Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет

информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра Информатики

Дисциплина «Конструирование программ»

**ОТЧЕТ**

к лабораторной работе №4

на тему:

**«Целочисленные арифметические операции. Обработка массивов числовых данных.»**

БГУИР 6-05-0612-02 39

|  |
| --- |
| Выполнил студент группы 353503  ДЕРГУН Николай Олегович |
|  |
| (дата, подпись студента) |
| Проверил ассистент каф. Информатики  РОМАНЮК Максим Валерьевич |
|  |
| (дата, подпись преподавателя) |

Минск 2024

# 1 Индивидуальное задание

**Задание 1. Вариант 19.** Ввести матрицу целых чисел размерностью 5х6 элементов. Найти номера столбцов с минимальной суммой элементов. Числовые данные вводятся с клавиатуры в виде строк символов (по умолчанию используется десятичная система счисления), при этом требуется производить проверку на переполнение разрядной сетки числа (по умолчанию используются знаковые 16-ти битные данные), для знаковых данных – знак требуется хранить в представлении самого числа (в дополнительном коде).

# 2 Выполнение работы

Задание предполагает множественный повторный ввод данных в однотипном формате, поэтому разумно вынести код обработки ввода в процедуры для простого их вызова в цикле (см. листинг 1).

Пользователь может вводить либо индексы ячейки матрицы и её значение, либо символ E на английской раскладке, который инициализирует не описанные пользователем ячейки нулями. Прочий ввод в консоль будет помечаться ошибочным и приводит к повторному запросу на ввод.

Листинг кода 1 – Процедуры, обрабатывающие ввода

manage\_input proc; return in bx if the input in buf is correct

; 0 - wrong input, 1 - E input, 2 - indexes

push di

push dx

lea di, buf + 2

call skip\_spaces

cmp [di], 'E'

mov dh, 1

mov bx, 1

je check\_ending

mov dh, 0

cmp [di], '0' ; indexes check

jb wrong\_in

cmp [di], '0' + rows - 1

ja wrong\_in

mov dl, [di]

sub dl, '0'

mov offset indexes, dl

inc di

call skip\_spaces

cmp [di], '0'

jb wrong\_in

cmp [di], '0' + columns - 1

ja wrong\_in

mov dl, [di]

sub dl, '0'

mov offset indexes + 1, dl

check\_ending:

inc di

call skip\_spaces

cmp [di], 0dh

jne wrong\_in

cmp dh, 1

je mi\_ret

mov bx, 2

jmp mi\_ret; indexes are all right

wrong\_in:

mov bx, 0

mi\_ret:

pop dx

pop di

ret

manage\_input endp

check\_number proc; return in bx if the number in buf is correct (1 or 0)

; write number in dw number

push cx

push si

push di

push dx

push ax

lea di, buf + 2

call skip\_spaces

mov si, di

cmp [si], '-'

mov dl, 0 ; dl shows if the number is negative

jne pos\_num\_in

mov dl, 1

inc si

pos\_num\_in:

push si

mov cx, 0 ; cntr for num chars

cnt\_chars:

cmp [si], 0dh

je end\_of\_num

cmp [si], ' '

je space\_check

cmp [si], '0'

jb wrong\_num\_si

cmp [si], '9'

ja wrong\_num\_si

inc si

inc cx

jmp cnt\_chars

space\_check:

cmp cx, 0

je wrong\_num\_si

mov di, si

call skip\_spaces

mov si, di

cmp [si], 0dh

je end\_of\_num

jmp wrong\_num\_si

end\_of\_num:

pop si

cmp cx, 0

je wrong\_num

cmp cx, 5

ja wrong\_num

je max\_check

parse\_to\_hex:

push dx

mov ax, si

add ax, cx

dec ax

mov si, ax

mov ax, 0

mov di, 1

mov bx, 0 ; result

parse\_loop:

mov ax, [si]

mov ah, 0

sub ax, '0'

mul di

add bx, ax

mov ax, di

push bx

mov bx, 10

mul bx

pop bx

mov di, ax

dec si

loop parse\_loop

pop dx

cmp dl, 1

push dx

je mk\_neg

jmp save\_num

mk\_neg:

not bx

add bx, 1

jmp save\_num

max\_check:

push cx

lea di, max\_nmbr\_str

push si

repe cmpsb

pop si

pop cx

ja wrong\_num

jmp parse\_to\_hex

save\_num:

pop dx

mov [number], bx

mov bx, 1

jmp cn\_ret

wrong\_num\_si:

pop si

wrong\_num:

mov bx, 0

cn\_ret:

pop ax

pop dx

pop di

pop si

pop cx

ret

check\_number endp

Был создан также ряд вспомогательных процедур для вывод и ввода данных. Были заданы константы rows и columns, определяющие размер матрицы. Макросы не использовались. Было определенно множество меток, в частности для матрицы, массива сумм столбцов, буфера ввода и так далее.

Таким образом основная логика программы, реализующая требуемый в задании алгоритм, имеет вид, представленный в листинге 2.

Листинг кода 2 – Основная логика программы

.model tiny

.code

org 100h

rows equ 5

columns equ 6

main proc

lea dx, entr\_str

call printl

main\_loop:

lea dx, wrong\_in\_str

call printl

call read\_in\_buf

call new\_line

call manage\_input

cmp bx, 0

je fail

cmp bx, 1

je find\_mins

nmbr\_entr:

lea dx, num\_in\_str

call printl

call read\_in\_buf

call new\_line

call check\_number

cmp bx, 0

je fail

mov bx, 0

mov dl, [indexes + 1]

mov dh, 0

add bx, dx

add bx, dx

mov dl, [indexes]

mov ax, columns \* 2

mul dx

add bx, ax ; final index

mov dx, [number]

mov si, offset matrix

add si, bx

mov [si], dx

jmp done

done:

lea dx, done\_str

call printl

jmp main\_loop

fail:

lea dx, fail\_str

call printl

jmp main\_loop

find\_mins:

mov cx, 0 ; outer cntr

mins\_loop\_o:

mov dx, 0 ; inner cntr

mov ax, 0 ; sum h

mov bx, 0 ; sum l

mins\_loop\_i:

mov si, offset matrix

push ax

push dx

mov ax, columns \* 2

mul dx

add si, ax

mov ax, cx

mov dx, 2

mul dx

add si, ax

pop dx

pop ax

add bx, [si]

adc ax, 0

inc dx

cmp dx, rows

jb mins\_loop\_i

mov si, offset sums

push ax

push dx

mov ax, cx

mov dx, 4

mul dx

add si, ax

pop dx

pop ax

mov [si], bx

add si, 2

mov [si], ax

inc cx

cmp cx, columns

jb mins\_loop\_o

mov dx, 0 ; min sum

mov cx, 1

cmp\_min:

call set\_sdi\_main

mov ax, [si]

cmp ax, [di]

jge not\_new\_min

jmp new\_min

not\_new\_min:

sub si, 2

sub di, 2

mov ax, [si]

cmp ax, [di]

jge cmp\_end

new\_min:

mov dx, cx

cmp\_end:

inc cx

cmp cx, columns

jb cmp\_min

; now dx contains one of the min sum

mov cx, 0

mov bx, 0 ; offset for mins array

cmp\_with\_min:

call set\_sdi\_main

mov ax, [di]

cmp ax, [si]

jne not\_equal

sub si, 2

sub di, 2

mov ax, [di]

cmp ax, [si]

jne not\_equal

mov si, offset mins

add si, bx

mov [si], cl

inc bx

not\_equal:

inc cx

cmp cx, columns

jb cmp\_with\_min

lea dx, res\_str

call printl

mov cx, 0

res\_out\_loop:

mov si, offset mins

add si, cx

mov dl, [si]

cmp dl, -1

je m\_ret

mov ax, '0'

add ax, [si]

mov dl, al

inc cx

mov ah, 02h

int 21h

mov dl, ','

int 21h

mov dl, ' '

int 21h

jmp res\_out\_loop

m\_ret:

ret

main endp

Корректность ввода выводится в консоль для лучшего опыта пользователя (см. рисунок 1).

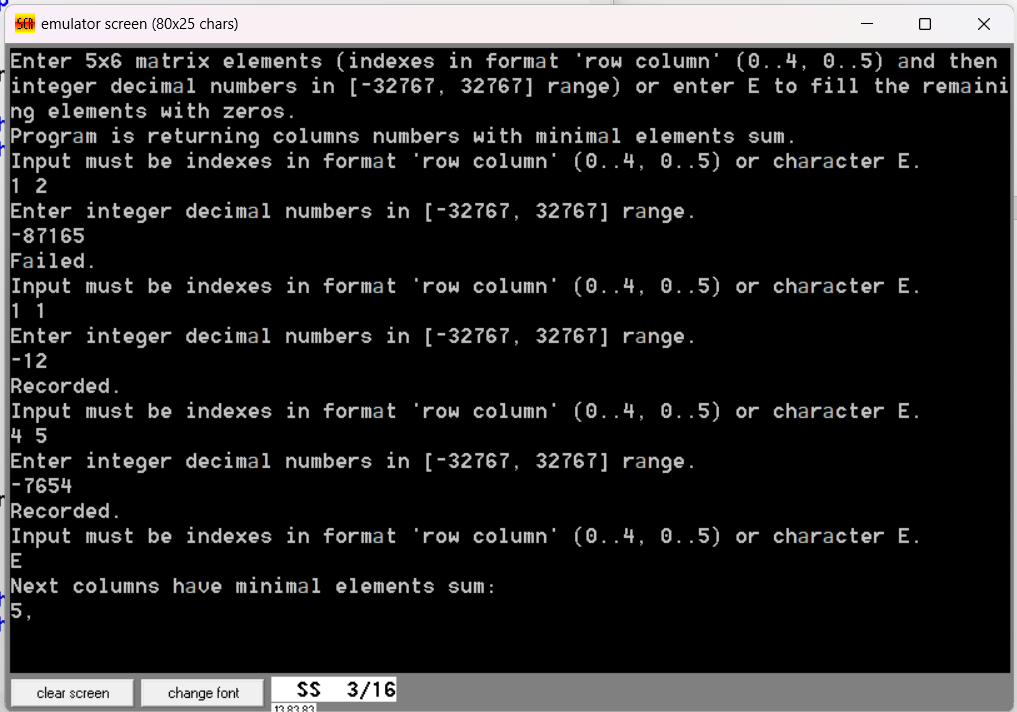


Рисунок 1 – Пример работы программы

# Вывод

В ходе выполнения лабораторной работы были получены навыки использования целочисленных арифметических операций и обработки числовых данных.