МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ

УНИВЕРСИТЕТ им. Р.Е.АЛЕКСЕЕВА

Институт радиоэлектроники и информационных технологий

Кафедра вычислительные системы и технологии

Лабораторная работа № 3

Разработка программы с псевдографическим режимом работы для ввода-вывода и обработки числовой информации

ОТЧЕТ

по лабораторной работе

по дисциплине

Принципы и методы  
организации системных программных средств

РУКОВОДИТЕЛЬ:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Викулова Е.Н.

СТУДЕНТ:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Игнаков К. М.

19-В-2

Работа защищена «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_

С оценкой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Нижний Новгород 2021

Цель работы.

Разработать программное обеспечение для расчета некоторых числовых данных с использованием псевдографики.

Задание.

Псевдографическая часть.

Программное обеспечение должно содержать псевдографический режим работы и содержать следующие окна для работы:

1. Основное окно – окно для ввода массива чисел;
2. “Тень” основного окна – окно, расположенное под основным окном с единичным смещением вниз и влево;
3. Окно вывода результата – расположено в центре терминала;
4. Окно ошибок – расположено в центре терминала, в случае неверного ввода, переполнения или прочих ошибок, связанных с вводом или расчетом данных.

Необходимо добавить поддержку перемещения основного окна внутри окна терминала за счет клавиш: “W” – вверх, “A” – влево, “S” – вниз, “D” – вправо, “Enter” – закончить перемещение окна и начать работу основной программы. Важно предусмотреть, чтобы основное окно не пропадало за границы окна терминала во время его перемещения.

Вычислительная часть.

Во время работы всей программы необходимо проверять корректность её вычислении результатов и делать проверку на вводимые пользователем данные. В случае возникновения ошибки вывести окно ошибки с информацией о ней и завершить работу программы. Расчеты производятся с учетом знаков чисел.

* Ввод 5 целых чисел в массив с произвольным знаком;
* Вычислить сумму чисел в массиве;
* Вычислить произведения чисел в массиве;
* Найти среднее значение всех чисел в массиве;
* Найти процент положительных чисел в массиве.

Структура программы.

Сегменты:

Программа является многосегментной.

1. Сегмент данных – содержит информацию для вывода, переменные для вычислений, цвета окон, сообщения для вывода и пр.
2. Сегмент кода – содержит основную логику работы программы: работа с окнами, ввод/вывод данных, вычисления и пр.
3. Стек – не несет информационной нагрузки на выполняемую задачу.
4. Макросы – в программе создано множество макросов для различных задач, облегчающие разработку продукта. В их числе есть:
   1. Print – для вывода информации на экран;
   2. Input – для ввода данных;
   3. drawWindow, drawShadow, clearWindow, drawError, drawAnswerWindow и drawAnswerWindow – для отрисовки окон под различные задачи. В каждый макрос принимает координаты отрисовки, кроме окна тени – принимает координаты отрисовки основного окна. Отрисовка создана по единому шаблону, отличает только цвет окон;
   4. printStringInWindow – напечатать текст по заданным координатам;
   5. pressButtonW (A, S, D) – перемещение основного окна внутри окна терминала с проверкой на выходы за границы.

Шаблон окон:

; xStart - левый верхний угол - столбец

; yStart - левый верхний угол - строка

; xEnd - правый нижний угол - столбец

; yEnd - правый нижний угол - строка

drawWindow macro xStart, yStart, xEnd, yEnd, color

mov ah, 06

mov al, 00

mov ch, yStart ; левый верхний угол - строка

mov cl, xStart ; левый верхний угол - столбец

mov dh, yEnd ; правый нижний угол - строка

mov dl, xEnd ; правый нижний угол - столбец

mov bh, <color> ; установка цвета фона и цвета букв

int 10h ; прерывание отрисовки

endm

Перемещение окна внутри окна консоли:

; y1 – верхняя координата окна

; y2 – нижняя координата окна

; pointToMove – куда “перепрыгнуть” после выполнения

pressButtonW macro y1, y2, pointToMove

cmp y1, 0d ; проверяем что окно не выйдет за границу

je pointToMove

dec y1 ; уменьшаем координаты

dec y2

jmp pointToMove

endm

Теоретическая справка.

Функция 6 int10h

Вход: ah=6

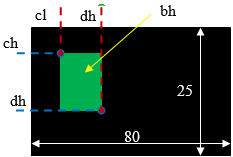
al=0

ch, cl – строка и столбец верхнего левого угла

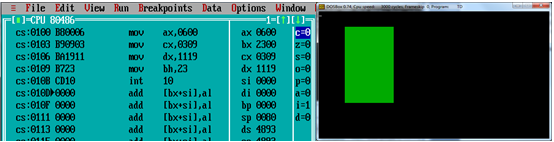
dh, dl – строка и столбец правого нижнего угла

bh – атрибут (цвет фона+цвет символа)

Выход: прямоугольник на экране



Пробуем в отладчике (выполнить F7, посмотреть Alt-F5)



Установить курсор в полученное окно:

mov ah,2

mov dx,0309

mov bh,0

int 10h

Вывести символы в текущую позицию курсора:

mov ah,9

mov al,31

mov cx,5

mov bl,45

mov bh,0

int 10h

Функция 0 int16h

Ввод символа с ожиданием без отображения на экране

Вход: al=0

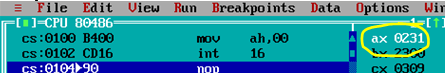
Выход: код нажатой клавиши в ax.

Для буквенно-цифровых клавиш: в al - ASCII-код символа, в ah – скан-код клавиши (№ клавиши), например, если нажмем «1» получим ax =0231h (клавиша №2, ASCII-код=31);

Для остальных: в al – префикс скан-кода (или 0 , если префикса нет), в ah – расширенный ASCII –код, например для  ax =4D00h.

Проверим в отладчике.

Так будет после нажатия «1»:



А так, после нажатия 🡪



Используя данную функцию, можно реализовать реакцию программы на определенные клавиши. Для этого достаточно сравнить возвращаемое значение с кодом клавиши, выбранной в качестве HotKey.

Листинг.

; -------------- Macro --------------

; Macros for output data

print macro srting

mov dx, offset srting

mov ah, 09h

int 21h

endm

; Macros for input data

input macro srting

mov dx, offset srting

mov ax, 0A00h

int 21h

endm

; Macro Draw Window

; xStart - left-top - col

; yStart - left-top - row

; xEnd - rigth-down - col

; yEnd - rigth-down - row

drawWindow macro xStart, yStart, xEnd, yEnd

push ax

push dx

mov ax, 0600h

mov ch, yStart ;????? ??????? ???? - ??????

mov cl, xStart ;????? ??????? ???? - ???????

mov dh, yEnd ;?????? ?????? ???? - ??????

mov dl, xEnd ;?????? ?????? ???? - ???????

mov bh, mainColor ;????????? ????? ???? ? ????? ????

int 10h ;?????????? ?????????

pop dx

pop ax

endm

drawShadow macro xStart, yStart, xEnd, yEnd

push ax

push dx

inc xStart

inc yStart

inc xEnd

inc yEnd

mov ax, 0600h

mov ch, yStart ;????? ??????? ???? - ??????

mov cl, xStart ;????? ??????? ???? - ???????

mov dh, yEnd ;?????? ?????? ???? - ??????

mov dl, xEnd ;?????? ?????? ???? - ???????

mov bh, shadowColor ;????????? ????? ???? ? ????? ????

int 10h ;?????????? ?????????

dec xStart

dec yStart

dec xEnd

dec yEnd

pop dx

pop ax

endm

clearWindow macro

mov ax, 0003

int 10h

endm

drawErrorWindow macro xStart, yStart, xEnd, yEnd

mov ax, 0600h

mov ch, yStart ;????? ??????? ???? - ??????

mov cl, xStart ;????? ??????? ???? - ???????

mov dh, yEnd ;?????? ?????? ???? - ??????

mov dl, xEnd ;?????? ?????? ???? - ???????

mov bh, 01000000b ;????????? ????? ???? ? ????? ????

int 10h ;?????????? ?????????

endm

drawAnswerWindow macro xStart, yStart, xEnd, yEnd

push ax

push dx

mov ax, 0600h

mov ch, yStart ;????? ??????? ???? - ??????

mov cl, xStart ;????? ??????? ???? - ???????

mov dh, yEnd ;?????? ?????? ???? - ??????

mov dl, xEnd ;?????? ?????? ???? - ???????

mov bh, 01110001b ;????????? ????? ???? ? ????? ????

int 10h ;?????????? ?????????

pop dx

pop ax

endm

printStringInWindow macro string, col, row

push ax

push dx

mov ah, 2

mov dh, row

mov dl, col

mov bh, 0

int 10h

mov ah, 09h

mov dx, offset string

int 21h

pop dx

pop ax

endm

pressButtonW macro y1, y2, pointToMove

cmp y1, 0d

je pointToMove

dec y1

dec y2

jmp pointToMove

endm

pressButtonS macro y1, y2, pointToMove

cmp y2, 24d

je pointToMove

inc y2

inc y1

jmp pointToMove

endm

pressButtonA macro x1, x2, pointToMove

cmp x1, 0d

je pointToMove

dec x1

dec x2

jmp pointToMove

endm

pressButtonD macro x1, x2, pointToMove

cmp x2, 79d

je pointToMove

inc x2

inc x1

jmp pointToMove

endm

sleep macro time

mov al, 0

mov ah, 86h

mov cx, time

int 15h

endm

; -------------- Macro -------------- END

; -------------- Data --------------

.model small

.data

; MESSAGES

pressAnyKey db 'Press any key.', 10, '$'

mess1 db 10, 13, 'Move window: W, A, S, D.. Press <Enter>$'

lab3 db 'Laboratory work #3','$'

author db 13,10, 'Ignakov Konstantin 19-V-2'

perevod db 10,13,'$'

inputError db 'Input Error!', 10, '$'

divByZeroError db 'Divition by zero!', 10, '$'

overflow db 'Overflow!', 10, '$'

overflowSum db 'Overflow Sum!', 10, '$'

sumText db 'Sum of elements: ','$'

lenSumText = $ - sumText

mulText db 'Mul of elements: ','$'

lenMulText = $ - mulText

sredText db 'Srednee: ','$'

lenSredText = $ - sredText

prochText db 'Prochent: ','$'

lenProchText = $ - prochText

out\_str db 6 dup (' '),'$'

enter\_please db 'Input value: $'

; COLORS

mainColor db 50h

shadowColor db 30h

; COORDINATES

mainWindowX1 db 5d

mainWindowX2 db 30d

mainWindowY1 db 5d

mainWindowY2 db 15d

secondWindowX1 db 5d

secondWindowX2 db 35d

secondWindowY1 db 5d

secondWindowY2 db 15d

cursorX db 0d

cursorY db 0d

; VALUES FOR TASK

in\_str label byte ; String (only for 6 chars)

razmer db 7 ; Buffer (for 6 chars)

kol db (?) ; Count of input chars

stroka db 7 dup (?) ; Buffer input value

number dw 5 dup (0) ; Array of values

sumres dw 0 ; Sum of values in the array

mulres dw 0 ; Mul of values in the array

sredres dw 0 ; Avarage value of values in the array

prochres dw 0 ; Percent of positive value of values in the array

siz dw 5 ; Lenght of the array

flag\_err equ 1

.stack 0ffh

.code

.386

start:

mov ax, @data

mov ds, ax

; video-mode #3 text 80\*25

clearWindow

; print mess1

print mess1

; draw main window

drawShadow mainWindowX1, mainWindowY1, mainWindowX2, mainWindowY2

drawWindow mainWindowX1, mainWindowY1, mainWindowX2, mainWindowY2

; -------------- Move Main Window --------------

mov ah, 0

int 16h

moveWindow:

clearWindow

drawShadow mainWindowX1, mainWindowY1, mainWindowX2, mainWindowY2

drawWindow mainWindowX1, mainWindowY1, mainWindowX2, mainWindowY2

call hideCursor

mov ah, 0

int 16h

; if <Enter>

cmp ax, 1c0dh

je startInput

; press W

cmp al, 77h

jne mainWindowsCheckPressA

pressButtonW mainWindowY1, mainWindowY2, moveWindow

; press A

mainWindowsCheckPressA:

cmp al, 61h

jne mainWindowsCheckPressS

pressButtonA mainWindowX1, mainWindowX2, moveWindow

; press S

mainWindowsCheckPressS:

cmp al, 73h

jne mainWindowsCheckPressD

pressButtonS mainWindowY1, mainWindowY2, moveWindow

; press D

mainWindowsCheckPressD:

cmp al, 64h

jne moveWindow

pressButtonD mainWindowX1, mainWindowX2, moveWindow

; -------------- Move Main Window -------------- END

; -------------- Input Data --------------

startInput:

mov al, mainWindowX1

mov cursorX, al

mov al, mainWindowY1

mov cursorY, al

clearWindow

drawShadow mainWindowX1, mainWindowY1, mainWindowX2, mainWindowY2

drawWindow mainWindowX1, mainWindowY1, mainWindowX2, mainWindowY2

xor di, di

mov cx, siz ; cx - array size

inputValues:

push cx

; Enter value

printStringInWindow enter\_please, cursorX, cursorY

input in\_str ; ???? ????? ? ???? ??????

call diapazon

cmp bh, flag\_err ; if flag\_err = 1 - error

je inErr

call dopust

cmp bh, flag\_err ; if flag\_err = 1 - error

je inErr

call AscToBin ; Convert string to value

inc di

inc di

pop cx

inc cursorY

loop inputValues

jmp searchSum

inErr:

drawErrorWindow 27, 9, 53, 16

printStringInWindow inputError, 34d, 12d

printStringInWindow pressAnyKey, 33d, 14d

call hideCursor

mov ah, 0 ; wait input

int 16h

jmp endProgram

; -------------- Input Data -------------- END

; -------------- Count answer --------------

; Sum of elements

searchSum:

mov cx, siz ; cx - array size

mov si, offset number

searchLoop:

mov ax, [si]

addToRes:

add sumres, ax

jo overFlowErrSum

nextVal:

inc si

inc si

loop searchLoop

; Multiply of elements

searchNegMul:

mov cx, siz

mov si, offset number

mov ax, 1 ; mul always use ax (smth \* 1 = smth)

minusEl:

mov bx, [si]

imul bx ; multiply

jo overFlowErr ; chech overflow

plusEl:

inc si

inc si

loop minusEl

mov mulres, ax ; mow answer to variable

; Average value of elements

searchSrednee:

mov ax, sumres

cwd

idiv siz

mov sredres, ax

; Percent of positive elements

; Formula (count \* 100) / SIZE %

searchProchent:

mov cx, siz

mov si, offset number

mov ax, 0h

findPosElem:

mov bx, [si]

cmp bx, 0h

jl foundNegElem ; If value is negative - skip

add ax, 100d ; 1 peace \* 100 - check formula

jo overFlowErrSum

foundNegElem:

inc si

inc si

loop findPosElem

cwd

idiv siz

mov prochres, ax

jmp outputRes

; Print errors

overFlowErr:

drawErrorWindow 27, 9, 53, 16

printStringInWindow overflow, 34d, 12d

printStringInWindow pressAnyKey, 33d, 14d

call hideCursor

mov ah, 0 ; wait input

int 16h

jmp endProgram

zero:

drawErrorWindow 27, 9, 53, 16

printStringInWindow divByZeroError, 34d, 12d

printStringInWindow pressAnyKey, 33d, 14d

call hideCursor

mov ah, 0 ; wait input

int 16h

jmp endProgram

overFlowErrSum:

drawErrorWindow 27, 9, 53, 16

printStringInWindow overflowSum, 34d, 12d

printStringInWindow pressAnyKey, 33d, 14d

call hideCursor

mov ah, 0 ; wait input

int 16h

jmp outputRes

; -------------- Count answer -------------- END

; -------------- Output answer --------------

outputRes:

; clearWindow

drawAnswerWindow 20, 5, 50, 17

mov al, 21

mov cursorX, al

mov al, 6

mov cursorY, al

printStringInWindow sumText, cursorX, cursorY

add cursorX, lenSumText

mov ax, sumres

call BinToAsc

printStringInWindow out\_str, cursorX, cursorY

inc cursorY

mov al, 21

mov cursorX, al

mov cx, 6 ; clear buffer

xor si, si

clear1:

mov [out\_str+si], ' '

inc si

loop clear1

printStringInWindow mulText, cursorX, cursorY

add cursorX, lenMulText

mov ax, mulres

call BinToAsc

printStringInWindow out\_str, cursorX, cursorY

inc cursorY

mov al, 21

mov cursorX, al

mov cx,6 ; clear buffer

xor si, si

clear2:

mov [out\_str+si],' '

inc si

loop clear2

printStringInWindow sredText, cursorX, cursorY

add cursorX, lenSredText

mov ax, sredres

call BinToAsc

printStringInWindow out\_str, cursorX, cursorY

inc cursorY

mov al, 21

mov cursorX, al

mov cx,6 ; clear buffer

xor si,si

clear3:

mov [out\_str + si], ' '

inc si

loop clear3

printStringInWindow prochText, cursorX, cursorY

add cursorX, lenProchText

mov ax, prochres

call BinToAsc

printStringInWindow out\_str, cursorX, cursorY

jmp endProgram

;-------------- Output answer -------------- END

endProgram:

mov ah, 02

int 21h

mov ax, 4c00h

int 21h

;-------------- Procedures --------------

; Hide cursor

hideCursor PROC

mov ah,2

mov dh,26

mov dl,81

mov bh,0

int 10h

ret

ENDP

; Data in (-29999, +29999)

; Args:

; Buffer input - stroka

; Res:

; bh - error flag

DIAPAZON PROC

xor bh, bh

xor si, si

cmp kol, 05h ; Count is 5

jb dop

cmp stroka, 2dh ; If enter 5 and more, check for `-`

jne plus ; If first is not `-` - check count of numbers

cmp kol, 06h ; If first `-` and values less 6

jb dop

inc si ; Chech first value

jmp first

plus:

cmp kol, 6 ; If count of data is 6 and first is not `-`

je error1 ; ERROR

first:

cmp stroka[si], 32h ; Check first simbol with `2`

jna dop ; If first simbol <= '2' - check the validity of the characters

error1:

mov bh, flag\_err ; Else bh = flag\_err

dop:

ret

DIAPAZON ENDP

; Chech data is value or `-`

; Args:

; Buffer input - stroka

; si - number of simbol

; Res:

; bh - error flag

DOPUST PROC

xor bh, bh

xor si, si

xor ah, ah

xor ch, ch

mov cl, kol ; Count of input chars

m11:

mov al, [stroka + si] ; In `al` - first

cmp al, 2dh ; If first char is `-`

jne testdop ; If no

cmp si, 00h ; If yes

jne error2 ; If another - ERROR

jmp m13

testdop:

cmp al, 30h ; Data in [30h, 39h]

jb error2

cmp al, 39h

ja error2

m13:

inc si

loop m11

jmp m14

error2:

mov bh, flag\_err ; Another error

m14:

ret

DOPUST ENDP

; ASCII to number

; Args:

; cx - Count of input simbols

; bx - number of simbol from last

; Res:

; number - buffer numbers

; di - number of value in the array

AscToBin PROC

xor ch, ch

mov cl, kol

xor bh, bh

mov bl, cl

dec bl

mov si, 01h ; In si, the weight of the discharge

n1:

mov al, [stroka + bx]

xor ah, ah

cmp al, 2dh ; Check positive or negative

je otr ; if negative

sub al, 30h

mul si

add [number + di], ax

mov ax, si

mov si, 10

mul si

mov si, ax

dec bx

loop n1

jmp n2

otr:

neg [number + di] ; Show negative in `adding` code

n2:

ret

AscToBin ENDP

; Number to ASCII

; Args

; ax - number

; Res

; out\_str - numbers buffer

BinToAsc PROC

xor si, si

add si, 05h

mov bx, 0Ah

push ax

cmp ax, 00h

jnl mm1

neg ax

mm1:

cwd

idiv bx

add dl,30h

mov [out\_str + si], dl

dec si

cmp ax, 00h

jne mm1

pop ax

cmp ax, 00h

jge mm2

mov [out\_str + si], 2dh

mm2:

ret

BinToAsc ENDP

end start

Список портов ввода/вывода.

Список стандартных портов ввода-вывода

----------------------------------------

Порты - это логические номера, через которые ЦП обращается к внешним устройствам. Это аналог адресов ОЗУ, только относящиеся к внешним устройствам. Номер порта - это число от 0 до 65535 (FFFFh). Первые 256 адресов (FFh) зарезервированы под архитектуру компьютеров. Остальные порты могут использоваться сторонним оборудованием, хотя многие из них тоже стандартны.

000-00F Контроллер DMA

010-01F Контроллер DMA (PS/2)

020-02F Главный контроллер прерываний

030-03F Подчиненный контроллер прерываний

040-05F Programmable Interval Timer (PIT)

060-06F Контроллер клавиатуры

070-071 Часы реального времени

080-083 DMA Page Register

090-097 Programmable Option Select (PS/2)

0A0-0AF PIC #2

0C0-0CF DMAC #2

0E0-0EF

0F0-0FF Математический сопроцессор, PCJr Disk Controller

100-10F Programmable Option Select (PS/2)

110-16F

170-17F Hard Drive 1 (AT)

180-1EF

1F0-1FF Hard Drive 0 (AT)

200-20F Game Adapter

210-217 Expansion Card Ports

220-26F

278-27F Parallel Port 3

280-2A1

2A2-2A3 Сlock

2B0-2DF EGA/Video

2E2-2E3 Data Acquisition Adapter (AT)

2E8-2EF Последовательный порт COM4

2F0-2F7

2F8-2FF Последовательный порт COM2

300-31F Prototype Adapter, Periscope Hardware Debugger

320-32F

330-33F

340-35F

360-36F Network

370-377 Контроллер дисковода FDD

378-37F Параллельный порт LPT2

380-38F SDLC Adapter

390-39F Cluster Adapter

3A0-3AF

3B0-3BF Monochrome Adapter

3BC-3BF Параллельный порт LPT1

3C0-3CF EGA/VGA

3D0-3DF Color Graphics Adapter

3E0-3EF Последовательный порт COM3

3F0-3F7 Контроллер дисковода FDD

3F8-3FF Последовательный порт COM1

Вывод.

В ходе работы программы я научился работать с графическим режимом в языке ассемблер. “Создавать” HotKey и работать с цветами в консоли.