Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ  
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра электронных вычислительных машин

Лабораторная работа № 4

по дисциплине ПнаЯА

Вариант 1

Выполнил студент гр. 250501: Снитко Д.А.

Проверил: Туровец Н.О.

Минск 2023

Теоретические сведения:

1. Видеорежимы и управление ими.

2. Прямой доступ к видеопамяти.

3. Управление курсором и вывод данных на экран прерыванием BIOS.

4. Обработка нажатий кнопок клавиатуры прерыванием BIOS. Буфер клавиатуры.

5. Доступ к системным часам.

Задание:

Игра «Змейка». Цель: ползаем, едим случайно появляющиеся в свободных местах яблочки и растем от этого в длину, выход за границы экрана означает возврат в поле с противоположной стороны (круглый мир). Окончание: проигрыш – укус самого себя, выигрыш – нет. Информация: счет. Усложнение: смена уровня (увеличение скорости)

Код:

.model small

.stack 100h

.data

videobuffSize equ 0FA0h ; Размер буфера видеопамяти (25\*80\*2)

startPositionOffset dw 07CAh ; Начальная позиция змейки (строка 12, столбец 37)

videobuff dw 0B800h ; Буфер видеопамяти (начальный адрес)

videobuffHeight db 25 ; Высота видеобуфера в строках

videobuffLengthInBytes db 160 ; Длина строки видеобуфера в байтах

defaultDirection equ right ; Начальное направление движения змейки

pointsUntilSpeedup equ 02h ; Количество очков до увеличения скорости

defaultInterruptsForMove equ 03h ; Количество прерываний таймера для движения змейки

defaultSnakeLength equ 0005h ; Начальная длина змейки

maxSnakeLength equ 2000 ; Максимальная длина змейки

snakeCell equ 0009h ; White circle ; Символ для тела змейки (белый круг)

point equ 0B03h ; Cyan heart ; Символ для яблока (голубое сердце)

nTwo db 02h ; Константа для умножения/деления на два

welcomText db "Snake!",0Dh,0Ah,"Control:",0Dh,0Ah,'$' ; Приветственное сообщение

controlText db 18h,", ",19h,", -> , <-",0Dh,0Ah,"Backspace - restart",0Dh,0Ah,"Esc - exit",0Dh,0Ah,'$' ; Инструкции по управлению

startOrExitText db "Press any key to start, Esc to exit...$" ; Сообщение о начале игры

gameOverText db "Game over! Final score: $" ; Сообщение об окончании игры

newLine db 0Dh,0Ah,'$' ; Переход на новую строку

top equ 48h

right equ 4Dh

bottom equ 50h

left equ 4Bh

direction db defaultDirection ; Текущее направление движения змейки

maxFinalScoreDigits equ 5 ; Максимальное количество цифр для отображения счета

finalScore dw ? ; Переменная для хранения текущего счета

finalScoreString db maxFinalScoreDigits dup('$') ; Строка для отображения счета

pointPos dw ? ; Позиция яблока на экране

interruptsForMove db ? ; Количество прерываний для движения змейки

interruptsCounter db ? ; Счетчик прерываний таймера

currentPointsUntilSpeedup db ? ; Текущее количество очков до увеличения скорости

snakeLength dw 0000h ; Длина змейки

snake dw maxSnakeLength dup(?) ; Массив для хранения позиций ячеек змейки

snakeMoved db ? ; Флаг для определения движения змейки

exit db 00h ; Флаг для выхода из программы

endGame db ? ; Флаг окончания игры

restart db ? ; Флаг для перезапуска игры

.code

start:

; Инициализация сегментов данных

mov ax, data

mov ds, ax

mov es, ax

; Очистка экрана

call clearScreen

; Получение обработчика таймера

mov ax, 351Ch

int 21h

mov cs:oldTimerHandler, bx

mov cs:oldTimerHandler + 2, es

; Получение обработчика клавиатуры

mov ax, 3509h

int 21h

mov cs:oldKeyboardHandler, bx

mov cs:oldKeyboardHandler + 2, es

; Вывод стартовых сообщений

mov ax, ds

mov es, ax

call showStartMessages

mov ax, 0C08h

int 21h

cmp al, 1Bh

je closeApp ; Закрыть, если нажат Esc

startGame:

; Очистка экрана, создание начального состояния и вывод элементов

call clearScreen

call buildStartState

call outputStartElements

; Отключение прерываний

cli

mov ax, cs

mov ds, ax

; Установка нового обработчика клавиатуры

lea dx, newKeyboardHandler

mov ax, 2509h

int 21h

; Установка нового обработчика таймера

lea dx, newTimerHandler

mov ax, 251Ch

int 21h

; Восстановление сегментов данных

mov ax, es

mov ds, ax

; Установка флага прерывания на 1

sti

gameLoop:

; Игровой цикл

cmp endGame, 0

je gameLoop

; Обработка прерываний

cli

mov ax, cs:oldTimerHandler + 2

mov ds, ax

mov dx, cs:oldTimerHandler

mov ax, 251Ch

int 21h

mov ax, cs:oldKeyboardHandler + 2

mov ds, ax

mov dx, cs:oldKeyboardHandler

mov ax, 2509h

int 21h

mov ax, es

mov ds, ax

; Установка флага прерывания на 1

sti

; Проверка условий завершения игры

cmp exit, 01h

je closeApp

cmp restart, 01h

je startGame

; Вывод сообщений об окончании игры

call showGameOverMessages

mov ax, 0C08h

int 21h

cmp al, 1Bh

jne startGame

closeApp:

; Завершение работы программы

call clearScreen

mov ax, 4c00h

int 21h

ends

clearScreen proc

; Очистка экрана

mov ax, 0002h

int 10h

ret

clearScreen endp

outputString proc

; Вывод строки

mov ah, 9

int 21h

ret

outputString endp

showStartMessages proc

; Вывод стартовых сообщений

lea dx, welcomText

call outputString

lea dx, controlText

call outputString

lea dx, startOrExitText

call outputString

ret

showStartMessages endp

showGameOverMessages proc

; Вывод сообщений об окончании игры

lea dx, gameOverText

call outputString

mov ax, finalScore

lea si, finalScoreString

call numberToString

mov dx, si

call outputString

lea dx, newLine

call outputString

lea dx, startOrExitText

call outputString

ret

showGameOverMessages endp

buildStartState proc

; Создание начального состояния

cmp snakeLength, 0

je buildNewSnake

lea si, snake

mov cx, snakeLength

xor ax, ax

resetSnake:

mov [si], ax

add si, 2

loop resetSnake

buildNewSnake:

mov snakeLength, defaultSnakeLength

mov cx, defaultSnakeLength

lea si, snake

mov ax, startPositionOffset

buildSnake:

mov [si], ax

add si, 2

add ax, 2

loop buildSnake

randomPointPos:

call random

cmp bx, startPositionOffset

jl savePointPos

sub ax, 2

cmp bx, ax

jg savePointPos

shr bx, 1

savePointPos:

mov cx, maxFinalScoreDigits

lea si, finalScoreString

clearScoreBuffer:

mov [si], '$'

inc si

loop clearScoreBuffer

xor ax, ax

mov pointPos, bx

mov currentPointsUntilSpeedup, pointsUntilSpeedup

mov interruptsCounter, al

mov interruptsForMove, defaultInterruptsForMove

mov endGame, al

mov restart, al

mov direction, defaultDirection

mov snakeMoved, al

mov finalScore, ax

ret

buildStartState endp

numberToString proc

; Преобразование числа в строку

pusha

xor cx, cx

mov bx, 000Ah

numberToStringParse:

xor dx, dx

div bx

add dx, '0'

inc cx

push dx

cmp ax, 0

jne numberToStringParse

moveNumberToString:

pop ax

mov [si], al

inc si

loop moveNumberToString

popa

ret

numberToString endp

random proc

; Генерация случайного числа

push ax

push dx

push cx

mov ah, 2Ch

int 21h

xor ax, ax

mov al, dl

mul dh

test ax, 1

jz checkNumber

inc ax

checkNumber:

cmp ax, videobuffSize

jl setPointPos

shr ax, 1

test ax, 1

jz setPointPos

inc ax

setPointPos:

mov bx, ax

pop cx

pop dx

pop ax

ret

random endp

outputStartElements proc

; Вывод начальных элементов (змейки и точки)

mov ax, videobuff

mov ds, ax

lea di, snake

outputCycle:

mov si, es:di

cmp si, 0

je outputEnd

mov [si], snakeCell

add di, 2

jmp outputCycle

outputEnd:

lea di, pointPos

mov si, es:di

mov [si], point

mov ax, es

mov ds, ax

ret

outputStartElements endp

newTimerHandler proc far

cmp endGame, 01h

je timerOldInt

cmp interruptsForMove, 01h

je move

inc interruptsCounter

mov cl, interruptsCounter

cmp cl, interruptsForMove

jne timerOldInt

mov interruptsCounter, 00h

move:

mov snakeMoved, 01h

lea si, snake

mov ax, snakeLength

dec ax

mul nTwo

add si, ax

mov ax, [si]

mov dl, direction

cmp dl, top

jne moveRight

sub ax, 160 ; Размер одной строки

cmp ax, 0

jnl isCrash

add ax, videobuffSize

jmp isCrash

moveRight:

cmp dl, right

jne moveBottom

mov bx, ax ; Размер одной ячейки

add bx, 2

div videobuffLengthInBytes

mov dx, ax

mov ax, bx

div videobuffLengthInBytes

cmp al, dl

mov ax, bx

je isCrash

sub ax, 160

jmp isCrash

moveBottom:

cmp dl, bottom

jne moveLeft

add ax, 160

cmp ax, videobuffSize

jl isCrash

sub ax, videobuffSize

jmp isCrash moveLeft:

mov bx, ax ;One cell size

sub bx, 2

cmp bx, 0

jge checkLeftBorder

mov ax, 158

jmp isCrash

checkLeftBorder:

div videobuffLengthInBytes

mov dx, ax

mov ax, bx

div videobuffLengthInBytes

cmp al, dl

mov ax, bx

je isCrash

add ax, 160

isCrash:

lea si, snake

mov cx, snakeLength

isCrashCycle:

mov bx, [si]

cmp bx, ax

je crash

add si, 2

loop isCrashCycle

jmp isPoint

crash:

mov endGame, 01h

jmp timerOldInt

isPoint:

cmp ax, pointPos

je addNewCell

lea di, snake

deleteLastCell:

mov bx, [di]

mov dx, videobuff

mov ds, dx

mov si, bx

mov [si], 0000h

mov dx, es

mov ds, dx

mov si, di

mov cx, snakeLength

moveBuffCycle:

add di, 2

mov bx, [di]

mov [si], bx

add si, 2

loop moveBuffCycle

moveLastCell:

mov [si], ax

mov dx, videobuff

mov ds, dx

jmp moveSnake

addNewCell:

mov [si], ax

inc finalScore

inc snakeLength

cmp interruptsForMove, 01h

je randomNewPoint

dec currentPointsUntilSpeedup

mov bl, currentPointsUntilSpeedup

cmp bl, 0

jne randomNewPoint

dec interruptsForMove

mov currentPointsUntilSpeedup, pointsUntilSpeedup

randomNewPoint:

call random

lea si, snake

mov cx, snakeLength

checkPointPos:

cmp [si], bx

je changePoint

add si, 2

loop checkPointPos

jmp printNewPoint

changePoint:

mov dx, videobuffSize

sub dx, 2

sub dx, bx

shr dx, 1

test dx, 1

jz isEven

inc dx

isEven:

mov bx, dx

lea si, snake

mov cx, snakeLength

jmp checkPointPos

printNewPoint:

mov pointPos, bx

mov dx, videobuff

mov ds, dx

mov si, bx

mov [si], point

moveSnake:

mov si, ax

mov [si], snakeCell

mov ax, es

mov ds, ax

timerOldInt:

db 0EAh

oldTimerHandler dd ?

newTimerHandler endp

newKeyboardHandler proc far

; Обработчик новых событий клавиатуры

in al, 60h

cmp al, 01h ;Esc

je endGamePressed

cmp al, 0Eh ;Backspace

je restartGamePressed

cmp snakeMoved, 00h

je keyboardOldInt

cmp al, top

jne rightKey

cmp direction, bottom

je keyboardOldInt

mov direction, top

jmp changeDirectionPressed

rightKey:

cmp al, right

jne bottomKey

cmp direction, left

je keyboardOldInt

mov direction, right

jmp changeDirectionPressed

bottomKey:

cmp al, bottom

jne leftKey

cmp direction, top

je keyboardOldInt

mov direction, bottom

jmp changeDirectionPressed

leftKey:

cmp al, left

jne keyboardOldInt

cmp direction, right

je keyboardOldInt

mov direction, left

jmp changeDirectionPressed

changeDirectionPressed:

mov snakeMoved, 00h

jmp keyboardOldInt

restartGamePressed:

; Обработка нажатия клавиши "Backspace" (перезапуск игры)

mov restart, 01h

mov endGame, 01h

jmp keyboardOldInt

endGamePressed:

; Обработка нажатия клавиши "Esc" (завершение игры)

mov endGame, 01h

mov exit, 01h

keyboardOldInt:

; Вызов старого обработчика прерывания клавиатуры

db 0EAh

oldKeyboardHandler dd ?

newKeyboardHandler endp

end start

Результат работы программы:

