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 №7

на тему:

**«ИНТЕГРАЦИЯ АССЕМБЛЕРНЫХ ПРЕРЫВАНИЙ В ПРОЕКТЫ НА C++»**

БГУИР 6-05-0612-02 05

|  |
| --- |
| Выполнила студентка группы 353504  АНТОНОВА Лидия Сергеевна |
|  |
| (дата, подпись студента) |
| Проверил ассистент каф. Информатики  РОМАНЮК Максим Валерьевич |
|  |
| (дата, подпись преподавателя) |

Минск 2024

# 1 Индивидуальное задание

**Задание 1. Вариант 5.** На стороне Assembler:

Необходимо создать макросы для комбинаций клавиш, выполняя ассемблерные функции в ответ на активацию макроса.  
Макросы:

1 F1 - вывод всех макросов и их описание.

2 При нажатии комбинации s + d вызывается ассемблерная функция, которая активирует PC Speaker для издания звукового сигнала определенной частоты и длительности.

3 Длительность звукового сигнала определяется по времени зажатия клавиш s + d(Например: клавиши были зажаты 30 секунд, после отпускания клавиш активируется звуковой сигнал протяженностью 30 секунд).

4 При нажатии другой комбинации клавиш, например s + c, вызывается ассемблерная функция, которая изменяет цвет фона и/или текста в консоли.

5 При нажатии комбинации, например s + r, текст на экране начинает "вращаться" (циклически сдвигаться влево или вправо).

На стороне C++:

В главной функции создайте цикл, который будет проверять состояние клавиатуры с использованием функции, написанной на ассемблере. При обнаружении комбинации клавиш вызывайте соответствующую функцию, написанную на ассемблере.

Логирование: Записывайте все действия пользователя и реакции программы на них в файл лога. Это может быть полезно для отладки или просто для истории использования.

Используемые прерывания: int 16h, int 10h, int 1Ah.

# 2 Выполнение работы

Для написания данной лабораторной работы используется TASM и редактор Turbo C++ в среде DOS. Ниже представлен листинг кода.

#include "..\include\stdio.h"

#include "..\include\stdlib.h"

#include "..\include\time.h"

#include "..\include\stdarg.h"

extern "C" short func();

extern "C" void printHelp();

extern "C" short soundKey();

extern "C" void colorize();

extern "C" void rotate();

void log(FILE\* file, const char\* format, ...) {

time\_t currentTime;

time(&currentTime);

struct tm\* timeInfo = localtime(&currentTime);

char timeString[9];

strftime(timeString, sizeof(timeString), "%H.%M.%S", timeInfo);

fprintf(file, "[%s]: ", timeString);

va\_list args;

va\_start(args, format);

vfprintf(file, format, args);

va\_end(args);

fprintf(file, "\n");

}

int main() {

int duration = 0;

short key = 0;

FILE\* file = fopen("logs.txt", "w");

if (file == NULL) {

printf("Failed to open logs.txt\n");

return 0;

}

while (1) {

key = 0;

key = func();

if (key == 0) {

continue;

} else if (key == 1) {

printf("'F1' pressed\n`");

log(file, "'F1' pressed");

printHelp();

} else if (key == 2) {

log(file, "'s + d' pressed.");

duration = soundKey();

log(file, "Sound played. Duration: %d", duration);

} else if (key == 3) {

log(file, "'s + c' pressed");

colorize();

} else if (key == 4) {

log(file, "'s + r' pressed");

rotate();

} else if (key == 5) {

log(file, "'s + e' pressed");

return 0;

}

}

fclose(file);

return 0;

}

.model small

.data

duration dw 0

startTime dw 0

endTime dw 0

color db 7

n db 0

m db 0

row db 0

column db 0

buff db 2000 dup(?)

text1 db "Macro list: $"

text2 db "S + D: Make a sound with the duration.$"

text3 db "S + C: Change text or background color.$"

text4 db "S + R: Starts rotating text in console.$"

text5 db "S + E: Exit.$"

msgDur db "Duration: $"

msgLineEnd db 0Ah, 0Dh, '$'

.code

public \_func

\_func proc

push di

mov di, @data

waitKey:

mov ah, 01h

int 16h

jz waitKey

mov ah, 00h

int 16h

cmp al, 's'

je sPressed

cmp ah, 3Bh

je f1Pressed

jmp waitKey

f1Pressed:

mov ax, 1

jmp endCheck

sPressed:

mov ah, 0

int 1ah

mov startTime, dx

call checkIsKeyPress

jmp endCheck

endCheck:

pop di

ret

\_func endp

checkIsKeyPress proc

mov ah, 00h

int 16h

cmp al, 'd'

je dPressed

cmp al, 'c'

je cPressed

cmp al, 'r'

je rPressed

cmp al, 'e'

je ePressed

ret

dPressed:

mov ah, 0

int 1ah

mov endTime, dx

mov ax, endTime

sub ax, startTime

mov duration, ax

mov ax, 2

ret

cPressed:

mov ax, 3

ret

rPressed:

mov ax, 4

ret

ePressed:

mov ax, 5

ret

checkIsKeyPress endp

public \_printHelp

\_printHelp proc

push di

mov di,@data

mov ah, 09h

lea dx, text1

int 21h

lea dx, msgLineEnd

int 21h

lea dx, text2

int 21h

lea dx, msgLineEnd

int 21h

lea dx, text3

int 21h

lea dx, msgLineEnd

int 21h

lea dx, text4

int 21h

lea dx, msgLineEnd

int 21h

lea dx, text5

int 21h

lea dx, msgLineEnd

int 21h

dec color

call \_colorize

pop di

ret

\_printHelp endp

public \_soundKey

\_soundKey proc

push di

mov di, @data

push si

mov ah, 09h

lea dx, msgDur

int 21h

mov ax, duration

lea di, buff

call convertToStr

mov ah, 09h

mov dx, di

int 21h

mov ah, 09h

lea dx, msgLineEnd

int 21h

mov si, duration

; mov ax, 10

; mul si

; mov si, ax

generateSound:

in al, 61h

or al, 2

out 61h, al

call delay

in al, 61h

and al, 11111100b

out 61h, al

call delay

dec si

cmp si, 0

jg generateSound

mov ax, duration

dec color

call \_colorize

pop si

pop di

ret

\_soundKey endp

convertToStr proc

mov bx, 10

add di, 5

mov byte ptr [di], '$'

convertLoop:

dec di

mov dx, 0

div bx

add dl, '0'

mov byte ptr [di], dl

test ax, ax

jnz convertLoop

ret

convertToStr endp

delay proc

mov cx, 65535

waitDelay:

loop waitDelay

ret

delay endp

public \_colorize

\_colorize proc

;Arg color:byte

push di

mov di, @data

inc color

mov ah, 03h

mov bh, 0

int 10h

mov byte ptr n, dh

mov byte ptr m, dl

mov byte ptr row, 0

mov byte ptr column, 0

colorLoop:

mov ah, 02h

mov bh, 0

mov dh, byte ptr row

mov dl, byte ptr column

int 10h

mov ah, 08h

mov bh, 0

int 10h

mov ah, 09h

mov bh, 0

mov bl, byte ptr color

mov cx, 1

int 10h

inc byte ptr column

cmp byte ptr column, 80

jge nextRowColor

jmp colorLoop

nextRowColor:

inc byte ptr row

mov byte ptr column, 0

cmp byte ptr row, 25

jl colorLoop

mov ah, 02h

mov bh, 0

mov dh, byte ptr n

mov dl, byte ptr m

int 10h

pop di

ret

\_colorize endp

public \_rotate

\_rotate proc

;Arg color:byte

push di

mov di, @data

mov ah, 03h

mov bh, 0

int 10h

mov byte ptr n, dh

mov byte ptr m, dl

mainLoop:

lea si, buff

mov byte ptr row, 0

mov byte ptr column, 0

readLoop:

mov ah, 02h

mov bh, 0

mov dh, byte ptr row

mov dl, byte ptr column

int 10h

mov ah, 08h

mov bh, 0

int 10h

mov byte ptr [si], al

inc si

inc byte ptr column

cmp byte ptr column, 80

jge nextRowRotate

jmp readLoop

nextRowRotate:

inc byte ptr row

mov byte ptr column, 0

cmp byte ptr row, 25

jge endRead

jmp readLoop

endRead:

lea si, buff

mov byte ptr row, 0

mov byte ptr column, 1

mov di, 0

printLoop:

mov ah, 02h

mov bh, 0

mov dh, byte ptr row

mov dl, byte ptr column

int 10h

mov ah, 09h

mov bh, 0

mov bl, byte ptr color

mov cx, 1

mov al, byte ptr [si]

int 10h

inc si

inc di

inc byte ptr column

cmp byte ptr column, 80

jge \_nextRow

jmp printLoop

\_nextRow:

cmp di, 80

jl waited

inc byte ptr row

mov byte ptr column, 1

cmp byte ptr row, 25

jge endPrint

jmp printLoop

waited:

mov ah, 02h

mov bh, 0

mov dh, byte ptr row

mov dl, 0

int 10h

mov ah, 09h

mov bh, 0

mov bl, byte ptr color

mov cx, 1

mov al, byte ptr [si]

int 10h

inc si

mov di, 0

inc byte ptr row

mov byte ptr column, 1

cmp byte ptr row, 25

jge endPrint

jmp printLoop

endPrint:

mov ah, 01h

int 16h

jnz endRotate

call delay

call delay

call delay

call delay

call delay

call delay

jmp mainLoop

endRotate:

mov ah, 02h

mov bh, 0

mov dh, byte ptr n

mov dl, byte ptr m

int 10h

pop di

ret

\_rotate endp

end

Этот код, написанный на C++ с использованием ассемблера, предназначен для работы в текстовом режиме консоли. Он обрабатывает клавиатурные события и ведет логирование действий пользователя в файл log.txt.

Основная функция запускает бесконечный цикл, ожидая нажатия клавиш. При нажатии F1 выводится список команд, а комбинации с клавишей S выполняют различные действия, такие как генерация звука или изменение цвета текста.

Код использует ассемблерные вставки для низкоуровневой работы с клавиатурой и экраном, что обеспечивает быструю реакцию на действия пользователя. Однако он ориентирован на DOS-подобные системы, что ограничивает его использование на современных ОС.

# Вывод

В ходе лабораторной работы был изучен ассемблер TASM и интеграция его с языком C++ в среде Turbo C++, а также основы написания ассемблерного кода и методы его вызова из C++, что значительно расширяет функциональные возможности разработанной программы.