МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФГБОУ ВО АЛТАЙСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Институт цифровых технологий, электроники и физики

Кафедра вычислительной техники и электроники (ВТиЭ)

Лабораторная работа № 02

**Программирование ввода-вывода. Работа с консолью.**

Выполнил студент 595 гр.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Д.В. Осипенко

Проверил: к.т.н,, доцент каф. ВТиЭ

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Ю.Г. Скурыдин

Лабораторная работа защищена

«\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2020 г.

Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Барнаул 2020

1. **Формулировка задачи**

Разработать программу, которая при нажатии клавиши Enter выводит текстовую строку из центра консоли вправо-вверх, по нажатию клавиши F4 выводит текстовую строку из нижнего левого угла консоли вправо, по нажатию клавиши "пробел" - текстовую строку из правого нижнего угла консоли вверх. Выход из программы - нажатием клавиши "+" на дополнительной клавиатуре. На нажатие прочих клавиш программа не реагирует.

1. **Постановка задачи**

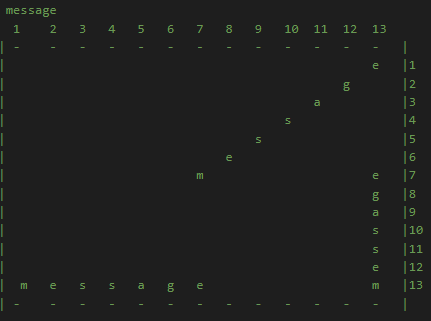
На вход программы последовательно поступают коды нажатых клавиш. При нажатии на клавиши F4, Enter и пробел программа выводит текстовые строки в заданной конфигурации. При нажатии на клавишу “+” происходит выход из программы. На нажатие остальных клавиш программа не реагирует.

1. **Математическая модель**

На вход программы последовательно поступают коды нажатых клавиш. Программа последовательно сравнивает поступившие коды с кодами клавиш F4, Enter, пробел и «+». Если код поступившей клавиши совпадает с кодом клавиши F4, то выводится строка из нижнего левого угла в правую часть экрана. Если код поступившей клавиши совпадает с кодом клавиши Enter, то выводится строка из центра экрана вправо-вверх. Если код поступившей клавиши совпадает с кодом клавиши пробел, то выводится строка из правого нижнего угла консоли вверх. Если код поступившей клавиши совпадает с кодом клавиши «+», то происходит завершение работы программы. Если поступивший код не совпадает ни с одним из кодов клавиш F4, Enter, пробел и «+», то программа ожидает повторного нажатия на клавишу.

В качестве текстовой строки выбрано слово «message», для вывода которого используется 2n-1 строк и 2n-1 столбцов (Где n – кол-во символов в сообщении).

* Для вывода сообщения из центра в право-вверх мы берем текст в обратном порядке (1я буква это последний символ строки, 2я это предпоследний и т.д.), где в первых n строках находятся пробелы (кол-во пробелов в строках - 1: (2n-1)-1; 2: (2n-1)-2; …; n: (2n-1)-n.), а затем символ. Последующие n-1 строк пустые.
* Для вывода сообщения из нижнего левого угла в право мы берем 2n-2 пустых строк и на следующей выводим n символов сообщения в стандартном порядке (1я буква в первом столбце, 2я во втором и т.д.).
* Для вывода сообщения из правого нижнего угла консоли вверх мы берем n-1 пустых строк, затем берем текст в обратном порядке и выводим оставшиеся n строк, где строка это 2n-2 пробелов и символ



1. **Описание алгоритма**

Начало

* 1. Поместить в регистр ah функцию 07h.
  2. Вызываем 21 прерывание. Происходит считывание кода нажатой клавиши и помещение его в регистр al.
  3. Сравниваем с кодом клавиши F4 содержимое регистра al. Если совпадают, то переход к п.4.7.
  4. Сравниваем с кодом клавиши Enter содержимое регистра al. Если совпадают, то переход к п.4.8.
  5. Сравниваем с кодом клавиши пробел содержимое регистра al. Если совпадают, то переход к п.4.9.
  6. Сравниваем с кодом клавиши «+» содержимое регистра al. Если совпадают, то переход к п.4.12. Иначе, к п.4.1.
  7. Загрузка в регистр dx адреса, где расположена строка msgBLtR. Переход к п.4.10.
  8. Загрузка в регистр dx адреса, где расположена строка msgCtTR. Переход к п.4.10.
  9. Загрузка в регистр dx адреса, где расположена строка msgBRtT. Переход к п.4.10.
  10. Поместить в регистр ah функцию 09h.
  11. Вызов 21 прерывания. Вывод строки, адрес которой сохранен в dx, на экран. Переход к п.4.1.
  12. Поместить в регистр ah функцию 4с. Вызов 21 прерывания.

Конец

1. **Опорный граф (блок-схема) алгоритма**
2. **Описание используемых процедур и инструментов**

Для обработки нажатия клавиш воспользуемся функцией номер 7 прерывания int 21, которая обрабатывает нажатие клавиш без вывода кода на экран. Для вывода строк воспользуемся функцией номер 9 того же прерывания. Для обработки завершения программы воспользуемся функцией номер 4с прерывания int 21, которая возвращает систему в исходное состояние после выполнения программы. Программу разработаем в формате exe.

1. **Текст программы**

MODEL MEDIUM

.STACK 200H

.DATA

msgCtTR db ' e',10,' g',10,' a',10,' s',10,' s',10,' e',10,' m',10,10,10,10,10,10,10,36

msgBLtR db 10,10,10,10,10,10,10,10,10,10,10,10,"message",10,36

msgBRtT db 10,10,10,10,10,10,' e',10,' g',10,' a',10,' s',10,' s',10,' e',10,' m',10,36

greetings db 10,"Press some of keys(Enter,F4,Space) or '+' key for exit.",10,36

.CODE

start:

mov ax, @data

mov ds, ax

lea dx, greetings

jmp print

check:

mov ah, 7

int 21h

cmp al, 13

je centerToTopRight

cmp al, 62;20

je bottomLeftToRight

cmp al, 32

je bottomRightToTop

cmp al, 43

je exit

jmp check

centerToTopRight:

lea dx, msgCtTR

jmp print

bottomLeftToRight:

lea dx, msgBLtR

jmp print

bottomRightToTop:

lea dx, msgBRtT

jmp print

print:

mov ah, 9

int 21h

jmp check

exit:

mov ah, 4ch

int 21h

END start

1. **Формулировка задачи**

Написать программу, которая выводит в центр экрана три символа звездочки (\*\*\*). При нажатии клавиш стрелки - вверх, вниз, влево, вправо - осуществляется перемещение группы символов по экрану в соответствующих направлениях.

1. **Постановка задачи**

На вход программы последовательно поступают коды нажатых клавиш стрелок. При нажатии на клавиши стрелки – вверх, вниз, вправо, влево осуществляется перемещение группы символов по экрану в соответствующем направлении. При нажатии на клавишу Enter происходит выход из программы. На нажатие остальных клавиш программа не реагирует.

1. **Математическая модель**

На вход программы последовательно поступают коды нажатых клавиш. Программа последовательно сравнивает поступившие коды с кодами клавиш стрелок – вверх, вниз, вправо, влево. Если код поступившей клавиши совпадает с кодом клавиши стрелка вверх, то значение смещения(offset) относительно адреса сегмента text mode (b800h) уменьшается на 160 байт (что эквивалентно длине одной строки): offset = offset – 160. Если код поступившей клавиши совпадает с кодом клавиши стрелка вниз, то значение смещения (offset) увеличивается на 160 байт: offset=offset+160. Если код поступившей клавиши совпадает с кодом клавиши стрелка вправо, то значение смещения (offset) увеличивается на 2 байта: offset=offset+2. Если код поступившей клавиши совпадает с кодом клавиши стрелка влево, то значение смещения (offset) уменьшается на 2 байта: offset=offset-2. После изменения значения смещения, на offset+0, offset+2, offset+4 присваиваются код символа «\*». Если код поступившей клавиши совпадает с кодом клавиши Enter, то происходит завершение работы программы. Если поступивший код не совпадает ни с одним из кодов клавиш стрелок или Enter, то программа ожидает повторного нажатия на клавишу.

1. **Описание алгоритма**

Начало

* 1. Поместить в регистр ax 0003h, вызываем 10h прерывание. Происходит переход в text display mode.
  2. Поместить в es адрес начала сегмента для text display mode.
  3. Поместить в bx значение 80\*21, начальное смещение.
  4. Поместить в ax значение регистра bx.
  5. Поместить в cx значение 3
  6. Поместить в es:[bx] значение 0
  7. Увеличить значение bx на 2
  8. Уменьшить значение cx на 1
  9. Если значение cx > 0, переход к п.4.6.
  10. Поместить значение ax в bx.
  11. Поместить в cx значение 3
  12. Поместить в es:[bx] значение 42
  13. Увеличить значение bx на 2
  14. Уменьшить значение cx на 1
  15. Если значение cx > 0, переход к п.4.12.
  16. Поместить в ax значение регистра bx.
  17. Поместить в регистр ah функцию 07h.
  18. Вызываем 21 прерывание. Происходит считывание кода нажатой клавиши и помещение его в регистр al.
  19. Сравниваем с кодом клавиши вниз содержимое регистра al. Если совпадают, то переход к п.4.24.
  20. Сравниваем с кодом клавиши вверх содержимое регистра al. Если совпадают, то переход к п.4.26.
  21. Сравниваем с кодом клавиши влево содержимое регистра al. Если совпадают, то переход к п.4.30.
  22. Сравниваем с кодом клавиши вправо содержимое регистра al. Если совпадают, то переход к п.4.28.
  23. Сравниваем с кодом клавиши Enter содержимое регистра al. Если совпадают, то переход к п.4.32. Иначе, к п.4.17.
  24. Помещение в ax значение регистра bx.
  25. Увеличить значение ax на 160. Переход к п.4.5.
  26. Помещение в ax значение регистра bx.
  27. Уменьшение значение ax на 160. Переход к п.4.5.
  28. Помещение в ax значение регистра bx.
  29. Увеличить значение ax на 2. Переход к п.4.5.
  30. Помещение в ax значение регистра bx.
  31. Уменьшение значение ax на 2. Переход к п.4.5.
  32. Поместить в регистр ah функцию 4с. Вызов 21 прерывания.

Конец

1. **Опорный граф (блок-схема) алгоритма**
2. **Описание используемых процедур и инструментов**

Для обработки нажатия клавиш воспользуемся функцией номер 7 прерывания int 21, которая обрабатывает нажатие клавиш без вывода кода на экран. Для вывода воспользуемся функцией номер 3 прерывания 10h. Для обработки завершения программы воспользуемся функцией номер 4с прерывания int 21, которая возвращает систему в исходное состояние после выполнения программы. Программу разработаем в формате exe.

1. **Текст программы**

MODEL MEDIUM

.STACK 200H

.DATA

ver\_offs dw 160

hor\_offs dw 2

color db 10

smb\_code db 42 ;star char code

.CODE

start:

mov ax, @data

mov ds, ax

mov ax, 0003h

int 10h

mov bx, 0b800h

mov es, bx

mov bx, 80\*21 ; start possition (center offset)

mov dh, 0 ; for clear screen

mov dl, smb\_code

mov ax, bx

jmp update\_scrn

check:

xor ax, ax

mov ah, 7

int 21h

cmp al, 50h ; key down

je down

cmp al, 48h ; key up

je up

cmp al, 4bh ; key left

je left

cmp al, 4dh ; key right

je right

cmp al, 13 ; key enter for exit

je exit

jmp check

down:

mov ax, bx

add ax,ver\_offs

jmp update\_scrn

up:

mov ax, bx

sub ax, ver\_offs

jmp update\_scrn

left:

mov ax, bx

sub ax, hor\_offs

jmp update\_scrn

right:

mov ax, bx

add ax, hor\_offs

jmp update\_scrn

update\_scrn:

mov cx, 3

clr\_loop:

mov es:[bx], dh

add bx, 2

loop clr\_loop

mov bx, ax

mov cx, 3

print\_loop:

mov es:[bx], dl

add bx, 2

loop print\_loop

mov bx, ax

jmp check

exit:

mov ah, 4ch

int 21h

END start