Федеральное государственное бюджетное образовательное учреждение высшего

образования

«Чувашский государственный университет им. И.Н. Ульянова»

Кафедра вычислительной техники

# **Лабораторная работа 9.**

**ВВОД-ВЫВОД ДАННЫХ В КОНСОЛЬНОМ ПРИЛОЖЕНИИ WIN32**

Вариант 9

Выполнил: Иванов В.С.

студент группы ИВТ-41-22

Проверила:

Доцент Андреева А.А.

Чебоксары, 2024

Цель работы: изучить функции windows API для ввода-вывода в консоль, применение их на практике.

Задание. Клавиатурный сервис. Озвучание клавиатуры (при нажатии клавиш) и индикация на экране NumLock, CapsLock и ScrollLock .  
Описание программы в соответствии с ГОСТ 19.402-78 ЕСПД.

Код представляет собой программу на ассемблере для операционной системы Windows, которая использует консольный интерфейс для отображения состояния клавиш CapsLock, NumLock и ScrollLock, а также проигрывает звуковые сигналы при каждом нажатии на эти клавиши. Программа циклично проверяет состояние этих клавиш и выводит их статус в консоль. Также программа реагирует на нажатие клавиши ESC для выхода из программы.

Функциональное назначение:

Программа отслеживает состояние клавиш CapsLock, NumLock и ScrollLock. Для каждой из этих клавиш программа:

* Проверяет, включена ли клавиша (и отображает это в консоли),
* Проигрывает звуковой сигнал (beep) при каждом нажатии клавиши,
* Позволяет завершить выполнение программы при нажатии клавиши ESC.

Описание логической структуры:

Инициализация:

* Программа начинает работу с инициализации консольного вывода и установки позиции курсора с помощью Windows API.
* На экране выводится строка с индикаторами состояния клавиш (CapsLock, NumLock, ScrollLock).
* Основной цикл работы программы (main\_loop):
* В цикле программа обновляет отображение состояния клавиш:
* Проверяется, включена ли каждая из клавиш (CapsLock, NumLock, ScrollLock) с помощью функции GetKeyState.
* Если клавиша включена, выводится символ \* в соответствующем месте строки.
* Программа проверяет нажатие каждой из клавиш через функцию GetAsyncKeyState.
* При каждом нажатии клавиши проигрывается звуковой сигнал (beep), соответствующий клавише.
* Программа будет продолжать работу до тех пор, пока не нажмут клавишу ESC для завершения.

Завершение работы:

Если клавиша ESC нажата, программа выводит сообщение "ESC pressed. Exiting..." и завершает выполнение.

Используемые технические средства:

Windows API:

* GetStdHandle — для получения дескриптора стандартного вывода (консоль).
* WriteConsoleA — для вывода текста в консоль.
* SetConsoleCursorPosition — для установки позиции курсора.
* GetKeyState — для получения состояния клавиш.
* GetAsyncKeyState — для проверки нажатия клавиш.
* Beep — для проигрывания звуковых сигналов.
* Sleep — для добавления задержки между циклами.
* ExitProcess — для завершения работы программы.

Код написан с использованием синтаксиса TASM для архитектуры x86.

Используется модель памяти flat, что позволяет обращаться к данным как к глобальным переменным.

Для обращения к системным вызовам используется техника вызова через stdcall.

Вызов и загрузка:

Программа вызывается через батник, батник содержит:  
tasm32 /ml 9lab.asm

pause

tlink32 /Tpe /ap /c /x 9lab.obj

pause

9lab.exe

pause

Программа работает в консольном режиме и не требует дополнительных входных данных.

Входные данные:

Программа не требует явных входных данных от пользователя. Она отслеживает состояния клавиш в реальном времени:

* CapsLock
* NumLock
* ScrollLock
* ESC (для завершения программы)

Выходные данные

Консольный вывод:

* В консоли выводится строка с состоянием клавиш (CapsLock, NumLock, ScrollLock).
* Для каждой клавиши, если она включена, на экране появляется символ \*.
* Каждый раз при нажатии клавиши проигрывается звуковой сигнал (beep)
* При нажатии клавиши ESC выводится сообщение о завершении работы программы: "ESC pressed. Exiting...".

Звуковые сигналы (beep):

Каждый звук сопровождается коротким сигналом длительностью 100 миллисекунд.

Примечания:

* Программа использует задержку в 100 миллисекунд в главном цикле для снижения нагрузки на процессор.
* Программа завершает выполнение, когда пользователь нажимает клавишу ESC.

Используемые API-функции:

Каждая программа в Win32 в конце своего выполнения ОБЯЗАТЕЛЬНО должна вызвать функцию ***ExitProcess***. После выполнения последней инструкции программы, дальше идёт пустота, даже если там что-то осмысленное, все равно у выделенной памяти есть конец, а после конца идёт невыделенная (т.е. несуществующая) память. Процессор начнёт выполнять непонятно что, пойдут глюки.

**Параметры**

[in] uExitCode

Код выхода для процесса и всех потоков.

**Возвращаемое значение** Нет

***GetStdHandle*** получает дескриптор буфера вывода консоли.

**Параметры**

*nStdHandle* [в]  
стандартном устройстве. Этот параметр может принимать одно из следующих значений.

| **Значение** | **Значение** |
| --- | --- |
| **STD\_INPUT\_HANDLE STD\_INPUT\_HANDLE** ((DWORD)-10) | Стандартное устройство ввода. Изначально это буфер ввода консоли, CONIN$. |
| **STD\_OUTPUT\_HANDLE - ДЕСКРИПТОР** ((DWORD)-11) | Стандартное устройство вывода. Изначально это активный буфер экрана консоли, CONOUT$. |
| **STD\_ERROR\_HANDLE - ОБРАБОТЧИК STD\_ERROR\_HANDLE** ((DWORD)-12) | Стандартное устройство вывода ошибок. Изначально это активный буфер экрана консоли, CONOUT$. |

**Возвращаемое значение**

Если функция выполнена успешно, возвращаемое значение представляет собой дескриптор указанного устройства или перенаправленный дескриптор, заданный предыдущим вызовом **[SetStdHandle](https://learn.microsoft.com/en-us/windows/console/setstdhandle)**. Дескриптор имеет права доступа **GENERIC\_READ** и **GENERIC\_WRITE**, если только приложение не использовало **SetStdHandle** для установки стандартного дескриптора с меньшими правами доступа

Для вывода текстовой информации используется API-функция ***WriteConsoleA***, параметры которой (слева направо) имеют следующий смысл. 1-й параметр — дескриптор буфера вывода консоли, который может быть получен при помощи функции GetStdHandle; 2-й параметр — указатель на буфер, где находится выводимый текст; 3-й параметр — количество выводимых символов; 192 Часть II. Простые программы, консольные приложения, обработка файлов 4-й параметр указывает на переменную DWORD, куда будет помещено количество действительно выведенных символов; 5-й параметр — резервный параметр, должен быть равен нулю.

**Возвращаемое значение**

Если функция выполняется успешно, возвращается ненулевое значение.

Если функция выполняется неудачно, возвращается нулевое значение.

Прошу читателя взять на вооружение весьма полезную функцию Sleep. Эта функция особенно часто используется именно в потоках, дабы несколько высвободить процессорное время. Единственным параметром функции является минимальное количество миллисекунд, которые поток будет находиться в состоянии простоя.

**Возвращаемое значение**

Нет

Установить позицию курсора в консоли можно при помощи функции ***SetConsoleCursorPosition*** со следующими параметрами: 1-й параметр — дескриптор входного буфера консоли; 2-й параметр — структура COORD: COORD STRUC X WORD ? Y WORD ? COORD ENDS

**Возвращаемое значение**

Если функция выполняется успешно, возвращается ненулевое значение.

Если функция выполняется неудачно, возвращается нулевое значение.

***Beep*** генерирует простые звуковые сигналы на динамике. Функция является синхронной; она выполняет ожидаемое ожидание и не возвращает управление вызывающей стороне до завершения воспроизведения звука.

**Параметры**

[in] dwFreq

Частота звука в герцах. Этот параметр должен находиться в диапазоне от 37 до 32 767 (от 0x25 до 0x7FFF).

[in] dwDuration

Длительность звука в миллисекундах.

**Возвращаемое значение**

Если функция завершается успешно, возвращаемое значение ненулевое.

Если функция не работает, возвращаемое значение равно нулю.

GetKeyState получает статус указанной виртуальной клавиши. Статус определяет, нажата ли клавиша вверх, вниз или переключается (включена, выключена — меняется при каждом нажатии клавиши).

**Параметры**

[in] nVirtKey

Тип: инт

Виртуальная клавиша. Если нужная виртуальная клавиша — это буква или цифра (от A до Z, от a до z или от 0 до 9), nVirtKey должен быть установлен на значение ASCII этого символа. Для других клавиш это должен быть код виртуальной клавиши.

Если используется неанглийская раскладка клавиатуры, для указания большинства клавиш символов используются виртуальные клавиши со значениями в диапазоне от A до Z и от 0 до 9 в кодировке ASCII. Например, для немецкой раскладки клавиатуры виртуальная клавиша со значением ASCII O (0x4F) соответствует клавише «o», а VK\_OEM\_1 соответствует клавише «o с умлаутом».

**Возвращаемое значение**

Тип: КОРОТКИЙ

Возвращаемое значение определяет состояние указанного виртуального ключа следующим образом:

Если старший бит равен 1, то клавиша нажата; в противном случае она не нажата.

Если младший бит равен 1, клавиша переключается. Клавиша, например клавиша CAPS LOCK, переключается, если она включена. Клавиша выключена и не переключается, если младший бит равен 0. Индикатор клавиши-переключателя (если он есть) на клавиатуре будет гореть, когда клавиша переключается, и гаснуть, когда клавиша не переключается.

***GetAsyncKeyState*** определяет, нажата ли клавиша вверх или вниз в момент вызова функции, а также была ли нажата клавиша после предыдущего вызова GetAsyncKeyState.

**Параметры**

[in] vKey

Тип: int

Код виртуального ключа. Дополнительную информацию см. в разделе Коды виртуальных ключей.

Для указания определённых ключей можно использовать константы, различающие левую и правую части. Дополнительную информацию см. в разделе «Примечания».

**Возвращаемое значение**

Тип: КОРОТКИЙ

Если функция выполнена успешно, возвращаемое значение указывает, была ли нажата клавиша с момента последнего вызова GetAsyncKeyState и нажата ли клавиша в данный момент. Если установлен старший бит, клавиша нажата, а если установлен младший бит, клавиша была нажата после предыдущего вызова GetAsyncKeyState. Однако не следует полагаться на это последнее поведение; дополнительные сведения см. в разделе «Примечания».

Возвращаемое значение равно нулю в следующих случаях:

Текущий рабочий стол не является активным рабочим столом

Поток переднего плана принадлежит другому процессу, и рабочий стол не позволяет перехватить или записать в журнал.

Текст программы.

.386

.model flat, stdcall

option casemap:none

include typefile.inc

.data

msgEscPressed db "ESC pressed. Exiting...", 0

buffer db 256 dup(0)

; Строки для вывода

line1 db "CapsLock NumPad ScrollLock", 0

line2 db " ", 0 ; Для вывода состояния клавиш (с \* или без)

; Координаты курсора

cursorPositionX dw 0

cursorPositionY dw 21

; Частоты для Beep

beepFrequency dw 500 ; Частота для Beep (500 Hz)

beepDuration dw 100 ; Длительность для Beep

.code

start:

; Получаем стандартный вывод

push STD\_OUTPUT\_HANDLE

call GetStdHandle

mov esi, eax ; Дескриптор консоли сохраняем в esi

; Отправляем первую строку (индикаторы)

push 0

push 38 ; Длина строки "CapsLock NumPad ScrollLock"

push offset line1

push esi

call WriteConsoleA

; Устанавливаем курсор в начало второй строки

mov ax, cursorPositionX

mov bx, cursorPositionY

push bx

push ax

push esi

call SetConsoleCursorPosition

main\_loop:

; Очищаем строку line2, устанавливая пробелы для каждого индикатора

mov byte ptr [line2+4], 20h ; space для CapsLock

mov byte ptr [line2+17], 20h ; space для NumLock

mov byte ptr [line2+33], 20h ; space для ScrollLock

; Проверка CapsLock

push VK\_CAPITAL

call GetKeyState

test eax, 1

jz skip\_capslock

mov byte ptr [line2+4], '\*' ; Добавляем \* если CapsLock включен

skip\_capslock:

; Проверка NumLock

push VK\_NUMLOCK

call GetKeyState

test eax, 1

jz skip\_numlock

mov byte ptr [line2+17], '\*' ; Добавляем \* если NumLock включен

skip\_numlock:

; Проверка ScrollLock

push VK\_SCROLL

call GetKeyState

test eax, 1

jz skip\_scrolllock

mov byte ptr [line2+33], '\*' ; Добавляем \* если ScrollLock включен

skip\_scrolllock:

; Отправляем обновленную строку

push 0

push 38 ; Длина строки для line2 (50 символов)

push offset line2

push esi

call WriteConsoleA

; Снова устанавливаем курсор в начало второй строки

mov ax, cursorPositionX

mov bx, cursorPositionY

push bx

push ax

push esi

call SetConsoleCursorPosition

; Проверяем, была ли нажата любая клавиша

; Проверяем CapsLock

push VK\_CAPITAL

call GetAsyncKeyState

test eax, 8000h

jz skip\_beep\_capslock

; Проигрываем звук при нажатии CapsLock

push 500 ; Частота для Beep

push 100 ; Длительность для Beep

call Beep

skip\_beep\_capslock:

; Проверяем NumLock

push VK\_NUMLOCK

call GetAsyncKeyState

test eax, 8000h

jz skip\_beep\_numlock

; Проигрываем звук при нажатии NumLock

push 600 ; Частота для Beep

push 100 ; Длительность для Beep

call Beep

skip\_beep\_numlock:

; Проверяем ScrollLock

push VK\_SCROLL

call GetAsyncKeyState

test eax, 8000h

jz skip\_beep\_scrolllock

; Проигрываем звук при нажатии ScrollLock

push 700 ; Частота для Beep

push 100 ; Длительность для Beep

call Beep

skip\_beep\_scrolllock:

; Проверяем ESC для выхода

push VK\_ESCAPE

call GetAsyncKeyState

test eax, 8000h

jnz exit\_program

; Небольшая задержка для снижения нагрузки на процессор

push 100

call Sleep

jmp main\_loop

exit\_program:

push 0

push 27 ; Длина строки msgEscPressed

push offset msgEscPressed

push esi

call WriteConsoleA

push 0

call ExitProcess

end start

Вывод в консоль:



Вывод: изучил функции windows API для ввода-вывода в консоль, применил их на практике.