
E:\PE\avs\microproject1\mcproj1.exe
Input count of line segments (400 >= x > 3): sadas
The size of the line count must be a positive integer
```

Введем размер массива меньше нижней границы:



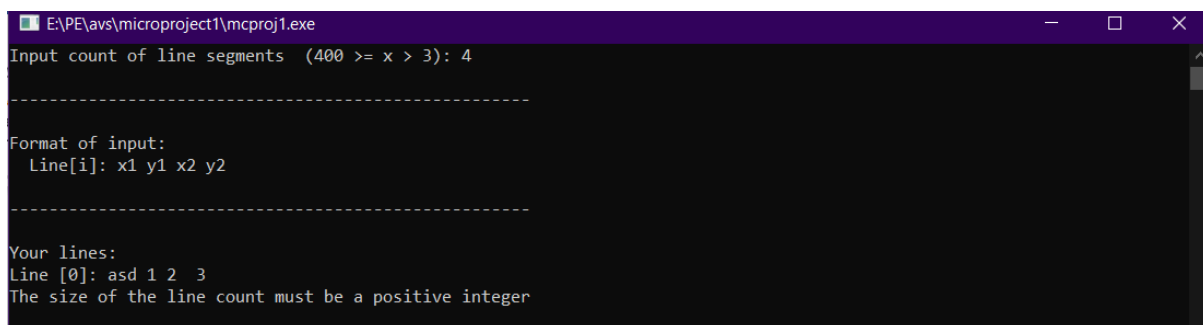
```
E:\PE\avs\microproject1\mcproj1.exe
Input count of line segments (400 >= x > 3): 3
Incorrect count of lines = 3
```

Введем размер массива больше верхней границы:



```
E:\PE\avs\microproject1\mcproj1.exe
Input count of line segments (400 >= x > 3): 401
Incorrect count of lines = 401
```

Введем строку вместо конца отрезка:



```
E:\PE\avs\microproject1\mcproj1.exe
Input count of line segments (400 >= x > 3): 4

-----
Format of input:
Line[i]: x1 y1 x2 y2
-----

Your lines:
Line [0]: asd 1 2 3
The size of the line count must be a positive integer
```

Введем конец отрезка выходящий за установленные границы:

```
E:\PE\avs\microproject1\mcproj1.exe
Input count of line segments (400 >= x > 3): 4

-----

Format of input:
Line[i]: x1 y1 x2 y2

-----

Your lines:
Line [0]: 1001 20 20 3
The ends of line segment must be integer between -1000 and 1000
```

Введем отрезки из которых нельзя сложить многоугольник:

```
E:\PE\avs\microproject1\mcproj1.exe
Input count of line segments (400 >= x > 3): 4

-----

Format of input:
Line[i]: x1 y1 x2 y2

-----

Your lines:
Line [0]: 1 1 -1 -1
Line [1]: 2 3 5 7
Line [2]: -2 5 9 3
Line [3]: 23 100 24 100

-----

You can not create a polygon from these line segments, because max legnth of line segments less than sum of others

-----

Lines legnth info:
MAX = 11.180340 Summ of all segments length = 20.008767
Line[0] legnth: 2.828427
Line[1] legnth: 5.000000
Line[2] legnth: 11.180340
Line[3] legnth: 1.000000
```

Введем отрезки из которых можно сложить многоугольник:

```
E:\PE\avs\microproject1\mcproj1.exe
Input count of line segments (400 >= x > 3): 5

-----

Format of input:
Line[i]: x1 y1 x2 y2

-----

Your lines:
Line [0]: 1 1 5 5
Line [1]: 3 4 7 8
Line [2]: 5 15 10 7
Line [3]: 10 23 1 2
Line [4]: 4 2 2 4

-----

You can create a polygon from these line segments

-----

Lines legnth info:
MAX = 22.847319 Summ of all segments length = 46.423436

Line[0] legnth: 5.656854
Line[1] legnth: 5.656854
Line[2] legnth: 9.433981
Line[3] legnth: 22.847319
Line[4] legnth: 2.828427
```

Введем много отрезков:

```
-----

You can create a polygon from these line segments

-----

Lines legnth info:
MAX = 2304.729919 Summ of all segments length = 414748.138270
Line[0] legnth: 1336.516741
```

(Данные для этого теста хранятся в репозитории на [github](#) файлы 8.txt и 8test.txt)