

МИКРОПРОЕКТ №1 ПО КУРСУ ABC

ВАРИАНТ 22

Пояснительная записка

Москва 2020

1. НАЗНАЧЕНИЕ ПРОГРАММЫ

1. Условие задания

Разработать программу, которая по параметрам трёх отрезков (задаются декартовыми координатами концов отрезков в виде машинных слов без знака) решает, могут ли являться заданные отрезки сторонами равнобедренного треугольника.

Назначение программы

По координатам трех точек — вершин треугольника, вводимыми пользователем, определять, является ли треугольник равнобедренным.

2. Описание алгоритма и функционирования программы и применяемых расчётных методов

Программа реализована в виде консольного приложения на языке FASM.

2.1 Текст программы

```
; Daria Tarasova, 197 group
format PE console
```

```
entry start
```

```
include 'win32a.inc'
include 'macro.inc'
```

```
section '.data' data readable writable
; Tools for input user numbers
digit2In      db '%d %d', 0
digitIn       db '%d', 0
; Messages for user
fail1Str      db 'These 3 vectors can not form a triangle.', 10, 0
fail2Str      db 'These 3 vectors can not form an isocleles triangle.', 10, 0
successStr    db 'These 3 vectors can form an isosceles triangle.', 10, 0
firstPoint    db 'Enter coordinates of the first point', 10, 0
secondPoint   db 'Enter coordinates of the second point', 10, 0
thirdPoint    db 'Enter coordinates of the third point', 10, 0
inputCoord    db 'Input X and Y coordinates separated by space: ', 10, 0
wrongCoord    db 'Wrong input: X and Y should be in range [-65536; 65535].', 10,
0
endMessage    db 'Press any key to exit...', 10, 0
; Tools for output of numbers
pointsStr     db '{%d, %d} {%d, %d}', 10, 0
newLine       db '', 10, 0

; Coordinates
p1x dd ?
p2x dd ?
p3x dd ?
p1y dd ?
p2y dd ?
p3y dd ?
; Temporary stack values
tmpStack dd ?
tmpCheckStack dd ?

s dd 0
a dd 0
b dd 0
c dd 0

; Procedures variables
x1 dd ?
x2 dd ?
y1 dd ?
y2 dd ?
l1 dd ?
l2 dd ?
temp dd ?
```

```

NULL = 0

;-----
section '.code' code readable executable
start:
    ; Points' coordinates input
    call getFirst
    call getSecond
    call getThird

    ; Check if vectors can form a triangle
    mov ebx, [p2y]
    sub ebx, [p3y]
    mov eax, [p1x]
    push eax
    push ebx
    call mult
    add [s], eax

    mov ebx, [p3y]
    sub ebx, [p1y]
    mov eax, [p2x]
    push eax
    push ebx
    call mult
    add [s], eax

    mov ebx, [p1y]
    sub ebx, [p2y]
    mov eax, [p3x]
    push eax
    push ebx
    call mult
    add [s], eax      ; s = p1x * (p2y - p3y) + p2x * (p3y - p1y) + p3x * (p1y
- p2y)

    cmp [s], 0      ; If s = 0 then the vectors can form a triangle
    jne checkIsosceles
    ; Can't form a triangle
    push fail1Str
    call [printf]
    call endProg

;-----
    ; Check if value is in range [0; 65535]
    checkValue:
        ; Check bottom border
        mov [tmpCheckStack], esp
        cmp eax, 65535
        jle checkBottom
        jmp wrongVal

        ; Print message about wrong input and exit program
    wrongVal:
        push wrongCoord
        call [printf]
        call endProg

        ; Check upper border
    checkBottom:

```

```

    cmp eax, -65536
    jl wrongVal
    mov esp, [tmpCheckStack]
    ret

```

```

;-----
; Input of coordinates of the first point - the end of the segment
getFirst:
    ; Prompt the user to enter values
    mov [tmpStack], esp ; memorizing the position in the stack
    push firstPoint
    call [printf]
    push inputCoord
    call [printf]
    ; Reading the entered values
    push p1y
    push p1x
    push digit2In
    call [scanf]
    ; Checking the entered value
    mov eax, [p1x]
    call checkValue
    mov eax, [p1y]
    call checkValue

    mov esp, [tmpStack] ; restoring the position in the stack
    ret

```

```

;-----
; Input of coordinates of the first point - the end of the segment
getSecond:
    ; Prompt the user to enter values
    mov [tmpStack], esp ; memorizing the position in the stack
    push secondPoint
    call [printf]
    push inputCoord
    call [printf]
    ; Reading the entered values
    push p2y
    push p2x
    push digit2In
    call [scanf]
    ; Checking the entered value
    mov eax, [p2x]
    call checkValue
    mov eax, [p2y]
    call checkValue

    mov esp, [tmpStack] ; restoring the position in the stack
    ret

```

```

;-----
; Input of coordinates of the first point - the end of the segment
getThird:
    ; Prompt the user to enter values
    mov [tmpStack], esp ; memorizing the position in the stack
    push thirdPoint
    call [printf]
    push inputCoord
    call [printf]
    ; Reading the entered values
    push p3y

```

```

push p3x
push digit2In
call [scanf]
; Checking the entered value
mov eax, [p3x]
call checkValue
mov eax, [p3y]
call checkValue

mov esp, [tmpStack] ; restoring the position in the stack
ret

```

;-----

checkIsosceles:

```

; Calculate square roots of the sides
lengthRoot p1x, p1y, p2x, p2y
mov [a], eax ; a = (p1x - p2x) ^ 2 + (p1y - p2y)

lengthRoot p1x, p1y, p3x, p3y
mov [b], eax ; b = (p1x - p3x) ^ 2 + (p1y - p3y)

lengthRoot p2x, p2y, p3x, p3y
mov [c], eax ; c = (p2x - p3x) ^ 2 + (p2y - p3y)

push [a]
push [b]
call areEqual
cmp eax, 1 ; if a = b then the vectors can form an isosceles triangles
je foundEqual

push [a]
push [c]
call areEqual
cmp eax, 1 ; if a = c then the vectors can form an isosceles triangles
je foundEqual

push [b]
push [c]
call areEqual
cmp eax, 1 ; if b = c then the vectors can form an isosceles triangles
je foundEqual

; Equal sides were not found
push fail2Str
call [printf]
call endProg

```

foundEqual:

```

push successStr
call [printf]
call endProg

```

;-----

; Ends the program

endProg:

```

push endMessage
call [printf]
call [getch]
push NULL
call [ExitProcess]

```

```

;-----
; The multiplication result is stored in eax
mult:
    mov [tmpStack], esp
    pop eax
    pop eax
    pop ebx
    xor ecx, ecx
    xor edx, edx

multLoop:
    cmp ecx, ebx
    jge endMultLoop
    add edx, eax
    inc ecx
    jmp multLoop

endMultLoop:
    mov eax, edx
    mov esp, [tmpStack]
    ret

;-----
; Stores in eax the absolute value of the top value on the stack
abs:
    mov [tmpStack], esp
    pop eax
    pop eax
    cmp eax, 0
    jl removeMinus
    mov esp, [tmpStack]
    ret
removeMinus:
    xor ebx, ebx ; ebx = 0
    sub ebx, eax
    mov eax, ebx
    mov esp, [tmpStack]
    ret

;-----
; Checks if the top 2 values on the stack are equal, if true eax = 1, else eax =
0
areEqual:
    mov [tmpStack], esp
    pop eax
    pop eax
    pop ebx
    cmp eax, ebx
    je areEqualRes
    mov eax, 0
    mov esp, [tmpStack]
    ret
areEqualRes:
    mov eax, 1
    mov esp, [tmpStack]
    ret

;-----

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

```

```

import kernel, \
    ExitProcess, 'ExitProcess'

import msvcrt, \
    printf, 'printf', \
    scanf, 'scanf', \
    getch, '_getch'

```

Код макроса lengthRoot

```

macro lengthRoot p1x, p1y, p2x, p2y{ ; stores in eax the square root of length
    mov ebx, [p1x]
    sub ebx, [p2x] ; ebx = p1x - p2x
    mov edx, [p1y]
    sub edx, [p2y] ; edx = p1y - p2y
    mov [l1], ebx ; l1 = ebx
    mov [l2], edx ; l2 = edx

    push [l1]
    call abs
    mov [l1], eax ; l1 = abs(l1)

    push [l2]
    call abs
    mov [l2], eax ; l2 = abs(l2)

    push [l1]
    push [l1]
    call mult
    mov [temp], eax
    push [l2]
    push [l2]
    call mult
    add [temp], eax ; temp = l1 * l1 + l2 * l2
    mov eax, [temp] ; eax = temp
}

```

2.2. Общая схема работы программы

Сначала происходит ввод трех точек – концов отрезков, образующих треугольника (вершин) с помощью меток getFirst, getSecond, getThird. Затем осуществляется проверка, могут ли точки сформировать треугольник. Затем - основная функция – проверка, является ли треугольник с введенными вершинами равнобедренным, после чего происходит вывод результата проверки.

Предусмотрен вывод уведомления об ошибке ввода.

2.2.1. Алгоритм проверки (getFirst, getSecond, getThird)

- 1) Запоминается позиция в стеке
- 2) В консоль выводится приглашение на ввод координат
- 3) Считываются введенные значения
- 4) Проверяются введенные значения (метка checkValue)
- 5) Восстанавливается позиция в стеке

2.2.2. Проверка допустимости значения

- 1) Проверяется верхняя граница (65535)
- 2) Если предыдущая проверка не пройдена, то выводится уведомление об ошибке и подсказка с диапазоном допустимых значений.
- 3) Если предыдущая проверка пройдена, осуществляется проверка нижней границы (-65536)
- 4) Если проверка нижней границы пройдена, работа программы продолжается, иначе выводится уведомление об ошибке и работа завершается

2.2.3. Проверка, могут ли точки сформировать треугольник

- 1) В переменной S вычисляем выражение $p1x * (p2y - p3y)$ – произведение координаты X первой точки на разность координат Y второй и третьей точки.
- 2) Прибавляем к S выражение $p2x * (p3y - p1y)$ – произведение координаты X второй точки на разность координат Y третьей и первой точки.
- 3) Прибавляем к S выражение $p3x * (p1y - p2y)$ – произведение координаты X третьей точки на разность координат Y первой и второй точки.
- 4) Если вычисленное в S значение равно 0, то точки могут сформировать треугольник, программа переходит к проверке на равнобедренность, иначе – завершается с соответствующим уведомлением.

2.2.4. Проверка на равнобедренность (checkIsosceles)

- 1) Вычисляются длины сторон (с помощью макроса lengthRoot)
- 2) Длины сторон проверяются на равенство
- 3) Если хотя бы одна пара сторон равна, выводится уведомление о том, что треугольник равнобедренный
- 4) Если ни одна пара не равна – выводится уведомление о том, что треугольник не равнобедренный

Пояснение: макрос lengthRoot применяется для вычисления квадратного корня длины стороны

2.2.5. Завершение работы программы

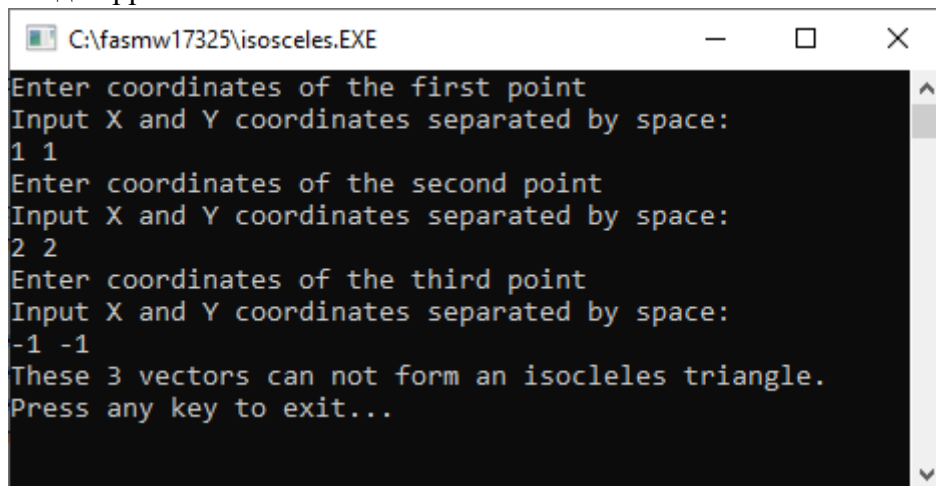
Пользователю выводится предложение нажать любую клавишу для выхода из консольного окна.

2.3. Ограничения

Программа принимает на вход в качестве координат точек целые числа в диапазоне от -65536 до 65535.

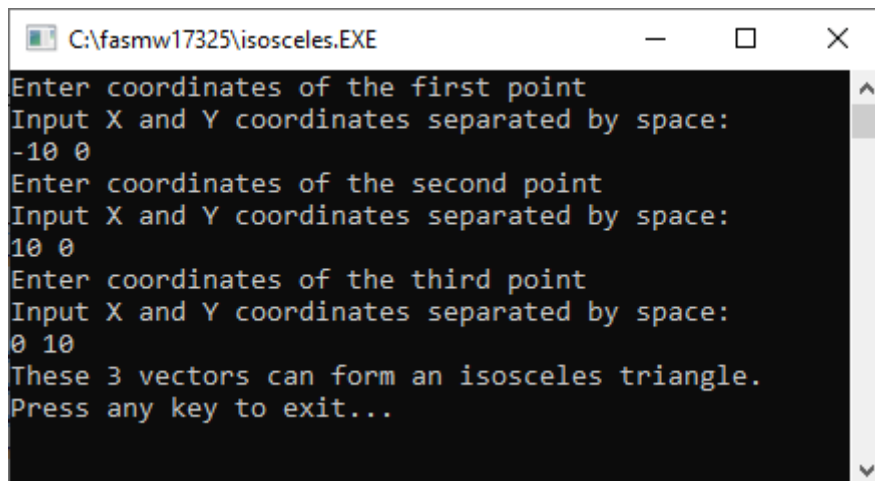
3. Тестирование программы

Ввод корректных значений:



```
C:\fasmw17325\isosceles.EXE
Enter coordinates of the first point
Input X and Y coordinates separated by space:
1 1
Enter coordinates of the second point
Input X and Y coordinates separated by space:
2 2
Enter coordinates of the third point
Input X and Y coordinates separated by space:
-1 -1
These 3 vectors can not form an isocleles triangle.
Press any key to exit...
```

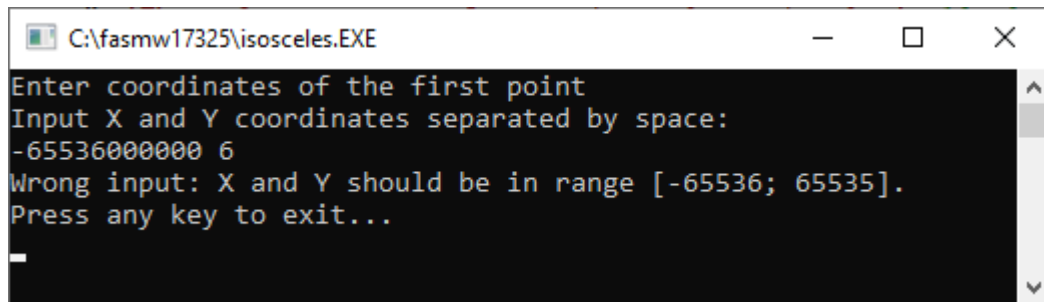
Рисунок 1. Неравнобедренный



```
C:\fasmw17325\isosceles.EXE
Enter coordinates of the first point
Input X and Y coordinates separated by space:
-10 0
Enter coordinates of the second point
Input X and Y coordinates separated by space:
10 0
Enter coordinates of the third point
Input X and Y coordinates separated by space:
0 10
These 3 vectors can form an isosceles triangle.
Press any key to exit...
```

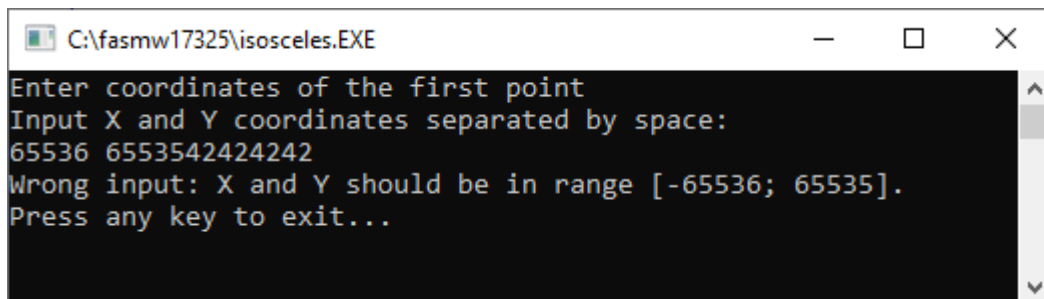
Рисунок 2. Равнобедренный

Ввод некорректных значений:



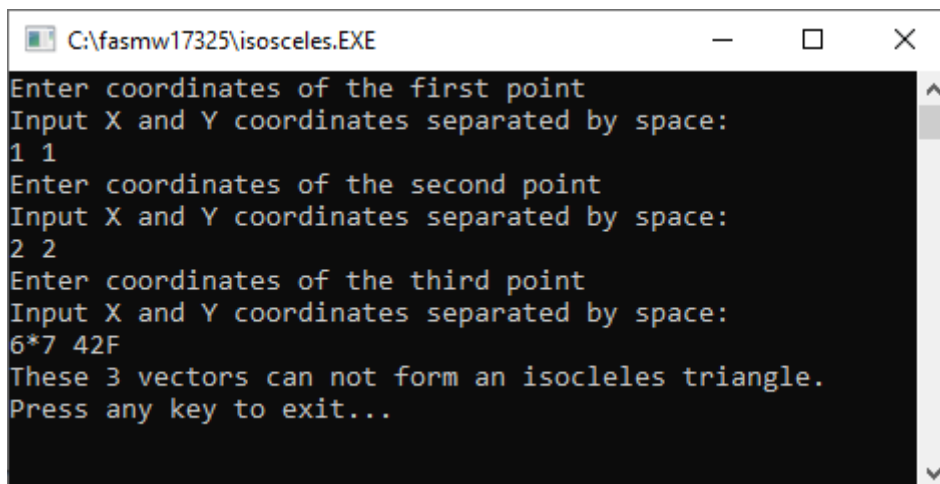
```
C:\fasmw17325\isosceles.EXE
Enter coordinates of the first point
Input X and Y coordinates separated by space:
-65536000000 6
Wrong input: X and Y should be in range [-65536; 65535].
Press any key to exit...
_
```

Рисунок 3. Меньше нижней границы



```
C:\fasmw17325\isosceles.EXE
Enter coordinates of the first point
Input X and Y coordinates separated by space:
65536 6553542424242
Wrong input: X and Y should be in range [-65536; 65535].
Press any key to exit...
```

Рисунок 4. Больше верхней границы



```
C:\fasmw17325\isosceles.EXE
Enter coordinates of the first point
Input X and Y coordinates separated by space:
1 1
Enter coordinates of the second point
Input X and Y coordinates separated by space:
2 2
Enter coordinates of the third point
Input X and Y coordinates separated by space:
6*7 42F
These 3 vectors can not form an isocleles triangle.
Press any key to exit...
```

Рисунок 5. При вводе не числа - уведомление о том, что не может быть сформирован треугольник