ЛАБОРАТОРНАЯ РАБОТА 4.
СОЗДАНИЕ ДИНАМИЧЕСКИХ БИБЛИОТЕК

Основные сведения

Использование динамических библиотек (DLL) – это способ осуществления модульности в период выполнения программы. Польза от этого несомненна. Использование динамических библиотек позволяет избежать перекомпиляции всей программы в случае необходимости перекомпиляции отдельной ее части. К динамической библиотеке возможен доступ сразу из нескольких исполняемых модулей. За счет того, что функция содержится в библиотеке единственный раз, экономится дисковое пространство. Кстати, файлы шрифтов – это тоже динамические библиотеки, единственным содержимым которых являются ресурсы.

Для понимания сущности динамических библиотек потребуется понятие «связывание». *Связывание* – это сопоставление имен, указанных в коде программы, и имен, расположенных во внешнем файле. В случае использования DLL связывание происходит во время выполнения модуля – это *позднее*, или *динамическое* связывание. Позднее связывание может также быть *явным* либо *неявным*, в зависимости от того, происходит оно с помощью API-функций или автоматически при запуске программы. Выгрузка динамической библиотеки из памяти всегда происходит автоматически, при завершении процесса. Следует отметить, что динамическая библиотека загружается в адресное пространство процесса, соответственно все данные процесса доступны из библиотеки и наоборот.

В любой динамической библиотеке следует определить точку входа в процедуру входа – по умолчанию это метка, указанная за директивой end. Процедура входа может быть и пустой.

Процедура входа вызывается каждый раз при загрузке или выгрузке библиотеки, получая через стек три параметра:

1. Идентификатор DLL-модуля.
2. Причина вызова. Возможны четыре значения:

0 (DLL_PROCESS_DETACH) – библиотека выгружается из адресного пространства процесса;

1 (DLL_PROCESS_ATTACH) – библиотека загружена в адрес-

ное пространство вызывающего процесса;

2 (DLL_THREAD_ATTACH) – вызывающий процесс создает новый поток;

3 (DLL_THREAD_DETACH) – вызывающий процесс уничтожает некий поток.

3. Резерв.

Динамическая библиотека, содержащая функцию перекодировки koi8 в 1251 имеет следующий вид. На самом деле, доступны четыре процедуры, позволяющие перекодировать один символ либо строку с передачей параметров через стек и регистр.

dllrus.asm:

```
; DLL для Win32 - перекодировщик из koi8 в cp1251
.386
.model flat
; функции, определяемые в этом DLL
  public koi2win_asm;koi2win_asm перекодирует символ ; в AL
 public koi2win; CHAR WINAPI koi2win(CHAR symbol)
 public koi2wins_asm;koi2wins_asm перекодирует
; строку в [EAX]
 public koi2wins ;VOID WINAPI koi2win(CHAR * string)
.const
```

```
; таблица для перевода символа из кодировки KOI8-r ;(RFC1489)
в кодировку Windows (cp1251)
; для символов 80h – FFh (т.е. надо вычесть 80h из символа,
;преобразовать его и снова добавить 80h)
k2w_tbl db 16 dup(0);символы, не существующие  в cp1251,
      db 16 dup(0)      ; перекодируются в 80h
      db   00h, 00h, 00h, 38h, 00h, 00h, 00h, 00h
      db   00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h
      db   00h, 00h, 00h, 28h, 00h, 00h, 00h, 00h
      db   00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h
      db   7Eh, 60h, 61h, 76h, 64h, 65h, 74h, 63h
      db   75h, 68h, 69h, 6Ah, 6Bh, 6Ch, 6Dh, 6Eh
      db   6Fh, 7Fh, 70h, 71h, 72h, 73h, 66h, 62h
      db   7Ch, 7Bh, 67h, 78h, 7Dh, 79h, 77h, 7Ah
      db   5Eh, 40h, 41h, 56h, 44h, 45h, 54h, 43h
```

```
      db    55h, 48h, 49h, 4Ah, 4Bh, 4Ch, 4Dh, 4Eh
      db    4Fh, 5Fh, 50h, 51h, 52h, 53h, 46h, 42h
      db    5Ch, 5Bh, 47h, 58h, 5Dh, 59h, 57h, 5Ah
.code
_start@12: ;процедура входа dll, в данном случае – пустая
;заглушка
  mov al,1    ; надо вернуть ненулевое число в EAX
  ret 12
; перекодировка символа, передача параметров через  стек
 koi2win proc
  pop ecx     ; обратный адрес в ECX
  pop eax;параметр в ECX (теперь стек очищен от параметров!)
  push ecx  ; обратный адрес вернуть в стек для RET. Здесь нет
;команды RET - управление передается следующей процедуре
 koi2win endp
; перекодировка символа, ввод: AL - код символа в KOI,
;вывод: AL - код этого же символа в WIN
 koi2win_asm    proc
 test al,80h; если символ меньше 80h (старший бит 0)
  jz dont_decode            ; не перекодировать,
  push ebx                  ; иначе -
  mov ebx,offset k2w_tbl
  sub al,80h                ; вычесть 80h
  xlat                      ; перекодировать
  add al,80h                ; и прибавить 80h
  pop ebx
dont_decode:
  ret                       ; выйти
 koi2win_asm endp
; перекодировка строки, передача параметров через стек
koi2wins proc
  pop ecx     ; адрес возврата из стека
  pop eax     ; параметр в EAX
  push ecx    ; адрес возврата в стек
 koi2wins      endp
; перекодировка строки, ввод: EAX - адрес строки,
;которую надо преобразовать из KOI в WIN
 koi2wins_asm   proc
```

```
  push esi; сохранить регистры, которые нельзя ;изменять
   push    edi
   push    ebx
   mov esi,eax ; приемник строк
   mov edi,eax ; и источник совпадают
   mov ebx,offset k2w_tbl
decode_string:
   lodsb              ; прочитать байт,
   test al,80h        ; если старший бит 0,
   jz  dont_decode2   ; не перекодировать,
   sub al,80h         ; иначе - вычесть 80h,
   xlat               ; перекодировать
   add al,80h         ; и добавить 80h
dont_decode2:  stosb              ; вернуть байт на место,
   test al,al         ; если байт - не ноль,
   jnz  decode_string  ; продолжить
   pop ebx
   pop edi
   pop esi
   ret
 koi2wins_asm   endp
   end    _start@12
```

>tasm/ml dllrus.asm -> dllrus.obj (517 b)
> tlink32/Tpd c dllrus.obj,,,,dllrus.def -> dllrus.dll (4096 b)

Внимания заслуживает то, что экспортируемые функции объявлены как **public**. При линковке на необходимость создания DLL указывает ключ /Tpd, а также DEF-файл, который содержит список экспортируемых функций:
dllrus.def:

```
EXPORTS koi2win_asm koi2win koi2wins koi2wins_asm
```

Известно, связывание может быть явным и неявным. При неявном связывании необходимо дополнительно создать программой **implib** и подключить при компиляции статическую библиотеку импорта dllrus.lib. При явном – загрузить библиотеку API-функцией LoadLibrary и определить адрес необходимой функции с помощью GetProcAdress. В этом случае компиляция не отличается от компиляции обычных программ. Считается, что явное

связывание дает большую гибкость для изменений. Ниже приведены оба варианта:

dlldemo.inc:

```
includelib     import32.lib
            extrn   MessageBoxA:near
            extrn   ExitProcess:near
            extrn   LoadLibraryA:near
            extrn   FreeLibrary:near
            extrn   GetProcAddress:near
      MessageBox      equ     MessageBoxA
      LoadLibrary     equ     LoadLibraryA
MB_OK          equ     0000H
```

dlldemo1.def:

```
; неявное подключение dllrus.dll
; выводит строку в KOI8 и затем в cp1251, ;перекодированную
функцией koi2wins
include dlldemo.inc
includelib     dllrus.lib
      extrn   koi2win_asm:near
      extrn   koi2win:near
      extrn   koi2wins_asm:near
      extrn   koi2wins:near
.386
.model FLAT,STDCALL
.const
title_string1  db   'koi2win demo: string in KOI8',0
title_string2 db 'koi2win demo: translate to cp1251',0
.data
koi_string      db      'є ╘╥╔╫╘ ыящ-8',0
.code
_start:
   call MessageBox,0,offset koi_string,\
         offset title_string1, MB_OK
   call koi2wins,offset koi_string
   call MessageBox,0,offset koi_string,\
            offset title_string2,MB_OK
```

```
  call ExitProcess,0     ; конец программы
end     _start
```

>tasm/ml dlldemo1.asm -> dlldemo1.obj (597 b)
>implib dllrus.lib dllrus.dll -> dllrus.lib (1024 b)
>tlink32/Tpe/aa/x/c dlldemo1.obj -> dlldemo1.exe (4096 b)

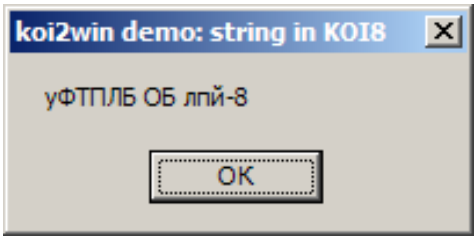
dlldemo2.def:

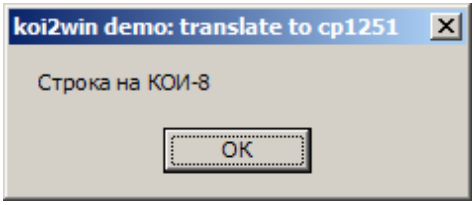
```
; явное подключение dllrus.dll
; выводит строку в KOI8 и затем в cp1251, ;перекодированную
функией koi2wins
include dlldemo.inc
.386
.model FLAT,STDCALL

.const
title_string1  db    'koi2win demo: string in KOI8',0
title_string2 db 'koi2win demo: translate to cp1251',0
libr          db      'dllrus.dll',0
nameproc      db      'koi2wins',0
.data
koi_string      db      'є ╘╥╔╫╨ ыящ-8',0
hlib            dd      ?
koi2wins        dd      ?

.code
_start:
  call LoadLibrary,offset libr  ;загрузка библиотеки
  mov  hlib,eax
  call GetProcAddress,hlib,offset nameproc ;получение
;адреса функции
  mov     koi2wins,eax

  call MessageBox,0,offset koi_string,\
          offset title_string1,MB_OK
  call koi2wins,offset koi_string
  call MessageBox,0,offset koi_string,\
          offset title_string2,MB_OK
  call ExitProcess,0     ; конец программы
```

```
end     _start
```

>tasm/ml dlldemo2.asm -> dlldemo2.obj (689b)
> tlink32/Tpe/aa/x/c dlldemo2.obj -> dlldemo2.exe (4096b)

Результат:





Задание к лабораторной работе

Разработать динамическую библиотеку, реализующую функции в соответствии с заданным вариантом, и программу для демонстрации ее возможностей. Использовать как явное, так и неявное связывание. Примечание: pchar – строка ascii-символов, заканчивающаяся нулем.

1. function HexL(L: longint): pchar.

Возвращает шестнадцатеричное символьное представление числа L.

2. function BinaryL(L: longint): pchar.

Возвращает двоичное символьное представление числа L.

3. function OctalL(L: longint): pchar.

Возвращает восьмиричное символьное представление числа L.

4. function Long2Str(L: longint): pchar.

Возвращает десятичное символьное представление числа L.

5. function Str2Long(S: pchar; var L: longint): boolean. Переводит символьное представление числа S в длинное целое L. Возвращает true, если формат числа в S правильный (например, не содержится недопустимых символов), иначе – false.

6. function StrUpCase(S: pchar): pchar.

Возвращает строку, в которой все строчные русские и латинские буквы заменены на прописные. Для перевода русских букв используются их порядковые номера: буквам «А».. «Я», «а»..«п», «р»..«я» соответствуют десятичные номера 128..159, 160..175, 224..239.

7. function StrLoCase(S:pchar): pchar.

Возвращает строку, в которой все прописные русские и латинские буквы заменены на строчные. Перевод аналогичен используемому в задании 6.

8. function DelBlanks(var S: pchar; Len: byte): boolean. Равномерно удаляет пробелы между словами в строке S до получения длины строки Len. Для этого S циклически просматривается и после каждого слова в ней удаляется 1 пробел (нельзя удалять последний пробел между словами). Если больше нет возможности удалять, а строка все еще длиннее чем Len, то функция завершается с результатом false. Аналогично при длине S меньше или равной Len. При неуспешном завершении результат функции равен true.

9. function PadCh(S: pchar; C: char; Len: byte): pchar. Возвращает строку, в которой S смещена влево, а остаток строки заполнен символами C. Для этого знаки C включаются справа от конца S до тех пор, пока общая длина строки не станет равной Len. Если S длиннее чем Len, то строка не изменяется. Если S - пустая строка, то возвращается строка из Len символов C.

10. function LeftPadCh(S: pchar; C: char; Len: byte):pchar. Возвращает строку, в которой S смещена вправо, а начало строки заполнено символами C. Знаки C включаются слева от начала строки, пока общая длина строки не станет равной Len. Остальное аналогично заданию 9.

11. function TrimLead(S: pchar): pchar.

Возвращает строку, в начале которой удалены все цифры 0 и символы с кодом меньше пробела.

12. function TrimTrail(S: pchar): pchar.

Возвращает строку, в конце которой удалены все цифры 0 и символы с кодом меньше пробела.

13. function DeleteCh(S: pchar; C: char): pchar. Возвращает строку, в которой удалены все вхождения символа C.

14. function CenterCh(S: pchar; C: char; Len: byte):pchar. Возвращает строку длиной Len, в которой содержимое S сцентрировано с помощью символов C . Для этого слева и справа от S равномерно добавляются символы C до получения длины Len. Остальное аналогично заданию 9.

15. function DeTab8(S: pchar): pchar.

Возвращает строку, в которой все вхождения кода горизонтальной табуляции Ht=9 расширены пробелами. Для этого строка просматривается слева направо и встречающиеся коды Ht заменяются на требуемое в данной позиции количество пробелов. Количество пробелов можно определить по формуле, связывающей индекс произвольной текущей позиции *i* с индексом следующей позиции табуляции *j*:

$$j = (((i-1)/8)+1)*8+1,$$

где *i, j* – целые и отсчитываются от единицы.

16. function EnTab8(S: pchar): pchar.

Возвращает строку, в которой пробелы по возможности заменены на коды горизонтальной табуляции (см. задание 15).

17. function WordCount(S: pchar; C: char): byte.

Возвращает количество слов в строке S ограниченных символами C.

18. function ExtractWord(I: byte; S: pchar; C: char):pchar. Возвращает слово, ограниченное символами C, поиск которого в S начинается с позиции I. Символы C в слово не включаются, а при отсутствии такого слова возвращается пустая строка.

19. function CompPchar(S1, S2: pchar): byte.

Возвращает 0, если S1=S2, 1 – если S1>S2, 2 – если S1<S2. Сравнение строк производится так: если длины строк неодинаковы, то результатом будет сравнение длин; если длины одинаковы, то строки сравниваются посимвольно и результатом будет либо значение первого несравнения символов, либо S1=S2, когда достигнут конец строки. Заметим, что результат сравнения не всегда совпадает с принятым для строк в Turbo Pascal.

20. procedure MakeLettersSet(S: pchar; var Letters: Set of char):byte. Возвращает множество Letters вхождений символов кода ASCII в строку S. Множество рассматривается как последовательность из 32 байтов, каждый бит которой соответствует одному символу кода. Если некоторый символ присутствует в S, то соответствующий ему по номеру бит в Letters равен 1, иначе – 0. Перед построением множества его следует инициализировать нулями.

21. function Pos(SubS, S: pchar): byte.

Возвращает позицию первого вхождения подстроки SubS в строку S или 0, если вхождений нет.

22. procedure Delete(var S: pchar; Start, Len: byte). Удаляет в строке S символы с позиции Start и длинной Len. Если Start, больше длины S, то ничего не изменяется.

23. procedure Insert(SubS: pchar; var S:pchar; Start:byte). Вставляет подстроку SubS в строку S, начиная с позиции Start. Если Start, больше длины S, то ничего не изменяется.

24. function Copy(S: pchar; Start, Len: byte): pchar. Возвращает подстроку длиной Len, выделенную из строки S начиная с позиции Start. Если Start, больше длины S, то возвращается пустая строка.

25. function InsBlanks(S: pchar; Len: byte): pchar. Возвращает строку длиной Len, в которую равномерно добавлены пробелы. Для этого строка циклически просматривается и после каждого слова в ней добавляется по одному пробелу до тех пор, пока не будет равенства длины строки и Len. Если длина S сразу больше или равна Len, либо S – пустая строка, то ничего не изменяется.

26. Функции прямого и обратного преобразования Хемминга (циклического кодирования, исправляющего любую одиночную ошибку).

27. Функции шифрования и дешифрования в соответствии с любым существующим алгоритмом.

28. Функции перекодировки текстового файла – Dos, Win, Koi8 и т.д.

29. Функции для работы с матрицами – транспонирование, нахождение обратной и т.д.

30. Функции быстрой сортировки массива элементов.

31. Функции быстрого поиска в массиве элементов.

Порядок выполнения работы

1. Изучить основные сведения.
2. В соответствии с заданием составить внешнюю подпрограмму и основную программу на языке ассемблера.
3. Выполнить ввод, трансляцию, построение кода программы и получить результаты ее работы для подготовленных вариантов исходных данных.

Содержание отчета о работе

1. Цель работы.
2. Текст задания и общая схема решения задачи.
3. Тексты программы и подпрограммы.
4. Результаты работы программы.
5. Выводы по работе.

СПИСОК РЕКОМЕНДУЕМОЙ ЛИТЕРАТУРЫ

1. Андреева А.А. и др. Программирование на языке ассемблера в операционной системе Windows: лабораторный практикум. Чебоксары, Чуваш. ун-т, 2006. 104 с.
2. Зубков С.В. Ассемблер для DOS, Windows и UNIX. М.: ДМК Пресс, 2015. 608 с.
3. Пирогов В.Ю. Ассемблер для Windows. СПб.: БХВ-Петербург, 2011. 864 с.
4. Аблязов Р. Программирование на ассемблере на платформе x86-64. М.: ДМК Пресс, 2016. 302 с.