|  |  |  |
| --- | --- | --- |
|  | | |
| Федеральное государственное бюджетное  образовательное учреждение высшего образования «Новосибирский государственный технический университет» | | |
|  | | |
| Кафедра теоретической и прикладной информатики | | |
| Практическое задание № 2 | | |
| по дисциплине «Языки программирование и методы трансляции» | | |
| ЦЕПОЧЕЧНЫЕ КОМАНДЫ | | |
|  | | |
|  | Бригада 9 | ПМ-13 исакин даниил |
| Вариант 9 | ПМ-13 вострецова екатерина |
|  |  |
|  |  |
|  |  |
|  |  |
| Преподаватель | дворецкая виктория константиновна |
|  |  |
| Новосибирск, 2024 | | |

**1. Задание**

Изучить и приобрести практические навыки работы с цепочечными командами (командами обработки строк символов).

**2. Алгоритм**

1)Считать текстовую строку и поместить её в буфер

2) Сохранить в переменные первый и второй символы в начале строки

3) Загрузить в al значение первой переменной. Вызвать команду repne scasb.

4) Если символы совпали, то заменить на второй символ

5) Если ecx>=0 повтроить действия 3-5

6) Вывести итоговую строку на экран

**3. Используемые средства**

**GetStdHandle** – функция для получения дескрипторов потока ввода, вывода или ошибок

**WriteConsoleA** – функция вывода информации из буфера в консоль

**ReadConsoleA** - функция вывода информации из консоли в буфер

**lstrlenA** - функция определения длины строки

**ExitProcess** - функция выхода из программы

**mov <приёмник> <источник>** – инструкция копирования данных из источника в приёмник

**push** <источник> - инструкция копирования источник в стек

**pop** <приёмник> - инструкция копирования источник в стек

**jmp** <имя> - переход на указанную метку

**call** <имя> - инструкция, вызывающая процедуру

**cmp** <число1> <число2> - инструкция сравнения двух чисел

**add** <число1> <число2> - инструкция сложения двух чисел

**sub** <число1> <число2> - инструкция вычитания двух чисел

**ret** <число> - инструкция возврата из процедуры

**lea** <регистр><память> - загрузить исполнительный адрес. Пересылает смещение ячейки памяти в 32-битовый регистр

**repne**  - префикс повторения, необходимый для многократного повторения команды

**cld**  - очистить флаг напрвления

**scasb** - Команды, реализующие операцию-примитив сканирования цепочек, производят поиск некоторого значения в цепочке.

**4. Текст программы**

.386

.MODEL FLAT, STDCALL

; прототипы внешних функций (процедур) описываются директивой EXTERN,

; после знака @ указывается общая длина передаваемых параметров,

; после двоеточия указывается тип внешнего объекта – процедура

EXTERN GetStdHandle@4: PROC

EXTERN WriteConsoleA@20: PROC

EXTERN CharToOemA@8: PROC

EXTERN ReadConsoleA@20: PROC

EXTERN ExitProcess@4: PROC

EXTERN lstrlenA@4: PROC

.CONST

mul\_coef dd 10 ; Коэффициент шага при переводе

.DATA ; data segment

tmp\_local\_var dd 0 ; переменная для хранения внутри функции, каких-либо промежуточных значений

din dd ? ; Дескриптор ввода

dout dd ? ; Дескриптор вывода dd - резервирет память объемом 4 байта ? - используется для не

strn db "Введите строку: ", 13, 10, 0; Строка для вывода

outbuf db 100 dup(0); Размер буфера на выход

buf db 100 dup(0) ; Размер буфера на вход программы

first\_ch db 0 ; Первый символ в строке

second\_ch db 0 ; Втророй символ в строке. На него мы будем минять

.CODE

; Инициализация потока ввода данных

; Дескриптор будет сохранен в переменную din

OpenRead proc

; получить десериптор ввода

push -10

call GetStdHandle@4

mov din, eax

ret

OpenRead endp

; Получить дескриптор вывода данных

; Дескриптор будет сохранен в переменную dout

OpenWrite proc

; получить дескриптор выводы

push -11

call GetStdHandle@4

mov dout, eax

ret

OpenWrite endp

; Функция перекодировки строки для вывода информации

; eax - Адрес строки для перекодировки.

; Результат сохранен в переменную буфер, которая была передана в регистре EAX

ConvertToDOSStr proc

push eax

push eax

call CharToOemA@8 ; Перекодировки

ret

ConvertToDOSStr endp

; Вывод строки в консоль

; ebx - Указатель на буфер

; ret - eax - Количество действительно выведенных переменных

WriteToConsole proc

; Получаем длину строки. Длина строки будет в регистре eax

push ebx

call lstrlenA@4

push 0 ; Резервный параметр

push offset tmp\_local\_var ; Адрес переменной куда будут записано количество фактически выведенных значений

push eax ; Размер выводимой строки

push ebx ; Адрес буфера

push dout ; Дескриптор консоли для вывода

call WriteConsoleA@20

mov eax, tmp\_local\_var ; Поместить количество выведенных символов в eax

ret

WriteToConsole endp

; Функция чтения данных из консоли.

; eax - Указатель на буфер куда читаем

; ebx - Максимальный размер буфера - его длина

; ret - В случае успеха в переменной eax будет хранится значение фактически прочитанных символов c учетом перевода каретки и строки

ReadFromConsole proc

push 0 ; Резервный параметр

push offset tmp\_local\_var ; Адрес переменной, в которую будет помещено еоличество действительно прочитанных символов

push ebx ; Длина буфер

push eax ; Указатель на буфер

push din ; Дескриптор ввода

call ReadConsoleA@20 ; Вызво операции чтения

mov eax, tmp\_local\_var ; Положить фактически прочитанное количество символов в eax

ret

ReadFromConsole endp

MAIN PROC

; Базовая настройка

call OpenRead ;Открываем дескриптор чтения

call OpenWrite ;Открываем дескриптор записи в консоль

mov eax , offset strn; Значение второго операнда перемещается в первый, offset заберает адрес

call ConvertToDOSStr

; Вызов функции для вывода строки приглошения ввести строку

mov ebx, offset strn

call WriteToConsole

; Операция ввода строки

mov eax, offset buf

mov ebx, 100 ; Размер буфера

call ReadFromConsole ; В регистре eax - фактический размер прочитанной строки с учетом 2 символов

; Валидация размера строки

; В регистре eax содержится количество байт которые были фактически прочитаны

cmp eax, 4

jb end\_replace

mov ecx, eax

sub ecx, 2

; Загрузим в esi источник строки

; Загрузим в edi примемник. В сущности это одна и та же переменная

lea edi, buf

; Сохронили значения 1-ого и 2-ого символа в строке

mov al, [edi]

mov first\_ch, al

mov al, [edi + 1]

mov second\_ch, al

; Сдвинулм на вторую 3-ий элемент в цепочке

inc edi

inc edi

sub ecx, 2

mov al, first\_ch

replace\_str:

repne scasb

je equel

jne end\_replace

equel:

mov al, second\_ch

dec edi

stosb

inc ecx

mov al, first\_ch

jecxz end\_replace

jmp replace\_str

end\_replace:

; Вывод результата работы программы

mov ebx, offset buf

call WriteToConsole

; return 0;

push 0

call ExitProcess@4

MAIN ENDP

END MAIN

**5. Тесты**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Входные данные | Результат | Описание |
| 1 |  |  | Пустая строка |
| 2 | 1 | 1 | Один символ |
| 3 | 12 | 12 | Два символа |
| 4 | 0111110 | 0111111 | Символ в конце |
| 5 | 1111111 | 1111111 | Строка одинаковых символов |
| 6 | 010000000000 | 011111111111 | Замена всей строки |
| 7 | &\*wifh&&&&3uieb\*\*\*& | &\*wifh\*\*\*\*3uieb\*\*\*\* | Английские символы, а так же произвольные, отличные от букв. |

**6. Выводы**

1) В ходе работы научились применять цепочечные команды. В частности, команду **scasb** для проверки на равенство символов, загруженных в регистр al.

2) Научились применять цепочечные команды для обработки строк.