ЛАБОРАТОРНАЯ РАБОТА 9.
ВВОД-ВЫВОД ДАННЫХ В КОНСОЛЬНОМ ПРИЛОЖЕНИИ WIN32

Основные сведения

Консольное приложение используется, как правило, в случаях, когда графический интерфейс не нужен. Выглядит оно как обычная DOS-программа, но ему доступны все возможности 32-битного режима, включая вызов функций API. Для вывода текстовой информации используется API-функция WriteConsole. Прототип этой функции выглядит следующим образом:

```
BOOL WriteConsole(
          HANDLE Ошибка! Недопустимый объект гиперссылки.,//
дескриптор буфера
// консоли. Можно получить GetStdHandle
          CONST VOID *Ошибка! Недопустимый объект гиперссылки.,//
указатель на
// буфер, где находится выводимый текст
          DWORD Ошибка! Недопустимый объект гиперссылки.,
//количество
// выводимых символов
          LPDWORD Ошибка! Недопустимый объект гиперссылки.,
//сюда будет помещено число реально выведенных
// символов
          LPVOID Ошибка! Недопустимый объект гиперссылки.
// резервный параметр, должен быть 0
             );
```

Для чтения из буфера консоли используется функция `ReadConsole`. Значения параметров этой функции (слева направо) следующие:

1) дескриптор входного буфера;

2) адрес буфера, куда будет помещена вводимая информация;

3) длина этого буфера;

4) количество фактически прочитанных символов;

5) зарезервировано.

Установить позицию курсора в консоли можно при помощи функции SetConsoleCursorPosition со следующими параметрами:

1) дескриптор входного буфера консоли;

2) структура COORD:

```
COORD STRUC
      X WORD ?
      Y WORD ?
COORD ENDS
```

Вторым параметром является не указатель на структуру, а именно структура. Для ассемблера это просто двойное слово (DWORD), у которого младшее слово – координата *X*, а старшее слово – координата *Y*.

Установить цвет выводимых букв можно с помощью функции SetConsoleTextAttribute. Первым параметром этой функции является дескриптор выходного буфера консоли, а вторым – цвет букв и фона. Цвет получается путем комбинации (сумма или операция «ИЛИ») двух или более из представленных ниже констант, причем возможна «смесь» не только цвета и интенсивности, но и цветов:

```
   FOREGROUND_BLUE equ 1h; синий цвет букв;
   FOREGROUND_GREEN equ 2h; зеленый цвет букв;
   FOREGROUND_RED equ 4h; красный цвет букв;
   FOREGROUND_INTENSITY equ 8h; повышенная
интенсивность;
   BACKGROUND_BLUE equ 10h; синий свет фона;
   BACKGROUND_GREEN equ 20h; зеленый цвет фона;
   BACKGROUND_RED equ 40h; красный цвет фона;
   BACKGROUND_INTENSITY equ 80h; повышенная
интенсивность.
```

Для определения заголовка окна консоли используется функция SetConsoleTitle, единственным параметром которой является адрес строки с нулем на конце. Здесь следует оговорить следующее: если для вывода в само окно консоли требовалась DOS-кодировка, то для установки заголовка требуется Windows-кодировка.

Для перекодировки существует специальная функция CharToOem. Первым параметром этой функции является указатель на строку, которую следует перекодировать, вторым

параметром – указатель на строку, куда следует поместить результат. Причем поместить результат можно и в строку, которая перекодируется.

В основе получения информации о клавиатуре и мыши в консольном режиме лежит функция `ReadConsoleInput`. Параметры этой функции:

1) дескриптор входного буфера консоли;

2) указатель на структуру (или массив структур), в которой содержится информация о событиях, происшедших в консоли;

3) количество получаемых информационных записей (структур);

4) указатель на двойное слово, содержащее количество реально полученных записей.

Структура, в которой содержится информация о консольном событии, такова: в начале этого блока данных идет двойное слово, младшее слово которого определяет тип события. В зависимости от значения этого слова последующие байты (максимум 18) будут трактоваться по-разному.

Всего системой зарезервировано пять типов событий:

```
KEY_EVENT equ 1h; клавиатурное событие
MOUSE_EVENT equ 2h; событие с мышью
WINDOW_BUFFER_SIZE_EVENT equ 4h; изменился размер окна
MENU_EVENT equ 8h; зарезервировано
FOCUS_EVENT equ 10h; зарезервировано
```

Значения других байт структуры в зависимости от происшедшего события описаны в следующих таблицах.

Событие `KEY_EVENT`

Смещение	Длина	Значение
+4	4	При нажатии клавиши значение поля больше нуля
+8	2	Количество повторов при удержании клавиши
+10	2	Виртуальный код клавиши
+12	2	Скан-код клавиши
+14	2	Для функции `ReadConsoleInputA`-младший байт равен ASCII-коду клавиши.

		Для функции `ReadConsoleInputW` слово содержит код клавиши в двухбайтной кодировке (Unicode)

+16	4	Содержится состояния управляющих клавиш. Может являться суммой следующих констант:
		`RIGHT_ALT_PRESSED equ 1h`
		`LEFT_ALT_PRESSED equ 2h`
		`RIGHT_CTRL_PRESSED equ 4h`
		`LEFT_CTRL_PRESSED equ 8h`
		`SHIFT_PRESSED equ 10h`
		`NUMLOCK_ON equ 20h`
		`SCROLLLOCK_ON equ 40h`
		`CAPSLOCK_ON equ 80h`
		`ENHANCED_KEY equ 100h`

Событие `MOUSE_EVENT`

Смещение	Длина	Значение
+4	4	Младшее слово - *X*-координата курсора мыши, старшее слово - *Y*-координата мыши
+8	4	Описывает состояние кнопок мыши. Первый бит - левая кнопка, второй бит - правая кнопка, третий бит - средняя кнопка. Бит установлен - кнопка нажата
+12	4	Состояние управляющих клавиш. Аналогично предыдущей таблице
+16	4	Может содержать следующие значения: `MOUSE_MOV equ 1h`; было движение мыши; `DOUBLE_CL equ 2h`; был двойной щелчок

Событие `WINDOW_BUFFER_SIZE_EVENT`

По смещению +4 находится двойное слово, содержащее новый размер консольного окна. Младшее слово – это размер по *X*, старшее слово – размер по *Y*. Следует иметь в виду, что когда речь идет о консольном окне, все размеры и координаты даются в «символьных» единицах.

Пример простой консольной программы:

console.inc:

```
includelib      import32.lib
; имена используемых функций из kernel32.dll
                extrn   ExitProcess:near
                extrn   WriteConsoleA:near
                extrn   GetStdHandle:near
; присваивания для облегчения читаемости кода
        WriteConsole    equ     WriteConsoleA
; имена используемых функций из mpr.dll
                extrn   WNetGetUserA:near
                extrn   WNetOpenEnumA:near
                extrn   WNetEnumResourceA:near
                extrn   WNetCloseEnum:near
; присваивания для облегчения читаемости кода
        WNetGetUser     equ     WNetGetUserA
        WNetOpenEnum    equ     WNetOpenEnumA
        WNetEnumResource equ    WNetEnumResourceA
; определения констант и типов
NO_ERROR                        equ     0
ERROR_NO_MORE_ITEMS             equ     259
RESOURCEUSAGE_CONNECTABLE       equ     1
RESOURCETYPE_ANY                equ     0
RESOURCE_CONNECTED              equ     1
STD_OUTPUT_HANDLE               equ     -11
NTRESOURCE struc
        dwScope         dd      ?
        dwType          dd      ?
        dwDisplayType   dd      ?
        dwUsage         dd      ?
        lpLocalName     dd      ?
        lpRemoteName    dd      ?
        lpComment       dd      ?
        lpProvider      dd      ?
NTRESOURCE ends
```

console.asm:

```
;Консольное приложение для Win32, перечисляющее
;сетевые ресурсы
include console.inc
.386
.model FLAT,STDCALL
```

```
.const
greet_message   db 'Example Win32 console program'
                db 0Dh, 0Ah,0Dh,0Ah,0
error1_message  db 0Dh,0Ah,'Couldnot get current
user name',
                                db 0Dh,0Ah,0
error2_message  db 0Dh,0Ah,'Could not enumerate',\
 0Dh,0Ah,0
good_exit_msg db 0Dh,0Ah,0Dh,0Ah,'Normal \
termination',
                           db 0Dh,0Ah,0
enum_msg1       db      0Dh,0Ah,'Local ',0
enum_msg2       db      ' remote - ',0
.data
user_name db 'List of connected resources for user '
user_buff  db  64 dup (?); буфер для WNetGetUser
user_buff_l dd $-user_buff;размер буфера
;для WNetGetUser
enum_buf_l  dd 1056; длина enum_buf в байтах
enum_entries dd 1 ; число ресурсов, которые
;в нём помещаются
.data?
enum_buf NTRESOURCE <?,?,?,?,?,?,?,?>; буфер
;для WNetEnumResource
         dd 256 dup (?); 1024 байт для строк
message_l dd ?     ; переменная для WriteConsole
enum_handle dd ?; идентификатор для
;WNetEnumResource
.code
_start:
; получим от системы идентификатор буфера вывода
;stdout
      push STD_OUTPUT_HANDLE
      call GetStdHandle
;возвращает идентификатор STDOUT в eax
   mov  ebx,eax ; а мы будем хранить его в EBX
; выведем строку greet_message на экран
      mov esi,offset greet_message
      call output_string
; определим имя пользователя, которому принадлежит
;наш процесс
      mov esi,offset user_buff
```

```
      push offset user_buff_l; адрес переменной
; с длиной буфера
      push esi                ; адрес буфера
      push 0                  ; NULL
      call WNetGetUser
      cmp  eax,NO_ERROR; если произошла ошибка
      jne  error_exit1; выйти
      mov esi,offset user_name; иначе - выведем
;строку на экран
      call    output_string
; начнём перечисление сетевых ресурсов
      push offset enum_handle; идентификатор
;для WNetEnumResource
      push    0
      push    RESOURCEUSAGE_CONNECTABLE
; все присоединяемые ресурсы
      push    RESOURCETYPE_ANY  ресурсы любого
;типа
      push    RESOURCE_CONNECTED; только
;присоединённые сейчас
      call    WNetOpenEnum; начать перечисление
      cmp eax,NO_ERROR; если произошла ошибка
      jne     error_exit2    ; выйти
enumeration_loop:       ; цикл перечисления
ресурсов
      push offset enum_buf_l ; длина буфера в
байтах
      push offset enum_buf         ; адрес буфера
      push offset enum_entries     ; число ресурсов
      push dword ptr enum_handle   ; идентификатор
;от WNetOpenEnum
      call    WNetEnumResource
     cmp eax,ERROR_NO_MORE_ITEMS; если
они ;закончились
      je end_enumeration; завершить перечисление
      cmp eax,NO_ERROR; если произошла ошибка
      jne error_exit2; выйти с сообщением об ошибке
; вывод информации ресурсе на экран
      mov esi,offset enum_msg1; первая часть строки
      call output_string           ; на консоль
      mov esi,dword ptr enum_buf.lpLocalName ;
; локальное имя устройства
```

```
      call    output_string   ; на консоль
      mov esi,offset enum_msg2; вторая часть строки
      call    output_string   ; на консоль
      mov esi,dword ptr enum_buf.lpRemoteName
; удалённое имя устройства
      call output_string    ; туда же
  jmp short enumeration_loop;продолжим перечисление
end_enumeration:
      push dword ptr enum_handle
      call WNetCloseEnum; конец перечисления
      mov esi,offset good_exit_msg
exit_program:
      call output_string; выведем строку
      push 0            ; код выхода
      call    ExitProcess; конец программы
; выходы после ошибок
error_exit1:
      mov esi,offset error1_message
      jmp short exit_program
error_exit2:
      mov esi,offset error2_message
      jmp short exit_program

; процедрура output_string выводит на экран строку
; ввод: esi - адрес строки, ebx – идентификатор
;stdout или другого консольного буфера
output_string proc near
      cld               ; определим длину строки
      xor eax,eax
      mov edi,esi
      repne scasb
      dec edi
      sub edi,esi
      push 0                ; пошлём её на консоль
 push offset message_l;сколько байт выведено
; на консоль
     push edi; сколько байт надо вывести на консоль
      push esi; адрес строки для вывода на консоль
      push ebx     ; идентификатор буфера вывода
      call    WriteConsole
; WriteConsole(hConsoleOutput, lpvBuffer,
; cchToWrite, ;lpcchWritten, lpvReserved)
```

```
      ret
output_string endp
      end _start
```

>tasm32/ml console.asm -> console.obj (976b)
>tlink32/Tpe/ap/c/x console.obj -> console.exe (4096b)

Следует обратить внимание на ключ /ap – именно он указывает на то, что приложение будет консольным.

Результат:

```
D:\PC>console
Example Win32 console program

List of connected resources for user

Normal termination
D:\PC>
```

Задание к лабораторной работе

Написать программу для варианта задания, соответствующего порядковому номеру студента в группе.

1. Вывод набранного в консоли текста в файл. Имя файла указывается в командной строке. Признаком конца ввода принять символ с кодом 26 (1Ah), соответствующий комбинации клавиш ctrl-Z.
2. Копирование набранной в консоли строки в буфер обмена Windows.
3. Вывод на консоль информации о системе, наподобие результата выполнения SystemInfo.exe.
4. Вывод на консоль указанного в командной строке текстового файла в кодировке Windows.
5. Отслеживание нажатия клавиши. Вывод в центре экрана консоли скан-кода и количества повторов при удержании. Выход по Esc.
6. Вывод в центре экрана консоли списка нажатых в текущий момент управляющих клавиш (L/R-Alt, L/R-Control, Shift, CapsLock, NumLock, ScrollLock).
7. Вывод в центре экрана консоли координат курсора и состояния клавиш мыши.

8. Вывод на консоль звездочки через заданное в командной строке число секунд. Выход при нажатии любой клавиши.

9. Клавиатурный сервис. Озвучание клавиатуры (при нажатии клавиш) и индикация на экране NumLock, CapsLock и ScrollLock .

10. Управление клавиатурой. Установка задержки клавиатуры (перед первым повторением и между повторениями символа).

Порядок выполнения работы

1. Изучить основные сведения.
2. Изучить необходимые для выполнения задания функции Win API.
3. Разработать алгоритм и программу решения задачи на языке ассемблера, подготовить тестовые примеры.
4. Выполнить ввод, трансляцию, построение кода программы и получить результаты ее работы.

Содержание отчета о работе

1. Цель работы.
2. Текст задания.
3. Описание программы в соответствии с ГОСТ 19.402-78 ЕСПД.
4. Описание используемых API-функций.
5. Текст программы.
6. Выводы по работе.

Список рекомендуемой литературы

1. Андреева А.А. и др. Программирование на языке ассемблера в операционной системе Windows: лаб. практикум. Чебоксары: Изд-во Чуваш. ун-та, 2006. 104 с.

2. Зубков С.В. Ассемблер для DOS, Windows и UNIX. М.: ДМК Пресс, 2015. 638 с.

3. Пирогов В.Ю. Ассемблер для Windows. СПб.: БХВ-Петербург, 2007. 896 с.

4. Юров В. И. Assembler: практикум. СПб.: Питер, 2007. 400 с.

5. Аблязов Р. Программирование на ассемблере на платформе x86-64. М.: ДМК Пресс, 2016. 302 с.

Приложение. ГОСТ 19.402-78 Единая система программной документации (ЕСПД). Описание программы

Составление информационной части (аннотации и содержания) является обязательным.

Описание программы должно содержать следующие разделы:

общие сведения;

функциональное назначение;

описание логической структуры;

используемые технические средства;

вызов и загрузка;

входные данные;

выходные данные.

В зависимости от особенностей программы допускается вводить дополнительные разделы или объединять отдельные разделы.

В разделе "Общие сведения" должны быть указаны:

обозначение и наименование программы;

программное обеспечение, необходимое для функционирования программы;

языки программирования, на которых написана программа.

В разделе "Функциональное назначение" должны быть указаны классы решаемых задач и (или) назначение программы и сведения о функциональных ограничениях на применение.

В разделе "Описание логической структуры" должны быть указаны:

алгоритм программы;

используемые методы;

структура программы с описанием функций составных частей и связи между ними;

связи программы с другими программами.

Описание логической структуры программы выполняют с учетом текста программы на исходном языке.

В разделе "Используемые технические средства" должны быть указаны типы электронных вычислительных машин и устройств, которые используются при работе программы.

В разделе "Вызов и загрузка" должны быть указаны:

способ вызова программы с соответствующего носителя данных;

входные точки в программу

Допускается указывать адреса загрузки, сведения об использовании оперативной памяти, объем программы.

В разделе "Входные данные" должны быть указаны:

характер, организация и предварительная подготовка входных данных;

формат, описание и способ кодирования входных данных.

В разделе "Выходные данные" должны быть указаны: характер и организация выходных данных

формат, описание и способ кодирования выходных данных.

Допускается содержание разделов иллюстрировать пояснительными примерами, таблицами, схемами, графиками.

В приложение к описанию программы допускается включать различные материалы, которые нецелесообразно включать в разделы описания