

Правительство Российской Федерации

**Федеральное государственное автономное образовательное
учреждение высшего образования «Национальный исследовательский
университет «Высшая школа экономики»**

Факультет компьютерных наук
Департамент программной инженерии

**Отчет к домашнему заданию
По дисциплине
«Архитектура вычислительных систем»**

Работу выполнил:
Студент группы БПИ-193 Штанько Е.О.

Москва 2020

Задание

Разработать программу, решающую вопрос о принадлежности заданных 4-х точек одной окружности(использовать FPU).

Решение

Проверка принадлежности четырех точек одной окружности будет осуществляться по следующему условию:

Если $AC \cup BD = P$ (ходы AC и BD пересекаются в точке P)
и $AP * PC = DP * PB$, то точки A, B, C, D лежат на одной окружности.

Пошаговое решение задачи:

1. Ввод

В цикле вводим 4 точки (используем scanf). Формат ввода - %g (То есть, в зависимости от того, какой формат короче, применяется либо %e, числа с плавающей запятой в экспоненциальной форме записи, либо %f (Десятичное число с плавающей точкой).

При каждом вводе осуществляем проверку данных на корректность(если введено не число - сообщение об ошибке и завершение программы).

Начиная со второй пары координат, осуществляем проверку на повторы (предполагается, что пользователь не может ввести координаты одной и той же точки более одного раза). Сравнение осуществляем с помощью proc Compare(рис. 1). Сначала сравниваем X точек, если они равны, сравниваем Y. При равенстве X и Y двух точек - сообщение об ошибке и завершение программы.

```
; Сравнение
proc Compare
    fcomip    st0, st1;Сравнить вещественные значения, извлечь одно из стека и по результату установить флаги
    fstp     st0      ;Сохранить вещественное значение с извлечением из стека
    ret
endp
```

Рисунок 1-proc Compare

2. Поиск точки пересечения.

Для начала, сортируем координаты по X.

Отсортированные точки условно обозначим как 1 2 3 4.

Из них формируем комбинацию 1 4 2 3 (для этого меняем местами вторую и третью точки (1 2 3 4) -> (1 3 2 4), а затем - вторую и четвертую точки (1 3 2 4) -> (1 4 2 3))

Попробуем найти точку пересечения при такой комбинации.

Для нахождения точки пересечения применяется алгоритм подробно описанный здесь (<https://users.livejournal.com/-winnie/152327.html>)

```
bool intersection(Point2f start1, Point2f end1, Point2f start2, Point2f end2, Point2f *out_intersection)
{
    Point2f dir1 = end1 - start1;
    Point2f dir2 = end2 - start2;

    //считаем уравнения прямых проходящих через отрезки
    float a1 = -dir1.y;
    float b1 = +dir1.x;
    float d1 = -(a1*start1.x + b1*start1.y);

    float a2 = -dir2.y;
    float b2 = +dir2.x;
    float d2 = -(a2*start2.x + b2*start2.y);

    //подставляем концы отрезков, для выяснения в каких полуплоскостях они
    float seg1_line2_start = a2*start1.x + b2*start1.y + d2;
    float seg1_line2_end = a2*end1.x + b2*end1.y + d2;

    float seg2_line1_start = a1*start2.x + b1*start2.y + d1;
    float seg2_line1_end = a1*end2.x + b1*end2.y + d1;

    //если концы одного отрезка имеют один знак, значит он в одной полуплоскости и пересечения нет.
    if (seg1_line2_start * seg1_line2_end >= 0 || seg2_line1_start * seg2_line1_end >= 0)
        return false;

    float u = seg1_line2_start / (seg1_line2_start - seg1_line2_end);
    *out_intersection = start1 + u*dir1;

    return true;
}
```

_Winnie C++ Colorizer

Рисунок 2 - код к алгоритму нахождения точки пересечения(<https://users.livejournal.com/-winnie/152327.html>)

Если точка пересечения при рассмотрении комбинации (1 4 2 3) не была найдена, формируем новую комбинацию (1 3 2 4) (меняем местами вторую и четвертую точки (1 4 2 3) -> (1 3 2 4)), повторяем попытку.

Если точку пересечения не удалось найти ни при комбинации (1 4 2 3), ни при (1 3 2 4) - выводим соответствующее сообщение и завершаем программу.

3. Расчет расстояний.

Расстояние обрабатываемых точек до точки пересечения хорд находим как гипотенузу (рис 3 и 4).

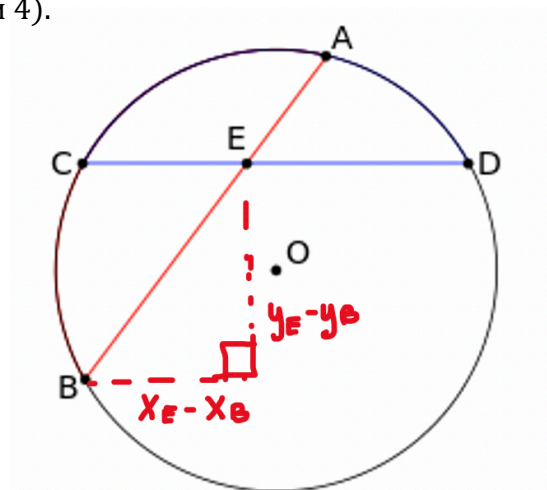


Рисунок 3 - иллюстрация нахождения расстояния

```

proc GetHipotenuza
; адрес 1 точки esi
; адрес 2 точки edi
fld dword [esi]
fld dword [edi]
; катет по X
call GetLength
fmul st0, st0

fld dword [esi + N]
fld dword [edi + N]
; катет по Y
call GetLength
fmul st0, st0

; сумма квадратов катетов
faddp
fsqrt
ret
endp

; расстояние между 2 координатами
proc GetLength
; делаем вычитание
fsubp
fabs
ret
endp

```

Рисунок 4 - proc GetHipotenuza и proc GetLength, используемые для нахождения расстояния

4. Осуществляется проверка.

Находим произведение отрезков обеих хорд и сравниваем их. Если они равны выводим сообщение "yes the points belong to the circle", иначе "no, the points do not belong to the circle".

Текст программы

```
format PE console
entry start
include 'win32a.inc'
```

N = 4 ; 1-е использование: 4 точки в массиве. 2-е использование: смещение к Y

M = 8 ; смещение в массиве к другой точке

```
;-----
section '.data' data readable writable
    msgIn    db 'input points', 10, 0
    msgerr   db 'input error', 10, 0
    msgrep   db 'repeated values', 10, 0
    points   rd  M + N ;координаты введенных точек и точки пересечения

    msgOK    db 'yes the points belong to the circle', 10, 0
    msgNO    db 'no, the points do not belong to the circle' , 10, 0
    fmt      db '%g', 0 ;В зависимости от того, какой формат короче, применяется либо
                        ;%e числа с плавающей запятой в экспоненциальной форме записи ), либо
                        ;%f(Десятичное число с плавающей точкой)

    a1       dd ? ; (Определяем неинициализированную переменную размером 4 байта)
    b1       dd ?
    d1       dd ?

    a2       dd ?
    b2       dd ?
    d2       dd ?

    dir1     rd 2
    dir2     rd 2
    s1       rd 2
    s2       rd 2

    u        dd ?

    scale1   dd 10000.0
    scale2   dd 0.0001
```

```
;-----
section '.code' code readable executable
start:
```

```
; Считываем координаты четырех точек, при этом осуществляя проверку на повторы
;(Предполагается, что пользователь не может ввести координаты одной
; и той же точки дважды)
```

```
;приглашение к вводу
    invoke printf, msgIn
;адреса массива под X
    mov esi, points
;адрес смещения под Y
    mov edi, points + N
```

```
;в цикле вводим 4 точки
    mov ebx, 0;счетчик
.input:
;ввод X
    cinvoke scanf, fmt, esi
;проверка ввода
    cmp eax, 0
    je .error
```

```
;ввод Y
    cinvoke scanf, fmt, edi
;проверка ввода
    cmp eax, 0
    je .error
```

```
    inc ebx;увеличиваем значение счетчика на единицу
;если это первые введенные координаты(значение счетчика = 1) проверка на повторы не нужна
    cmp ebx, 1
    je .inp
```

```

; если это не первые введенные координаты,
; ищем повторы. (Предполагается, что пользователь не может ввести координаты одной
; и той же точки дважды)
push edi
mov edi, points

mov ecx, ebx
dec ecx ;то, сколько будет повторяться цикл repeat. А повторяться он должен (кол-во введенных
;элементов(ebx) - 1)
.repeat:
;сравниваем x
fld dword [edi] ;загрузка вещественного значения в стек регистров FPU.
fld dword [esi]
call Compare
;если x не равны, то можно не сравнивать y
jne .next_val
fld dword [edi + N]
fld dword [esi + N]
call Compare
je .repeated

.next_val:
add edi, M ;сдвигаемся к новому значению
loop .repeat

pop edi

.inp:
;сдвигаемся дальше по массиву
add esi, M
add edi, M

cmp ebx, N
;если введено 4 точки - идем дальше, иначе - продолжаем ввод.
jne .input

;-----

; Вычисление по условию пересечения хорд
; Если  $AC \cdot U \cdot BD = P$ ,  $AP \cdot PC = DP \cdot PB$ , то A, B, C, D лежат на одной окружности

;-----

; Поиск точки пересечения

; Упорядочим вершины
; Сортируем точки по X
; Отсортированные точки условно обозначим как 1 2 3 4
mov ecx, N
.begin_sort:
push ecx
mov esi, points
mov edi, points + M ;сдвигаемся к следующей точке

mov ecx, N-1
.sort:
fld dword [esi]
fld dword [edi]
call Compare
;если больше, идем дальше
ja .next
; иначе делаем обмен
call swap
.next:
add esi, M
add edi, M
loop .sort

pop ecx
loop .begin_sort

; формируем комбинацию 1 4 2 3

; меняем местами вторую и третью точки ( 1 2 3 4 ) -> ( 1 3 2 4 )
mov esi, points + 1*M

```

```

mov edi, points + 2*M
call swap

; меняем местами вторую и четвертую точки ( 1 3 2 4 ) -> ( 1 4 2 3 )
mov esi, points + 1*M
mov edi, points + 3*M
call swap

mov ebx, -1
.next_pair:
inc ebx
cmp ebx, 2
;если точку пересечения не удалось найти ни при комбинации ( 1 4 2 3 ), ни при ( 1 3 2 4 )
;выводим соответствующее сообщение
je .no
cmp ebx, 1
je .next_combination
jmp .calc

;Если точка пересечения при рассмотрении комбинации ( 1 4 2 3 ) не была найдена,
;формируем новую комбинацию ( 1 3 2 4 ) и повторяем попытку.
.next_combination:
; меняем местами вторую и четвертую точки ( 1 4 2 3 ) -> ( 1 3 2 4 )
mov esi, points + 1*M
mov edi, points + 3*M
call swap

.calc:
; Алгоритм нахождения точки пересечения описан в ПЗ

; start1 - точка "начала" рассматриваемой хорды.
; end1 - точка "конца" рассматриваемой хорды.

; Point2f dir1 = end1 - start1;
mov esi, points + 0*M
mov edi, points + 1*M

fld dword [edi] ;Загрузить вещественное значение
fsub dword [esi] ;Вычитание
fstp dword [dir1] ;Сохранить вещественное значение с извлечением из стека

fld dword [edi + N]
fsub dword [esi + N]
fstp dword [dir1 + N]

; Point2f dir2 = end2 - start2;
mov esi, points + 2*M
mov edi, points + 3*M

fld dword [edi]
fsub dword [esi]
fstp dword [dir2]

fld dword [edi + N]
fsub dword [esi + N]
fstp dword [dir2 + N]

;считаем уравнения прямых проходящих через отрезки

; float a1 = -dir1.y;
fldz ;Загрузить константу +0.0
fld dword [dir1+N] ;Загрузить вещественное значение dir1.y
fsubp ;Вычитание с извлечением из стека
fstp dword [a1] ;Сохранить вещественное значение с извлечением из стека

; float b1 = +dir1.x;
fld dword [dir1]
fabs ;Абсолютное значение
fstp dword [b1]

; float d1 = -(a1*start1.x + b1*start1.y);
mov esi, points + 0*M
fld dword [esi]
fmul dword [a1]
fld dword [esi+N]
fmul dword [b1]
faddp ;Сложение с извлечением из стека

```

```

fldz                ;Загрузить константу +0.0
fxch                ;Обменять содержимое регистров
fsubp                ;Вычитание с извлечением из стека
fstp dword [d1]

; float a2 = -dir2.y;
fldz
fld dword [dir2+N]
fsubp
fstp dword [a2]

; float b2 = +dir2.x;
fld dword [dir2]
fabs
fstp dword [b2]

; float d2 = -(a2*start2.x + b2*start2.y);
mov esi, points + 2*M
fld dword [esi]
fmul dword [a2]
fld dword [esi+N]
fmul dword [b2]
faddp
fldz
fxch
fsubp
fstp dword [d2]

; подставляем концы отрезков, для выяснения в каких полуплоскостях они

; float seg1_start = a2*start1.x + b2*start1.y + d2;
mov esi, points + 0*M
fld dword [esi]
fmul dword [a2]

fld dword [esi + N]
fmul dword [b2]

fadd dword [d2]
faddp
fstp dword [s1]

; float seg1_end = a2*end1.x + b2*end1.y + d2;
mov esi, points + 1*M
fld dword [esi]
fmul dword [a2]

fld dword [esi + N]
fmul dword [b2]

fadd dword [d2]
faddp
fstp dword [s1 + N]
; float seg2_start = a1*start2.x + b1*start2.y + d1;
mov esi, points + 2*M
fld dword [esi]
fmul dword [a1]

fld dword [esi + N]
fmul dword [b1]

fadd dword [d1]
faddp
fstp dword [s2]
; float seg2_end = a1*end2.x + b1*end2.y + d1;
mov esi, points + 3*M
fld dword [esi]
fmul dword [a1]

fld dword [esi + N]
fmul dword [b1]

fadd dword [d1]
faddp
fstp dword [s2 + N]

; если концы одного отрезка имеют один знак, значит он в одной полуплоскости и пересечения нет.

```



```

; seg1_start * seg1_end >= 0
fld dword [s1]
fmul dword [s1 + N]
fldz
fxch
call Compare
jae .next_pair

; seg2_start * seg2_end >= 0
fld dword [s2]
fmul dword [s2 + N]
fldz
fxch
call Compare
jae .next_pair

; float u = seg1_start / (seg1_start - seg1_end);
mov esi, s1
fld dword [esi]
fld dword [esi]
fsub dword [esi + N]
fdivp
fstp dword [u]

;out_intersection = start1 + u*dir1;
mov esi, points + 4*M
fld dword [u]
fmul dword [dir1]
fadd dword [points]
fld dword [scale1]
call scale ;избавляемся от неточности
fld dword [scale2]
call scale
fstp dword [esi]

mov esi, points + 4*M
fld dword [u]
fmul dword [dir1 + N]
fadd dword [points + N]
fld dword [scale1] ;scale1 = 10000.0
call scale
fld dword [scale2] ;scale2 = 0.0001
call scale
fstp dword [esi + N]

;-----
; расчет расстояний

mov esi, points + 0*M
mov edi, points + 4*M
call GetHipotenuza

mov esi, points + 4*M
mov edi, points + 1*M
call GetHipotenuza

fmulp ; произведение пересечения первой хорды
fld dword [scale2]
call scale

mov esi, points + 2*M
mov edi, points + 4*M
call GetHipotenuza

mov esi, points + 4*M
mov edi, points + 3*M
call GetHipotenuza

fmulp ; произведение пересечения второй хорды
fld dword [scale2]
call scale

;проверка равенства
call Compare
jne .next_pair

.ok:

```

```

    invoke printf, msgOK
jmp .exit
.no:
    invoke printf, msgNO
jmp .exit
.repeated:
    invoke printf, msgrep
    jmp .exit
.error:
    invoke printf, msgerr
.exit:
    invoke getch
    invoke ExitProcess, 0

```

```

;-----

```

```

; обмен координат в массиве

```

```

proc swap
; обмен X
fld dword [esi] ;Загрузить вещественное значение в стек
fld dword [edi]
fstp dword [esi] ;Сохранить вещественное значение с извлечением из стека
fstp dword [edi]

```

```

; обмен Y
fld dword [esi + N]
fld dword [edi + N]
fstp dword [esi + N]
fstp dword [edi + N]

```

```

ret
endp

```

```

proc GetHipotenuza

```

```

; адрес 1 точки esi
; адрес 2 точки edi
fld dword [esi]
fld dword [edi]
; катет по X
call GetLength
fmul st0, st0

```

```

fld dword [esi + N]
fld dword [edi + N]
; катет по Y
call GetLength
fmul st0, st0

```

```

; сумма квадратов катетов
faddp
fsqrt
ret
endp

```

```

; расстояние между 2 координатами

```

```

proc GetLength
;делаем вычитание
fsubp
fabs
ret
endp

```

```

; Сравнение

```

```

proc Compare
fcomip st0, st1;Сравнить вещественные значения, извлечь одно из стека и по результату установить флаги
fstp st0 ;Сохранить вещественное значение с извлечением из стека
ret
endp

```

```

; Чтобы избавиться от неточности

```

```

proc scale
fmulp
call ceil
ret
endp

```

```

proc ceil

```

```

fld st0      ;Дублирует вершину стека(наше значение)
fld1         ;Загрузить константу +1.0
fxch
fprem        ;Найти частичный остаток деления на 1.0(числа после запятой)
fsubp st2, st0 ;Вычитаем из изначального значения "округленное"
fstp st0
ret
endp

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'

include 'api\kernel32.inc'
import msvcrt, printf, 'printf', scanf, 'scanf', getch, '_getch'

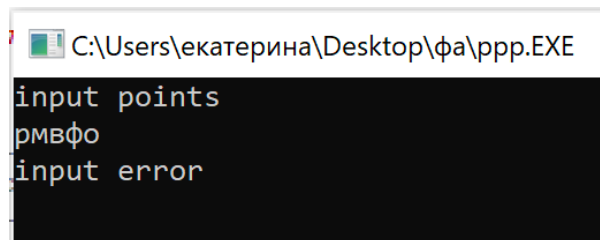
```

Тестирование

1. Рассмотрим случаи некорректного ввода.

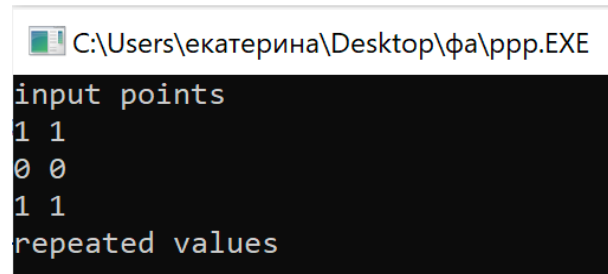
Когда программа получает на вход не число - она сообщает об этом "input error" (рис.5)

Если координаты точки введены более одного раза - программа сообщает "repeated values" (рис.6)



```
C:\Users\екатерина\Desktop\фа\ppp.EXE
input points
рмвфо
input error
```

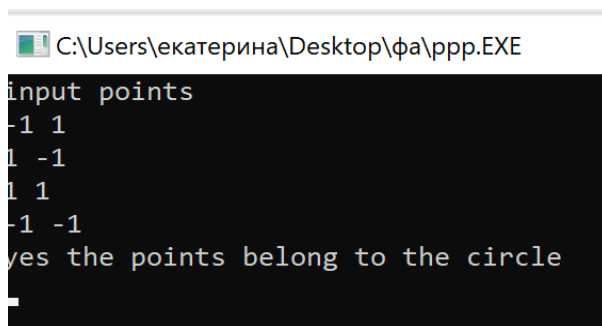
Рисунок 5 - пример некорректного ввода. Не число.



```
C:\Users\екатерина\Desktop\фа\ppp.EXE
input points
1 1
0 0
1 1
repeated values
```

Рисунок 6 - пример некорректного ввода. Повтор координат.

1. Рассмотрим примеры работы программы при вводе корректных данных.



```
C:\Users\екатерина\Desktop\фа\ppp.EXE
input points
-1 1
1 -1
1 1
-1 -1
yes the points belong to the circle
```

Рисунок 7-простейший пример корректной работы

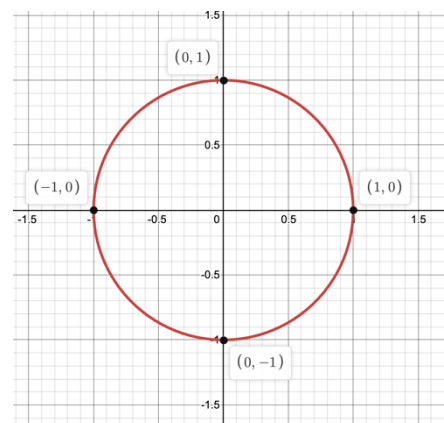


Рисунок 8-иллюстрация к примеру из рис. 7

```

C:\Users\екатерина\Desktop\фа\ppp.EXE

input points
2 0
0 2
-4 0
0 -4
yes the points belong to the circle

```

Рисунок 9-простейший пример корректной работы

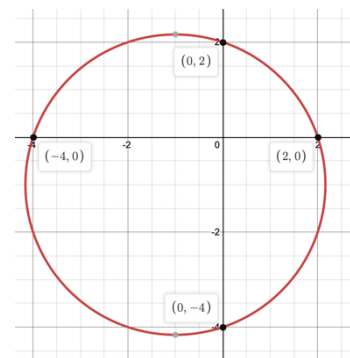


Рисунок 10 - иллюстрация к примеру из рис.9

```

C:\Users\екатерина\Desktop\фа\ppp.EXE

input points
0 5
0 -1
-2 5
-2.8 4.6
yes the points belong to the circle

```

Рисунок 11 - пример корректной работы

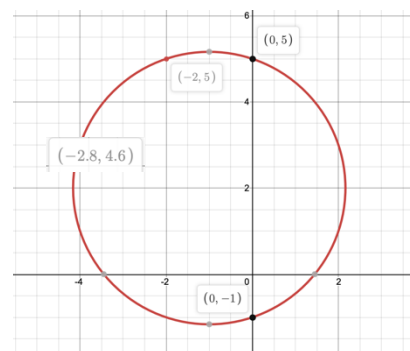


Рисунок 12 - иллюстрация к примеру из рис.11

```

C:\Users\екатерина\Desktop\фа\ppp.EXE

input points
-0000000.6 0000000.6
-0000000.6 -0000000.6
0000000.6 0000000.6
0000000.6 -0000000.6
yes the points belong to the circle

```

Рисунок 13 - пример корректной работы

```

C:\Users\екатерина\Desktop\фа\ppp.EXE

input points
1.1e+44 1.1e+44
-1.1e+44 -1.1e+44
1.1e+44 -1.1e+44
-1.1e+44 1.1e+44
yes the points belong to the circle

```

Рисунок 14 - пример корректной работы. Числа в экспоненциальной форме записи.



```

C:\Users\екатерина\Desktop\фа\ppp.EXE

input points
4.6 -2.74
4.74 -2.82
5 -3.09
5.1 -3.9
yes the points belong to the circle

```

Рисунок 15 - пример корректной работы

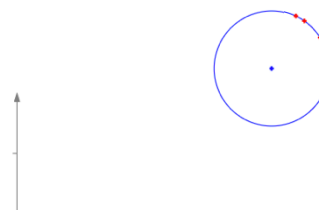


Рисунок 16 - иллюстрация к примеру из рис.15

```
C:\Users\екатерина\Desktop\фа\ppp.EXE
input points
-5.34 4.9
-7.6 5
-9.12 3.91
-9.53 3.19
yes the points belong to the circle
```

Рисунок 17 - пример корректной работы

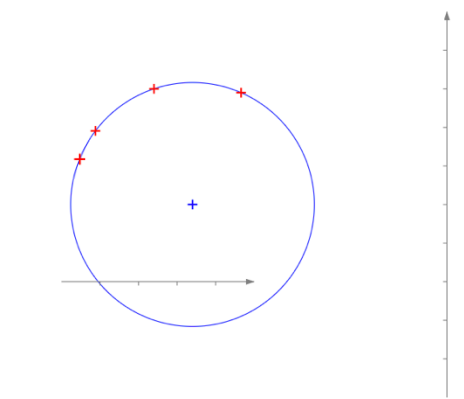


Рисунок 18 - иллюстрация к примеру из рис.17

```
C:\Users\екатерина\Desktop\фа\ppp.EXE
input points
-1.9 -3.4
-3.76 -1.45
-2.3 -2.6
-3.1 -1.8
yes the points belong to the circle
```

Рисунок 19 - пример корректной работы. Все точки находятся в одной четверти.

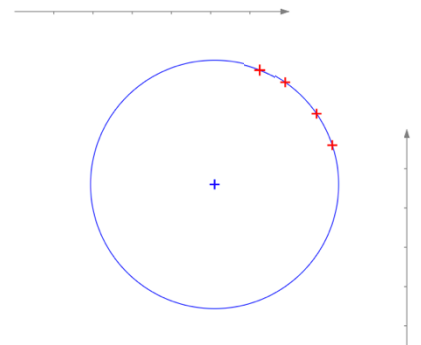


Рисунок 20 - иллюстрация к примеру из рис.19

```
C:\Users\екатерина\Desktop\фа\ppp.EXE
input points
0 0
-2 -2
-3 -3
4 4
no, the points do not belong to the circle
```

Рисунок 21 - пример корректной работы

Список используемых источников

1. Омский государственный университет «Справочник. Окружности» (<http://www.univer.omsk.su/omsk/Edu/Rusanova/circles.htm>) Просмотрено: 16.10.2020
2. LIVEJOURNAL (2007) «Геометрические Алгоритмы. Пересечение двух отрезков на плоскости» (<https://users.livejournal.com/-winnie/152327.html>) Просмотрено 16.10.2020
3. Легалов А.И.(2020) «Разработка программ на ассемблере. Использование макроопределений» (<http://softcraft.ru/edu/comparch/practice/asm86/04-macro/>) Просмотрено: 18.10.2020
4. Легалов А.И.(2020) «Разработка программ на ассемблере. Использование сопроцессора с плавающей точкой» (<http://softcraft.ru/edu/comparch/practice/asm86/05-fpu/>) Просмотрено: 17.10.2020
5. YouTube “Яша добрый хакер” (2018) «Канал Яша добрый хакер» (<https://www.youtube.com/user/yashechka85/videos>) Просмотрено: 19.10.2020
6. Сайт "Клуб 155" (<http://www.club155.ru>)
7. flat assembler. Documentation and tutorial (https://flatassembler.net/docs.php?article=fasmg_manual)