



**«Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет
имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)»**

ФАКУЛЬТЕТ ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ
КАФЕДРА КОМПЬЮТЕРНЫЕ СИСТЕМЫ И СЕТИ

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.03.01 Информатика и вычислительная техника**

О Т Ч Е Т

по лабораторной работе № 4

Дисциплина: Машинно-зависимые языки и основы компиляции

Название лабораторной работы: Обработка массивов и матриц

Вариант: 17

Студент гр. ИУ6-43Б

26.03.2022
(Подпись, дата)

М.А. Мяделец
(И.О. Фамилия)

Преподаватель

(Подпись, дата)

М.В. Широкова
(И.О. Фамилия)

Москва, 2022

Обработка массивов и матриц

Цель работы: изучение приемов моделирования обработки массивов и матриц в языке ассемблера.

Задание

Рассмотрим задание для лабораторной работы 4 (смотри рисунок 1).

Лабораторная работа №4. Программирование обработки массивов и матриц.

Дана матрица 6x5. Вычеркнуть столбец с заданным номером. Организовать ввод матрицы и вывод результатов.

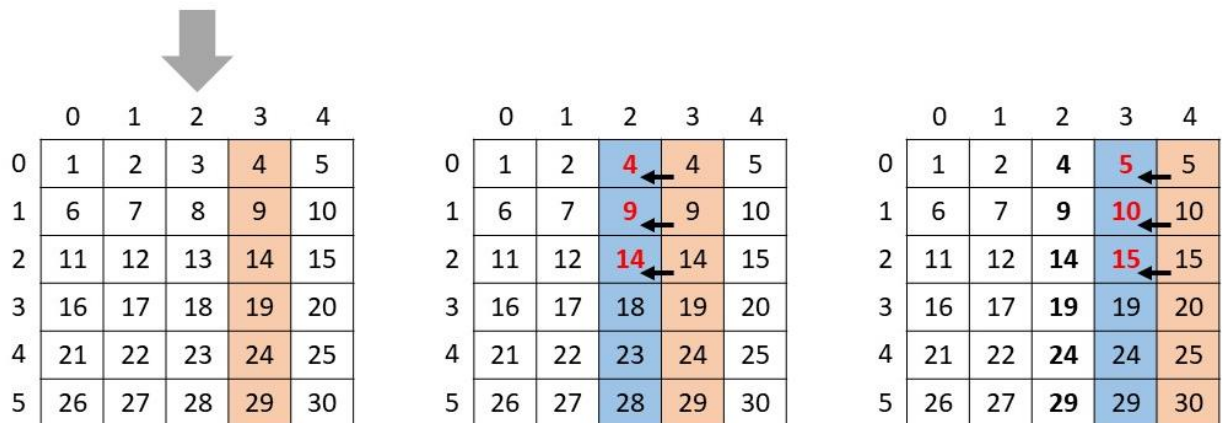
Рисунок 1 - Условие задания

Краткие моменты решения:

- 1) Попросим пользователя ввести $n * m$ целочисленных элементов матрицы построчно. При наличии иных символов, кроме цифр, программа напечатает “Error...” и завершится.
- 2) Параллельно будем записывать все введенные элементы в область памяти, начало которой символизирует переменная `mas` (цикл `loop_input`).
- 3) Попросим пользователя ввести номер столбца для удаления, причем отрицательные или несуществующие столбцы будут восприняты как ошибочные – программа напечатает “Error...” и завершится.
- 4) «Удалим» столбец и выведем матрицу без указанного столбца. Удаление столбца – значит, сдвиг элементов следующих столбцов на один столбец влево. Из номера удаляемого столбца и общего количества столбцов поймем, сколько столбцов необходимо сдвинуть (счетчик цикла `outer_loop_output`).

Рассмотрим для примера матрицу 6 на 5, заполненную числами от 1 до 30. Пользователь намерен удалить третий столбец, поэтому вводит в консоль цифру 3. Т.к. в программе нумерацию столбцов ведем с нуля, необходимо удалить второй столбец (смотри на серую стрелочку на рисунке 2). Соответственно, для удаления второго столбца, нужно сдвинуть следующие за ним, т.е. третий и

четвертый. Пользователь ввел цифру три, значит, с третьего столбца и начинаем сдвигать (смотри рисунок 2).



	0	1	2	3	4
0	1	2	3	4	5
1	6	7	8	9	10
2	11	12	13	14	15
3	16	17	18	19	20
4	21	22	23	24	25
5	26	27	28	29	30

	0	1	2	3	4
0	1	2	4	4	5
1	6	7	9	9	10
2	11	12	14	14	15
3	16	17	18	19	20
4	21	22	23	24	25
5	26	27	28	29	30

	0	1	2	3	4
0	1	2	4	5	5
1	6	7	9	10	10
2	11	12	14	15	15
3	16	17	19	19	20
4	21	22	24	24	25
5	26	27	29	29	30

Рисунок 2 – Процесс удаления третьего столбца матрицы

В результате сдвига столбцов, получим результирующую матрицу, которую необходимо выводить в консоль без последнего столбца (смотри рисунок 3).

	0	1	2	3	4
0	1	2	4	5	5
1	6	7	9	10	10
2	11	12	14	15	15
3	16	17	19	20	20
4	21	22	24	25	25
5	26	27	29	30	30

Рисунок 3 – Матрица после удаления столбца

Для визуализации работы алгоритма была спроектирована блок-схема алгоритма. Схема наглядно демонстрирует все этапы работы алгоритма без использования синтаксиса ассемблера (смотри рисунок 4).

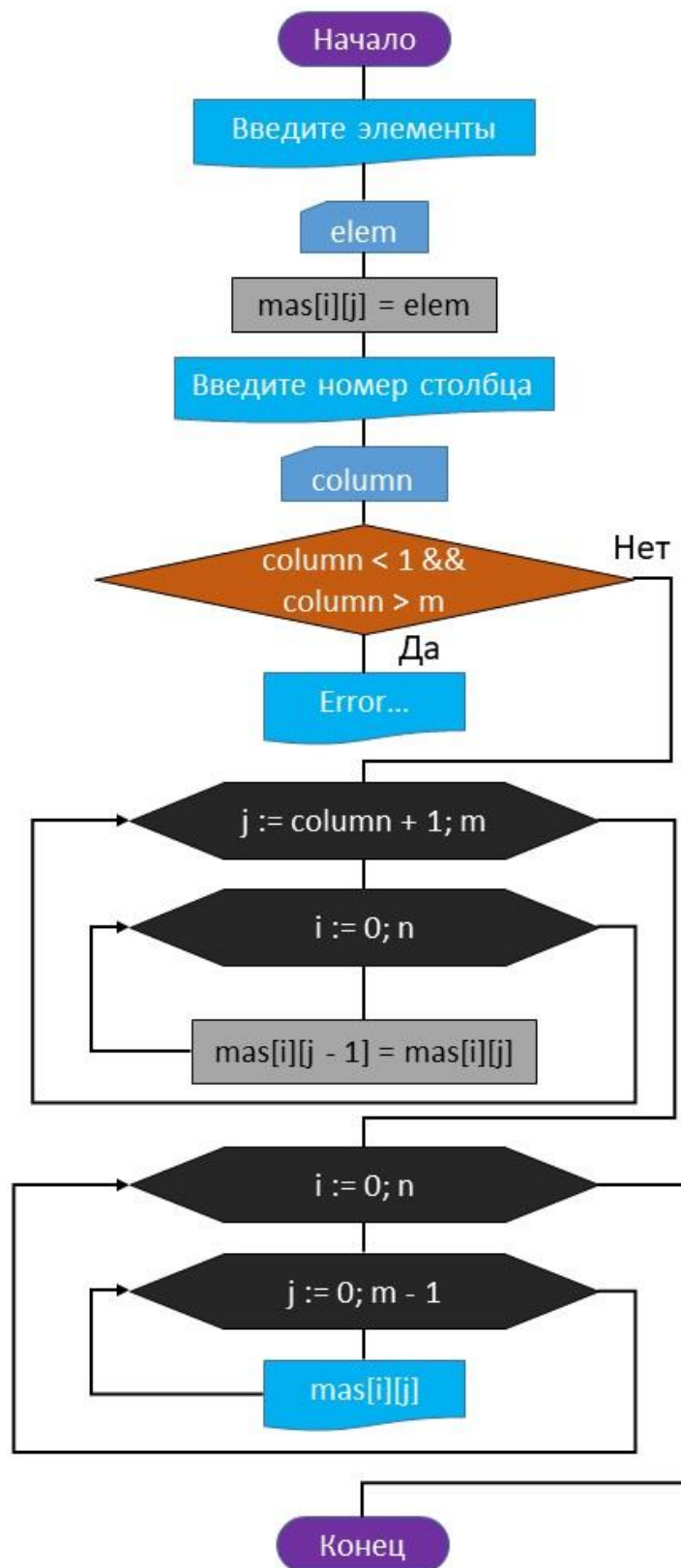


Рисунок 4 – Блок-схема алгоритма

Замечание. Размерность матрицы зависит от переменных, задаваемых в области инициализированных данных (n и m – количество строк и столбцов

соответственно), поэтому для удобства тестирования будем работать с матрицей 2 на 3. Тестирование проведем для всех крайних случаев, а также для неверно введенных данных (смотри таблицу 1).

Таблица 1 – Результаты тестирования

Исходные данные		Результат
n	2	Введем числа от 1 до 6, удалим столбец 2.
m	3	
n	2	Введем числа от 1 до 6, удалим столбец 3.
m	3	
n	2	Введем числа от 1 до 6, удалим столбец 1.
m	3	

		<pre> mikhail@DESKTOP-2LSJ1H4:/mnt/c/МГТУ/4 семестр/МЗЯ/prog_asm/Lab_4\$./lab_4 Enter number of rows and columns 2 3 Enter elements of matrix n x m 1 2 3 4 5 6 1 2 3 4 5 6 Enter column number 1 2 3 5 6 </pre>
n	2	Введем числа от 1 до 6, удалим столбец -1, 0, 4.
m	3	
		<pre> Enter column number -1 Error... Enter column number 0 Error... Enter column number 4 Error... </pre>
n	2	Введем вместо элемента матрицы строчку.
m	3	
		<pre> Enter elements of matrix n x m 1 2 h Error... </pre>
n	2	Введем вместо номера столбца строчку.
m	3	
		<pre> Enter column number h Error... </pre>
n	6	Зададим размер матрицы, заданный по условию. Введем числа от 1 до 30, удалим столбец 3.
m	5	

		<pre> 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 Enter column number 2 1 3 4 5 6 8 9 10 11 13 14 15 16 18 19 20 21 23 24 25 26 28 29 30 </pre>
--	--	---

Вывод: была сделана лабораторная работа 4, которая была направлена на работу с массивами и матрицами. В ходе выполнения задания были применены приемы обработки матриц построчно и по столбцам, более детально рассмотрены механизмы ввода / вывода, вложенные циклы, работа со стеком, условные и безусловные переходы. Также в процессе написания программы были написаны обработчики для неверных данных. Результаты тестирования корректны и оформлены в виде таблицы (смотри таблицу 1). Работа проведена успешно!

Контрольные вопросы

1) Почему в ассемблере не определены понятия «массив», «матрица»?

Ассемблер взаимодействует напрямую с памятью, а массив во внутреннем представлении — это последовательность элементов в памяти. Ассемблер взаимодействует с «массивом» через адрес первого элемента и смещений.

2) Как в ассемблере моделируются массивы?

В ассемблере массивом является переменная, под которую программа выделила место, кратное размеру переменной.

3) Поясните фрагмент последовательной адресации элементов массива?

Почему при этом для хранения частей адреса используют регистры?

Так как в ассемблере нет понятия массив, а значит и индекса, то получить доступ к элементу возможно только через обращение к его ячейке памяти. Для запоминания ячейки можно пользоваться регистром EBX, который будут хранить величину смещения относительно начала массива (смотри рисунок 5).

```

; вывод
mov ecx, [n]      ; вывод матрицы построчно
mov ebx, 0
cycleOut1:
    push ecx
    mov ecx, [m]
cycleOut2:
    push ecx
    mov eax, [mas + ebx * 4]

    push ebx
    mov esi, OutBuf
    call IntToStr

    mov     ebx, 1
    mov     ecx, esi
    mov     edx, eax
    mov     eax, 4
    int     80h

    pop ebx
    pop ecx
    inc ebx
    loop cycleOut2

    push ebx

; вывод перехода на новую строку
mov     eax, 4
mov     ebx, 1
mov     ecx, newline
mov     edx, 1
int     80h

    pop ebx

    pop ecx
loop cycleOut1
```

Рисунок 5 – Фрагмент последовательной адресации к элементам массива.

4) Как в памяти компьютера размещаются элементы матриц?

Матрицы могут храниться построчно и по столбцам в зависимости от индивидуальной задачи.

5) *Чем моделирование матриц отличается от моделирования массивов? В каких случаях при выполнении операций для адресации матриц используется один регистр, а в каких – два?*

Моделирование матриц отличается от моделирования массивов тем, что у матриц есть столбцы и строки, в то время как у массива только строка. Поэтому для массива достаточно иметь один регистр, хранящий смещение от начала массива, а для матрицы требуется два регистра: один считает смещение от первой строки, а второй считает смещение от первого столбца.

ПРИЛОЖЕНИЕ

Текст программы

```
        section .data
n        dd 6      ; количество строк
m        dd 5      ; количество столбцов
bit      dd 4      ; разрядность системы
message  db "Enter column number", 10
lenMsg   equ $-message
msginput  db "Enter elements of matrix n x m", 10
lenMsgInput equ $-msginput
msgError  db "Error...", 10
lenMsgError equ $-msgError
endline   db 10

        section .bss
nm        resd 1      ; количество элементов матрицы
mas       resd 100
InBuf     resb 10     ; введенный элемент матрицы
lenIn     equ $-InBuf

        section .text
global _start
_start:
    mov     eax, 4
    mov     ebx, 1
    mov     ecx, msginput
    mov     edx, lenMsgInput
    int     80h

    mov     eax, [nm]
    mul     dword[nm]          ; в EAX - количество всех элементов матрицы 6 * 5 =
30
    mov     dword[nm], eax     ; заносим 30 в память [nm]

    mov     ecx, [nm]          ; номер текущего элемента (index)
    jcxz    end_loop_input
                                ; цикл ввода 6 * 5 чисел и занесение в память mas
loop_input:
    push     ecx               ; stack <

                                ; чтение числа с консоли
    mov     eax, 3
```

```

mov ebx, 0
mov ecx, InBuf
mov edx, lenIn
int 80h

; ввели "124\n", далее нужно преобразовать в числовое значение

mov esi, ecx      ; адрес строки, которую нужно конвертировать в
число
call StrToInt     ; преобразованное из строки число находится в EAX

cmp EBX, 0
jne error

; если EBX = 0, то произошла ошибка, 1 в обратном
случае
; (недопустим. символ, выход за границы разрядной
сетки)

pop ecx           ; stack >

mov edx, dword[nm]
sub edx, ecx      ; edx - индекс массива, в который нужно записать
введенное значение
; количество чисел - номер итерации

mov [mas + edx * 4], eax

loop loop_input
end_loop_input:

; вывод сообщения "Enter column number"

mov     eax, 4
mov     ebx, 1
mov     ecx, message
mov     edx, lenMsg
int     80h

; ввод номера столбца, который необходимо удалить

mov     eax, 3
mov     ebx, 0
mov     ecx, InBuf

```

```

mov     edx, lenIn
int     80h

mov esi, ecx          ; адрес строки с номером удаляемого столбца

call StrToInt          ; в eax - целочисленный номер столбца
                        ; (при нумерации с нуля - номер следующего столбца)

cmp EBX, 0
jne error

cmp eax, dword[m]      ; ввели номер столбца, которого не существует -
ошибка
jg error
cmp eax, 1              ; ввели отрицательный номер столбца - ошибка
jl error

mov ecx, dword[m]      ; ecx = количество столбцов
sub ecx, eax            ; кол-во столбцов - номер удаляемого столбца = кол-
во столбцов после удаляемого (ecx)

; внешний цикл - итерируемся по столбцам
; внутренний цикл - итерируемся по строкам

jcxz end_loop_logic

outer_loop_logic:
    push ecx
    mov ebx, 0          ; ebx - смещение по строкам
    mov ecx, [n]

inner_loop_logic:
    mov edx, [ebx + eax * 4 + mas]          ; пересылаем
значение из столбца в регистр edx
    dec eax                                ; переходим на
предыдущий столбец
    mov dword[ebx + eax * 4 + mas], edx      ; заносим значение
edx в предыдущий столбец
    inc eax                                ; переходим на
исходный столбец

; переход на новую строку

```

```

        push eax          ; stack <
        mov eax, [m]
        mul dword[bit]    ; eax = [n] * 4 = смещение на следующую
строку
        add ebx, eax      ; смещаемся на следующую строку (5
элементов * 4 байта)
        pop eax           ; stack >

        loop inner_loop_logic

```

```

        inc eax           ; перемещаемся на следующий столбец
        pop ecx

```

```

        loop outer_loop_logic

```

```

end_loop_logic:

```

```

; выводим пустую строку (отсуп для результатов)

```

```

mov     eax, 4
mov     ebx, 1
mov     ecx, endlrline
mov     edx, 1
int     80h

```

```

mov edi, 0          ; edi - смещение по строкам
mov ecx, [n]

```

```

outer_loop_output:

```

```

        push ecx
        mov ecx, [m]
        dec ecx      ; количество столбцов уменьшилось на 1, поэтому
значение декрементируем
        mov ebx, 0    ; ebx - номер столбца

```

```

inner_loop_output:

```

```

        push ecx          ; stack <

        mov eax, [edi + ebx * 4 + mas] ; пересылаем элемент из
массива (итерируемся построчно)
        call IntToStr     ; конвертируем элемент в
строку

```

```

        push ebx                ; stack <

        ; ВЫВОДИМ ЧИСЛО
        mov     eax, 4
        mov     ebx, 1
        mov     ecx, esi
        mov     edx, 4
        int     80h

        pop ebx                ; stack >
        inc ebx                ; переход к следующему столбцу

        pop ecx                ; stack >
        loop inner_loop_output

; выводим пустую строку (как обозначение завершения строки матрицы)

        mov     eax, 4
        mov     ebx, 1
        mov     ecx, endl
        mov     edx, 1
        int     80h

; переход на новую строку

        mov     eax, [m]
        mul     dword[bit]      ; eax = [n] * 4 = смещение на следующую строку
        add     edi, eax        ; смещаемся на следующую строку (5 элементов * 4
байта)

        pop     ecx
        loop outer_loop_output

        jmp     exit

error:
        ; вывод сообщения "Error"
        mov     eax, 4
        mov     ebx, 1
        mov     ecx, msgError
        mov     edx, lenMsgError
        int     80h

```

```
exit:
    ; exit
    mov     eax, 1          ; системная функция 1 (exit)
    xor     ebx, ebx        ; код возврата 0
    int     80h             ; вызов системной функции

#include "./lib.asm"
```