**ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ**

**НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ**

**"ВЫСШАЯ ШКОЛА ЭКОНОМИКИ"**

Факультет компьютерных наук

Департамент программной инженерии

|  |  |  |
| --- | --- | --- |
|  | **Микропроект №1  Пояснительная записка**  Исполнитель  Студент группы БПИ194  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/В.Д. Хворостяной /  "\_\_\_"\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2020 г. | |
|  |  |

СОДЕРЖАНИЕ

[1. УСЛОВИЕ ЗАДАНИЯ 3](#_Toc55303198)

[1.1. Текст задания 3](#_Toc55303199)

[2. МЕТОДЫ И АЛГОРИТМЫ ПРОГРАММЫ 4](#_Toc55303200)

[2.1. Методы работы с данными 4](#_Toc55303201)

[2.2. Алгоритм нахождения произведения матриц 4](#_Toc55303202)

[3. ОПИСАНИЕ ДАННЫХ В ПРОГРАММЕ 5](#_Toc55303203)

[3.1. Входные данные 5](#_Toc55303204)

[3.2. Выходные и промежуточные данные 5](#_Toc55303205)

[4. ТЕСТИРОВАНИЕ ПРОГРАММЫ 6](#_Toc55303206)

[4.1. Положительные тесты 6](#_Toc55303207)

[4.2. Отрицательные тесты 8](#_Toc55303208)

[4.3. Тесты с переполнением 9](#_Toc55303209)

[4.4. Тесты с прерыванием программы 9](#_Toc55303210)

[ПРИЛОЖЕНИЕ 1 11](#_Toc55303211)

[ПРИЛОЖЕНИЕ 2 (ТЕКСТ ПРОГРАММЫ) 12](#_Toc55303212)

1. УСЛОВИЕ ЗАДАНИЯ

Текст задания

Разработать программу умножения матриц порядка N=4 при условии размещения элементов матриц в линейном массиве по строкам.

1. **МЕТОДЫ И АЛГОРИТМЫ ПРОГРАММЫ**

Методы работы с данными

Для ввода и обработки данных в программе используются циклы на основе условных и безусловных переходов, заменяющие собой циклы while – jmp+jle/jl/jge, a также условную инструкцию – je/jl/jge. Также в программе используются section data с данными использующими в программе.   
  
Основные структуры - массивы, строки, числа.

Алгоритм нахождения произведения матриц

Для нахождения произведения двух матриц применяется прямое определение: A\*B=C, Cij = ∑nk=1Aik\*Bkj. Для вычислений значений используются циклы.

1. ОПИСАНИЕ ДАННЫХ В ПРОГРАММЕ

Входные данные

Фактически входные данные ограничены 10 байтами для каждого введенного целого числа. В противном случае происходит переполнение. Значения будут некорректными.

В программе описаны ограничения для 10000 по модулю. В таких пределах данные всегда будут корректные. Фактическое ограничение несколько больше, но смысловой нагрузки незначительное повышение ограничений не имеет, поскольку нет необходимости вычислять матрицы с большими числовыми значениями.

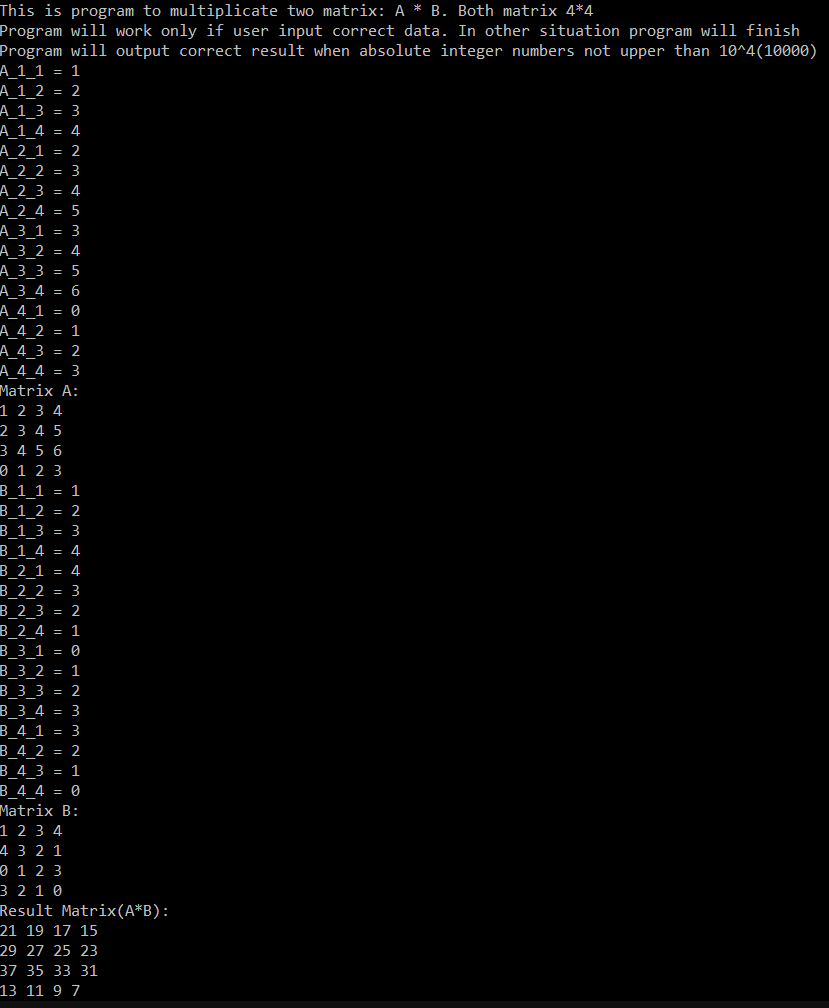
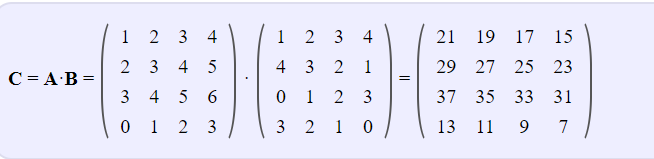
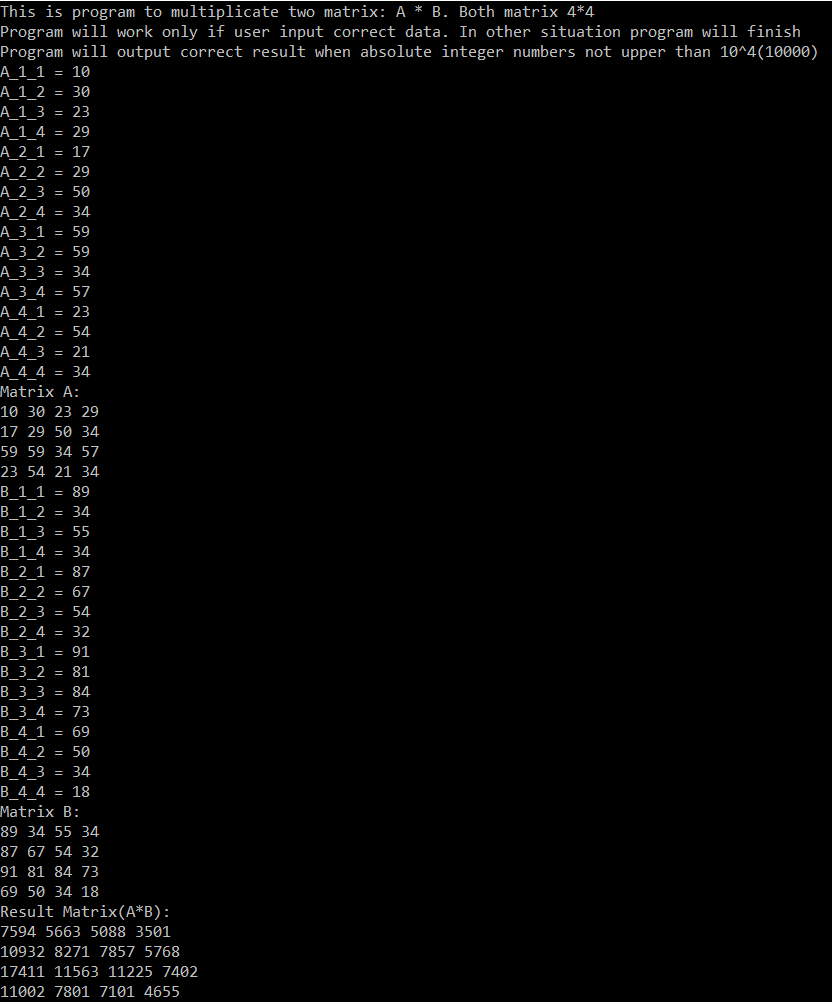
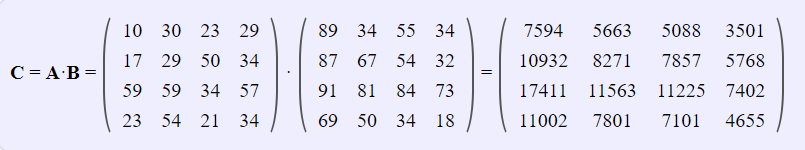
При вводе нечисловых символов программа завершит работу заполним все оставшиеся входные значения 0.

Выходные и промежуточные данные

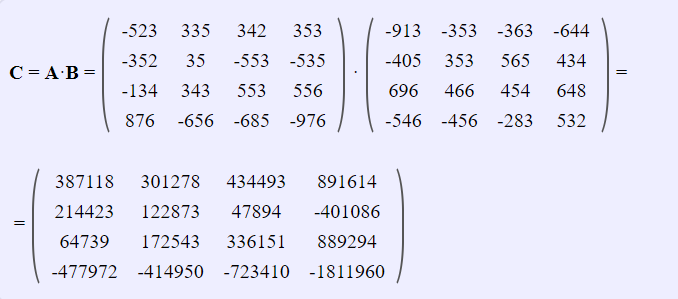
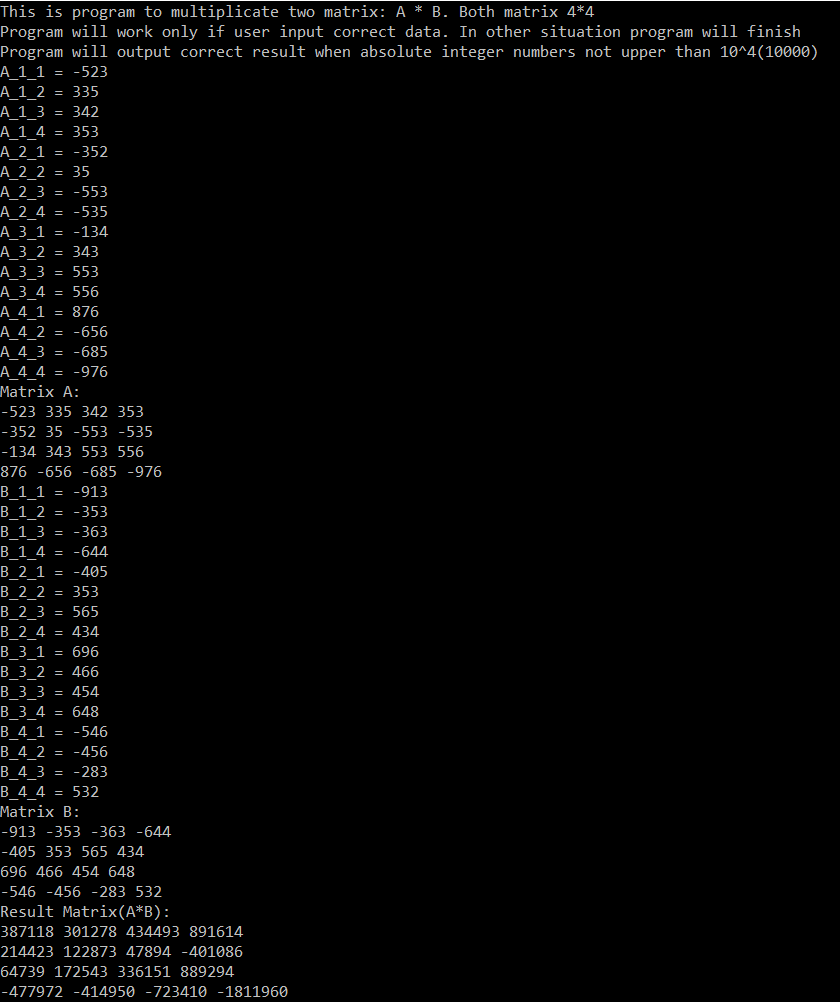
С учетом ограничений по вводу чисел в 10000 промежуточные значения как и выходные не превышают 4 \* 10^8 по модулю. При вводе больших чисел, чем описано в программе приведет к увеличению промежуточных и выходных данных, а также может привести их к переполнению, и как следствие некорректным данным.

1. ТЕСТИРОВАНИЕ ПРОГРАММЫ

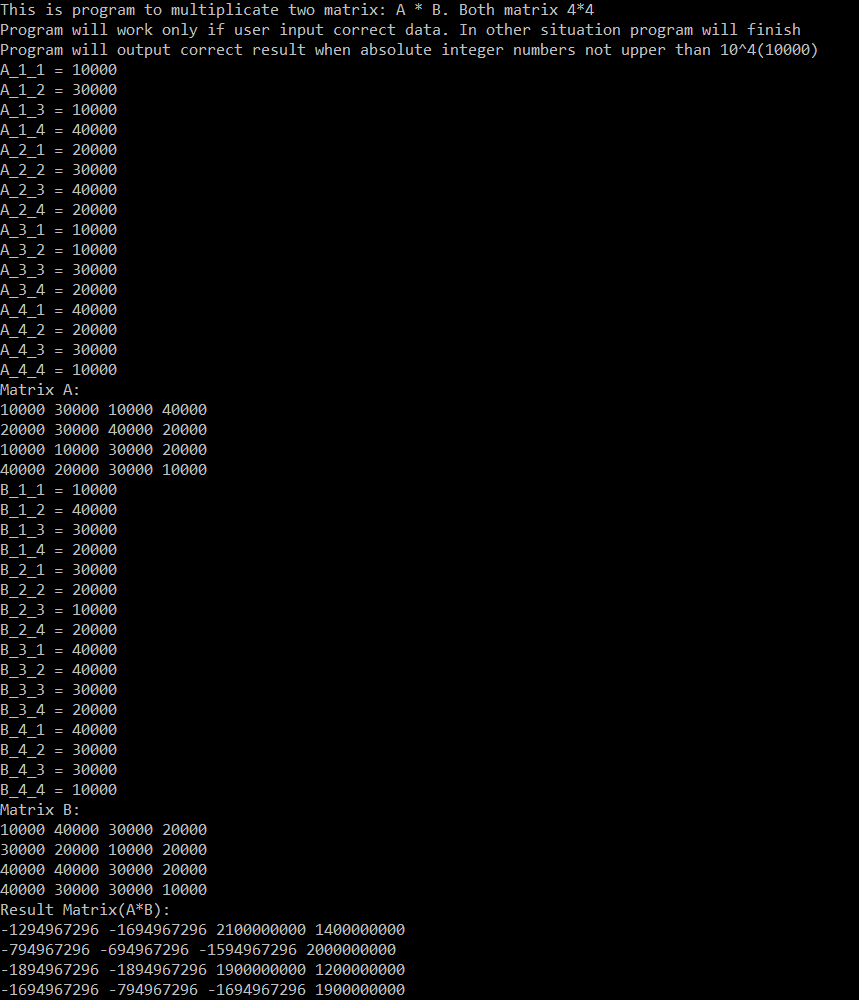
Положительные тесты

Проверка  
Проверка

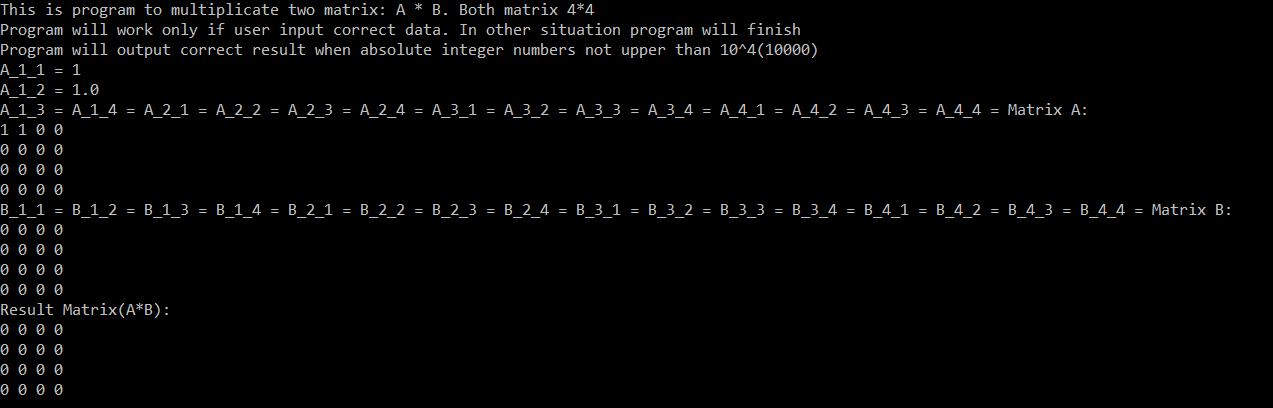
Отрицательные тесты

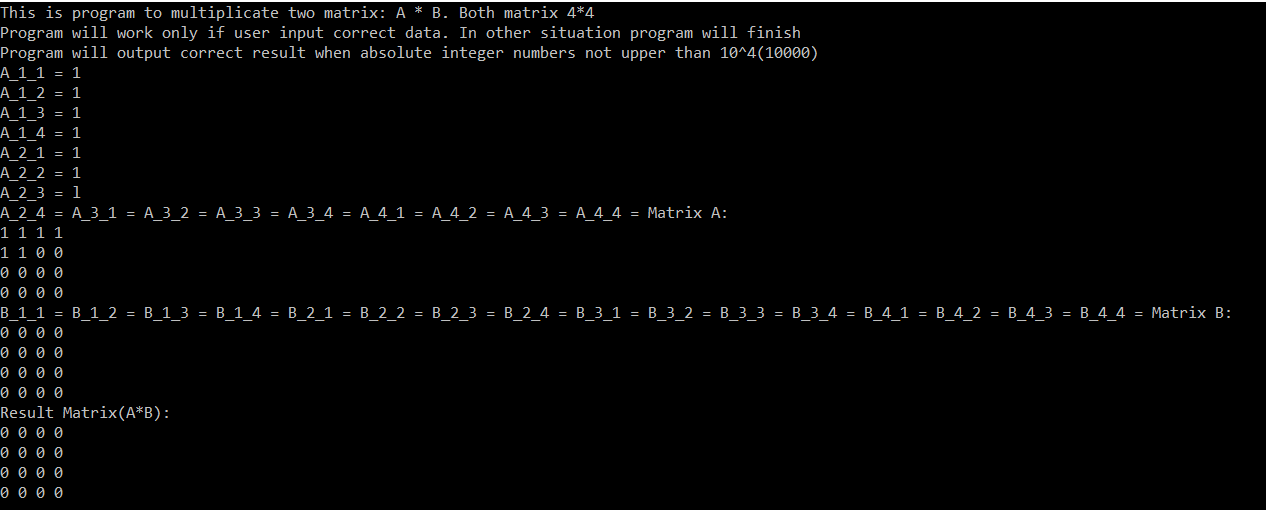


Тесты с переполнением



Тесты с прерыванием программы





ПРИЛОЖЕНИЕ 1

**СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ**

1. ГОСТ 19.404-79 Пояснительная записка. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
2. <http://softcraft.ru/edu/comparch/practice/asm86/03-subprog/sum1-32/sum.asm> - Пример программы на языке ассемблера.
3. <http://osinavi.ru/index.php?param1=4&param2=18&hidden=0&supp=1> - Теоритический материал по условным и безусловным переходам.

ПРИЛОЖЕНИЕ 2 (ТЕКСТ ПРОГРАММЫ)

**ТЕКСТ ПРОГРАММЫ**

format PE console

entry start

include 'C:\Fasm\INCLUDE\win32a.inc'

section '.data' data readable writable

strVecElemA db 'A\_%d\_%d = ', 0 ;для информации о вводе элемента в матрице А

strVecElemB db 'B\_%d\_%d = ', 0 ;для информации о вводе элемента в матрице В

strScanInt db '%d', 0 ;\

strVecElemOutEnter db '%d ', 0 ;/для ввода чисел

strStartInfo db 'This is program to multiplicate two matrix: A \* B', 10, 0 ;информация о программа

strMatrixSize db "Both matrix 4\*4", 10, 0 ;информация о размере матриц

strInstract db 'Program will work only if user input correct data. In other situation program will finish', 10, 0 ;информация касаемо ввода чисел в программе

strNumbers db 'Program will output correct result when absolute numbers not upper than 10^4(10000) ', 10, 0 ;информация для пользовател об ограничениях

strMatrixA db 'Matrix A:', 10, 0 ;информация о матрице А

strMatrixB db 'Matrix B:', 10, 0 ;информация о матрице В

strMatrixRes db 'Result Matrix(A\*B):', 10, 0 ;информация о результирующе матрице

newLine db '', 10, 0 ;перенос строки

vec\_size dd 16 ;постоянное значение длины массивов

sum dd 0 ;переменная для суммы

i dd ? ;индекс первого уровня

tmp dd ? ;дополнительная переменная для хранения

vec rd 16 ;массив А

vecB rd 16 ;массив В

vecRes rd 16 ;итоговый массив

j dd ? ;индекс 2ого уровня

cicleUp dd ? ;счетчик внешнего цикла

cicleDown dd ? ;счетчик внутреннего цикла

section '.code' code readable executable

start:

push strStartInfo ;\

call [printf] ; \

push strMatrixSize ; \

call [printf] ; \

; basic information to user

push strInstract ; /

call [printf] ; /

push strNumbers ; /

call [printf] ;/

mov eax, [vec\_size]

cmp eax, 0

jg getVectorA ;start work with matrix

;input matrix A

getVectorA:

mov eax, 1

mov ebx, vec ; ebx = &vec

getVecLoopA:

mov ecx, 1

getOneElemA:

mov [tmp], ebx

mov [i], ecx

push ecx

mov [j], eax

push eax

push strVecElemA

call [printf] ;print elem info

push ebx

push strScanInt

call [scanf] ;scan to get elem

mov ecx, [i]

mov eax, [j]

inc ecx

mov ebx, [tmp]

add ebx, 4

cmp ecx, 5

jl getOneElemA ;update cicle data

inc eax

cmp eax, 5

jl getVecLoopA

cmp eax, 5

jge endInputMatrixA ;continue or finish input

endInputMatrixA:

push strMatrixA

call [printf] ;matrix A info

mov eax, 0

mov ebx, vec

;print matrix A

printLineA:

mov ecx, 0

printElemA:

mov [tmp], ebx

cmp eax, 16

jge getVectorB ;finish print A

mov [i], ecx

mov [j], eax

push dword [ebx]

push strVecElemOutEnter

call [printf] ;print elem

mov ecx, [i]

mov eax, [j]

inc ecx

inc eax

mov ebx, [tmp]

add ebx, 4

cmp ecx, 4

jl printElemA ;cicle to print elem in line

mov [i], ecx

mov [j], eax

push newLine

call [printf] ;\n

mov ecx, [i]

mov eax, [j]

cmp ecx, 4

je printLineA ;cicle to print - new line

;input Matrix B

getVectorB:

mov eax, 1

mov ebx, vecB ; ebx = &vec

getVecLoopB:

mov ecx, 1

getOneElemB:

mov [tmp], ebx

mov [i], ecx

push ecx

mov [j], eax

push eax

push strVecElemB

call [printf] ;print info about new elem

push ebx

push strScanInt ;scan to get elem

call [scanf]

mov ecx, [i]

mov eax, [j]

inc ecx

mov ebx, [tmp]

add ebx, 4

cmp ecx, 5

jl getOneElemB ;cicle to get elems

inc eax

cmp eax, 5

jl getVecLoopB

cmp eax, 5

jge endInputMatrixB ;continue cicle or finish

endInputMatrixB:

push strMatrixB ;base data to printB

call [printf]

mov eax, 0

mov ebx, vecB

;printB

printLineB:

mov ecx, 0

printElemB:

mov [tmp], ebx

cmp eax, 16

jge MulMatrix ;end print

mov [i], ecx

mov [j], eax

push dword [ebx]

push strVecElemOutEnter ;print elem

call [printf]

mov ecx, [i]

mov eax, [j]

inc ecx

inc eax

mov ebx, [tmp]

add ebx, 4

cmp ecx, 4

jl printElemB ;cicle to print elem in line

mov [i], ecx

mov [j], eax

push newLine

call [printf] ;\n

mov ecx, [i]

mov eax, [j]

cmp ecx, 4

je printLineB ;cicle to print - new line

MulMatrix: ;base data od cicles

mov edx, 0

mov esi, 0

cmp esi, 64

jge endMul

MulLine: ;base data of cicleUp

mov [cicleUp], 0

mov edi, 0

cmp esi, 64

jge endMul

MulElem: ;base data of cicleDown

cmp esi, 64

jge endMul

mov [cicleDown], 0

mov [sum], 0

cmp [cicleUp], 4

jge endLineMul

Multi:

cmp [cicleDown], 4

jge endElemMul

mov ecx, [vec+edx]

imul ecx, [vecB+edi]

add [sum], ecx ;sum elements

add edi, 16

add edx, 4

inc [cicleDown]

jmp Multi ;update cicleUp parametrs

endElemMul:

mov ecx, [vecRes+esi]

mov ecx, [sum]

mov [vecRes+esi], ecx ;update res matrix

add esi, 4

sub edx, 16

sub edi, 60

inc [cicleUp]

add ebx, 4

jmp MulElem ;update cicleUp parametrs

endLineMul:

add edx, 16 ;end line mul

jmp MulLine

endMul: ;start to print result

push strMatrixRes

call [printf]

mov eax, 0

mov ebx, vecRes

printLineRes: ;point to refresh ecx

mov ecx, 0

printElemRes: ;print result matrix

mov [tmp], ebx

cmp eax, 16

jge endOutputRes ;to out from program

mov [i], ecx

mov [j], eax

push dword [ebx]

push strVecElemOutEnter ;print element

call [printf]

mov ecx, [i]

mov eax, [j]

inc ecx

inc eax

mov ebx, [tmp]

add ebx, 4

cmp ecx, 4

jl printElemRes ;return if cicle is end

mov [i], ecx

mov [j], eax

push newLine ;\n

call [printf]

mov ecx, [i]

mov eax, [j]

cmp ecx, 4

je printLineRes ;return if cicle is end

endOutputRes:

finish:

call [getch]

push 0

call [ExitProcess]

;-------------------------------third act - including HeapApi--------------------------

section '.idata' import data readable

library kernel, 'kernel32.dll',\

msvcrt, 'msvcrt.dll',\

user32,'USER32.DLL'

include 'C:\Fasm\INCLUDE\api\user32.inc'

include 'C:\Fasm\INCLUDE\api\kernel32.inc'

import kernel,\

ExitProcess, 'ExitProcess',\

HeapCreate,'HeapCreate',\

HeapAlloc,'HeapAlloc'

include 'C:\Fasm\INCLUDE\api\kernel32.inc'

import msvcrt,\

printf, 'printf',\

scanf, 'scanf',\

getch, '\_getch'