Министерство образования Республики Беларусь

Учреждение образования

«Брестский государственный технический университет»

Кафедра ИИТ

Лабораторная работа №2

По дисциплине: «Компьютерные системы и сети»

Тема: «Язык Ассемблер. Обработка символьных данных»

Выполнил:

Студент 2 курса

Группы ПО-6

Осьмушников А.Ю.

Проверил:

Бойко Д. О

Брест, 2022

Лабораторная работа №2

Язык Ассемблер. Обработка символьных данных

Вариант 6

**Цель:** требуется написать на ассемблере программу обработки текста.

**Задание:**

Дан текст – непустая последовательность не длиннее ста символов. Признаком конца ввода является точка, в сам текст точка не входит.

Проверить, удовлетворяет ли текст заданному условию. Если условие выполнено, преобразовать текст по одному правилу, в противном случае – по другому правилу. Преобразованный текст напечатать.

Проверяемое условие и правила обработки текста определяются конкретным вариантом задания.

Если введенная последовательность символов не является текстом, преобразовывать ее не нужно, а следует напечатать соответствующее сообщение.

Ввод текста, проверка условия, обработка текста и печать результата должны выполняться последовательно, отдельными частями программы.

**Проверяемое условие:**

Текст содержит равное количество строчных и прописных латинских букв.

**Первое правило преобразования:**

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

**Второе правило преобразования:**

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

**Текст программы:**

#include <iostream>

#include <Windows.h>

using namespace std;

int checkCondition(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 равное количество строчных и прописных латинских букв

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

mov al, [esi]

cmp al, 65

jb case1 //если меньше, то переходим в case1

jge case2 //если больше или равно, то переходим в case2

case1:

inc esi //увеличиввем esi

loop start

jmp check //переходим в check

case2:

cmp al,90

jg case3 //если больше, то переходим в case3

jbe case4 //если меньше или равно, то переходим в case4

case3:

cmp al,97

jge case5 //если больше или равно, то переходим в case25

jb case1 //если меньше, то переходим в case1

case4:

inc ebx //увеличиввем ebx

jmp case1 //переходим в case 1

case5:

cmp al, 122

jbe case6 //если меньше или равно, то переходим в case6

jg case1 //если больше, то переходим в case1

case6:

inc edx //увеличиввем edx

jmp case1 //переходим в case 1

//проверка на равенство прописных и строчных латинских букв

check :

cmp ebx, edx //сравниваем ebx и edx

je rule1 //если равны,то переходим в rule1

jne rule2 //если не равны, то переходим в rule2

rule1 : //выполняем первое правило преобразования

mov eax, 1 //помещаем в eax значение 1

mov checker, eax //помещаем в checker eax

jmp ext //выходим

rule2 : // выполняем второе правило преобразования

mov eax, 0 //помещаем в eax значение 0

mov checker, eax //помещаем в checker eax

jmp ext //выходим

ext : //выход

}

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 << "Переворот первых K символов:" << endl;

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

cout << "Переворот оставшихся символов:" << endl;

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

cout << "Переворот всей строки:" << 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 << "Введите строку (обязательно с '.' на конце): " << endl;

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

cin >> str[i];

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

str[i] = NULL;

break;

}

}

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

cout << "Выполнение первого правила:" << endl;

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

}

else {

cout << "Выполнение второго правила:" << endl;

secondRule(str, strlen(str));

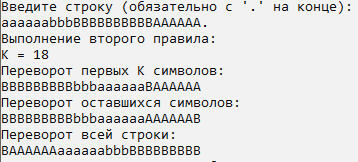
}

system("pause");

return 0;

}

**Результаты тестирования:**

****

**Вывод:** В данной лабораторной работе написала на ассемблере программу обработки текста