|  |  |  |
| --- | --- | --- |
|  | | |
| Федеральное государственное бюджетное  образовательное учреждение высшего образования FPMI_ngtu_neti_rgb_polya«Новосибирский государственный технический университет» | | |
|  | | |
| Кафедра прикладной математики | | |
| Практическое задание № 2 | | |
| по дисциплине «Языки программирования и методы трансляции» | | |
| **Цепочечные команды** | | |
|  | | |
|  | Группа ПМ-04 | Плешкова Дарья |
| Бригада 11 | Сальников дмитрий |
|  |  |
|  |  |
|  |  |
|  |  |
| Преподаватель | сивак мария алексеевна |
|  | петров роман владимирович |
| Новосибирск, 2022 | | |

**1. Задание**

Написать программу, удаляющую точки в конце исходной строки.

**2. Алгоритм**

1. Получить дескрипторы ввода и вывода
2. Вывести сообщение для ввода строки
3. Считывание строки
4. Сравнение каждого элемента с точкой с конца строки

4.1. Если элемент не является точкой, переход к пункту 5

4.2. Если элемент является точкой, он проверяется на пустоту

4.2.1. Если элемент не пуст, счетчик количества символов в выходной строке увеличить на 1, переход к пункту 4

4.2.2. Если элемент пуст, проверяется на пустоту входная строка

4.2.2.1. Если строка пуста, переход к пункту 8

4.2.2.2. Если строка не пуста, переход к пункту 6

1. Запись в LEND резултата вычитания LENS – LEND, где LEND – будет являтся колличеством симолов которые необходимо вывести с начала строки, а LENS – общая длина строки.
2. Количество символов в выходной строке приравнивается к длине исходной строки
3. В выходную строку записывается полученное количество символов изсходной строки
4. Вывести сообщение для вывода строки
5. Вывести полученную строку

**3. Программные средства**

* PUSH – команда «положить в стек»
* CALL – команда вызова внешней функции
* MOV – команда, пересылающая байт, слово или двойное слово между регистром и ячейкой памяти или между двумя регистрами
* CMP – команда условного сравнения
* JE – команда перехода по условию «приемник равен источнику»
* JNE – команда перехода по условию «приемник не равен источнику»
* ADD – команда, помещающая сумму приемника и источника в приемник
* INC – команда, прибавляющая единицу к регистру или ячейке памяти
* CLD – команда установки флага для движения по возрастанию адресов
* LEA – команда, пересылающая смещение ячейки памяти в регистр
* MOVSB – цепочечная команда, копирующая часть цепочки-источника в цепочку-приемник
* REP – префикс повторения, заставляющий команды выполняться, пока содержимое регистра ЕСХ не равно нулю
* DEC - команда, вычитающая единицу к регистру или ячейке памяти

**4. Тесты**

|  |  |  |  |
| --- | --- | --- | --- |
| **№** | **Исходные данные** | **Результат** | **Комментарий** |
| **1.** | Привет! | Привет! | Строка с символами и без точек |
| **2.** | Привет!... | Привет! | Строка с символами и точками |
| **3.** | … | (пустая строка) | Строка без символов и с точками |
| **4.** | (пустая строка) | (пустая строка) | Пустая строка |
| **5** | A.B.C.D... | A.B.C.D | Строка с точками в середине строки |
| **6** | ...ABCD | ...ABCD | Строка сточками в начале строки |

**5. Текст программы**

.386 ; директива, предписывающая Ассемблеру использовать набор операций для процессора 80386.

.MODEL FLAT, STDCALL ; плоская модель, вызов процедур по соглашению в Windows x32, далее - процедуры:

EXTRN GetStdHandle@4:PROC ; стандартный дескриптор ввода-вывода

EXTRN WriteConsoleA@20:PROC ; вывод текста

EXTRN CharToOemA@8:PROC ; перекодировка

EXTRN ReadConsoleA@20:PROC ; ввод

EXTRN ExitProcess@4:PROC ; выход

EXTRN lstrlenA@4:PROC ; определение длинны строки

.DATA ; сегмент данных

DOUT DD ? ; дескриптор вывода

DIN DD ? ; дескриптор ввода

user\_messaege DB "Введите строку: ",13,10,0 ; Строка на прием данных

out\_message DB "Результат удаления символов в конце строки: ",13,10,0 ; Строка на вывод данных

BUF DB 100 dup (?) ; достаточный буфер для вводимых/выводимых строк

out\_string DB 100 dup (?) ; измененная строка

LENS DD ? ; количество символов во введеной строке

LEND DD ? ; количество символов в строке без точек

.CODE ; сегмент кода

MAIN PROC

; метка точки входа

START:

; получим дескриптор ввода

PUSH -10 ; константа ввода

CALL GetStdHandle@4

MOV DIN, EAX ; перемещение результата из регистра EAX в ячейку памяти с именем DIN (для дескриптора ввода)

; получим дескриптор вывода

PUSH -11 ; константа вывода

CALL GetStdHandle@4

MOV DOUT, EAX ; перемещение результата из регистра EAX в ячейку памяти с именем DOUT (для дескриптора вывода)

; перекодируем строку user\_messaege

PUSH OFFSET user\_messaege ; параметры функции помещаются в стек командой PUSH, OFFSET — операция, возвращающая адрес (смещения)

PUSH OFFSET user\_messaege

CALL CharToOemA@8

NULL:

; вывод строки

PUSH OFFSET user\_messaege ; в стек помещается указатель на строку

CALL lstrlenA@4 ; длина в EAX

; вызов функции WriteConsoleA для вывода строки user\_messaege

PUSH 0 ; 5-й параметр (Конец аргументов)

PUSH OFFSET LENS ; 4-й параметр (Место куда поместить кол-во выведенных символов)

PUSH EAX ; 3-й параметр (Длина выводимой строки)

PUSH OFFSET user\_messaege ; 2-й параметр (Адрес выводимой строки)

PUSH DOUT ; 1-й параметр (Дескриптор вывода)

CALL WriteConsoleA@20

; ввод строки

PUSH 0 ; 5-й параметр (Конец аргументов)

PUSH OFFSET LENS ; 4-й параметр (Длина введенных символов с учетом 2 служебных)

PUSH 100 ; 3-й параметр (Длина строки)

PUSH OFFSET BUF ; 2-й параметр (Куда записать введенную строку)

PUSH DIN ; 1-й параметр (Дескриптор ввода)

CALL ReadConsoleA@20

SUB LENS, 2 ; вычитаение из длины исходной строки двух служебных символов

MOV ECX, LENS

MOV LEND, 0 ; инициализация счетчика символов выходной строки

MOV ESI, OFFSET BUF

DOT:

INC ESI

LOOP DOT

DEC ESI

MOV EAX, LENS ; загрузка длины исходной строки в регистр EAX

MOV BL, [ESI]

CMP BL, '.' ; каждый элемент строки сравнивается с точкой

JE NOT\_EQUAL ; переход на метку "если равно точке"

JNE EQUAL ; переход на метку "если не равно точке"

NOT\_EQUAL:

CMP BL, 0 ; проверка символа на пустоту

JE END\_STR ; переход на метку "если пуст" - значит, строка пройдена до конца

JNE NOT\_NULL ; переход на метку "если не пуст"

END\_STR:

CMP EAX, 0 ; проверка на пустоту исходной строки

JE NULL ; переход на метку "если пуста"

JNE FULL ; переход на метку "если не пуста" - значит, строка без точек

NOT\_NULL:

ADD LEND, 1 ; увеличение количества символов в выходной строке

DEC ESI ; смещение на следующий символ входной строки

MOV BL, [ESI]

CMP BL, '.' ; повторное сравнение символа с точкой

JE NOT\_EQUAL ; повторный переход а метку "если равно точке"

JNE EQUAL ; повторный переход а метку "если не равно точке"

EQUAL:

MOV EAX, LEND

SUB LENS, EAX

MOV EAX, LENS

MOV LEND, EAX

FULL:

MOV LEND, EAX ; запись в количество символов выходной строки длины исходной строки

CLD ; установка флага для прохода с начала строки

LEA ESI, BUF ; загрузка адреса входной строки в регистр ESI

LEA EDI, out\_string ; загрузка адреса выходной строки в регистр EDI

MOV ECX, LEND ; загрузка количества символов выходной строки в регистр ECX

REP MOVSB ; копирование LEND символов из входной строки BUF в выходную строку out\_string

; перекодируем строку user\_messaege

PUSH OFFSET out\_message ; параметры функции помещаются в стек командой PUSH, OFFSET — операция, возвращающая адрес (смещения)

PUSH OFFSET out\_message

CALL CharToOemA@8

; вывод строки

PUSH OFFSET out\_message ; в стек помещается указатель на строку

CALL lstrlenA@4 ; длина в EAX

; вызов функции WriteConsoleA для вывода строки user\_messaege

PUSH 0 ; 5-й параметр (Конец аргументов)

PUSH OFFSET LENS ; 4-й параметр (Место куда поместить кол-во выведенных символов)

PUSH EAX ; 3-й параметр (Длина выводимой строки)

PUSH OFFSET out\_message ; 2-й параметр (Адрес выводимой строки)

PUSH DOUT ; 1-й параметр (Дескриптор вывода)

CALL WriteConsoleA@20

; вывод строки

PUSH OFFSET out\_string ; в стек помещается указатель на строку

CALL lstrlenA@4 ; длина в EAX

; вызов функции WriteConsoleA для вывода строки user\_messaege

PUSH 0 ; 5-й параметр (Конец аргументов)

PUSH OFFSET LEND ; 4-й параметр (Место куда поместить кол-во выведенных символов)

PUSH EAX ; 3-й параметр (Длина выводимой строки)

PUSH OFFSET out\_string ; 2-й параметр (Адрес выводимой строки)

PUSH DOUT ; 1-й параметр (Дескриптор вывода)

CALL WriteConsoleA@20

; выход из программы

PUSH 0 ; код выхода

CALL ExitProcess@4

MAIN ENDP

END MAIN