|  |  |
| --- | --- |
| **Gerb-BMSTU_01** | **Министерство науки и высшего образования Российской Федерации**  Калужский филиал  федерального государственного бюджетного  образовательного учреждения высшего образования  ***«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»***  ***(КФ МГТУ им. Н.Э. Баумана)*** |

**ФАКУЛЬТЕТ** ***ИУК «Информатика и управление»***

**КАФЕДРА** \_\_***ИУК4 «Программное обеспечение ЭВМ, информационные технологии»***

**ДОМАШНЯЯ РАБОТА №1**

**«Обработка двухмерных массивов целых чисел»**

**ДИСЦИПЛИНА: «Машинно-зависимые языки программирования»**

|  |  |  |
| --- | --- | --- |
| Выполнил: студент гр. ИУК4-32Б | | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ( Карельский М.К. )  (Подпись) |
| Проверил: | | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ( Амеличева К.А. )  (Подпись) |
| Дата сдачи (защиты):  Результаты сдачи (защиты): | | |
|  | - Балльная оценка:  - Оценка: | |

Калуга, 2021

**Цель:** изучить особенности обработки двухмерных массивов на языке ассемблер.

**Вариант 9**

1. Ввести с клавиатуры и вывести на экран матрицу m × n (матрица не обязательно должна быть квадратная и может содержать нулевые и отрицательные элементы, если это предусмотрено условиями задания варианта);
2. Реализовать простейший интерфейс взаимодействия с пользователем для выполнения задания варианта
   * Транспонировать матрицу, результат вынести на экран;
   * Обработка элементов матрицы, результат выполнения отобразить на экране;
     1. В каждом столбце матрицы найти количество элементов, находящихся в диапазоне между двумя заданными числами.
     2. Найти среднее арифметическое в каждой ненулевой строке.
     3. Найдите первый положительный элемент среди элементов матрицы, выделенных черным цветом (матрица квадратная).
   * Завершение выполнение программы.

**Макросы:**

|  |  |  |
| --- | --- | --- |
| Номер | Название | Страница |
| 1 | mSetColorMode | 8 |
| 2 | mClearScreen | 8 |
| 3 | mClearPlace frsRow, frsCol, lstRow, lsrCol | 8 |
| 4 | mPrintString string | 9 |
| 5 | mPrintSymbol symbol | 9 |
| 6 | mMoveCursor row, column | 9 |
| 7 | mShiftCursor row, column | 9 |
| 8 | mIncCursor | 10 |
| 9 | mNextLine | 10 |
| 10 | mInput buffer, size | 10-12 |
| 11 | mOutput | 12 |
| 12 | mInputMatrix matrix, rows, columns | 12-13 |
| 13 | mOutputMatrix matrix, rows, columns | 13-14 |
| 14 | mTranspose matrix, rows, columns | 14-15 |
| 15 | mFindQuantity matrix, rows, columns | 15-16 |
| 16 | mFindMean matrix, rows, columns | 16-17 |
| 17 | mFindPlus matrix, dimension | 17-18 |

**Таблица 1.** Макросы

1. mSetColorMode – назначает 7-ой бит атрибута яркостью фона.
2. mClearScreen – очищает экран и устанавливает курсор в верхний левый угол.
3. mClearPlace frsRow, frsCol, lstRow, lsrCol – очищает выбранный участок экрана. Имеет следующие параметры:
   * frsRow – строка верхнего левого угла очищаемого участка
   * frsCol – столбец верхнего левого угла очищаемого участка
   * lstRow – строка нижнего правого угла очищаемого участка
   * lstCol – строка нижнего правого угла очищаемого участка
4. mPrintString string – выводит строку, указанную в параметре string. При этом курсор устанавливается на первую позицию после строки.
5. mPrintSymbol symbol – вывод символ, указанный в параметре symbol. При этом курсор остается на той же позиции.
6. mMoveCursor row, column – устанавливает курсор на позицию, которая задается параметрами row (строка) и column (столбец).
7. mShiftCursor row, column – находит текущую позицию курсора и сдвигает его на row позиций вправо и на column позиций вниз.
8. mIncCursor – находит текущую позицию курсора и сдвигает его на 1 вправо.
9. mNextLine – находит текущую позицию курсора и передвигает его в начало следующей строки.
10. mInput buffer, size – ввод числа. Сперва определяет знак числа и приравнивает число его модулю, затем его размер. Далее рассчитывается множитель – 10 в степени самого левого разряда. Затем каждая цифра переводится из символьного представления в числовое и умножается на множитель. После этого полученное число добавляется к общей сумме, а множитель делится на 10. Конечная сумма умножается на знак, и по итогу введенное число хранится в регистре ах. В параметр buffer передается массив, минимальный размер которого равен size + 3. В size передается максимальный размер положительного числа (соответственно, размер отрицательного числа будет на 1 меньше).
11. mOutput – вывод числа, хранящегося в регистре ах. Сперва выводит знак минуса и умножает число на -1, если оно отрицательное. Затем число делится на 10, целые части от деления заносятся в стек. Затем они извлекаются из стека, переводятся в символьную форму и выводятся в консоль.
12. mInputMatrix matrix, rows, columns – ввод матрицы. Работа происходит в 2 цикла: во внешнем перебираются строки, а во внутреннем – столбцы. Также на каждой итерации внутреннего цикла вводится и заносится в матрицу число. В параметр matrix передается двумерный массив размером rows (количество строк) × columns (количество столбцов).
13. mOutputMatrix matrix, rows, columns – вывод матрицы. Работа происходит в 2 цикла: во внешнем перебираются строки, а во внутреннем – столбцы. Также на каждой итерации внутреннего цикла выводится соответствующее число. При этом происходит расчет количества отступов: если число однозначное и положительно, то отступ равен 2, иначе – 1. Параметры аналогичны mInputMatrix.
14. mTranspose – транспонирование и вывод матрицы. Параметры и принцип работы аналогичены mOutputMatrix с той лишь разницей, что во внешнем цикле перебираются столбцы, а во внутреннем – строки.
15. mFindQuantity matrix, rows, columns – задача «а». Сначала происходит ввода нижней и верхней границ сравнения. Затем происходит работа, аналогичная mTranspose с той лишь разницей, что вместо вывода на каждой итерации внутреннего цикла происходит проверка числа на вхождение в данный диапазон, а на каждой итерации внешнего цикла выводится количество таких чисел. Параметры также аналогичны.
16. mFindMean matrix, rows, columns – задача «b». Работа происходит в 2 цикла: во внешнем перебираются строки (начиная с 1, а не с 0), а во внутреннем – столбцы. На каждой итерации внутреннего цикла текущее число прибавляется к общей сумме, а на каждой итерации внешнего цикла полученная сумма делится на число столбцов. Получившееся число (с учетом остатка, если таковой имеется) выводится в консоль. Параметры аналогичны mInputMatrix.
17. mFindPlus matrix, dimension – задача «c». Работа происходит в 2 цикла: во внешнем перебираются строки, а во внутреннем – столбцы. Максимальное число итераций внешнего цикла равно размеру матрицы, а внутреннего – номеру итерация внешнего цикла начиная с 1. Если текущее число положительно, происходит выход из обоих циклов, а самое число выводится. Если внешний цикл закончил свою работу без прерывания, выводится сообщение о том, что положительного числа нет. В параметр matrix передается двумерный массив размером dimension × dimension.

**Блок-схемы:**



**Рисунок 1.1.** Задача «а»



**Рисунок 1.2.** Задача «а»



**Рисунок 2.** Задача «b»



**Рисунок 3.** Задача «c»

**Листинг:**

.model small

.stack 100h

.386

.data

buffer db 5 dup(0)

matrix db 5 dup(5 dup(0))

rows db 0

columns db 0

s\_rows db 'Input number of rows (3-5): ', '$'

s\_columns db 'Input number of columns (3-5): ', '$'

s\_inputMatrix db 'Input matrix:', 10, 13, '$'

s\_range db 'a <= x[i] <= b', 10, 13, '$'

s\_input\_a db 'Input a: ', '$'

s\_input\_b db 'Input b: ', '$'

s\_column db ' column: ', '$'

s\_mean db ' line mean: ', '$'

s\_positive db 'First positive: ', '$'

s\_menu db '1. Transpose matrix', 10, 13

db '2. Numbers in the range', 10, 13

db '3. Arithmetic mean', 10, 13

db '4. Positive number', 10, 13

db '0. Exit', 10, 13

db '>>> ', '$'

s\_error db 'The matrix is not square', '$'

mSetColorMode macro

push ax

push bx

mov ah, 10h

mov al, 3

sub bl, bl

int 10h

pop bx

pop ax

endm

mClearScreen macro

push ax

push bx

push cx

push dx

mov ah, 6

sub al, al

mov bh, 0F0h

sub cx, cx

mov dx, 184Fh

int 10h

pop dx

pop cx

pop bx

pop ax

endm

mClearPlace macro frsRow, frsCol, lstRow, lstCol

push ax

push bx

push cx

push dx

mov ah, 6

sub al, al

mov bh, 0F0h

mov ch, frsRow

mov cl, frsCol

mov dh, lstRow

mov dl, lstCol

int 10h

pop dx

pop cx

pop bx

pop ax

endm

mPrintString macro string

push ax

push dx

mov ah, 09h

mov dx, offset string

int 21h

pop dx

pop ax

endm

mPrintSymbol macro symbol

push ax

push bx

push cx

mov ah, 9

mov al, symbol

mov bl, 0F0h

mov cx, 1

int 10h

pop cx

pop bx

pop ax

endm

mMoveCursor macro row, column

push ax

push bx

push dx

mov ah, 2

mov dh, row

mov dl, column

sub bh, bh

int 10h

pop dx

pop bx

pop ax

endm

mShiftCurcor macro row, column

push ax

push bx

push cx

push dx

mov ah, 3

sub bh, bh

int 10h

add dh, row

add dl, column

mov ah, 2

int 10h

pop dx

pop cx

pop bx

pop ax

endm

mIncCursor macro

push ax

push bx

push cx

push dx

mov ah, 3

sub bh, bh

int 10h

inc dl

mov ah, 2

int 10h

pop dx

pop cx

pop bx

pop ax

endm

mNextLine macro

push ax

push bx

push cx

push dx

mov ah, 3

sub bh, bh

int 10h

inc dh

mov dl, 0

mov ah, 2

int 10h

pop dx

pop cx

pop bx

pop ax

endm

mInput macro buffer, size

LOCAL NEGATIVE

LOCAL POSITIVE

LOCAL CALCULATION

LOCAL POWER

LOCAL CYCLE

LOCAL END\_CALCULATION

LOCAL END\_INPUT

push bx

push cx

push dx

push si

mov ah, 3

mov bh, 0

int 10h

push dx

mov buffer, size

inc buffer

mov ah, 0Ah

lea dx, buffer

int 21h

pop dx

mov ah, 2

int 10h

sub ax, ax

cmp buffer+1, 0

je END\_INPUT

mov ax, 1

sub dx, dx

mov dl, buffer+1

dec dx

mov bx, 10

cmp buffer+2, '-'

je NEGATIVE

jmp POSITIVE

NEGATIVE:

push -1

mov si, 3

dec dx

jmp POWER

POSITIVE:

push 1

mov si, 2

POWER:

cmp dx, 0

jle CALCULATION

mul bl

dec dx

jmp POWER

CALCULATION:

mov cx, ax

sub bx, bx

CYCLE:

sub ax, ax

mov al, buffer[si]

sub al, 30h

mul cx

add bx, ax

mov ax, cx

mov cx, 10

div cx

mov cx, ax

inc si

mov dx, si

sub dl, buffer+1

cmp dl, 2

je END\_CALCULATION

jmp CYCLE

END\_CALCULATION:

pop dx

mov ax, bx

imul dx

END\_INPUT:

pop si

pop dx

pop cx

pop bx

endm

mOutput macro

LOCAL NEGATIVE

LOCAL SPLIT

LOCAL OUTPUT

push dx

push si

push di

mov si, 10

sub di, di

cmp ax, 0

jl NEGATIVE

jmp SPLIT

NEGATIVE:

neg ax

push ax

mPrintSymbol '-'

mIncCursor

pop ax

SPLIT:

inc di

sub dx, dx

div si

add dx, 30h

push dx

cmp ax, 0

jg SPLIT

OUTPUT:

pop ax

mPrintSymbol al

mIncCursor

dec di

cmp di, 0

jg OUTPUT

pop di

pop si

pop dx

endm

mInputMatrix macro matrix, rows, columns

LOCAL ROW

LOCAL COLUMN

push ax

push bx

push cx

push si

sub bx, bx

sub cx, cx

mov cl, rows

JUMPS

ROW:

push cx

sub si, si

sub cx, cx

mov cl, columns

COLUMN:

mInput buffer, 2

mov matrix[bx][si], al

inc si

mShiftCurcor 0, 3

loop COLUMN

add bl, columns

mNextLine

pop cx

loop ROW

NOJUMPS

pop si

pop cx

pop bx

pop ax

endm

mOutputMatrix macro matrix, rows, columns

LOCAL ROW

LOCAL COLUMN

LOCAL ONE\_SPACE

LOCAL TWO\_SPACES

LOCAL END\_COLUMN

push ax

push bx

push cx

push si

sub bx, bx

sub cx, cx

mov cl, rows

JUMPS

ROW:

push cx

sub si, si

sub cx, cx

mov cl, columns

COLUMN:

sub ax, ax

mov al, matrix[bx][si]

cbw

mOutput

sub ax, ax

mov al, matrix[bx][si]

cmp ax, 10

jl TWO\_SPACES

jmp ONE\_SPACE

TWO\_SPACES:

mShiftCurcor 0, 2

jmp END\_COLUMN

ONE\_SPACE:

mIncCursor

END\_COLUMN:

inc si

loop COLUMN

add bl, columns

mNextLine

pop cx

loop ROW

NOJUMPS

pop si

pop cx

pop bx

pop ax

endm

mTranspose macro matrix, rows, columns

LOCAL ROW

LOCAL COLUMN

LOCAL ONE\_SPACE

LOCAL TWO\_SPACES

LOCAL END\_ROW

push ax

push bx

push cx

push si

sub si, si

sub cx, cx

mov cl, columns

COLUMN:

push cx

sub bx, bx

sub cx, cx

mov cl, rows

ROW:

mov al, matrix[bx][si]

cbw

mOutput

sub ax, ax

mov al, matrix[bx][si]

cmp ax, 10

jl TWO\_SPACES

jmp ONE\_SPACE

TWO\_SPACES:

mShiftCurcor 0, 2

jmp END\_ROW

ONE\_SPACE:

mIncCursor

END\_ROW:

add bl, columns

dec cx

cmp cx, 0

jg ROW

mNextLine

inc si

pop cx

dec cx

cmp cx, 0

jg COLUMN

pop si

pop cx

pop bx

pop ax

endm

mFindQuantity macro matrix, rows, columns

LOCAL ROW

LOCAL COLUMN

LOCAL CMP\_B

LOCAL CMP\_END

LOCAL INCREMENT

push ax

push bx

push cx

push dx

push si

mPrintString s\_range

mPrintString s\_input\_a

mInput buffer, 2

mov dh, al

mNextLine

mPrintString s\_input\_b

mInput buffer, 2

mov dl, al

mNextLine

mNextLine

sub si, si

sub cx, cx

mov cl, columns

COLUMN:

push cx

sub bx, bx

sub ax, ax

sub cx, cx

mov cl, rows

ROW:

cmp dh, matrix[bx][si]

jle CMP\_B

jmp CMP\_END

CMP\_B:

cmp matrix[bx][si], dl

jle INCREMENT

jmp CMP\_END

INCREMENT:

inc ax

CMP\_END:

add bl, columns

dec cx

cmp cx, 0

jg ROW

push ax

mov ax, si

mOutput

mPrintString s\_column

pop ax

mOutput

mNextLine

inc si

pop cx

dec cx

cmp cx, 0

jg COLUMN

pop si

pop dx

pop cx

pop bx

pop ax

endm

mFindMean macro matrix, rows, columns

LOCAL ROW

LOCAL COLUMN

LOCAL REMAINDER

LOCAL END\_ROW

push ax

push bx

push cx

push dx

push si

sub bx, bx

mov bl, columns

sub cx, cx

mov cl, rows

dec cx

ROW:

push cx

sub si, si

sub ax, ax

sub cx, cx

mov cl, columns

COLUMN:

mov dl, matrix[bx][si]

xchg ax, dx

cbw

xchg ax, dx

add ax, dx

inc si

dec cx

cmp cx, 0

jg COLUMN

pop cx

push ax

sub ax, ax

mov al, rows

sub al, cl

mOutput

mPrintString s\_mean

pop ax

mov dl, columns

idiv dl

push ax

cbw

mOutput

pop ax

cmp ah, 0

jne REMAINDER

jmp END\_ROW

REMAINDER:

mIncCursor

mov al, ah

cbw

mOutput

mPrintSymbol '/'

mIncCursor

sub ax, ax

mov al, columns

mOutput

END\_ROW:

mNextLine

add bl, columns

dec cx

cmp cx, 0

jg ROW

pop si

pop dx

pop cx

pop bx

pop ax

endm

mFindPlus macro matrix, dimension

LOCAL ROW

LOCAL COLUMN

LOCAL FOUND

LOCAL END\_MAC

push ax

push bx

push cx

push si

mPrintString s\_positive

sub bx, bx

sub cx, cx

mov cl, dimension

ROW:

push cx

sub si, si

mov ax, cx

mov cl, dimension

sub cx, ax

inc cx

COLUMN:

cmp byte ptr matrix[bx][si], 0

jg FOUND

inc si

dec cx

cmp cx, 0

jg COLUMN

add bl, dimension

pop cx

dec cx

cmp cx, 0

jg ROW

mPrintSymbol '-'

jmp END\_MAC

FOUND:

mov al, matrix[bx][si]

cbw

mOutput

END\_MAC:

mNextLine

pop si

pop cx

pop bx

pop ax

endm

.code

Start:

mov ax, @data

mov ds, ax

sub ax, ax

mSetColorMode

mClearScreen

mMoveCursor 0, 0

mPrintString s\_rows

mInput buffer, 1

mov rows, al

mNextLine

mPrintString s\_columns

mInput buffer, 1

mov columns, al

mNextLine

mPrintString s\_inputMatrix

mInputMatrix matrix, rows, columns

mClearScreen

mMoveCursor 0, 0

mOutputMatrix matrix, rows, columns

mNextLine

mPrintString s\_menu

mov ah, 3

sub bh, bh

int 10h

CYCLE:

mClearPlace dh, dl, dh, 6

mInput buffer, 1

inc dh

mClearPlace dh, 0, 18h, 4FH

dec dh

mNextLine

mNextLine

cmp al, 1

je TASK\_1

cmp al, 2

je TASK\_2

cmp al, 3

je TASK\_3

cmp al, 4

je TASK\_4

cmp al, 0

je END\_PROG

jmp CYCLE

TASK\_1:

mTranspose matrix, rows, columns

mMoveCursor dh, dl

jmp CYCLE

TASK\_2:

mFindQuantity matrix, rows, columns

mMoveCursor dh, dl

jmp CYCLE

TASK\_3:

mFindMean matrix, rows, columns

mMoveCursor dh, dl

jmp CYCLE

TASK\_4:

mov al, rows

cmp al, columns

je SQUARE

mPrintString s\_error

mMoveCursor dh, dl

jmp CYCLE

SQUARE:

mFindPlus matrix, rows

mMoveCursor dh, dl

jmp CYCLE

END\_PROG:

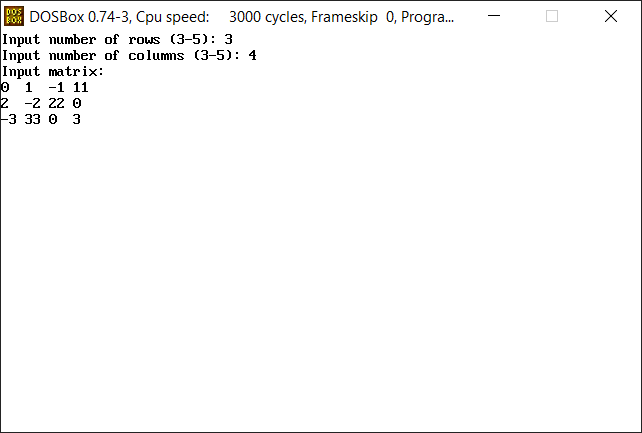
mov ax, 4C00h

int 21h

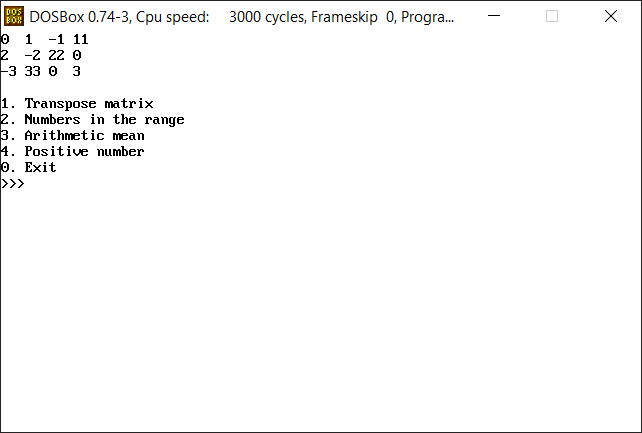
end start

END

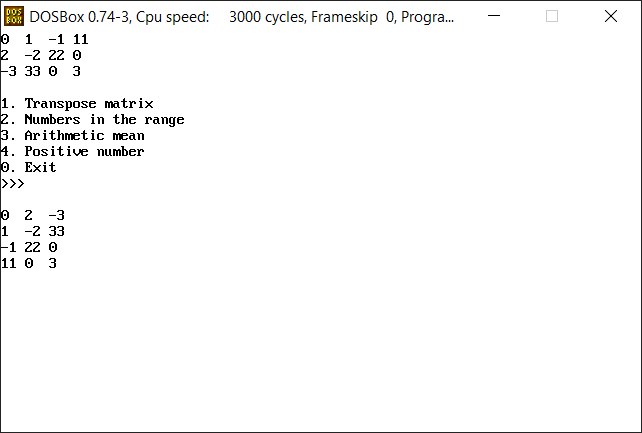
**Результаты работы:**



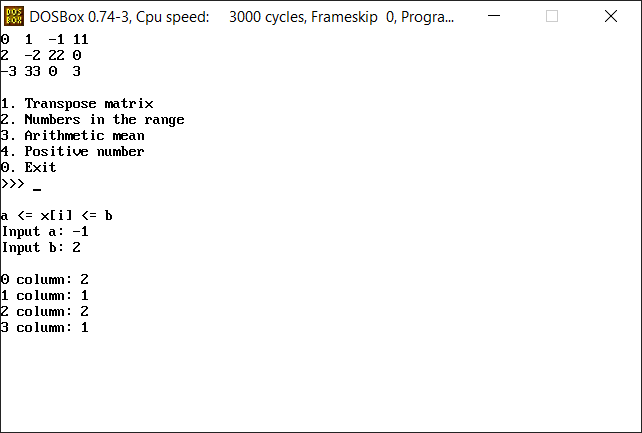
**Рисунок 4.** Ввод матрицы



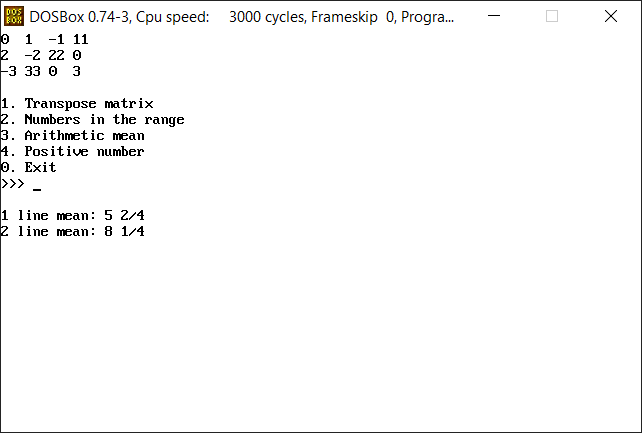
**Рисунок 5.** Главное меню



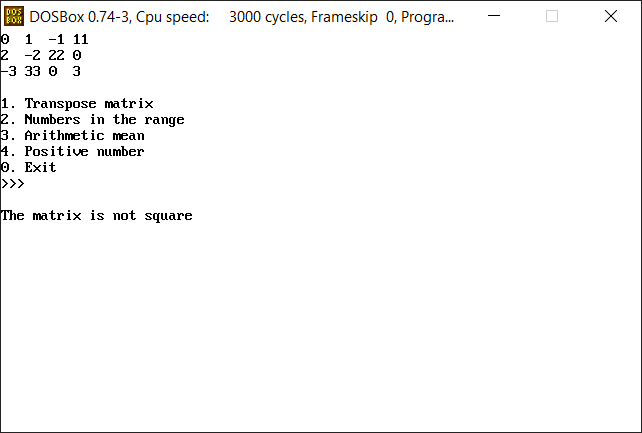
**Рисунок 6.** Транспонирование



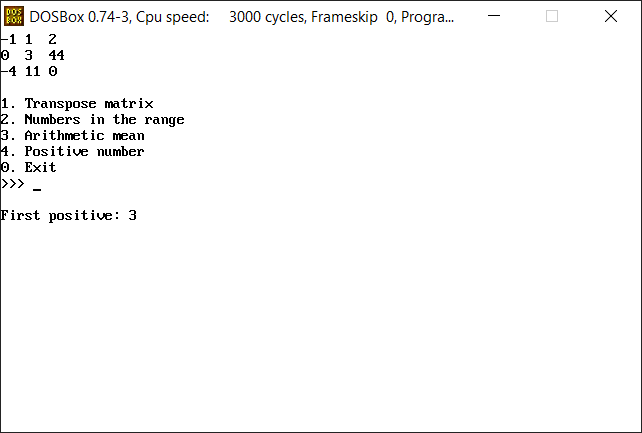
**Рисунок 7.** Задача «а»



**Рисунок 8.** Задача «b»



**Рисунок 9.1.** Задача «c»



**Рисунок 9.2.** Задача «c»

**Вывод:** в ходе выполнения лабораторной работы были получены навыки реализации ввода, вывода, транспонирования и обработки двумерных массивов.

**ЛИТЕРАТУРА**

**Основная литература**

1. Калашников О.А. Ассемблер – это просто. Учимся программировать [Текст] / О.А. Калашников. – СПб. БХВ-Петербург, 2012. – 336 с.
2. Кирнос В. Н. Введение в вычислительную технику: основы организации ЭВМ и программирование на Ассемблере [Электронный ресурс]: учеб. пособие /В. Н. Кирнос. – Томск: Эль Контент, 2011. – 172с. URL://biblioclub.ru/index.php?page=book\_red&id=208652

**Дополнительная литература**

1. Юров В. И. ASSEMBLER [Текст]. Учебник для вузов /В. И. Юров. 2-е

изд. – Спб.:Питер 2010. – 637с.: ил.

1. Юров В. И. ASSEMBLER [Текст]. Практикум. / В. И. Юров. 2-е изд. –

Спб.:Питер 2007. – 399 с.

1. Зубков С.В. ASSEMBLER для DOS, WINDOWS, UNIX [Текст] / С.В.

Зубков-3-е изд., М.:ДМК Пресс; 2004. – 608 с.: ил.

**Электронные ресурсы:**

1. Научная электронная библиотека <http://eLIBRARY.ru>.
2. Электронно-библиотечная система <http://e.lanbook.com>.
3. Электронно-библиотечная система «Университетская библиотека онлайн» http://biblioclub.ru.