

Учреждение образования
«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ»
Кафедра информатики

Отчет по лабораторной работе №5

Работа с двумерными массивами

Выполнил: студент гр. 053502
Герчик Артём Вадимович

Руководитель: ст. преподаватель
Шиманский В.В.

Минск 2021

СОДЕРЖАНИЕ

1. Введение
2. Постановка задачи
3. Программная реализация
4. Выводы

Литература

Приложение

1. Введение

Целью данной работы является изучить следующий материал:

- 1) Представление двумерного массива в языке Ассемблер.
- 2) Понятие индексной адресации.
- 3) Использование регистров BX, SI, DI для адресации данных.
- 4) Организация вложенных циклов в языке Ассемблер

2. Постановка задачи

2.1. Текст задания

Пятое задание посвящается двумерным массивам. Каждый из них является прямоугольным, если не указано обратное (количество строк может быть отличным от количества столбцов). Для задания массива необходимо ввести количество строк, затем количество столбцов и затем все элементы. При выводе на экран элементы одного столбца матрицы необходимо показывать друг под другом.

2.2. Условие задания

С клавиатуры вводятся размерность N и квадратная матрица размерности $N \times N$. Найти в верхнем треугольнике, образуемом диагоналями максимальный элемент, а в нижнем треугольнике - минимальный. Затем увеличить минимальный элемент на 30%, а максимальный уменьшить на 30% от разности найденных максимального и минимального элементов. Повторять процедуру до тех пор, пока разность между элементами превышает 10%.

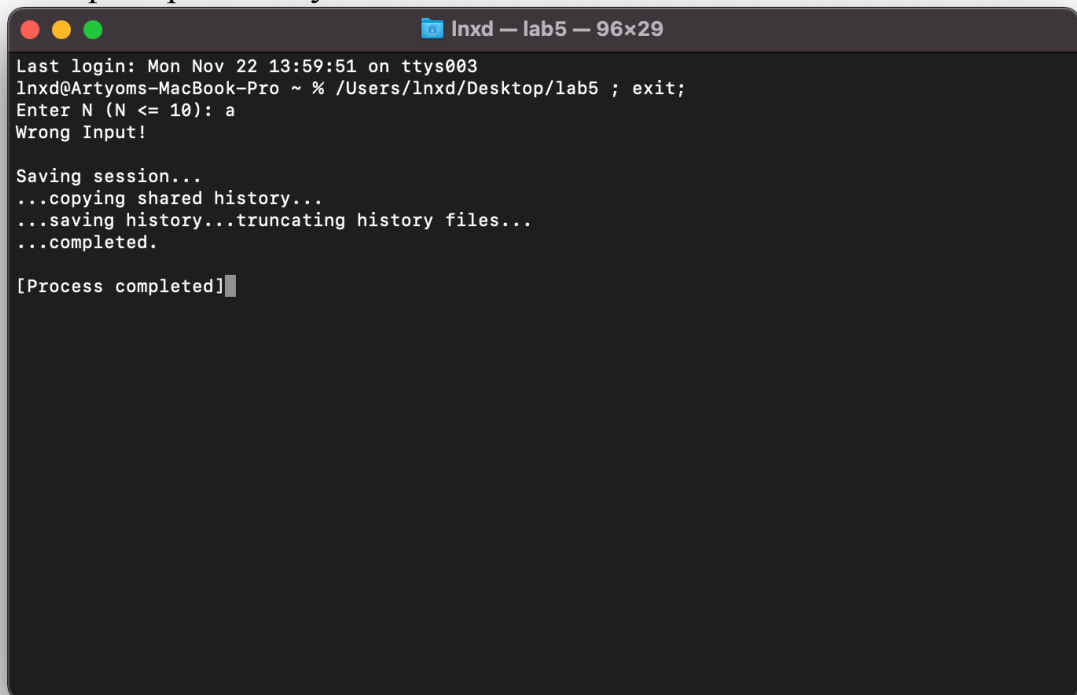
3. Программная реализация

3.1. В главной программе вводится размерность квадратной матрицы, после вводятся значения этой матрицы, затем идет вывод верхнего и нижнего треугольников, затем идет выполнение основного задания.

3.2. Результат можно видеть в окне консоли

3.3. Примеры:

3.3.1 Пример ввода буквы:



```
lnxd — lab5 — 96x29
Last login: Mon Nov 22 13:59:51 on ttys003
lnxd@Artyoms-MacBook-Pro ~ % /Users/lnxd/Desktop/lab5 ; exit;
Enter N (N <= 10): a
Wrong Input!

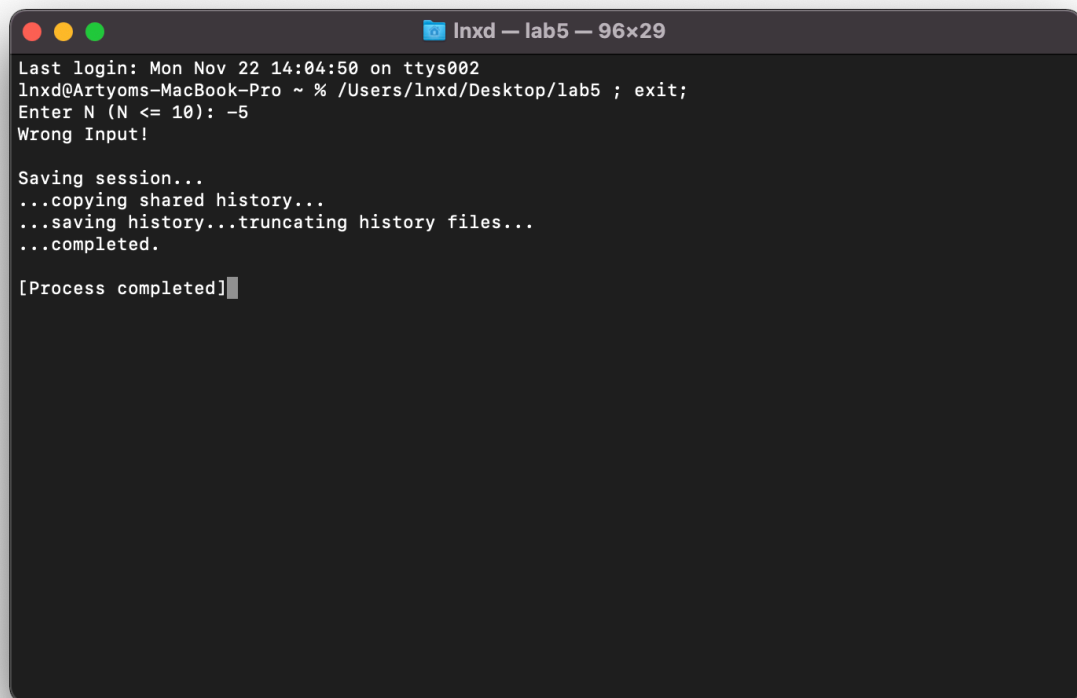
Saving session...
...copying shared history...
...saving history...truncating history files...
...completed.

[Process completed]
```

Результат: вывод сообщения об ошибке и завершение программы.

3.3.2 Пример ввода невалидного числа:

N = -5 (размерность не может быть меньше, либо равной нулю)

A screenshot of a macOS terminal window titled "lnxd — lab5 — 96x29". The window shows the following text: "Last login: Mon Nov 22 14:04:50 on ttys002", "lnxd@Artyoms-MacBook-Pro ~ % /Users/lnxd/Desktop/lab5 ; exit;", "Enter N (N <= 10): -5", "Wrong Input!", "Saving session...", "...copying shared history...", "...saving history...truncating history files...", "...completed.", and "[Process completed]". The terminal has a dark background and standard macOS window controls at the top.

```
lnxd — lab5 — 96x29
Last login: Mon Nov 22 14:04:50 on ttys002
lnxd@Artyoms-MacBook-Pro ~ % /Users/lnxd/Desktop/lab5 ; exit;
Enter N (N <= 10): -5
Wrong Input!

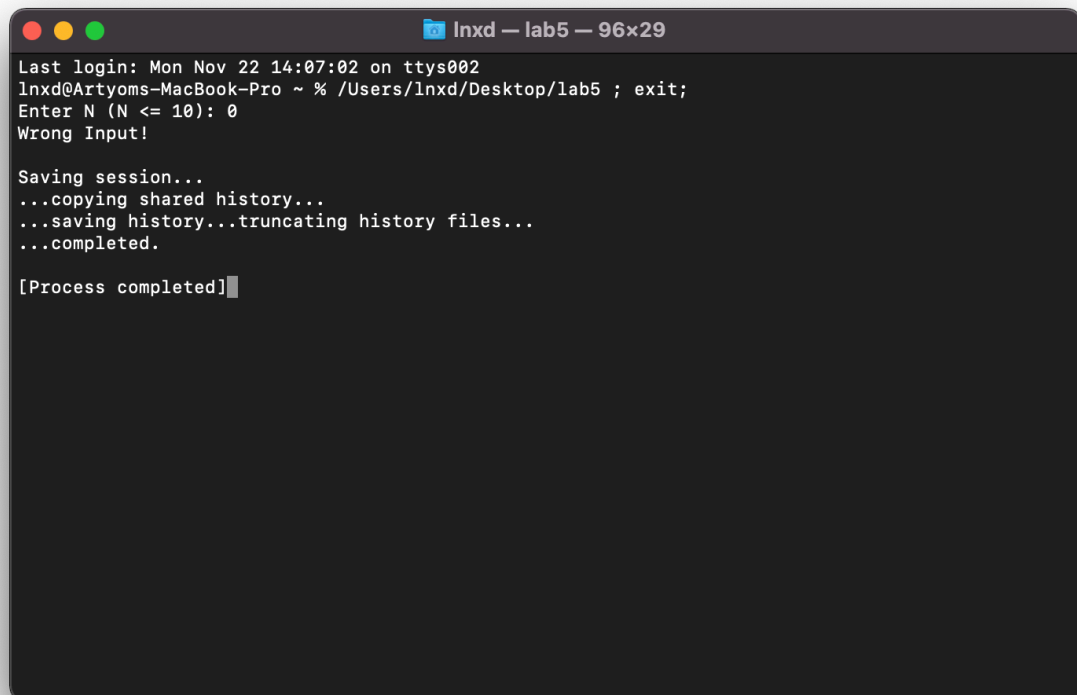
Saving session...
...copying shared history...
...saving history...truncating history files...
...completed.

[Process completed]
```

Результат: вывод сообщения об ошибке и завершение программы.

3.3.3 Пример ввода невалидного числа:

$N = 0$ (размерность не может быть меньше либо равной нулю)

A terminal window titled 'lnxd — lab5 — 96x29' with a dark background and light text. The window shows the following text: 'Last login: Mon Nov 22 14:07:02 on ttys002', 'lnxd@Artyoms-MacBook-Pro ~ % /Users/lnxd/Desktop/lab5 ; exit;', 'Enter N (N <= 10): 0', 'Wrong Input!', 'Saving session...', '...copying shared history...', '...saving history...truncating history files...', '...completed.', and '[Process completed]'.

```
lnxd — lab5 — 96x29
Last login: Mon Nov 22 14:07:02 on ttys002
lnxd@Artyoms-MacBook-Pro ~ % /Users/lnxd/Desktop/lab5 ; exit;
Enter N (N <= 10): 0
Wrong Input!

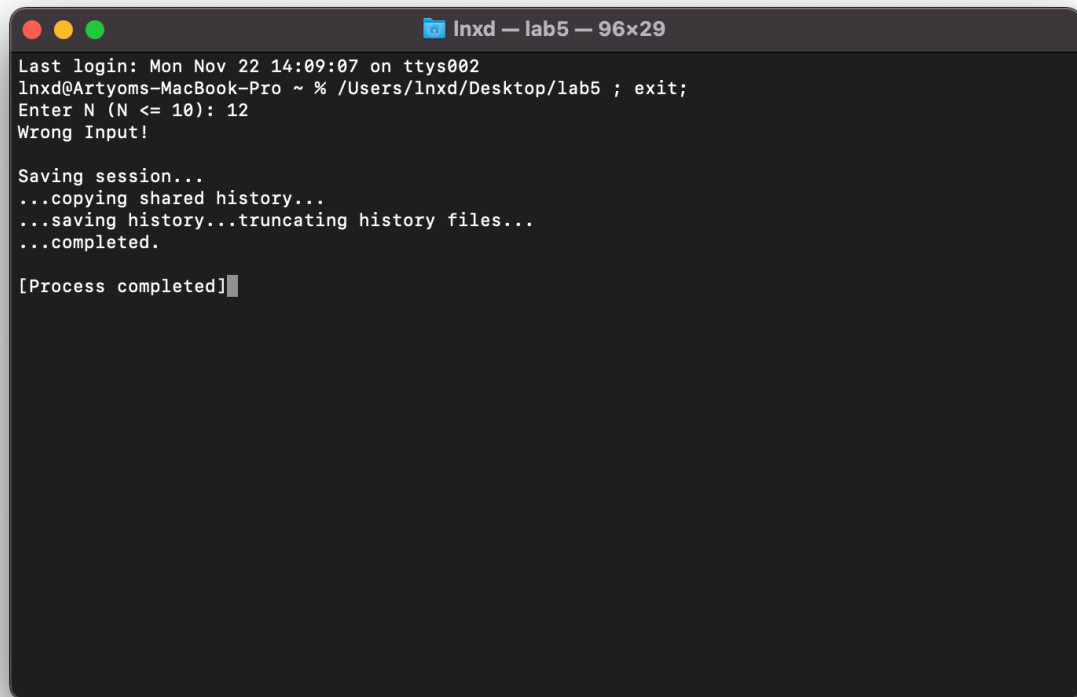
Saving session...
...copying shared history...
...saving history...truncating history files...
...completed.

[Process completed]
```

Результат: вывод сообщения об ошибке и завершение программы.

3.3.4 Пример ввода невалидного числа:

N = 12

A terminal window titled 'lnxd — lab5 — 96x29' with standard macOS window controls. The terminal text shows a login session, a command to run a program, and the program's execution. The program prompts for 'Enter N (N <= 10):' and receives the input '12'. It then outputs 'Wrong Input!', followed by session saving messages, and finally '[Process completed]' with a cursor.

```
lnxd — lab5 — 96x29
Last login: Mon Nov 22 14:09:07 on ttys002
lnxd@Artyoms-MacBook-Pro ~ % ./Users/lnxd/Desktop/lab5 ; exit;
Enter N (N <= 10): 12
Wrong Input!

Saving session...
...copying shared history...
...saving history...truncating history files...
...completed.

[Process completed]
```

Результат: вывод сообщения об ошибке и завершение программы.

3.3.5 Пример правильного исполнения программы:

N = 5

```
lnxd@Artyoms-MacBook-Pro ~ % /Users/lnxd/Desktop/lab5 ; exit;
Enter N (N <= 10): 5

[1][1] = 3
[1][2] = 5
[1][3] = 4
[1][4] = 2
[1][5] = 5
[2][1] = 8
[2][2] = 10
[2][3] = 3
[2][4] = 5
[2][5] = 1
[3][1] = 4
[3][2] = 6
[3][3] = 1
[3][4] = 4
[3][5] = 5
[4][1] = 3
[4][2] = 11
[4][3] = 2
[4][4] = 5
[4][5] = 6
[5][1] = 5
[5][2] = 7
[5][3] = 9
[5][4] = 8
[5][5] = 1

Matrix:
  3  5  4  2  5
  8 10  3  5  1
  4  6  1  4  5
  3 11  2  5  6
  5  7  9  8  1

Upper Triangle:  5  4  2  5  3  5  1  4  5  6
Max element at upper triangle: 6

Lower Triangle:  8  4  6  3 11  2  5  7  9  8
Min element at lower triangle: 2

10% is: 0.4

Max - 30%: 4.2
Min + 30%: 2.6

Max - 30%: 2.94
Min + 30%: 3.38

Difference lower than 10%

Saving session...
...copying shared history...
...saving history...truncating history files...
```

Результат: когда разница становится меньше 10% выводится сообщение об этом.

4. Выводы

На практике было изучено и опробовано, в соответствии с поставленной задачей:

Представление двумерного массива в языке Ассемблер.

Понятие индексной адресации.

Использование регистров BX, SI, DI для адресации данных.

Организация вложенных циклов в языке Ассемблер

Литература

1. Юров В.И. – «Assembler. Учебник для вузов. 2-ое издание, 2003 год».
2. Юров В.И. – «Assembler. Практикум. 2-ое издание, 2006 год».
3. Калашников О.А. – «Ассемблер - это просто. 2-ое издание, 2011 год».

Приложение

.model small

.data

matrix dw 10 dup(10 dup(?))

temp_matrix dw 10 dup(10 dup(?))

matrix_size dw ?

det dw 0

i dw 0

j dw 0

for_index dw 0

temp_index dw 0

minus_one dw -1

d dw 0

count dw 0

end_line db 13, 10, '\$'

maxlen db 3

len db 0

buffer db 6 dup(0)

string db 255 dup(0)

tempmsg db "Enter N (N <= 10): \$"

temp1msg db "Upper Triangle: \$"

temp2msg db "Lower Triangle: \$"

temp3msg db "Max Element at Upper Triangle: \$"

temp4msg db "Min Element at Lower Triangle: \$"

temp3msg db "Difference lower than 10%\$"

res_msg db "Matrix: \$"

enter_msg db "Enter matrix size: \$"

read_msg db "Reading from file\$"

write_msg db "Writing to file\$"

space db " \$"

;input_file db "input.txt", 0

;output_file db "output.txt", 0

;input_txt db "23 14 11 10 -7 -1 -5 -7 17 -5 16 1 14 -8 5 14 0 5 22 -6
11 13 5 12 16 15 8 24 8 17 20 10 16 22 -3 11 22 24 14 18 0 -1 19 4 17 24"

14 7 -3 -5 10 14 17 -3 23 4 -2 -5 10 1 -4 7 12 -1 5 16 14 24 17 0 20 -9 -2
22 -2 3 -3 -5 20 16 8 "

input db "3 5 4 2 5 8 10 3 5 1 4 6 1 4 5 3 11 2 5 6 5 7 9 8 1 1 4 5 6 7 7 6
5 5 7 5 4 6 7 8 7 5 4 2 1 23 3 34 43 2 21 1 1 2 43 45 6 5 7 9 8 1 1 4 5 6 7 7
6 5 5 7 5 4 6 7 8 7 5 4 2 1 23 3 34 43 2 21 1 1 2 43 45 6 5 7 9 8 1 1 4 5 6 7
7 6 5 5 7 5 4 6 7 8 7 5 4 2 1 23 3 34 43 2 21 1 1 2 43 45"

.code

to_string proc

push bx

push cx

push dx

push si

push di

mov di, 10

mov cx, 0

lea bx, buffer

test ax, ax

jns to_string_cycle

neg ax

mov [bx], '-'

inc bx

to_string_cycle:

push ax

mov al, len

inc ax

mov len, al

pop ax

inc cx

mov dx, 0

div di

add dx, 30h

push dx

cmp ax, 0

jne to_string_cycle

to_string_cycle_2:

pop [bx]

inc bx

```
loop to_string_cycle_2
inc bx
mov [bx], '$'
pop di
pop si
pop dx
pop cx
pop bx
ret
to_string endp
```

```
convert proc
push bx
push cx
push dx
push si
push di
convert_start:
lea bx, len
mov cx, [bx]
xor ch, ch
mov di, 0
mov si, 1
convert_cycle:
push si
mov si, cx
mov ax, [bx + si]
pop si
cmp al, '0'
jl convert_check
cmp al, '9'
jg convert_check
xor ah, ah
sub ax, 30h
mul si
add di, ax
mov ax, si
mov si, 10
```

```
mul si
jo convert_check_2
mov si, ax
loop convert_cycle
convert_plus:
mov ax, di
cmp ax, 32767
ja convert_error_point
jmp convert_return
convert_minus:
mov ax, di
cmp ax, 32768
ja convert_error_point
xor ax, ax
sub ax, di
convert_return:
pop di
pop si
pop dx
pop cx
pop bx
ret
convert_check:
cmp cx, 1
jne convert_error_point
cmp al, '-'
jne convert_error_point
cmp di, 0
jne convert_minus
convert_error_point:
jmp convert_start
convert_check_2:
cmp cx, 1
je convert_plus
dec cx
cmp cx, 1
jne convert_error_point
mov si, cx
```

```
mov bl, [bx + si]
cmp bl, '-'
je convert_minus
jmp convert_error_point
convert endp
```

```
input proc
push bx
push cx
push dx
push si
push di
in_start:
lea dx, maxlen
mov ah, 0ah
int 21h
lea bx, len
mov cx, [bx]
xor ch, ch
mov di, 0
mov si, 1
in_cycle:
push si
mov si, cx
mov ax, [bx + si]
pop si
cmp al, '0'
jl check
cmp al, '9'
jg check
xor ah, ah
sub ax, 30h
mul si
add di, ax
mov ax, si
mov si, 10
mul si
jo check_2
```



```
mov si, ax
loop in_cycle
plus:
mov ax, di
cmp ax, 32767
ja error_point
jmp return
minus:
mov ax, di
cmp ax, 32768
ja error_point
xor ax, ax
sub ax, di
return:
pop di
pop si
pop dx
pop cx
pop bx
ret
check:
cmp cx, 1
jne error_point
cmp al, '-'
jne error_point
cmp di, 0
jne minus
error_point:
jmp in_start
check_2:
cmp cx, 1
je plus
dec cx
cmp cx, 1
jne error_point
mov si, cx
mov bl, [bx + si]
cmp bl, '-'
```

```
je minus
jmp error_point
input endp
```

```
output proc
push bx
push cx
push dx
push si
push di
mov di, 10
mov cx, 0
lea bx, buffer
test ax, ax
jns out_cycle
neg ax
mov [bx], '-'
inc bx
out_cycle:
inc cx
mov dx, 0
div di
add dx, 30h
push dx
cmp ax, 0
jne out_cycle
out_cycle_2:
pop [bx]
inc bx
loop out_cycle_2
inc bx
mov [bx], '$'
lea dx, buffer
mov ah, 09h
int 21h
pop di
pop si
pop dx
```

```
pop cx
pop bx
ret
output endp
```

```
index proc
push dx
mov ax, i
mul for_index
add ax, j
add ax, j
pop dx
ret
index endp
```

```
show_matrix proc
push ax
push cx
push dx
push di
push si
mov di, 0
show_matrix_:
mov cx, 0
show_matrix_2:
mov i, di
mov j, cx
call index
mov si, ax
mov ax, [bx + si]
cmp ax, 10
jge skip2
cmp ax, 0
jl skip2
push ax
lea dx, space
mov ah, 09h
int 21h
```

```
pop ax
skip2:
call output
inc cx
cmp cx, matrix_size
jne show_matrix_2
lea dx, end_line
mov ah, 09h
int 21h
inc di
cmp di, matrix_size
jne show_matrix_
pop si
pop di
pop dx
pop cx
pop ax
ret
show_matrix endp
```

```
create proc
push i
push j
push ax
push bx
push cx
push dx
push si
push di
mov ax, d
add ax, 2
mov d, ax
mov bx, 1
;inc bx
new:
mov dx, 0
new2:
cmp j, dx
```

```
je skip
mov i, bx
push j
mov j, dx
call index
pop j
mov si, ax
push [matrix + si]
mov si, temp_index
pop [temp_matrix + si]
add si, 2
mov temp_index, si
skip:
mov ax, matrix_size
dec ax
cmp dx, ax
jne next
mov si, temp_index
add si, d
mov temp_index, si
next:
inc dx
cmp dx, matrix_size
jne new2
inc bx
cmp bx, matrix_size
jne new
```

```
mov bx, 0
mov cx, matrix_size
cycle:
push cx
mov cx, matrix_size
mov dx, 0
cycle2:
mov i, bx
mov j, dx
call index
```

```
mov si, ax
push [temp_matrix + si]
pop [matrix + si]
inc dx
loop cycle2
pop cx
inc bx
loop cycle
pop di
pop si
pop dx
pop cx
pop bx
pop ax
pop j
pop i
ret
create endp
```

```
find_det proc
mov ax, matrix_size
cmp ax, 1
jne contin
mov ax, matrix
ret
contin:
mov bx, 0
sum:
mov i, 0
mov j, bx
call index
mov si, ax
mov ax, -1
imul minus_one
mov minus_one, ax
mov ax, [matrix + si]
imul minus_one
mov cx, ax
```

```
push det
push cx
push matrix_size
mov di, 0
mov si, 0
push ax
mov ax, matrix_size
mul ax
mov dx, ax
pop ax
pushing:
push [matrix + si]
add si, 2
inc di
cmp di, dx
jne pushing
push si
push di
push d
push temp_index
push minus_one
call create
jmp sum3
sum2:
jmp sum
sum3:
push ax
mov ax, matrix_size
dec ax
mov matrix_size, ax
mov ax, 0
mov temp_index, ax
mov det, ax
mov ax, -1
mov minus_one, ax
pop ax
push bx
```

```
call find_det
pop bx
pop minus_one
pop temp_index
pop d
pop di
pop si
sub si, 2
popping:
pop [matrix + si]
sub si, 2
dec di
cmp di, 0
jne popping
pop matrix_size
pop cx
imul cx
pop det
```

```
add det, ax
inc bx
cmp bx, matrix_size
jne sum2
mov ax, det
ret
find_det endp
```

```
main:
mov ax, @data
mov ds, ax
```

```
mov ah, 3Ch
mov cx, 7
lea dx, input_file
int 21h
```

```
mov ax, 3D01h
lea dx, input_file
```


int 21h

mov bx, ax
mov ah, 40h
lea dx, input_txt
mov cx, 254
int 21h

mov ah, 3Eh
int 21h

lea dx, enter_msg
mov ah, 09h
int 21h
call input
push ax
lea dx, end_line
mov ah, 09h
int 21h
pop ax
mov matrix_size, ax
mov ax, matrix_size
mov bx, 2
mul bx
mov for_index, ax

lea dx, read_msg
mov ah, 09h
int 21h
lea dx, end_line
mov ah, 09h
int 21h

mov ax, 3D00h
lea dx, input_file
int 21h

mov bx, ax

```
mov ah, 3Fh
mov cx, 254
lea dx, string
int 21h
```

```
mov ah, 3Eh
int 21h
```

```
lea di, buffer
lea bx, string
mov ax, matrix_size
mul ax
mov count, ax
mov cx, 0
mov ax, 0
mov si, 0
split:
mov dl, [bx]
cmp dl, 32
je fill
push [bx]
pop [di]
inc di
inc ax
mov len, al
jmp no_fill
fill:
call convert
mov matrix + si, ax
add si, 2
lea di, buffer
mov ax, 0
inc cx
no_fill:
inc bx
cmp cx, count
jne split
```

```
lea bx, matrix
call show_matrix
```

```
call find_det
push ax
lea dx, res_msg
mov ah, 09h
int 21h
pop ax
call output
lea dx, end_line
mov ah, 09h
int 21h
```

```
mov bx, 0
edit:
mov cx, 0
edit2:
mov i, bx
mov j, cx
call index
mov si, ax
mov ax, [matrix + si]
cmp ax, det
je increase
jg reduce
jmp point
increase:
cmp ax, 0
jl for_minus
mov di, 7
imul di
mov di, 5
idiv di
mov [matrix + si], ax
jmp point
for_minus:
mov di, 3
```

```
imul di
mov di, 5
idiv di
mov [matrix + si], ax
jmp point
reduce:
cmp ax, 0
jl for_minus2
mov di, 4
imul di
mov di, 5
idiv di
mov [matrix + si], ax
jmp point
for_minus2:
mov di, 6
imul di
mov di, 5
idiv di
mov [matrix + si], ax
jmp point
edit3:
jmp edit
point:
inc cx
cmp cx, matrix_size
jne edit2
inc bx
cmp bx, matrix_size
jne edit3

lea dx, write_msg
mov ah, 09h
int 21h
lea dx, end_line
mov ah, 09h
int 21h
```

```
lea bx, matrix
call show_matrix
```

```
mov bx, 0
mov len, bl
lea di, string
write:
mov cx, 0
write2:
mov i, bx
mov j, cx
call index
mov si, ax
mov ax, matrix + si
cmp ax, 0
jge write_skip
push ax
mov al, len
inc al
mov len, al
pop ax
write_skip:
call to_string
push bx
lea bx, buffer
write_rep:
cmp [bx], '$'
je write_next
push [bx]
pop [di]
inc di
inc bx
jmp write_rep
write_next:
pop bx
mov al, len
inc al
mov len, al
```

```
inc cx
cmp cx, matrix_size
jne write2
mov [di], 10
inc di
mov al, len
inc al
mov len, al
inc bx
cmp bx, matrix_size
jne write
```

```
mov ah, 3Ch
mov cx, 7
lea dx, output_file
int 21h
```

```
mov ax, 3D01h
lea dx, output_file
int 21h
```

```
mov bx, ax
mov ah, 40h
mov cl, len
lea dx, string
int 21h
```

```
mov ah, 3Eh
int 21h
```

```
exit:
mov ah, 4ch
int 21h
end main
```