Министерство образования Республики Беларусь

Учреждение образования

“Брестский государственный технический университет”

**Лабораторная работа №2**

**По дисциплине КСиС за 4 семестр**  
**Тема: «Язык ассемблера. Обработка символьных данных»**

**Выполнил:**

Студент группы ПО-6(1)  
 2-го курса

Мартынович Даниил

**Проверил:**

Бойко Д. О.

Брест 2022

Лабораторная работа №3

Язык ассемблера. Обработка символьных данных

Цель работы: требуется написать на ассемблере программу обработки текста.

**Задание:**

Проверяемое условие:

Текст содержит равное количество прописных латинских и прописных русских букв.

Первое правило преобразования:

Заменить каждую строчную латинскую букву соответствующей прописной буквой, а прописную – строчной.

Первое правило преобразования:

Циклически сдвинуть текст на K (константа) позиций влево без использования дополнительной памяти, реализовав следующий алгоритм: перевернуть под массив из первых K символов; перевернуть оставшийся под массив; перевернуть весь текст.

**Код программы:**

#include <iostream>

#include <Windows.h>

using namespace std;

int checker(char\* str, int len) {

int checker = 0;

\_asm

{

//для проверки какое правило преобразования вызывать

mov eax, checker

//проверка правильности ввода текста

mov esi, str //записываем указатель на текст в регистр esi

mov ecx, len //устанавливаем счётчик цикла равным длине строки

xor ebx, ebx //очищаем ebx

xor edx, edx //очищаем ebx

//проверка размера

mov al, [esi]

cmp al, 0 //если строка пустая

je ext //выход

//Проверяемое условие

//Текст содержит содержит равное количество прописных латинских и прописных русских букв

start: //(65-90 - Прописные латинские буквы, 192-223 - Прописные русские буквы)

mov al, [esi]

cmp al, 65

jb nxt //если меньше переход в nxt

jge s1 //если больше или равно переход в s1

nxt :

inc esi

loop start

jmp check

s1 :

cmp al, 90

jg s2 //если больше переход в s2

jbe s3 //если меньше или равно переход в s3

s2 :

cmp al, 192

jge s4 //если больше или равно переход в s4

jb nxt //если меньше переход в nxt

s3 :

inc ebx //увеличиваем счетчик прописных латинских букв

jmp nxt

s4 :

cmp al, 223

jbe s5 //если меньше или равно переход в s5

jg nxt //если больше переход в nxt

s5 :

inc edx //увеличиваем счетчик прописных русских букв

jmp nxt

//проверка на равенство прописных русских и латинских букв

check :

cmp ebx, edx

je rule1 //если равны переход в rule1

jne rule2 //если не равны переход в rule2

rule1 : //первое праивло преобразования выполнить

mov eax, 1

mov checker, eax

jmp ext

rule2 : // второе правило преобразования выполнить

mov eax, 0

mov checker, eax

jmp ext

ext :

}

return checker;

}

char\* firstRule(char\* str, int len) {

\_asm

{

mov esi, str //записываем указатель на текст в регистр esi

mov ecx, len //устанавливаем счётчик цикла равным длине строки

//Первое правило преобразования

//Заменить каждую строчную латинскую букву соответствующей прописной буквой, а прописную – строчной

rule1:

mov esi, str //записываем указатель на текст в регистр esi

mov ecx, len //устанавливаем счётчик цикла равным длине строки

jmp checkRule1

checkRule1 : //(65-90 - прописные латинские буквы, 97-122 - строчные латинские буквы)

mov al, [esi]

cmp al, 65

jb nxt //если меньше переход в h1

jge h2 //если больше или равно переход в h2

nxt :

inc esi

loop checkRule1

jmp ext

h2 :

cmp al, 90

jg h3 //если больше переход в h3

jbe h4 //если меньше или равно переход в h4

h4 :

add al, 32 //замена прописной латинской на строчную букву

mov[esi], al

jmp nxt

h3 :

cmp al, 97

jge h5 //если больше или равно переход в h5

jb nxt //если меньше переход в h1

h5 :

cmp al, 122

jbe h6 //если меньше или равно переход в h6

jg nxt //если больше переход в h1

h6 :

sub al, 32 //замена строчной латинской на прописную букву

mov[esi], al

jmp nxt

ext :

}

return str;

}

char\* invert(char\* str, int K, int strOffset) {

\_asm

{

jmp \_start

reversee :

mov bx, 2 //делим eax на 2 части,

div bx // чтобы поменять половину строки с другой

mov ecx, eax //поместим в ecx занчение eax

rev :

mov al, [edi]

movsb //скопируем байт из edi в esi

dec esi //уменьшим esi

mov[esi], al

dec esi //уменьшим esi

dec ecx //уменьшим ecx

cmp ecx, 0

jne rev //если не равно 0 переход в rev

ret //выход из подпрограммы

\_start :

mov ecx, str

mov eax, K //установим количество символов, которые нужно перевернуть

mov edx, strOffset //установим смещение

cmp edx, 0

jne m1 //если не равно переход в m1

m1 :

inc ecx //сместим строку на 1 символ вправо

dec edx //уменьшим счетчик смещения на 1

cmp edx, 0

jne m1 //если не равно переход в m1

jmp m2 //переход в m2

m2 :

mov edi, ecx //установим указатель на элемент строки, с которого начать смещение

mov esi, ecx //установим указатель на элемент строки, с которого начать смещение

add esi, eax

dec esi

call reversee //вызов ф смещения

jmp ext //выход

ext :

}

return str;

}

void secondRule(char\* str, int len) {

int K = rand() % (len - 1) + 1;

int secondPart = len - K;

cout << "K = " << K << endl;

cout << "Revert first L symbols" << endl;

cout << invert(str, K, 0) << endl;

cout << "Revert stay symbols" << endl;

cout << invert(str, secondPart, K) << endl;

cout << "Revert all strok" << endl;

cout << invert(str, len, 0) << endl;

}

int main() {

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

setlocale(LC\_ALL, "rus");

system("color f0");

int maxLength = 100;

char\* str = new char[maxLength];

cout << "Input strok c. in the end. " << endl;

for (int i = 0; i < maxLength; i++) {

cin >> str[i];

if (str[i] == '.') {

str[i] = NULL;

break;

}

}

if (checker(str, strlen(str)) == 1) {

cout << "First rule" << endl;

cout << firstRule(str, strlen(str)) << endl;

}

else {

cout << "Second rule" << endl;

secondRule(str, strlen(str));

}

system("pause");

return 0;

}