

**ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»**

Факультет компьютерных наук
Образовательной программы бакалавриата «Программная инженерия»

Пояснительная записка

Исполнитель:
студент группы БПИ198
Здор Андрей Максимович

Москва 2020

Задание

Вариант №11

Разработать программу вычисления определителя квадратной матрицы порядка $N \leq 4$ при условии размещения элементов матрицы в линейном массиве по строкам.

Применяемые расчетные методы

$$|A_{n \times n}| = \sum_{j=1}^n a_{1j} \times A_{1j}, \text{ где } A_{n \times n} - \text{квадратная матрица } n \text{ на } n, a_{1j} - \text{элемент матрицы } A, A_{1j} \\ - \text{алгебраическое дополнение элемента } a_{1j}.$$

Текст программы

```
format PE console
```

```
entry start
```

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

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

```
section '.data' data readable writable
```

```
;Ввод N
```

```
strVecSize db 'Input size of Matrix A([1 ; 4]): ', 0
```

```
;При неверном входном N
```

```
strIncorSize db 'Incorrect size of Matrix = %d', 10, 0
```

```
;Ввод элементов матрицы
```

```
strVecElemI db 'A[%d] = ', 0
```

```
;Считывание int
```

```
strScanInt db '%d', 0
```

```
;Вывод определителя
```

```
DetOut db 'DetA = %d', 0
```

```
;Размер массива
```

```
vec_size dd 0
```

;Временная переменная для сохранения счетчика при считывании массива

i dd ?

;Переменные для сохранения промежуточных вычислений определителя

tmp dd ?

tmp2 dd ?

tmp3 dd ?

;Матрица записанная по строкам

vec rd 100

;-----

section '.code' code readable executable

start:

; 1) Ввод матрицы

call VectorInput

mov eax, [vec_size]

; 2) Выбор нужной функции в соответствии с N

cmp eax, 1

je PrintDet1

cmp eax, 4

je PrintDet2

cmp eax, 9

je PrintDet3

cmp eax, 16

je PrintDet4

finish:

call [getch]

push 0

call [ExitProcess]

;-----

VectorInput:

```
push strVecSize
call [printf]
add esp, 4
```

```
push vec_size
push strScanInt
call [scanf]
add esp, 8
```

```
mov eax, [vec_size]
cmp eax, 0
jle FinishError
cmp eax, 4
jg FinishError
imul eax, eax
mov [vec_size], eax
jmp getVector
```

;If vec_size <= 0 || vec_size > 4

FinishError:

```
push [vec_size]
push strIncorSize
call [printf]
call [getch]
push 0
call [ExitProcess]
```

getVector:

```
xor ecx, ecx          ; ecx = 0
mov ebx, vec          ; ebx = &vec
```

getVecLoop:

```
mov [tmp], ebx
cmp ecx, [vec_size]
jge endInputVector      ; to end of loop
```

```
; input element
```

```
mov [i], ecx
push ecx
push strVecElemI
call [printf]
add esp, 8
```

```
push ebx
push strScanInt
call [scanf]
add esp, 8
```

```
mov ecx, [i]
inc ecx
mov ebx, [tmp]
add ebx, 4
jmp getVecLoop
```

```
endInputVector:
```

```
ret
```

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

```
PrintDet1:
```

```
;N = 1, Det = v[0]
```

```
push [vec]
push DetOut
call [printf]
jmp finish
```

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

PrintDet2:

;N = 2, Det = v[0]*v[3] - v[1]*v[2]

mov eax, [vec]

imul eax, [vec + 12]

mov [tmp], eax

mov eax, [vec + 8]

imul eax, [vec + 4]

sub [tmp], eax

push [tmp]

push DetOut

call [printf]

jmp finish

;-----

PrintDet3:

;N = 3, Det = v[0]*(v[4]*v[8]-v[5]*v[7])-v[1]*(v[3]*v[8]-v[6]*v[5])+v[2]*(v[3]*v[7]-v[4]*v[6])

mov eax, [vec + 4*4]

imul eax, [vec + 4*8]

mov [tmp], eax

mov eax, [vec + 4*5]

imul eax, [vec + 4*7]

sub [tmp], eax

mov eax, [tmp]

imul eax, [vec]

mov [tmp], eax

mov eax, [vec + 4*3]

imul eax, [vec + 4*8]

mov [tmp2], eax

mov eax, [vec + 4*5]

imul eax, [vec + 4*6]

sub [tmp2], eax

```

mov eax, [tmp2]
imul eax, [vec + 4]
sub [tmp], eax
mov eax, [vec + 4*3]
imul eax, [vec + 4*7]
mov [tmp2], eax
mov eax, [vec + 4*4]
imul eax, [vec + 4*6]
sub [tmp2], eax
mov eax, [tmp2]
imul eax, [vec + 4*2]
add [tmp], eax

```

```

push [tmp]
push DetOut
call [printf]
jmp finish

```

;-----

PrintDet4:

;N = 4, Det = $v[0]*A_0 + v[1]*A_1 + v[2]*A_2 + v[3]*A_3$, где A_i - алгебраическое дополнение элемента $v[i]$

```

mov eax, [vec + 4*10]
imul eax, [vec + 4*15]
mov [tmp], eax
mov eax, [vec + 4*14]
imul eax, [vec + 4*11]
sub [tmp], eax
mov eax, [tmp]
imul eax, [vec + 5*4]
mov [tmp], eax

```

```
mov eax, [vec + 4*9]
imul eax, [vec + 4*15]
mov [tmp2], eax
mov eax, [vec + 4*11]
imul eax, [vec + 4*13]
sub [tmp2], eax
mov eax, [tmp2]
imul eax, [vec + 4*6]
sub [tmp], eax
mov eax, [vec + 4*9]
imul eax, [vec + 4*14]
mov [tmp2], eax
mov eax, [vec + 4*10]
imul eax, [vec + 4*13]
sub [tmp2], eax
mov eax, [tmp2]
imul eax, [vec + 4*7]
add [tmp], eax
mov eax, [tmp]
imul eax, [vec]
mov [tmp3], eax
```

```
mov eax, [vec + 4*10]
imul eax, [vec + 4*15]
mov [tmp], eax
mov eax, [vec + 4*11]
imul eax, [vec + 4*14]
sub [tmp], eax
mov eax, [tmp]
imul eax, [vec + 4*4]
mov [tmp], eax
```



```
mov eax, [vec + 4*8]
imul eax, [vec + 4*15]
mov [tmp2], eax
mov eax, [vec + 4*11]
imul eax, [vec + 4*12]
sub [tmp2], eax
mov eax, [tmp2]
imul eax, [vec + 4*6]
sub [tmp], eax
mov eax, [vec + 4*8]
imul eax, [vec + 4*14]
mov [tmp2], eax
mov eax, [vec + 4*10]
imul eax, [vec + 4*12]
sub [tmp2], eax
mov eax, [tmp2]
imul eax, [vec + 4*7]
add [tmp], eax
mov eax, [tmp]
imul eax, [vec + 4]
sub [tmp3], eax
```

```
mov eax, [vec + 4*9]
imul eax, [vec + 4*15]
mov [tmp], eax
mov eax, [vec + 4*11]
imul eax, [vec + 4*13]
sub [tmp], eax
mov eax, [tmp]
imul eax, [vec + 4*4]
mov [tmp], eax
```

```
mov eax, [vec + 4*8]
imul eax, [vec + 4*15]
mov [tmp2], eax
mov eax, [vec + 4*11]
imul eax, [vec + 4*12]
sub [tmp2], eax
mov eax, [tmp2]
imul eax, [vec + 4*5]
sub [tmp], eax
mov eax, [vec + 4*8]
imul eax, [vec + 4*13]
mov [tmp2], eax
mov eax, [vec + 4*9]
imul eax, [vec + 4*12]
sub [tmp2], eax
mov eax, [tmp2]
imul eax, [vec + 4*7]
add [tmp], eax
mov eax, [tmp]
imul eax, [vec + 8]
add [tmp3], eax
```

```
mov eax, [vec + 4*9]
imul eax, [vec + 4*14]
mov [tmp], eax
mov eax, [vec + 4*10]
imul eax, [vec + 4*13]
sub [tmp], eax
mov eax, [tmp]
imul eax, [vec + 4*4]
mov [tmp], eax
```

```

mov eax, [vec + 4*8]
imul eax, [vec + 4*14]
mov [tmp2], eax
mov eax, [vec + 4*10]
imul eax, [vec + 4*12]
sub [tmp2], eax
mov eax, [tmp2]
imul eax, [vec + 4*5]
sub [tmp], eax
mov eax, [vec + 4*8]
imul eax, [vec + 4*13]
mov [tmp2], eax
mov eax, [vec + 4*9]
imul eax, [vec + 4*12]
sub [tmp2], eax
mov eax, [tmp2]
imul eax, [vec + 4*6]
add [tmp], eax
mov eax, [tmp]
imul eax, [vec + 3*4]
sub [tmp3], eax

push [tmp3]
push DetOut
call [printf]
jmp finish

```

```

;-----

```

```

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'
```

Ограничения входных данных

N должно быть целым числом от 1 до 4.

Элементы матрицы должны быть целыми числами от -97 до 97.

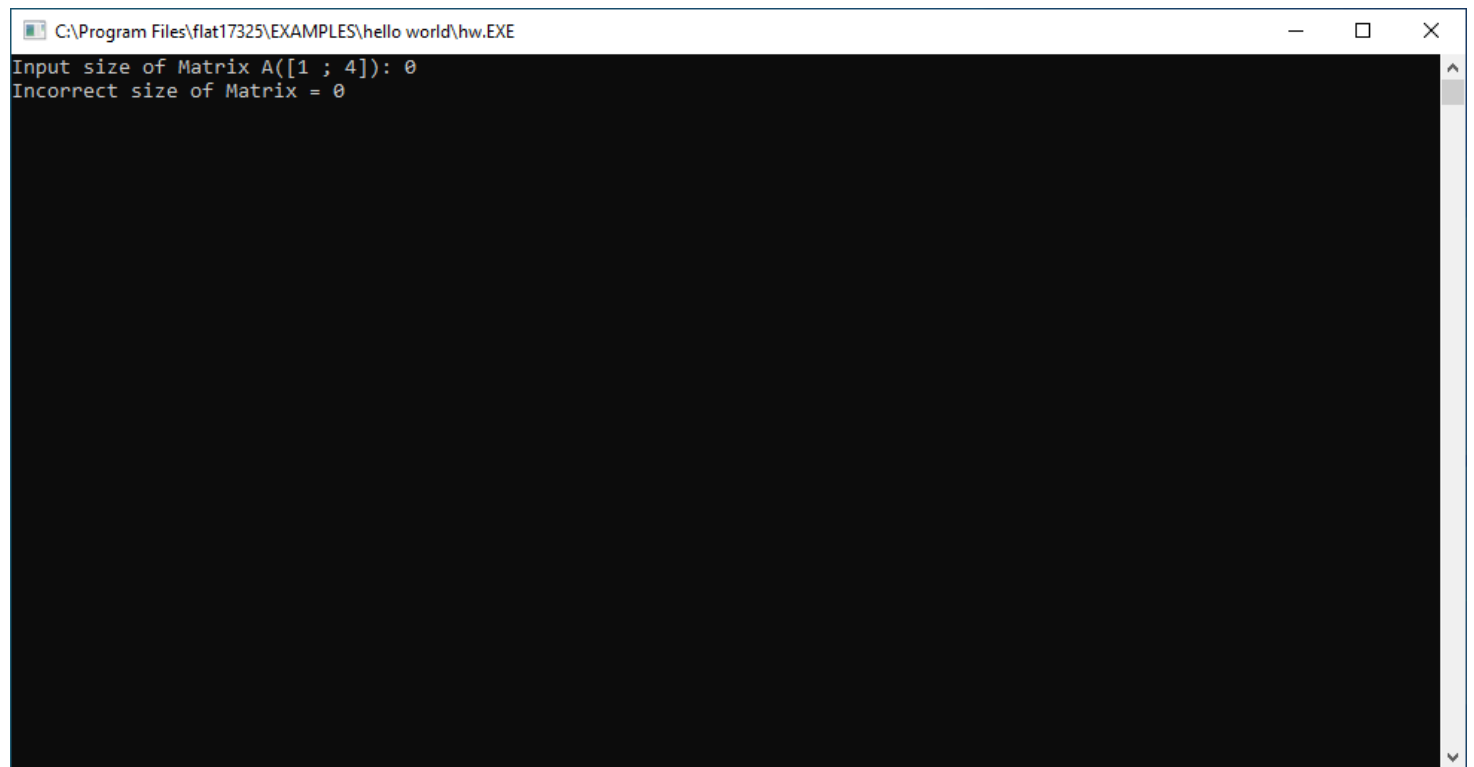
Элементы нумеруются и вводятся построчно.

Выходные данные

Результатом работы программы является строка вида: 'DetA = N' где N – определитель исходной матрицы.

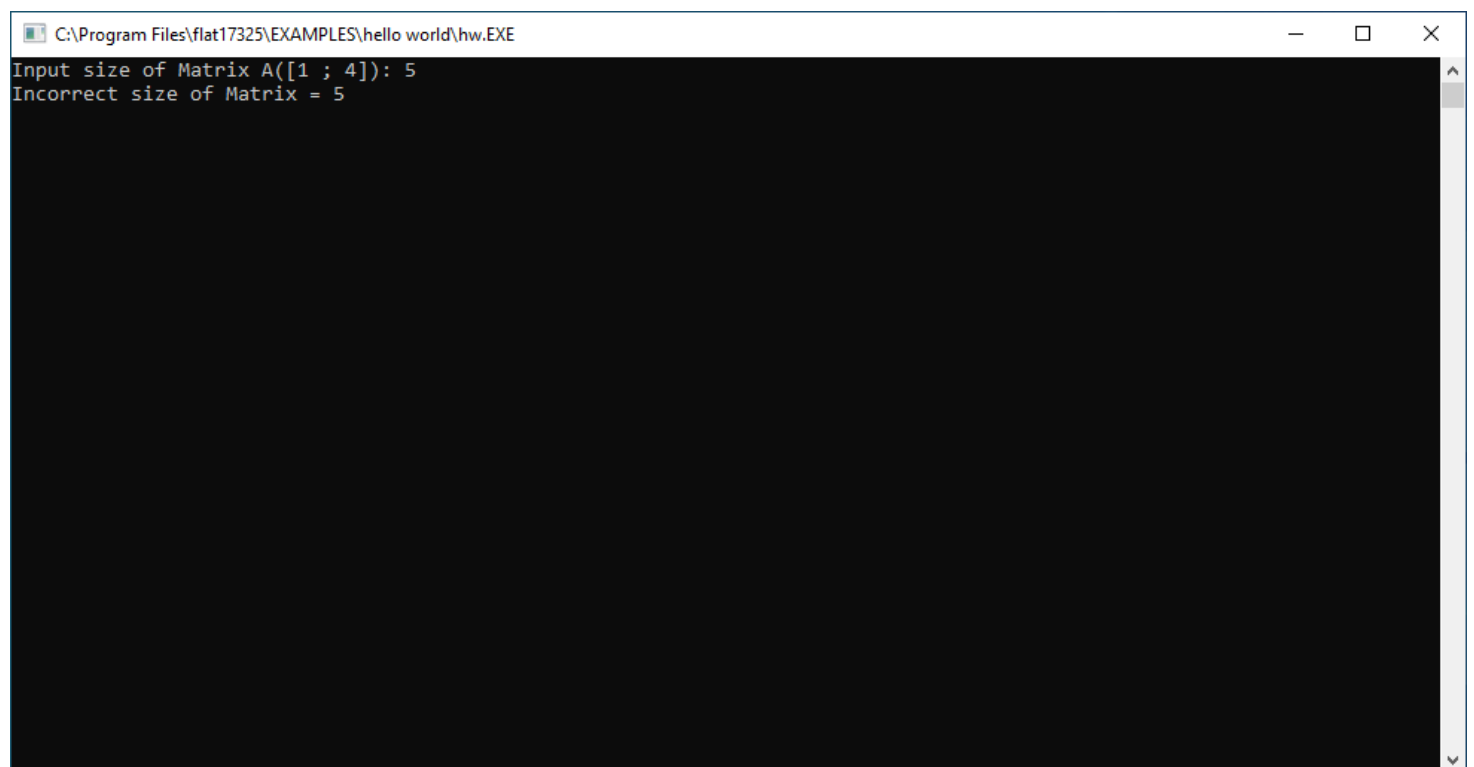
Тесты

No1



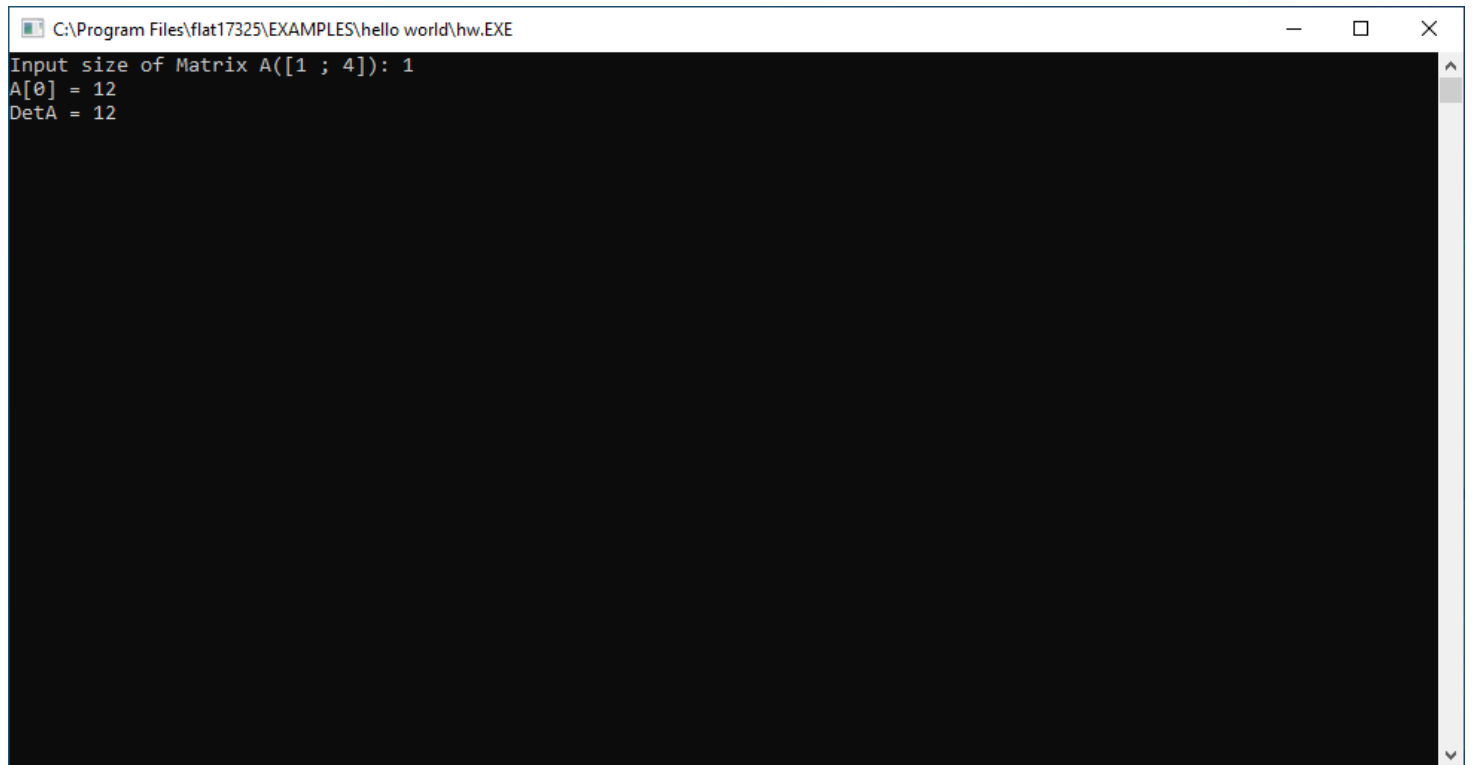
```
C:\Program Files\flat17325\EXAMPLES\hello world\hw.EXE
Input size of Matrix A([1 ; 4]): 0
Incorrect size of Matrix = 0
```

No2



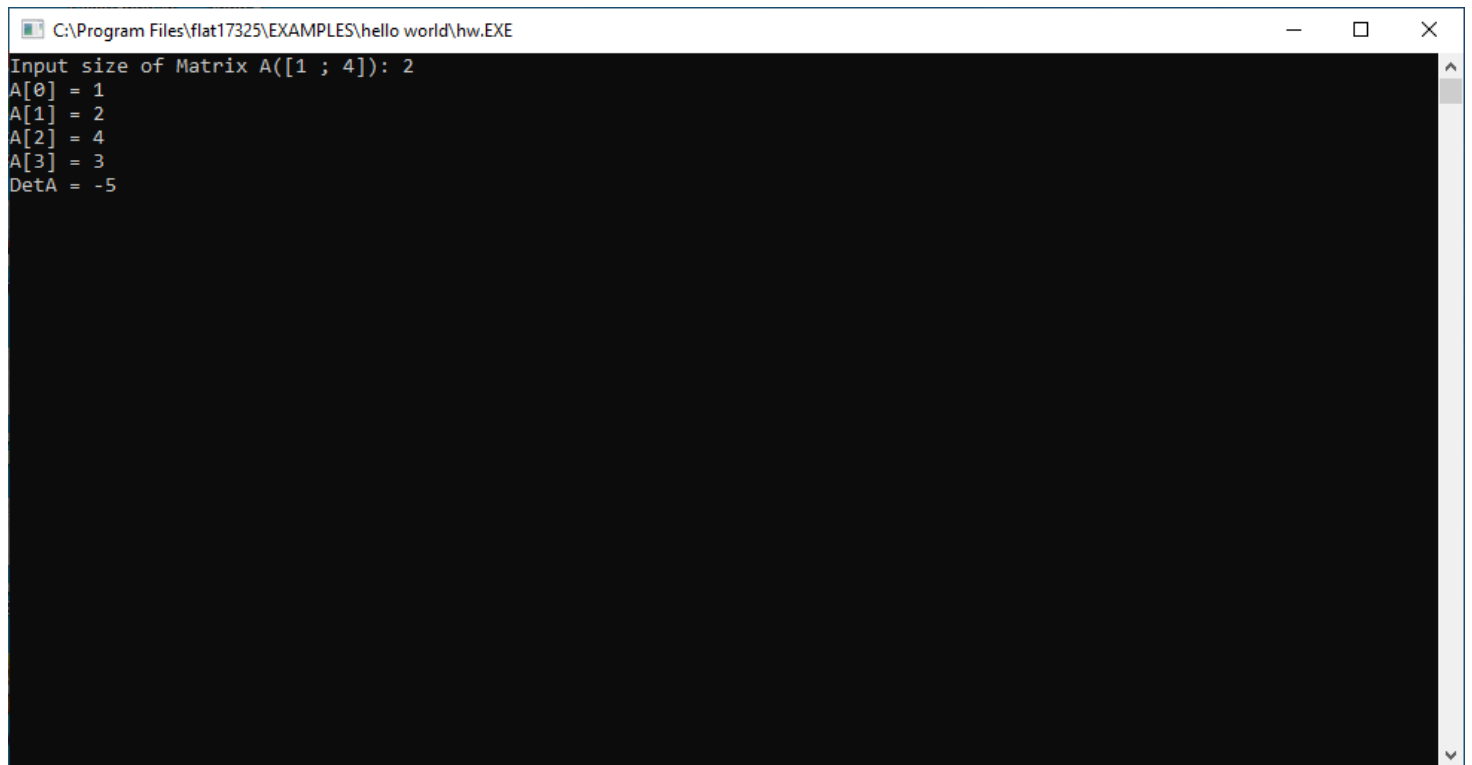
```
C:\Program Files\flat17325\EXAMPLES\hello world\hw.EXE
Input size of Matrix A([1 ; 4]): 5
Incorrect size of Matrix = 5
```

No3



```
C:\Program Files\flat17325\EXAMPLES\hello world\hw.EXE
Input size of Matrix A([1 ; 4]): 1
A[0] = 12
DetA = 12
```

No4



```
C:\Program Files\flat17325\EXAMPLES\hello world\hw.EXE
Input size of Matrix A([1 ; 4]): 2
A[0] = 1
A[1] = 2
A[2] = 4
A[3] = 3
DetA = -5
```

№5

```
C:\Program Files\flat17325\EXAMPLES\hello world\hw.EXE
Input size of Matrix A([1 ; 4]): 3
A[0] = 1
A[1] = 0
A[2] = 0
A[3] = 0
A[4] = 1
A[5] = 0
A[6] = 0
A[7] = 0
A[8] = 1
DetA = 1
```

№6

```
C:\Program Files\flat17325\EXAMPLES\hello world\hw.EXE
Input size of Matrix A([1 ; 4]): 4
A[0] = 1
A[1] = 2
A[2] = 3
A[3] = 4
A[4] = 8
A[5] = 7
A[6] = 6
A[7] = 5
A[8] = 1
A[9] = 0
A[10] = 1
A[11] = 0
A[12] = 0
A[13] = 1
A[14] = 0
A[15] = 1
DetA = 0
```