**ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ**

**НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИ УНИВЕРСИТЕТ «ВЫСШАЯ**

**ШКОЛА ЭКОНОМИКИ»**

Факультет компьютерных наук

Департамент программной инженерии

Программа, которая решает вопрос о нахождении пар перпендикулярных отрезков из N=4 отрезков, заданных координатами концевых точек

**Исполнитель**

Студент группы БПИ195

Баранова Е. В.

**Задание**

Разработать программу, которая решает вопрос о нахождении пар перпендикулярных отрезков из N=4 отрезков, заданных координатами концевых точек

Составление​​ программы

Программа действует следующим образом:

1. По точкам строит векторы
2. Для каждой пары векторов с помощью скалярного произведения вычисляется, являются ли они перпендикулярными
3. Результат выводить на экран после проверки всех пар векторов

В программе выделены две функции:

dot:

Функция ищет скалярное произведение двух векторов

is\_perpendicular:

Функция возвращает 1 если два вектора перпендикулярны или 0 в ином случае.

Текст программы:

format PE GUI

include 'win32ax.inc'

.data

;First line segment

a1\_x dd 0

a1\_y dd 0

b1\_x dd 1

b1\_y dd 0

;Second line segment

a2\_x dd 0

a2\_y dd 0

b2\_x dd 0

b2\_y dd 1

;Third line segment

a3\_x dd 0

a3\_y dd 0

b3\_x dd -1

b3\_y dd 0

;Fourth line segment

a4\_x dd 0

a4\_y dd 0

b4\_x dd 0

b4\_y dd -1

;First vector

v1\_x dd 0

v1\_y dd 0

;Second vector

v2\_x dd 0

v2\_y dd 0

;Third vector

v3\_x dd 0

v3\_y dd 0

;Fourth vector

v4\_x dd 0

v4\_y dd 0

;Perpendicularity flags

p12 dd 0

p13 dd 0

p14 dd 0

p23 dd 0

p24 dd 0

p34 dd 0

;Title string

title db 'Perpendicular line segments', 0

;Output format string

output\_format db 'Line segments:', 10, '(%d;%d),(%d;%d)', 10, '(%d;%d),(%d;%d)', 10, '(%d;%d),(%d;%d)', 10, '(%d;%d),(%d;%d)', 10, 10, 'Perpendicularity:', 10, '1-2: %d', 10, '1-3: %d', 10, '1-4: %d', 10, '2-3: %d', 10, '2-4: %d', 10, '3-4: %d', 0

;Output string

output db 256 dup(?)

.code

start:

;Calculate the first vector

mov eax, [b1\_x]

sub eax, [a1\_x]

mov [v1\_x], eax

mov eax, [b1\_y]

sub eax, [a1\_y]

mov [v1\_y], eax

;Calculate the second vector

mov eax, [b2\_x]

sub eax, [a2\_x]

mov [v2\_x], eax

mov eax, [b2\_y]

sub eax, [a2\_y]

mov [v2\_y], eax

;Calculate the third vector

mov eax, [b3\_x]

sub eax, [a3\_x]

mov [v3\_x], eax

mov eax, [b3\_y]

sub eax, [a3\_y]

mov [v3\_y], eax

;Calculate the fourth vector

mov eax, [b4\_x]

sub eax, [a4\_x]

mov [v4\_x], eax

mov eax, [b4\_y]

sub eax, [a4\_y]

mov [v4\_y], eax

;Find wheather the first and the second vectors are perpendicular

push [v1\_x]

push [v1\_y]

push [v2\_x]

push [v2\_y]

call is\_perpendicular

mov [p12], eax

;Find wheather the first and the third vectors are perpendicular

push [v1\_x]

push [v1\_y]

push [v3\_x]

push [v3\_y]

call is\_perpendicular

mov [p13], eax

;Find wheather the first and the fourth vectors are perpendicular

push [v1\_x]

push [v1\_y]

push [v4\_x]

push [v4\_y]

call is\_perpendicular

mov [p14], eax

;Find wheather the second and the third vectors are perpendicular

push [v2\_x]

push [v2\_y]

push [v3\_x]

push [v3\_y]

call is\_perpendicular

mov [p23], eax

;Find wheather the second and the fourth vectors are perpendicular

push [v2\_x]

push [v2\_y]

push [v4\_x]

push [v4\_y]

call is\_perpendicular

mov [p24], eax

;Find wheather the third and the fourth vectors are perpendicular

push [v3\_x]

push [v3\_y]

push [v4\_x]

push [v4\_y]

call is\_perpendicular

mov [p34], eax

;Format the output

cinvoke wsprintf, output, output\_format, [a1\_x], [a1\_y], [b1\_x], [b1\_y], [a2\_x], [a2\_y], [b2\_x], [b2\_y], [a3\_x], [a3\_y], [b3\_x], [b3\_y], [a4\_x], [a4\_y], [b4\_x], [b4\_y], [p12], [p13], [p14], [p23], [p24], [p34]

;Create the output window

invoke MessageBox, 0, output, title, 0

;Exit the program

invoke ExitProcess, 0

.end start

;Calculates a dot product of two vectors

;Call convention - stdcall

;Used registers - EAX, EBX

;Parameters - coordinates of two vectors on the stack

;Return value - a dot product of two vectors in the EAX register

proc dot

mov eax, [esp + 16]

imul eax, [esp + 8]

mov ebx, eax

mov eax, [esp + 12]

imul eax, [esp + 4]

add eax, ebx

ret 16

endp

;Finds wheather two vectors are perpendicular

;Call convention - stdcall

;Used registers - EAX, EBX

;Parameters - coordinates of two vectors on the stack

;Return value - 1 if two vectors are perpendicular or 0 in the EAX register

proc is\_perpendicular

mov eax, esp

push dword [eax + 16]

push dword [eax + 12]

push dword [eax + 8]

push dword [eax + 4]

call dot

cmp eax, 0

je perpendicular

not\_perpendicular:

mov eax, 0

ret 16

perpendicular:

mov eax, 1

ret 16

endp

Тестирование программы



