Лабораторная работа №1
Программное извлечение USB-накопителя

1. Цель работы

Практическое овладение навыками составления программ, работающих с USB-накопителями информации.

2. Теоретический материал

При отключении USB-накопителей информации от компьютера производителями таких устройств рекомендуется предварительно выполнять программное «безопасное извлечение устройства». Данная процедура информирует операционную систему о скором прекращении работы устройства, чтобы она смогла своевременно завершить все операции с устройством. В случае работы каких-либо процессов с содержимом разделов накопителя, операционная система информирует пользователя об этом и прекращает процесс извлечения устройства.

В операционных системах семейства Windows для этого имеется функция «Безопасное извлечение устройства», которую можно вызвать командой

rundll32.exe shell32.dll, Control_RunDLL hotplug.dll

Однако данную операцию можно выполнить и своей программой. Существует несколько способов программного извлечения устройства, например, используя:

- функции библиотеки SetupAPI (CM_Request_Device_Eject);
- функции Win API (CreateFile, DeviceIOControl);
- функции прямого обращения к драйверу.

В зависимости от выбранного способа и реализации при извлечении устройства операционная система информируется об этом или нет, что может привести к извлечению используемого в этот момент устройства.

Далее приведён листинг программы на языке Ассемблера, которая позволяет безопасно извлекать из системы первый попавшийся USB-накопитель. Для поиска USB-накопителя в списке всех существующих в системе накопителей информации проверяется PnP-идентификатор экземпляра устройства (см. далее). Извлечение осуществляется по дескриптору физического устройства, потомком которого является накопитель. Блок-схема алгоритма программы представлена на рисунках 1.1. и 1.2.

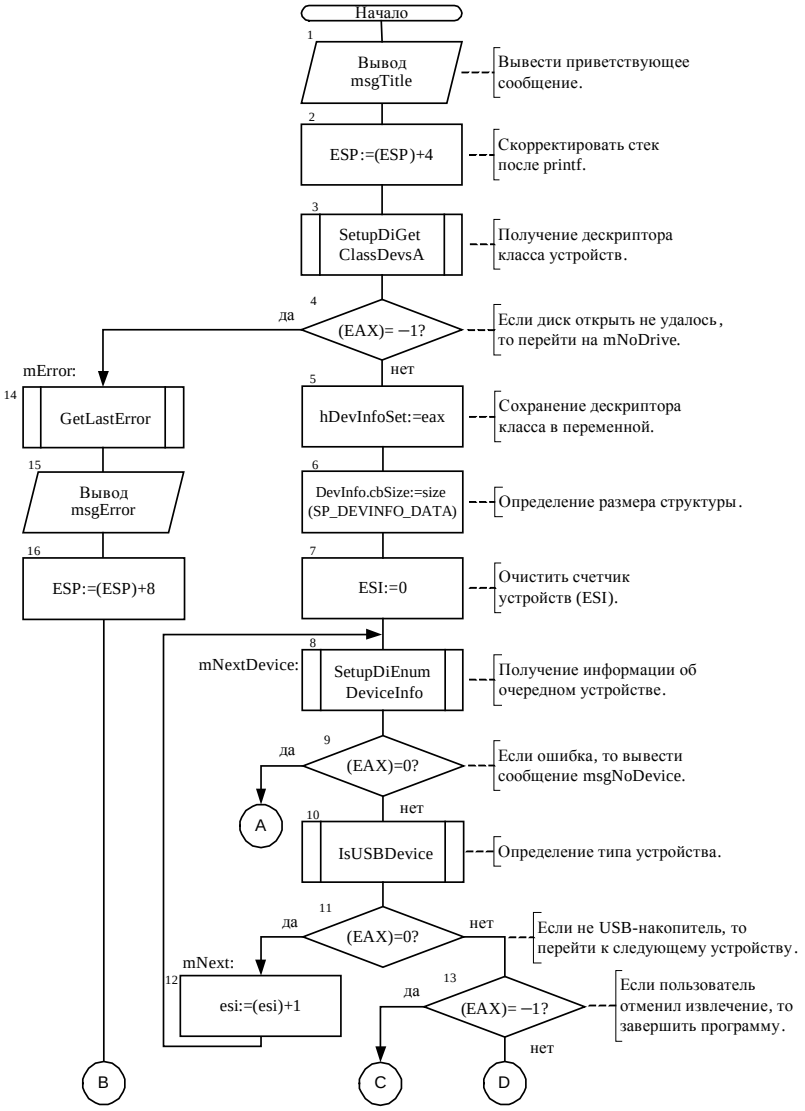

Рис. 1.1 – Алгоритм безопасного извлечения USB-накопителя

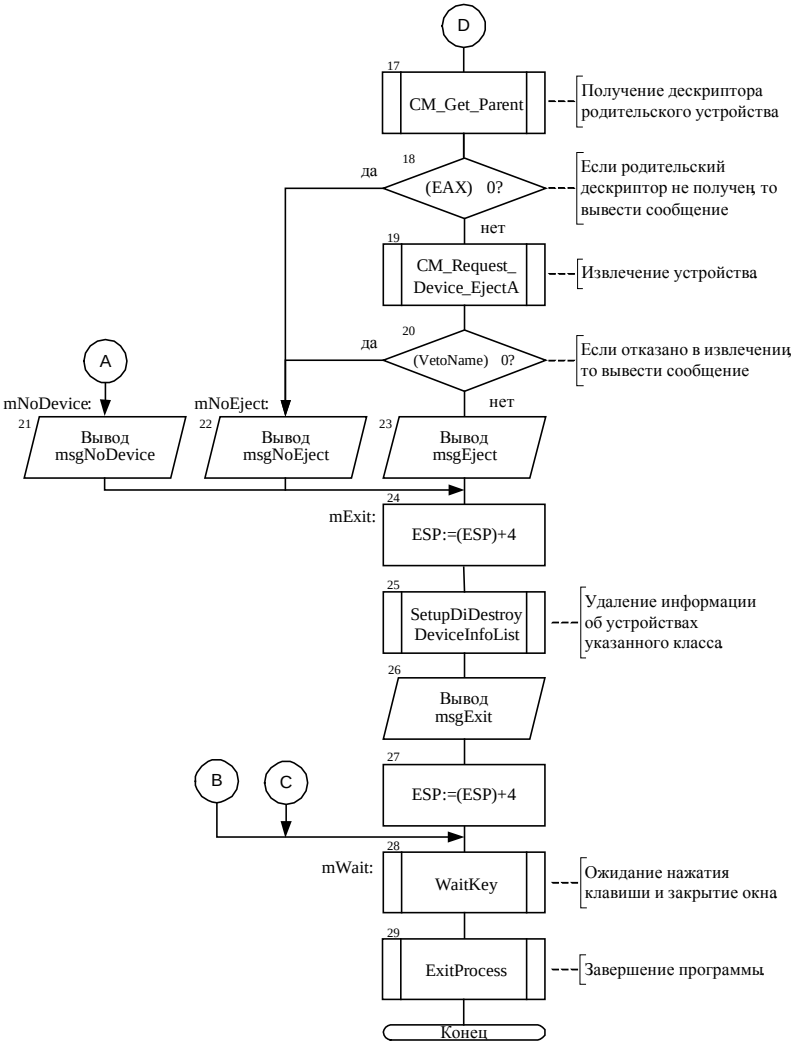

Рис. 1.2 – Продолжение алгоритма безопасного извлечения USB-накопителя

2.1. Библиотека SetupAPI.dll

SetupAPI это системный компонент, содержащий функции для установки драйвера устройства, и связывающий пользовательское приложение с устройством.

В дальнейшем рассмотрим некоторые функции из двух групп SetupAPI: *Device Installation Functions* и *PnP Configuration Manager Functions*. Первая группа включает функции установки устройств в Microsoft Windows, вторая – дополняет её функциями конфигурирования устройств (CM_xxx).

Функция **SetupDiGetClassDevs** возвращает дескриптор (манипулятор или handle) класса устройства, заданного в качестве параметра. Функция имеет следующий формат:

SetupDiGetClassDevs, *ClassGuid, Enumerator, hwndParent, Flags,*

где *ClassGuid* – глобальный уникальный идентификатор класса устройств (для жестких дисков и USB-накопителей ClassGuid = {4d36e967-e325-11ce-bfc1-08002be10318});
Enumerator – системный компонент, определяющий PnP-идентификатор устройства;
hwndParent – дескриптор родительского окна;
Flags – флаг управления функцией. Может принимать пять значений, мы будем использовать флаг по умолчанию DIGCF_DEFAULT = 2.

При успешном выполнении функция возвращает в регистр EAX дескриптор класса устройств, в противном случае вернёт -1 (INVALID_HANDLE_VALUE).

Функция **SetupDiEnumDeviceInfo** возвращает структуру с информацией об очередном устройстве указанного класса. Если функция вернула в EAX значение 1 (TRUE), то информация извлечена успешно, а если 0 (FALSE), то, в большинстве случаев, устройства в списке закончились. Функция имеет следующий формат:

SetupDiEnumDeviceInfo, *hDeviceInfoSet, MemberIndex, DeviceInfoData,*

где *hDeviceInfoSet* – дескриптор класса устройств;
MemberIndex – порядковый номер в списке устройств указанного класса;
DeviceInfoData – возвращаемая структура с информацией об устройстве, состав которой показана в таблице 1.1

Функция **SetupDiGetDeviceRegistryProperty** позволяет получить PnP свойства устройства.

SetupDiGetDeviceRegistryProperty, *hDeviceInfoSet, DeviceInfoData, Property, PropertyRegDataType, PropertyBuffer, PropertyBufferSize, RequiredSize,*

Таблица 1.1

Структура с информацией об устройстве

Имя структуры записи	Элемент структуры	Размер элемента, байт	Описание
SP_DEVINFO _DATA	cbSize	4	размер структуры в байтах
	ClassGuid	16	GUID класса устройств
	DevInst	4	Дескриптор устройства
	Reserved	4	Зарезервировано

где *hDeviceInfoSet* – дескриптор класса устройств;
DeviceInfoData – указатель на структуру с информацией об устройстве;
Property – параметр, указывающий какое именно свойство требуется получить. Для получения строки с описанием устройства необходимо указать константу SPDRP_DEVICEDESC (00000000h) или SPDRP_FRIENDLYNAME (0000000Ch). Для получения идентификатора оборудования (HardwareID) необходимо указать константу SPDRP_HARDWAREID (00000001h);
PropertyRegDataType – указатель на переменную, в которую помещается тип возвращаемых функцией данных;
PropertyBuffer – указатель на буфер, в который возвращается значение указанного свойства. Если этот параметр указа как null (0) и PropertyBufferSize указан как 0, то функция возвращает в RequiredSize необходимый размер буфера;
PropertyBufferSize – размер буфера для получения значения свойства;
RequiredSize – дополнительный параметр для получения размера буфера, если не используется, то null (0).

Функция **SetupDiDestroyDeviceInfoList** удаляет всю информацию об устройствах указанного класса, и очищаем память. В качестве параметра этой функции указывается дескриптор класса устройств *(hDeviceInfoSet)*, который предварительно был получен функцией SetupDiGetClassDevs.

Функция **CM_Get_Device_ID_Size** возвращает размер строки идентификатора устройства.

CM_Get_Device_ID_Size, *pulLen, dnDevInst, ulFlags*,

где *pulLen* – указатель на переменную для записи длины строки;
dnDevInst – дескриптор устройства;
ulFlags – не используется, должен быть нулем.

Функция **CM_Get_Device_ID** возвращает текстовый идентификатор экземпляра устройства ID.

CM_Get_Device_ID, *dnDevInst, Buffer, BufferLen, ulFlags,*

где *dnDevInst* – дескриптор устройства;
Buffer – указатель на буфер для записи строки идентификатора устройства;
BufferLen – длина строки идентификатора устройства;
ulFlags – не используется, должен быть нулем.

Функция **CM_Get_Parent** получает дескриптор родительской ветки в дереве устройств локальной машины.

CM_Get_Parent, *pdnDevInst, dnDevInst, ulFlags,*

где *pdnDevInst* – возвращаемый указатель на дескриптор родительского устройства;
dnDevInst – дескриптор устройства;
ulFlags – не используется, должен быть нулём.

Функция **CM_Request_Device_Eject** выполняет безопасное извлечение устройства, а если это невозможно, то возвращает информацию об ошибке.

CM_Request_Device_Eject, *dnDevInst, pVetoType, pszVetoName, ulNameLength, ulFlags,*

где *dnDevInst* – дескриптор устройства;
pVetoType – указатель на переменную, которая будет содержать код ошибки в случае отказа в извлечении устройства;
pszVetoName – указатель на буфер для возвращения текстового описания ошибки, в случае отказа в извлечении устройства;
ulNameLength – максимальная длина текстового описания ошибки;
ulFlags – не используется, должен быть нулем.

Функция **CM_Locate_DevNode** позволяет получить дескриптор устройства по строке идентификатора.

CM_Locate_DevNode, *pdnDevInst, pDeviceID, ulFlags,*

где *pdnDevInst* – указатель на возвращаемый дескриптор устройства;
pDeviceID – указатель на строку идентификатора устройства;
ulFlags – флаг управления функцией. Может принимать четыре значения, мы будем использовать CM_LOCATE_DEVNODE_NORMAL = 0.

Функция **CM_Get_DevNode_Status** позволяет получить статус устройства, по которому можно определить, можно ли извлечь данное устройство. Если в статусе (pulStatus) возвращается флаг DN_REMOVABLE (4000h), то устройство можно извлечь.

CM_Get_DevNode_Status, *pulStatus, pulProblemNumber, dnDevInst, ulFlags,*

где *pulStatus* – указатель на переменную со статусом устройства;
pulProblemNumber – указатель на переменную с номером ошибки;
dnDevInst – дескриптор устройства, у которого необходимо проверить статус;
ulFlags – не используется, должен быть нулем.

В отличии от функций устнановки (SetupXXX) функции конфигурирования устройств (CM_xxx) при успешном выполнении возвращают в регистр EAX значение CR_SUCCESS (EAX=0).

2.2. PnP-идентификаторы USB-накопителей

Каждое USB-устройство, спроектированное по спецификации PnP, должно иметь идентификатор, который однозначно определяет модель данного устройства.

Идентификатор устройства должен иметь строго определённый для данного класса устройств формат. Для USB-устройства идентификатор имеет следующий формат:

USB\Vid_vvvv&Pid_dddd&Rev_rrrr,

где *vvvv* – код идентификатора производителя, зарегистрированного в Комитете USB-производителей;
dddd – идентификатор, присвоенный производителем данный модели USB-устройства;
rrrr – номер версии разработки.

Все эти поля вводятся как шестнадцатеричные числа.

При идентификации USB-накопителя драйвером порта, последний создаёт новый *идентификатор устройства* (device ID) и набор *идентификаторов оборудования* (hardware ID). Первая строка идентификатора оборудования должна совпадать с идентификатором устройства. Идентификатор оборудования имеет следующий формат

USBSTOR\t*v(8)p(16)r(4)
USBSTOR\t*v(8)p(16)
USBSTOR\t*v(8)
USBSTOR\v(8)p(16)r(1)
v(8)p(16)r(1)
USBSTOR\GenDisk
GenDisk,

где *t** – тип устройства (для USB-накопителей DISK);
v(8) – идентификатор производителя из 8 символов;
p(16) – идентификатор продукта;
r(4) – идентификатор версии разработки.

Идентификатор оборудования необходим для точного подбора драйвера для устройства.

Кроме идентификатора устройства (device ID) каждый накопитель имеет *идентификатор экземпляра* (instance ID), который отличает устройство от других устройств того же типа. Идентификатор экземпляра может определять устройство относительно шины (например, учитывать USB-порт, к которому подключено устройство) или представлять глобально уникальный дескриптор (например, серийный номер устройства). Идентификаторы устройства и экземпляра дают *идентификатор экземпляра устройства* (device instance ID, DIID или код экземпляра устройства), который однозначно идентифицирует экземпляр устройства в системе.

Просмотреть код экземпляра устройства можно в диспетчере устройств, выбрав интересующее дисковое устройство и перейдя на вкладку «Сведений» в свойствах этого устройства.

Например: код экземпляра устройства **USBSTOR\DISK&VEN_JETFLASH&PROD_TS256MJF2A/120&REV_8.07\6&38D7AE47&0&7ZNDZ4S6&0** содержит идентификатор устройства **USBSTOR\DISK&VEN_JETFLASH&PROD_TS256MJF2A/120&REV_8.07** и идентификатор экземпляра **6&38D7AE47&0&7ZNDZ4S6&0**. Идентификатор оборудования для того же устройства содержит:

USBSTOR\DiskJetFlashTS256MJF2A/120__8.07
USBSTOR\DiskJetFlashTS256MJF2A/120__
USBSTOR\DiskJetFlash
USBSTOR\JetFlashTS256MJF2A/120__8
JetFlashTS256MJF2A/120__8
USBSTOR\GenDisk
GenDisk

3. Подготовка к работе

3.1. Изучить методические указания и рекомендованную литературу.
3.2. Подготовить ответы на контрольные вопросы.
3.3. Подготовить флеш-диск с интерфейсом USB.

4. Задание на выполнение работы

4.1. Создать и отладить исполняемый модуль программы **EjectFlesh**, исходный код которой приведён ниже, предназначенной для безопасного извлечения первого попавшегося USB-накопителя.
4.2. Отредактировать исходный модуль программы EjectFlesh таким образом, чтобы безопасно извлекался только Ваш USB-накопитель.

Если в системе отсутствует Ваш накопитель или имеется другой программа должна корректно завершить свою работу не извлекая устройства.

```
 TITLE EjectFlesh
;32-х разрядное приложение для безопасного извлечения первого
; попавшегося USB-накопителя.

.386
; Плоская модель памяти и стандартная модель вызова подпрограмм.
.MODEL FLAT, STDCALL

; Директивы компоновщику для подключения библиотек.
INCLUDELIB import32.lib          ; Работа с библиотекой ОС Kernel32.
INCLUDELIB SetupAPI.lib          ; Работа с библиотекой ОС SetupAPI.

;--- Внешние WinAPI-функций ---------------------------------------------------
EXTRN FreeConsole:        PROC   ; Освободить текущую консоль
EXTRN AllocConsole:       PROC   ; Создать свою консоль
EXTRN GetStdHandle:       PROC   ; Получить дескриптор текущей консоли
EXTRN printf:             PROC   ; Вывести по шаблону
EXTRN RtlZeroMemory:      PROC   ; Очистить память
EXTRN ReadConsoleInputA: PROC    ; Получить события консоли
EXTRN GetLastError:       PROC   ; Получить код ошибки
EXTRN CloseHandle:        PROC   ; Закрыть дескриптор
EXTRN ExitProcess:        PROC   ; Завершить процесс

;--- Внешние SetupAPI-функции -------------------------------------------------
; Получить дескриптор класса устройств
EXTRN SetupDiGetClassDevsA:           PROC
; Получить информацию об устройстве
EXTRN SetupDiEnumDeviceInfo:          PROC
; Удалить информацию об устройствах указанного класса
EXTRN SetupDiDestroyDeviceInfoList: PROC
; Получить размер строки идентификатора устройства
EXTRN CM_Get_Device_ID_Size:          PROC
; Получить текстовый идентификатор экземпляра устройства
EXTRN CM_Get_Device_IDA:              PROC
; Получить дескриптор родительской ветки в дереве устройств
EXTRN CM_Get_Parent:                  PROC
; Извлеч устройство
EXTRN CM_Request_Device_EjectA:       PROC
```

```
;--- Константы ----------------------------------------------------------------
STD_INPUT_HANDLE    = -10  ; консольное окно для ввода данных
Key_Event           = 1    ; тип клавиатурного события
CR_SUCCESS          = 0    ; успешное выполнение функций CM_xxx

;--- Структуры данных ---------------------------------------------------------
; Структура с информацией об устройстве
SP_DEVINFO_DATA STRUC
  cbSize      dd ?          ; Размер структуры в байтах
  ClassGuid   dd 4 dup (?)  ; GUID класса устройств
  DevInst     dd ?          ; Дескриптор устройства
  Reserved    dd ?          ; Зарезервировано
SP_DEVINFO_DATA ENDS

; Структура с информацией о событиях от клавиатуры
KeyEvent STRUC
  KeyDown          dd ?     ; Признак нажатия/отпускания клавиши
  RepeatCount      dw ?     ; Кол-во повторов при удержании клавиши
  VirtualKeyCode   dw ?     ; Виртуальный код клавиши
  VirtualScanCode  dw ?     ; Скан-код клавиши
  ASCIIChar        dw ?     ; ASCII-код клавиши
  ControlKeyState  dd ?     ; Состояние управляющих клавиш
KeyEvent ENDS

; Структура буфера событий консоли
InputRecord  STRUC
    EventType              dw ?          ; Тип события
                           dw 0          ; для выравнивания поля
    KeyEventRecord     KeyEvent <>
    MouseEventRecord       dd 4 dup (?)
    WindowsBufferSizeRecord dd ?
    MenuEventRecord        dd ?
    FocusEventRecord       dd ?
InputRecord  ENDS

;--- Данные -------------------------------------------------------------------
.DATA
; Выводимые сообщения
  msgTitle     db ' Безопасное извлечение USB-накопителя', 13,10,13,10, 0
  msgError     db 13,10, 'Ошибка: %u', 13,10,0
  msgQuery     db 'Извлечь USB-накопитель? '
```

```
            db '(y - да; любая другая клавиша - нет): ', 0
  msgKey    db '%c', 13,10,0
  msgNoDevicedb 'USB-накопители не найдены.', 13,10,0
  msgNoEject db 'Устройство сейчас используется. '
            db 'В извлечении отказано.', 13,10,0
  msgCancel db 'Пользователь отменил операцию. '
            db 'Для выхода нажмите любую клавишу...', 13,10,0
  msgEject  db 'Устройство безопасно извлечено из системы.', 13,10,0
  msgExit   db 'Для выхода нажмите любую клавишу...', 0

; GUID класса накопителей
GUID_DEVCLASS_DISKDRIVE dd 4D36E967h
                        dw 0E325h
                        dw 11CEh
                        db 0BFh, 0C1h, 8, 0, 2Bh, 0E1h, 3, 18h

; Переменные
  hIn           dd ?            ; Дескриптор окна ввода данных
  NumEvent      dd ?            ; Число событий
  Key           dd ?            ; Код нажатой клавиши
  bReturned     dd ?            ; Число байт, возвращённых функцией
  hDevInfoSet   dd ?            ; Дескриптор класса устройств
  hParent       dd ?            ; Дескриптор родительского устройства
  IDLen         dd ?            ; Длинна текстового идентификатора
  ID            db 200 dup (?)  ; Буфер для идентификатора устройства
  NameStore     db 'USBSTOR'
  VetoName      db 260 dup (?)  ; Текстовый буфер для описания отказа

; Указатели на структуры
  CBuffer       InputRecord <>              ; Буфер консоли
  DevInfo       SP_DEVINFO_DATA <>          ; Информация об устройстве

.CODE
;--- Проверка типа (USB-накопителем) устройства -------------------------
; hDevice – дескиптор устройства
; Возвращает EAX =  1 – успешное завершение, -1 – отмена пользователем,
                    ;0 – не является USB-накопителем
IsUSBDevice PROC
        ARG   hDevice: DWORD
        uses  esi, edi, ecx
```

```
; Получение размера строки идентификатора
        call    CM_Get_Device_ID_Size, offset IDLen, hDevice, 0
        cmp     eax, CR_SUCCESS  ; Проверка успешного выполнения команды
        jnz     mNoUSBDevice     ; В случае ошибки выход из процедуры
        cmp     IDLen, 0
        jz      mNoUSBDevice
        inc     IDLen

; Очистка буфера под идентификатор и его получение
        call    RtlZeroMemory, offset ID, IDLen
        call    CM_Get_Device_IDA, hDevice, offset ID, IDLen, 0
        cmp     eax, CR_SUCCESS
        jnz     mNoUSBDevice

        cld                      ; Сравнение вперед
        push    ds
        pop     es
        lea     esi, ID
        lea     edi, NameStore
        mov     ecx, 7
        repe    cmpsb
        jne     mNoUSBDevice
        call    printf, offset msgQuery
        add     esp, 4*1

        call    WaitKey
        call    printf, offset msgKey, Key
        add     esp, 4*2

; Определение нажатия клавиши "y"
        cmp     Key, 121
        jne     mEjectCancel
        mov     eax, 1
        jmp     IUDExit

mEjectCancel:
        call    printf, offset msgCancel
        add     esp, 4*1
        mov     eax, -1
        jmp     IUDExit
```

```
mNoUSBDevice:
        mov     eax, 0

IUDExit:
        ret
IsUSBDevice ENDP

;--- Ожидание нажатия клавиши ----------------------------------------------
; Возвращает в Key – код нажатой клавиши
WaitKey PROC

mWaitKey:
        call    ReadConsoleInputA, hIn, offset CBuffer, 1, offset NumEvent
        cmp     CBuffer.EventType, Key_Event
        jne     mWaitKey
        cmp     CBuffer.KeyEventRecord.KeyDown, 0
        jz      mWaitKey
        mov     ax, CBuffer.KeyEventRecord.ASCIIChar
        mov     Key, dword ptr eax
        ret
WaitKey ENDP

;--- Основной код ----------------------------------------------------------
Start:
        call    FreeConsole
        call    AllocConsole
        call    GetStdHandle, STD_INPUT_HANDLE
        mov     hIn, eax
        call    printf, offset msgTitle
        add     esp, 4*1                ; Корректировка стека после printf

; Получение дескриптора класса устройств
        call    SetupDiGetClassDevsA,                                   offset
GUID_DEVCLASS_DISKDRIVE, 0, 0, 2
        test    eax, eax
        js      mError
; Сохранение дескриптора клааса устройств в переменной
        mov     hDevInfoSet, eax
; Определение размера структуры
        mov     DevInfo.cbSize, size SP_DEVINFO_DATA
```

```
; Цикл по всем устройствам
        mov     esi, 0                  ; Счётчик устройст
mNextDevice:
        call    SetupDiEnumDeviceInfo, hDevInfoSet, esi, offset DevInfo
        test    eax, eax
        jz      mNoDevice

        call    IsUSBDevice, DevInfo.DevInst
        test    eax, eax
        jz      mNext       ; Если eax=0 переход к следующему устройству
        js      mWait       ; Если eax=-1 завершение работы программы

; Получение дескриптора родительского устройства
        call    CM_Get_Parent, offset hParent, DevInfo.DevInst, 0
        cmp     eax, CR_SUCCESS
        jnz     mNoEject

; Извлечение устройства
        call    CM_Request_Device_EjectA, hParent, 0, offset VetoName,
size VetoName, 0
        cmp     VetoName, 0         ; Проверка отказа в извлечении
        jnz     mNoEject

; Вывод сообщения об успешном извлечении устройства
        call    printf, offset msgEject
        jmp     mExit

mNext:
        inc     esi
        jmp     mNextDevice

mError:                             ; Вывод сообщения об ошибке
        call    GetLastError
        call    printf, offset msgError, eax
        add     esp, 4*2
        jmp     mWait

mNoEject:                   ; Вывод сообщения о занятости устройства
        call    printf, offset msgNoEject
        jmp     mExit
```

```
mNoDevice:                  ; Вывод сообщения об отсутствии устройств
        call    printf, offset msgNoDevice

mExit:
        add     esp, 4*1
        call    SetupDiDestroyDeviceInfoList, hDevInfoSet
        call    printf, offset msgExit
        add     esp, 4*1

mWait:
        call    WaitKey
        call    CloseHandle, hIn
        call    FreeConsole
        call    ExitProcess, 0
END Start
```

5. Требования к отчёту

Отчёт должен содержать:

– титульный лист с указанием названия ВУЗа, кафедры, номера и темы лабораторной работы, а также Ф.И.О. студента, подготовившего отчёт;

– цель работы;

– блок-схема алгоритма программы с кратким описанием;

– листинг отредактированной программы EjectFlesh с результатами её работы.

6. Контрольные вопросы

6.1. Логическая и физическая архитектура USB?

6.2. Что является инициатором транзакции USB?

6.3. Почему не желательно вынимать USB-накопитель из разъёма без использования безопасного отключения?

6.4. Какой формат имеет PnP-идентификатор USB-устройств?

6.5. Какой формат имеет идентификатор экземпляра устройства USB-накопителя?

6.6. Что делает и какой формат имеет функция SetupDiGetDeviceRegistryProperty?

6.7. Что делает и какой формат имеет функция CM_Get_Device_ID?

6.8. Какие функции из SetupAPI.dll используются для нахождения и отключения устройства?

6.9. Как определить строковый идентификатор производителя и продукта?

6.10. Как программа идентифицировать USB-устройства в системе?

7. Рекомендуемая литература

7.1. Агуров П.В. Практика программирования USB. – СПб.: БХВ-Петербург, 2006. с. 69…73, 332…341, 368…374, 566…567.

7.2. SetupAPI Reference [Электронный документ] / Microsoft Corporation. – Режим доступа: https://docs.microsoft.com/en-us/previous-versions//ff550897(v=vs.85) , свободный. – 15.04.2019.

7.3. GMax. Безопасное извлечение USB-устройств [Электронный документ] / GMax. – Режим доступа: https://vxlab.info/wasm/article.php-article=usb_eject.htm , свободный. – 15.04.2019.

7.4. Программное извлечение USB-диска [Электронный документ]. – Режим доступа: http://superadm.net/index.php?name=pages&op=view&id=126 . – 15.04.2019.

7.5. Аблязов Р. Работа с устройствами в Windows [Электронный документ] / Аблязов Р. – Режим доступа: http://pblog.ru/?p=105 , свободный. – 15.04.2019.