Национальный исследовательский университет

"Высшая школа экономики"

Факультет компьютерных наук

Программная инженерия

Микропроект №1.

Программирование на языке ассемблера.

Шкляр Михаил Игоревич, БПИ191

ВАРИАНТ 28. Разработать программу, которая решает вопрос о нахождении пар перпендикулярных отрезков среди N = 5 отрезков, заданных координатами концевых точек.

Решение задания

В секции данных мы объявляем все db переменные, необходимые нам для информирования пользователя о происходящем и dd переменные непосредственно для осуществления процесса нахождения ортогональных отрезков (ниже подробнее).

```
messageX db 'Input the abscissa of 1 point: ', 0
messageY db 'Input the ordinate of 1 point: ', 0
messageX1 db 'Input the abscissa of 2 point: ', 0
messageY1 db 'Input the ordinate of 2 point: ', 0
perpMes1 db 'A pair consisting of a segment defined by such coordinates of the end points: (', 0
perpMes2 db ' ; ', 0
perpMes3 db ') and (', 0
perpMes4 db ')',  10, 13, 0
perpMes41 db 'as well as a segment with such coordinates of the end points: (', 0
perpMes5 db ')", 10, 13, 0
perpMes6 db "contains perpendicular line segments.", 10, 13, 0
conclusionMessage1  db 'To sum up, there was/were ', 0
conclusionMessage2  db ' pair/pairs of perpendicular line segments in total.', 0
newStr db '',10, 13, 0
repEnter db 'The wrong coordinates input. Entered coordinate have to be less than -10^6 and more than 10^6.', 10, 13, 0
tryAgain db 'Try again: ', 0
digit db '%d', 0

c11        dd ?
c12        dd ?
c13        dd ?
c14        dd ?
c21        dd ?
c22        dd ?
c23        dd ?
c24        dd ?
c31        dd ?
c32        dd ?
c33        dd ?
c34        dd ?
c41        dd ?
c42        dd ?
c43        dd ?
c44        dd ?
c51        dd ?
c52        dd ?
c53        dd ?
c54        dd ?
tmpStack   dd ?
iInp       dd 1
perpNum    dd 0
```

В следующей таблице описаны все используемые переменные, их исходные значения и их предназначения:

C11 – C51	Не задано	Переменные для хранения ссылок на абсциссы первых точек, задающих отрезки.

C12 – C52	Не задано	Переменные для хранения ссылок на ординаты первых точек, задающих отрезки.
C13 – C53	Не задано	Переменные для хранения ссылок на абсциссы вторых точек, задающих отрезки.
C14 – C54	Не задано	Переменные для хранения ссылок на ординаты вторых точек, задающих отрезки.
tmpStack	Не задано	Переменная, которая будет необходима для корректной работы с процедурами (сохранение команды выхода из процедуры, которая может уйти из начального положения в стеке в связи работы с ним для ret).
iInp	1	Переменная, необходимая для вывода номера отрезка, координаты которого в данный момент необходимо ввести.
perpNum	0	Переменная для подсчета числа пар найденных перпендикулярных отрезков.

Все действия в программе происходят с использованием макроопределений. В секции с основным кодом происходит лишь вызов 3 макросов и

завершение программы (используем invoke ExitProcess, 0 для передачи нуля в качестве аргумента функции).

```
;------------------------------------------------------------------
section '.code' code readable executable
start:
        ReadAllCoord c11, c12, c13, c14, c21, c22, c23, c24, c31, c32, c33, c34, c41, c42, c43, c44, c51, c52, c53, c54    ; Считываем данные
        FindPerpendicular c11, c12, c13, c14, c21, c22, c23, c24, c31, c32, c33, c34, c41, c42, c43, c44, c51, c52, c53, c54    ; Ищем перпендикуляры
        Conclusion perpNum                                                                                                 ; Делаем вывод
finish:
        call [getch]
        invoke ExitProcess, 0
;------------------------------------------------------------------
```

Подключаем библиотеку с созданными макросами.

```
include 'win32a.inc'
include 'Find_perpendicular.inc'
```

Далее рассмотрим каждый макрос по-отдельности.

Первый макрос нужен для сокращения кода в основной секции.
С помощью него мы просто вызываем 2 макрос от ссылок на конкретные координаты.

```
;------------------------------------------------------------------
macro ReadAllCoord c11, c12, c13, c14, c21, c22, c23, c24, c31, c32, c33, c34, c41, c42, c43, c44, c51, c52, c53, c54 {
  cinvoke printf, messageStartInput0
  ReadCoord c11, c12, c13, c14
  ReadCoord c21, c22, c23, c24
  ReadCoord c31, c32, c33, c34
  ReadCoord c41, c42, c43, c44
  ReadCoord c51, c52, c53, c54
}
;------------------------------------------------------------------
```

Во втором макросе осуществляется ввод отрезка путём задания координат его концевых точек. С помощью: cinvoke printf, digit, dword [iInp] мы выводим номер текущего отрезка.
Также мы вызываем макрос RightScan для проверк корректности введённой координаты.

```
;------------------------------------------------------------------
macro ReadCoord x, y, x1, y1 {
        cinvoke printf, messageStartInput1
        cinvoke printf, digit, dword [iInp]
        cinvoke printf, messageStartInput2
        cinvoke printf, messageX
        RightScan x
        cinvoke printf, messageY
        RightScan y
        cinvoke printf, messageX1
        RightScan x1
        cinvoke printf, messageY1
        RightScan y1
        add [iInp], 1
        cinvoke printf, newStr
}
;------------------------------------------------------------------
```

<u>Третий</u> макрос необходим для проверки правильности входных данных. Хочу отметить использование локальных меток, которые я отмечал ..<Name> по код-стайлу. Локальные метки необходимы в макросах, если они выполняются 2 и более раз. Ведь в ином случае произойдет ошибка компиляции:
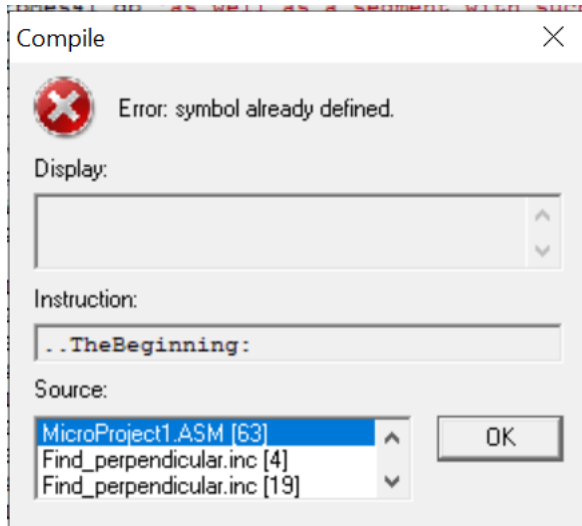
Кроме того, хочу отметить, что необходимо использование дополнительной метки в конце для пропуска части кода, которая не всегда должна выполняться, ведь в макросах выполняются все метки по порядку.

```
;--------------------------------------------------------------
macro RightScan scanAdres {
    local ..TheBeginning
    ..TheBeginning:
      mov [tmpStack], esp
      cinvoke scanf, digit, scanAdres
      local ..RepeatInput
      cmp [scanAdres], -1000000
      jl ..RepeatInput
      cmp [scanAdres], 1000000
      jg ..RepeatInput
      mov esp, [tmpStack]
      local ..Over
      jmp ..Over
    ..RepeatInput:
      cinvoke printf, newStr
      cinvoke printf, repEnter
      cinvoke printf, tryAgain
      jmp ..TheBeginning
    ..Over:
      ;ret
}
;--------------------------------------------------------------
```

<u>Четвертый</u> макрос так же, как и первый используется для сокращения главной секции с кодом. Отсюда происходит проверка перпендикулярности отрезков по парам. Количество таких проверок (то есть число неповторяющихся пар отрезков) высчитывается по формуле:

$$C_5^2 = \frac{5!}{2! \cdot (5-2)!} = \frac{5!}{2! \cdot 3!} = \frac{4 \cdot 5}{1 \cdot 2} = 10$$

.

```
;--------------------------------------------------------------------
macro FindPerpendicular c11, c12, c13, c14, c21, c22, c23, c24, c31, c32, c33, c34, c41, c42, c43, c44, c51, c52, c53, c54 {
  IsPerpendicular c11, c12, c13, c14, c21, c22, c23, c24
  IsPerpendicular c11, c12, c13, c14, c31, c32, c33, c34
  IsPerpendicular c11, c12, c13, c14, c41, c42, c43, c44
  IsPerpendicular c11, c12, c13, c14, c51, c52, c53, c54
  IsPerpendicular c21, c22, c23, c24, c31, c32, c33, c34
  IsPerpendicular c21, c22, c23, c24, c41, c42, c43, c44
  IsPerpendicular c21, c22, c23, c24, c51, c52, c53, c54
  IsPerpendicular c31, c32, c33, c34, c41, c42, c43, c44
  IsPerpendicular c31, c32, c33, c34, c51, c52, c53, c54
  IsPerpendicular c41, c42, c43, c44, c51, c52, c53, c54
}
;--------------------------------------------------------------------
```

В <u>пятом</u> макросе происходит главная часть программы – проверка ортогональности отрезков.

Здесь также будут использоваться локальные метки.

В данной части макроса:

```
macro IsPerpendicular x1, y1, x2, y2, x3, y3, x4, y4 {
      mov [tmpStack], esp
      mov eax, [x2]
      sub eax, [x1]
      mov ecx, [x4]
      sub ecx, [x3]
      imul ecx
      mov ebx, [y2]
      sub ebx, [y1]
      mov edx, [y4]
      sub edx, [y3]
      imul edx
      add eax, ebx
      cmp eax, 0
      local ..PerpendicularFound
      je ..PerpendicularFound
      mov esp, [tmpStack]
      local ..Over
      jmp ..Over
```

происходит проверка перпендикулярности. Она выполняется путём вычисления скалярного произведения и сравнения его с нулём.

Так как непересекающиеся отрезки тоже могут быть перпендикулярными, то мы вправе представлять их в виде векторов, вычислять координаты 1 и 2 векторов, а далее находить сумму перемножения их соответствующих координат. Если скалярное произведение равно 0, то переходим к метке ..PerpendicularFound, иначе идём в метку ..Over.

Далее приведен формат вывода сообщения о том, что найдена пара перпендикулярных отрезков:

```
..PerpendicularFound:
      cinvoke printf, perpMes1          ; Пара, состоящая из отрезока с координатами концевых точек: (
      cinvoke printf, digit, dword [x1]
      cinvoke printf, perpMes2          ; ;
      cinvoke printf, digit, dword [y1]
      cinvoke printf, perpMes3          ; ) и (
      cinvoke printf, digit, dword [x2]
      cinvoke printf, perpMes2
      cinvoke printf, digit, dword [y2]
      cinvoke printf, perpMes4          ; ), а также отрезка с координатами концевых точек: (
      cinvoke printf, perpMes41
      cinvoke printf, digit, dword [x3]
      cinvoke printf, perpMes2
      cinvoke printf, digit, dword [y3]
      cinvoke printf, perpMes3
      cinvoke printf, digit, dword [x4]
      cinvoke printf, perpMes2
      cinvoke printf, digit, dword [y4]
      cinvoke printf, perpMes5          ; ")", 10, 13, 0
      cinvoke printf, perpMes6          ; "содержит перпендикулярные отрезки.", 10, 13, 0
      add [perpNum], 1
..Over:
      ;ret
```

В <u>шестом</u> макросе осуществляется вывод информации о том, какое количество пар ортогональных отрезков было найдено.

```
;--------------------------------------------------------------------
macro Conclusion pNum {
  cinvoke printf, newStr
  cinvoke printf, conclusionMessage1          ; Всего нашлось
  cinvoke printf, digit, dword [pNum]
  cinvoke printf, conclusionMessage2          ; пар перпендикулярных отрезков.
}
```

Примеры запуска программы

1) Тестируем программу в случай с отсутствием пар перпендикулярных отрезков:


```
C:\Users\Mi\Downloads\fasmw17325\EXAMPLES\MicroProject1.EXE

Program for the search of the perpendicular segments amoung 5 ones. Input coordinates of segments` end points.
Entering coordinates of the 1 vector:
Input the abscissa of 1 point: 1
Input the ordinate of 1 point: 2
Input the abscissa of 2 point: 3
Input the ordinate of 2 point: 4

Entering coordinates of the 2 vector:
Input the abscissa of 1 point: 5
Input the ordinate of 1 point: 6
Input the abscissa of 2 point: 7
Input the ordinate of 2 point: 8

Entering coordinates of the 3 vector:
Input the abscissa of 1 point: 9
Input the ordinate of 1 point: 10
Input the abscissa of 2 point: 11
Input the ordinate of 2 point: 12

Entering coordinates of the 4 vector:
Input the abscissa of 1 point: 13
Input the ordinate of 1 point: 14
Input the abscissa of 2 point: 15
Input the ordinate of 2 point: 16

Entering coordinates of the 5 vector:
Input the abscissa of 1 point: 17
Input the ordinate of 1 point: 18
Input the abscissa of 2 point: 19
Input the ordinate of 2 point: 20


To sum up, there was/were 0 pair/pairs of perpendicular line segments in total._
```

2) Запустим программу, введя только 1 пару перпендикулярных отрезков + проверим на отрицательных, нулевых значениях и больших числах:

```
Program for the search of the perpendicular segments amoung 5 ones. Input coordinates of segments` end points.
Entering coordinates of the 1 vector:
Input the abscissa of 1 point: 12
Input the ordinate of 1 point: 0
Input the abscissa of 2 point: -12673
Input the ordinate of 2 point: 0

Entering coordinates of the 2 vector:
Input the abscissa of 1 point: 0
Input the ordinate of 1 point: 17891
Input the abscissa of 2 point: 0
Input the ordinate of 2 point: -178129

Entering coordinates of the 3 vector:
Input the abscissa of 1 point: 35124
Input the ordinate of 1 point: 234525
Input the abscissa of 2 point: -32432
Input the ordinate of 2 point: -332832

Entering coordinates of the 4 vector:
Input the abscissa of 1 point: 829222
Input the ordinate of 1 point: 29393
Input the abscissa of 2 point: -2933
Input the ordinate of 2 point: -2943

Entering coordinates of the 5 vector:
Input the abscissa of 1 point: 29239
Input the ordinate of 1 point: 0
Input the abscissa of 2 point: 28393
Input the ordinate of 2 point: 193922

A pair consisting of a segment defined by such coordinates of the end points: (12 ; 0) and (-12673 ; 0),
as well as a segment with such coordinates of the end points: (0 ; 17891) and (0 ; -178129)
contains perpendicular line segments.

To sum up, there was/were 1 pair/pairs of perpendicular line segments in total._
```

3) Теперь зададим нулевой отрезок. Он должен оказаться ортогональным всем остальным:

```
Program for the search of the perpendicular segments amoung 5 ones. Input coordinates of segments` end points.
Entering coordinates of the 1 vector:
Input the abscissa of 1 point: 1
Input the ordinate of 1 point: 0
Input the abscissa of 2 point: 3
Input the ordinate of 2 point: 0

Entering coordinates of the 2 vector:
Input the abscissa of 1 point: 5
Input the ordinate of 1 point: 5
Input the abscissa of 2 point: 5
Input the ordinate of 2 point: 5

Entering coordinates of the 3 vector:
Input the abscissa of 1 point: -1
Input the ordinate of 1 point: 0
Input the abscissa of 2 point: 3
Input the ordinate of 2 point: 6

Entering coordinates of the 4 vector:
Input the abscissa of 1 point: 0
Input the ordinate of 1 point: -1
Input the abscissa of 2 point: 0
Input the ordinate of 2 point: 5555

Entering coordinates of the 5 vector:
Input the abscissa of 1 point: 15514
Input the ordinate of 1 point: -2324
Input the abscissa of 2 point: 343
Input the ordinate of 2 point: -2324
```

A pair consisting of a segment defined by such coordinates of the end points: (1 ; 0) and (3 ; 0),
as well as a segment with such coordinates of the end points: (5 ; 5) and (5 ; 5)
contains perpendicular line segments.
A pair consisting of a segment defined by such coordinates of the end points: (1 ; 0) and (3 ; 0),
as well as a segment with such coordinates of the end points: (0 ; -1) and (0 ; 5555)
contains perpendicular line segments.
A pair consisting of a segment defined by such coordinates of the end points: (1 ; 0) and (3 ; 0),
as well as a segment with such coordinates of the end points: (15514 ; -2324) and (343 ; -2324)
contains perpendicular line segments.
A pair consisting of a segment defined by such coordinates of the end points: (5 ; 5) and (5 ; 5),
as well as a segment with such coordinates of the end points: (-1 ; 0) and (3 ; 6)
contains perpendicular line segments.
A pair consisting of a segment defined by such coordinates of the end points: (5 ; 5) and (5 ; 5),
as well as a segment with such coordinates of the end points: (0 ; -1) and (0 ; 5555)
contains perpendicular line segments.
A pair consisting of a segment defined by such coordinates of the end points: (5 ; 5) and (5 ; 5),
as well as a segment with such coordinates of the end points: (15514 ; -2324) and (343 ; -2324)
contains perpendicular line segments.
A pair consisting of a segment defined by such coordinates of the end points: (-1 ; 0) and (3 ; 6),
as well as a segment with such coordinates of the end points: (15514 ; -2324) and (343 ; -2324)
contains perpendicular line segments.
A pair consisting of a segment defined by such coordinates of the end points: (0 ; -1) and (0 ; 5555),
as well as a segment with such coordinates of the end points: (15514 ; -2324) and (343 ; -2324)
contains perpendicular line segments.

To sum up, there was/were 8 pair/pairs of perpendicular line segments in total.

4) Проверим некорректный ввод:

```
Program for the search of the perpendicular segments amoung 5 ones. Input coordinates of segments` end points.
Entering coordinates of the 1 vector:
Input the abscissa of 1 point: 1
Input the ordinate of 1 point: 1
Input the abscissa of 2 point: 1
Input the ordinate of 2 point: 1

Entering coordinates of the 2 vector:
Input the abscissa of 1 point: -2737
Input the ordinate of 1 point: 0
Input the abscissa of 2 point: 363737
Input the ordinate of 2 point: 0

Entering coordinates of the 3 vector:
Input the abscissa of 1 point: 0
Input the ordinate of 1 point: 4748
Input the abscissa of 2 point: 0
Input the ordinate of 2 point: 3

Entering coordinates of the 4 vector:
Input the abscissa of 1 point: -567891
Input the ordinate of 1 point: 0
Input the abscissa of 2 point: 18293
Input the ordinate of 2 point: 0

Entering coordinates of the 5 vector:
Input the abscissa of 1 point: -292802903

The wrong coordinates input. Entered coordinate have to be less than -10^6 and more than 10^6.
Try again: -1
Input the ordinate of 1 point: 5
Input the abscissa of 2 point: 3
Input the ordinate of 2 point: 6
```

```
A pair consisting of a segment defined by such coordinates of the end points: (1 ; 1) and (1 ; 1),
as well as a segment with such coordinates of the end points: (-2737 ; 0) and (363737 ; 0)
contains perpendicular line segments.
A pair consisting of a segment defined by such coordinates of the end points: (1 ; 1) and (1 ; 1),
as well as a segment with such coordinates of the end points: (0 ; 4748) and (0 ; 3)
contains perpendicular line segments.
A pair consisting of a segment defined by such coordinates of the end points: (1 ; 1) and (1 ; 1),
as well as a segment with such coordinates of the end points: (-567891 ; 0) and (18293 ; 0)
contains perpendicular line segments.
A pair consisting of a segment defined by such coordinates of the end points: (1 ; 1) and (1 ; 1),
as well as a segment with such coordinates of the end points: (-1 ; 5) and (3 ; 6)
contains perpendicular line segments.
A pair consisting of a segment defined by such coordinates of the end points: (-2737 ; 0) and (363737 ; 0),
as well as a segment with such coordinates of the end points: (0 ; 4748) and (0 ; 3)
contains perpendicular line segments.
A pair consisting of a segment defined by such coordinates of the end points: (-2737 ; 0) and (363737 ; 0),
as well as a segment with such coordinates of the end points: (-567891 ; 0) and (18293 ; 0)
contains perpendicular line segments.

To sum up, there was/were 6 pair/pairs of perpendicular line segments in total.
```

5) И последний пример с нетривиальными значениями перпендикулярных отрезков.

```
Entering coordinates of the 1 vector:
Input the abscissa of 1 point: -8
Input the ordinate of 1 point: -7
Input the abscissa of 2 point: -9
Input the ordinate of 2 point: -6

Entering coordinates of the 2 vector:
Input the abscissa of 1 point: 0
Input the ordinate of 1 point: 0
Input the abscissa of 2 point: 1
Input the ordinate of 2 point: 1

Entering coordinates of the 3 vector:
Input the abscissa of 1 point: 5
Input the ordinate of 1 point: 6
Input the abscissa of 2 point: 7
Input the ordinate of 2 point: 8

Entering coordinates of the 4 vector:
Input the abscissa of 1 point: 5
Input the ordinate of 1 point: 6
Input the abscissa of 2 point: 6
Input the ordinate of 2 point: 5

Entering coordinates of the 5 vector:
Input the abscissa of 1 point: 1232
Input the ordinate of 1 point: 1243
Input the abscissa of 2 point: -235
Input the ordinate of 2 point: 3

A pair consisting of a segment defined by such coordinates of the end points: (-8 ; -7) and (-9 ; -6),
as well as a segment with such coordinates of the end points: (0 ; 0) and (1 ; 1)
contains perpendicular line segments.
A pair consisting of a segment defined by such coordinates of the end points: (0 ; 0) and (1 ; 1),
as well as a segment with such coordinates of the end points: (5 ; 6) and (6 ; 5)
contains perpendicular line segments.
A pair consisting of a segment defined by such coordinates of the end points: (5 ; 6) and (7 ; 8),
as well as a segment with such coordinates of the end points: (5 ; 6) and (6 ; 5)
contains perpendicular line segments.

To sum up, there was/were 3 pair/pairs of perpendicular line segments in total.
```

ПРИЛОЖЕНИЕ 1. Текст программы.

Шкляр Михаил Игоревич
; БПИ191
; Вариант №28
; Условие: написать программу, находящую все пары перпендикулярных отрезков среди пяти введённых.

format PE console
entry start

include 'win32a.inc'
include 'Find_perpendicular.inc'

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

```
    messageStartInput0   db 'Program for the search of the perpendicular segments
amoung 5 ones. Input coordinates of segments` end points.', 10, 13, 0
    messageStartInput1   db 'Entering coordinates of the ', 0
    messageStartInput2   db ' vector:', 10, 13, 0
    messageX db 'Input the abscissa of 1 point: ', 0
    messageY db 'Input the ordinate of 1 point: ', 0
    messageX1 db 'Input the abscissa of 2 point: ', 0
    messageY1 db 'Input the ordinate of 2 point: ', 0
    perpMes1  db 'A pair consisting of a segment defined by such coordinates of the end
points: (', 0
    perpMes2  db ' ; ', 0
    perpMes3  db ') and (', 0
    perpMes4  db '),', 10, 13, 0
    perpMes41 db 'as well as a segment with such coordinates of the end points: (', 0
    perpMes5  db ")", 10, 13, 0
    perpMes6  db "contains perpendicular line segments.", 10, 13, 0
    conclusionMessage1  db 'To sum up, there was/were ', 0
    conclusionMessage2  db ' pair/pairs of perpendicular line segments in total.', 0
    newStr  db '',10, 13, 0
    repEnter db 'The wrong coordinates input. Entered coordinate have to be less than -
10^6 and more than 10^6.', 10, 13, 0
    tryAgain db 'Try again: ', 0
    digit db '%d', 0

    c11       dd ?
    c12       dd ?
    c13       dd ?
    c14       dd ?
    c21       dd ?
    c22       dd ?
    c23       dd ?
    c24       dd ?
```

```
        c31     dd ?
        c32     dd ?
        c33     dd ?
        c34     dd ?
        c41     dd ?
        c42     dd ?
        c43     dd ?
        c44     dd ?
        c51     dd ?
        c52     dd ?
        c53     dd ?
        c54     dd ?
    tmpStack    dd ?
    iInp        dd 1
    perpNum     dd 0


;----------------------------------------------------------------
section '.code' code readable executable
start:
    ReadAllCoord c11, c12, c13, c14, c21, c22, c23, c24, c31, c32, c33, c34, c41, c42, c43,
c44, c51, c52, c53, c54          ; Считываем данные
    FindPerpendicular c11, c12, c13, c14, c21, c22, c23, c24, c31, c32, c33, c34, c41, c42,
c43, c44, c51, c52, c53, c54       ; Ищем перпендикуляры
    Conclusion perpNum                                                              ;
Делаем вывод
finish:
    call [getch]
    invoke ExitProcess, 0


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

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'


  ;----------------------------------------------------------------------
```

```
macro ReadAllCoord c11, c12, c13, c14, c21, c22, c23, c24, c31, c32, c33, c34, c41, c42, c43,
c44, c51, c52, c53, c54 {
  cinvoke printf, messageStartInput0
  ReadCoord c11, c12, c13, c14
  ReadCoord c21, c22, c23, c24
  ReadCoord c31, c32, c33, c34
  ReadCoord c41, c42, c43, c44
  ReadCoord c51, c52, c53, c54
}

;-------------------------------------------------------------------------
macro ReadCoord x, y, x1, y1 {
    cinvoke printf, messageStartInput1
    cinvoke printf, digit, dword [iInp]
    cinvoke printf, messageStartInput2
    cinvoke printf, messageX
    RightScan x
    cinvoke printf, messageY
    RightScan y
    cinvoke printf, messageX1
    RightScan x1
    cinvoke printf, messageY1
    RightScan y1
    add [iInp], 1
    cinvoke printf, newStr
}

;-------------------------------------------------------------------------
macro RightScan scanAdres {
    local ..TheBeginning
    ..TheBeginning:
     mov [tmpStack], esp
     cinvoke scanf, digit, scanAdres
     local ..RepeatInput
     cmp [scanAdres], -1000000
     jl ..RepeatInput
     cmp [scanAdres], 1000000
     jg ..RepeatInput
     mov esp, [tmpStack]
     local ..Over
     jmp ..Over
    ..RepeatInput:
     cinvoke printf, newStr
     cinvoke printf, repEnter
     cinvoke printf, tryAgain
     jmp ..TheBeginning
    ..Over:
     ;ret
}
```

```
;-------------------------------------------------------------------------
macro FindPerpendicular c11, c12, c13, c14, c21, c22, c23, c24, c31, c32, c33, c34, c41, c42,
c43, c44, c51, c52, c53, c54 {
  IsPerpendicular c11, c12, c13, c14, c21, c22, c23, c24
  IsPerpendicular c11, c12, c13, c14, c31, c32, c33, c34
  IsPerpendicular c11, c12, c13, c14, c41, c42, c43, c44
  IsPerpendicular c11, c12, c13, c14, c51, c52, c53, c54
  IsPerpendicular c21, c22, c23, c24, c31, c32, c33, c34
  IsPerpendicular c21, c22, c23, c24, c41, c42, c43, c44
  IsPerpendicular c21, c22, c23, c24, c51, c52, c53, c54
  IsPerpendicular c31, c32, c33, c34, c41, c42, c43, c44
  IsPerpendicular c31, c32, c33, c34, c51, c52, c53, c54
  IsPerpendicular c41, c42, c43, c44, c51, c52, c53, c54
}
;-------------------------------------------------------------------------
macro IsPerpendicular x1, y1, x2, y2, x3, y3, x4, y4 {
    local ..Zero1
    local ..Continue1
    local ..Zero2
    local ..Continue2
    local ..PerpendicularFound
    mov [tmpStack], esp
    mov eax, [x2]
    sub eax, [x1]
    mov ecx, [x4]
    sub ecx, [x3]
    cmp ecx, 0
    je ..Zero1
    jmp ..Continue1
    ..Zero1:
      mov eax, 0
      mov ecx, 0
    ..Continue1:
      mul ecx
      mov ebx, [y2]
      sub ebx, [y1]
      mov edx, [y4]
      sub edx, [y3]
      cmp edx, 0
      je ..Zero2
      jmp ..Continue2
    ..Zero2:
      mov ebx, 0
      mov edx, 0
    ..Continue2:
      mul edx
      add eax, ebx
      cmp eax, 0
      je ..PerpendicularFound
```

```
        mov esp, [tmpStack]
        local ..Over
        jmp ..Over
..PerpendicularFound:
        cinvoke printf, perpMes1                ; Пара, состоящая из отрезока с координатами
концевых точек: (
        cinvoke printf, digit, dword [x1]
        cinvoke printf, perpMes2                ; ;
        cinvoke printf, digit, dword [y1]
        cinvoke printf, perpMes3                ; ) и (
        cinvoke printf, digit, dword [x2]
        cinvoke printf, perpMes2
        cinvoke printf, digit, dword [y2]
        cinvoke printf, perpMes4                ; ), а также отрезка с координатами концевых
точек: (
        cinvoke printf, perpMes41
        cinvoke printf, digit, dword [x3]
        cinvoke printf, perpMes2
        cinvoke printf, digit, dword [y3]
        cinvoke printf, perpMes3
        cinvoke printf, digit, dword [x4]
        cinvoke printf, perpMes2
        cinvoke printf, digit, dword [y4]
        cinvoke printf, perpMes5                ; ")", 10, 13, 0
        cinvoke printf, perpMes6                ; "содержит перпендикулярные отрезки.", 10, 13,
0
        add [perpNum], 1
..Over:
        ;ret
}

;------------------------------------------------------------------------
macro Conclusion pNum {
  cinvoke printf, newStr
  cinvoke printf, conclusionMessage1           ; Всего нашлось
  cinvoke printf, digit, dword [pNum]
  cinvoke printf, conclusionMessage2           ; пар перпендикулярных отрезков.
}
```

ПРИЛОЖЕНИЕ 2. Список используемых источников.

1. https://newtonov.ru/chemu-ravno-chislo-pi/#:~:text=%D0%A0%D1%8F%D0%B4%20%D0%9D%D0%B8%D0%BB%D0%B0%D0%BA%D0%B0%D0%BD%D1%82%D0%B0.&text=%CF%80%20%3D%203%20%2B%204%2F(2,%D0%9F%D0%BE%20%D1%8D%D1%82%D0%BE%D0%BC%D1%83%20%D0%B8%D0%B4%D0%B5%D0%BC%20%D0%B4%D0%B0%D0%BB%D1%8C%D1%88%D0%B

2. https://www.frolov-lib.ru/books/bsp/v02/ch12_4.htm

3. https://drive.google.com/file/d/1cYZ68FZQJwEwntH5b9n8LPpLmHcvX7pN/view?usp=sharing