Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет
имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ **ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ**

КАФЕДРА **КОМПЬЮТЕРНЫЕ СИСТЕМЫ И СЕТИ (ИУ6)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.03.01 Информатика и вычислительная техника**

О Т Ч Е Т

по лабораторной работе № 5

Название: Обработка символьной информации

Дисциплина: Программирование с использованием разно языковых модулей

Студент	ИУ6-42б	17.04.2021	И.С. Марчук
	(Группа)	(Подпись, дата)	(И.О. Фамилия)
Преподаватель			Г.С. Иванова
		(Подпись, дата)	(И.О. Фамилия)

2021 г.

Программирование с использованием разно языковых модулей

Задание:

Разработать программу, состоящую из трех модулей:

1) основной модуль на выбранном языке программирования общего назначения должен содержать диалоговый ввод необходимых данных, вызов функции или процедуры на ассемблере и вывод результатов;

2) второй модуль - функция или процедура на ассемблере, выполняющая заданную обработку и вызывающая для печати диагностических сообщений процедуру на выбранном языке программирования общего назначения;

3) третий модуль - процедура на выбранном языке общего назначения, обязательно получающая некоторые параметры из вызывающего модуля.

Вариант 2.27. Дан текст не более 255 символов. Слова отделяются друг от друга пробелами. Расположить слова текста по алфавиту по первой букве.

Второй модуль должен использовать конвенцию: stdcall.

Третий модуль должен использовать конвенцию: stdcall.

Цель работы:

Изучение конвенций о способах передачи управления и данных при вызове из программы, написанной на языке высокого уровня, подпрограмм, написанных на ассемблере, и наоборот.

Ход работы:

Схема алгоритма для решения поставленной задачи представлена на рисунке 1:

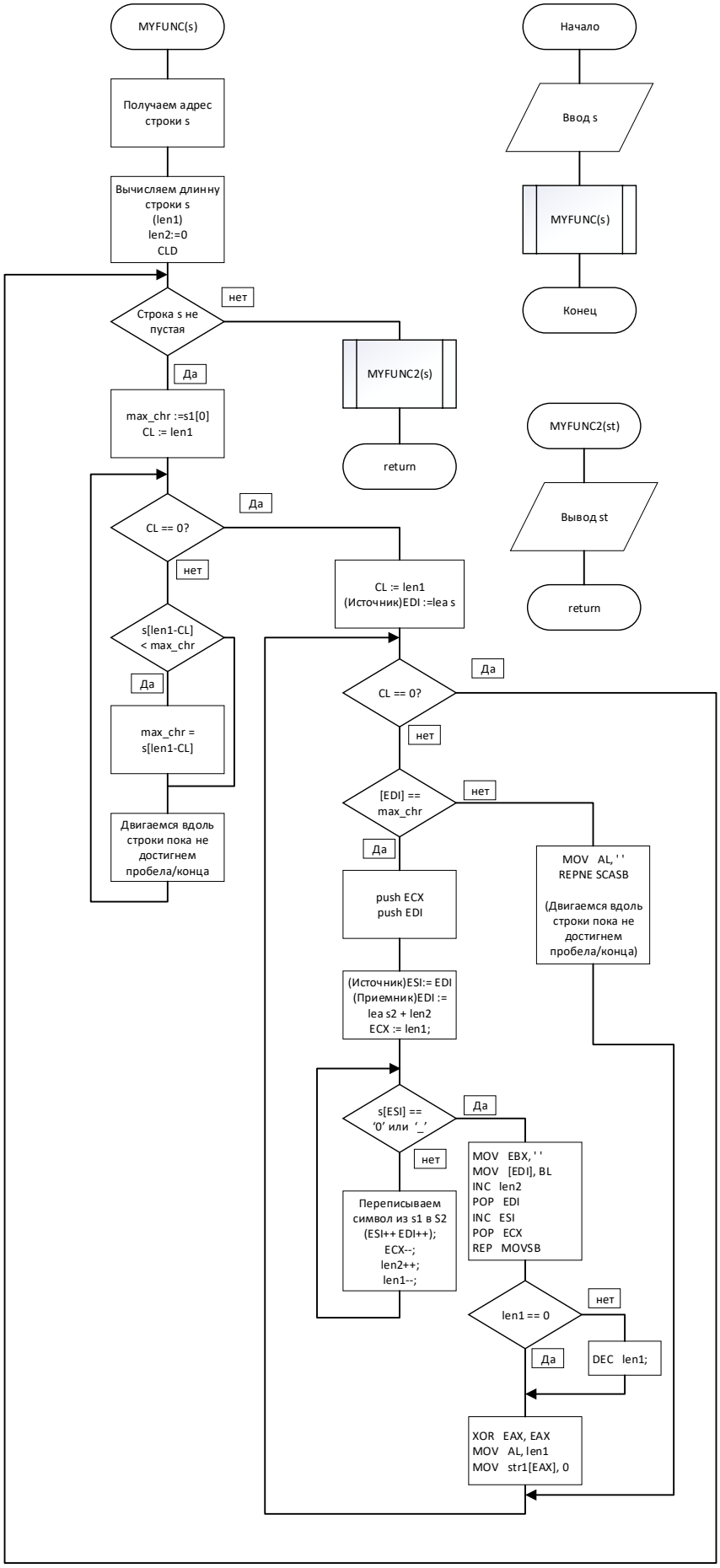

Рисунок 1 – Схема алгоритма

Текст программы:

Код на языке C++:

```
#include <iostream>
using namespace std;
extern "C" void __stdcall MYFUNC(char* a);
extern "C" void __stdcall MYFUNC2(char* a);
int main(){
	char *st;
	st = (char*) malloc(255);
	cout << "Hello! Enter some line\n";
	cin.getline(st, 255, '\n');
	cout << "entered: \"" << st << "\' \n";
	MYFUNC(st);
	free(st);
	system("pause");
}
void __stdcall MYFUNC2(char* a){
	cout << "answer: " << a << '\n';
}
```

Код на языке ассемблера:

```
	.586
	.MODEL  flat
;	OPTION CASEMAP:NONE
	.CONST
	.DATA
str1	BYTE  100 dup (0)
str2	BYTE  100 dup (0)
len2	BYTE  0

	.DATA?
inbuf	DB	100 DUP (?)
len1	BYTE	?
max_chr	BYTE	?
```

```
.CODE
public _MYFUNC@4
EXTERNDEF _MYFUNC2@4:near
_MYFUNC@4   PROC
push   EBP
mov    EBP, ESP
push   EAX
;get argument string length
mov    EDI, [EBP+8]
mov    ECX, 255
mov    EAX, 0
repne  scasb
mov    len1, 255
sub    len1, CL
dec    len1
;dec   len1
;dec   len1

;copy string from arguments to str1
mov    ESI, [EBP+8]
lea    EDI, str1
xor    ECX, ECX
mov    CL, len1
mov    str1[ECX], 0
REP    MOVSB

;main rewrite cycle
while_not_empty:
CMP    len1, 0
JE     while_not_empty_end
```

```
; find smallest first letter
MOV    CL, str1[0]
MOV    max_chr, CL  ;max_chr :=s1[0]
XOR    ECX, ECX
MOV    CL, len1     ;ECX     := len1

while_scan:
CMP    CL, 0
JZ     while_scan_end; CL == 0?
XOR    EBX, EBX
XOR    EAX, EAX
MOV    BL, len1
SUB    BL, CL
MOV    AL, str1[EBX]
CMP    AL, max_chr;s[len1-CL] > max_chr
JG if_no
MOV    max_chr, AL
if_no:
LEA    EDI, str1
ADD    EDI, EBX
MOV    EAX, ' '
REPNE SCASB

JMP    while_scan
while_scan_end:

XOR    ECX, ECX
MOV    CL, len1  ;CL   := len1
lea    EDI, str1 ;ESI := lea s

while_move:
CMP    CL, 0
JZ     while_move_end; CL == 0?
```

```
MOV    AL, [EDI]
CMP    AL, max_chr
JNE if_rewrite_else

PUSH   ECX
PUSH   EDI

MOV    ESI, EDI  ;ESI:= EDI
LEA    EDI, str2
XOR    EAX, EAX
MOV    AL, len2
ADD    EDI, EAX  ;EDI :=lea s2 + len2
XOR    ECX, ECX
MOV    CL, len1  ;ECX := len1

while_inner_move:
MOV    AL, [ESI]
CMP    AL, 0
JE     while_inner_move_end
CMP    AL, ' '
JE     while_inner_move_end
STOSB
DEC    ECX
INC    ESI
INC    len2
DEC    len1

JMP    while_inner_move
while_inner_move_end:

MOV    EBX, ' '
MOV    [EDI], BL
```

```
INC    len2; add whitespce
POP    EDI
INC    ESI
POP    ECX
REP    MOVSB
CMP    len1, 0
JZ     my_end
DEC    len1;
my_end:
XOR    EAX, EAX
MOV    AL, len1
MOV    str1[EAX], 0

JMP    if_rewrite_end
if_rewrite_else:

MOV    AL, ' '
REPNE  SCASB

if_rewrite_end:
jmp    while_move
while_move_end:
jmp    while_not_empty
while_not_empty_end:
; метод вывода
lea    EAX, str2
push   EAX
call   _MYFUNC2@4
pop    EAX
pop    EBP
ret    4
_MYFUNC@4   ENDP
END
```

Работа программы с тестовыми данными проиллюстрирована на рисунке 2, а результаты представлена в таблице 1:

Таблица 1 – Отладка программы

Исходные данные	Ожидаемый результат	Полученный результат
one two three four five six seven	four five one six seven two three	answer: four five one six seven two three
hello enter some line	enter hello line some	answer: enter hello line some
get some bucks and go to america	and america bucks get go some to	answer: and America bucks get go some to

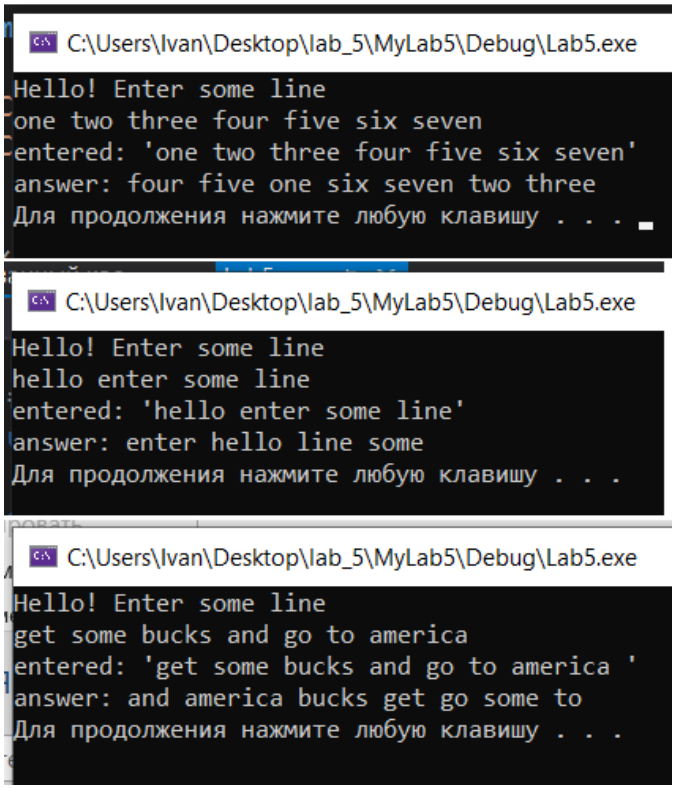

Рисунок 2 – Вывод программы при различных входных данных

Контрольные вопросы:

1) Дайте определение символьной строки.
   Под строкой символов понимается последовательность байт содержащих однотипные данные, идущих в памяти друг за другом.

2) Назовите основные команды обработки цепочек?

- пересылка цепочки:
  movs адрес_приемника,адрес_источника
  movsb
  movsw
  movsd
- сравнение цепочек:
  cmps адрес_приемника,адрес_источника
  cmpsb
  cmpsw
  cmpsd
- сканирование цепочки:
  scas адрес_приемника
  scasb
  scasw
  scasd
- загрузка элемента из цепочки:
  lods адрес_источника
  lodsb
  lodsw
  lodsd
- сохранение элемента в цепочке:
  stos адрес_приемника
  stosb
  stows
  stosd

Какие операции выполняют строковые команды MOVS? Какие особенности характерны для этих команд?

Эта команда выполняет примитивный перенос элемента из цепочки источника в цепочку приемник и увеличивает/уменьшает значение адреса источника и приемника на 1.

3) Какие операции выполняют строковые команды CMPS, SCAS? Какие особенности характерны для этих команд?

CMPS – сравнение двух цепочек, поиск несовпадающих/совпадающих элементов

SCAS – сканирование цепочки, поиск нужного элемента

4) Как обеспечить циклическую обработку строк?

Поставить REP/REPZ/REPNZ перед оператором обработки строки и задать максимальное количество повторов в регистре ECX.

5) Какова роль флага DF во флаговом регистре при выполнении команд обработки строк?

Направление обработки, 0 – вперед (увеличение адресов в ESI и EDI), 1 – назад (уменьшение).

6) Какие макрокоманды используются в среде RADASM для ввода и вывода строк?

- Процедура ввода:
  StdIn PROC lpszBuffer:DWORD, bLen:DWORD
- Процедура вывода завершающейся нулем строки в окно консоли:
  StdOut PROC lpszBuffer:DWORD
- Процедура замены маркера конца строки (0Dh,0Ah) нулем:
  StripLF PROC lpszBuffer:DWORD

7) Как правильно выбрать тестовые данные для проверки алгоритма обработки строки?

Тестовые данные для тестирования и отладки программы необходимо выбрать в соответствии с ее особенностями и в соответствии с заданием. Также необходимо учесть все важные аспекты работы программы и граничные значения.

Вывод:

Я изучил команды обработки цепочек, приемы обработки символьной информации, команды ввода вывода строк. Я изучил подключение и совместную работу в одной программе разноязыковых модулей, а также вызов разноязыковых подпрограмм в разных конвенциях и передачу параметров, в том числе строк.