МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ

«БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра ИИТ

Отчёт

о лабораторной работе №2

по дисциплине «Компьютерные системы и сети»

Тема: «Изучение аппаратно-программной

архитектуры процессоров семейства Intel – обработка

символьных данных»

Выполнил студент 2 курса

группы ПО-11Сымоник И.А.

Номер зачетной книжки: 220220

Проверил: Савицкий Ю.В.

**Цель работы**: изучение программной модели микропроцессора, практическая работа с подсистемой регистровой и оперативной памяти.

**Вариант 6**

**Ход работы**

**Задание:**

Постановка задачи. 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 указанному в варианте правилу; в противном случае выдать соответствующее сообщение. 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 прописных латинских и прописных русских букв.

**Правило преобразования:**

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

**Исходный код:**

#include <iostream>

#include <Windows.h>

#include <string>

#define NO\_ERR 0

#define ERR\_NOT\_STRING 1

#define ERR\_NOT\_PROCESSING 2

#define ERR\_EXCEEDS\_LIMIT 3

#define MAX\_LENGTH 100

void Process(const char arr[])

{

\_\_asm

{

mov esi, arr

mov ecx, 0

mov edx, 0

replaceLoop :

mov dl, [esi + ecx]

cmp dl, '.'

je end

jmp isUpperEng

isUpperEng :

cmp dl, 'A'

jl isLowerEng

cmp dl, 'Z'

jg isLowerEng

mov dl, [esi + ecx]

add dl, 32

mov[esi + ecx], dl

inc ecx

jmp replaceLoop

isLowerEng :

cmp dl, 'a'

jl notEng

cmp dl, 'z'

jg notEng

mov dl, [esi + ecx]

sub dl, 32

mov[esi + ecx], dl

inc ecx

jmp replaceLoop

notEng :

inc ecx

jmp replaceLoop

end :

}

}

void Count(const char arr[])

{

int res = NO\_ERR;

\_\_asm

{

mov esi, arr

mov eax, 0; eng

mov ebx, 0; rus

mov ecx, 0

mov edx, 0

countLoop:

mov dl, [esi + ecx]

test dl, dl

jz notString

cmp dl, '.'

je isEqual

jmp cheakEng

cheakEng :

cmp dl, 'A'

jl cheakRus

cmp dl, 'Z'

jg cheakRus

inc eax

inc ecx

jmp countLoop

cheakRus :

cmp dl, 'А'

jl notRus

cmp dl, 'Я'

jg notRus

inc ebx

inc ecx

jmp countLoop

notRus :

inc ecx

jmp countLoop

isEqual :

cmp eax, ebx

mov ecx, 0

je end

jmp noProcessing

notString:

mov eax, [res]

add eax, ERR\_NOT\_STRING

mov[res], eax

jmp end

noProcessing:

mov eax, [res]

add eax, ERR\_NOT\_PROCESSING

mov[res], eax

jmp end

end:

}

switch (res)

{

case NO\_ERR:

Process(arr);

break;

case ERR\_NOT\_STRING:

std::cout << "Последовательность символов не является строкой" << std::endl;

break;

case ERR\_NOT\_PROCESSING:

std::cout << "Обработка не требуется" << std::endl;

break;

default:

break;

}

}

void IsPossibleLength(const char arr[])

{

int res = NO\_ERR;

\_\_asm

{

mov esi, arr

mov ecx, 0

mov eax, 0

countLoop:

mov dl, [esi + ecx]

test dl, dl

jz end

inc eax

inc ecx

jmp countLoop

greater :

mov eax, [res]

add eax, ERR\_EXCEEDS\_LIMIT

mov [res], eax

jmp stop

end:

cmp eax, MAX\_LENGTH

jg greater

stop:

}

if (res != NO\_ERR)

{

std::cout << "Длина сообщения превышает 100 символов" << std::endl;

return;

}

Count(arr);

}

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

std::string arr;

std::cout << "Введите строку" << std::endl;

std::getline(std::cin, arr);

std::cout << std::endl;

std::cout << "Начальная строка" << std::endl;

std::cout << arr << std::endl;

std::cout << std::endl;

IsPossibleLength(arr.c\_str());

std::cout << "Преобразованная строка" << std::endl;

std::cout << arr << std::endl;

std::cout << std::endl;

return 0;

}

**Результат выполнения:**

Изображение выглядит как текст, снимок экрана, Шрифт, черный

Автоматически созданное описание

**Вывод:** изучили программную модель микропроцессора, практическая работа с подсистемой регистровой и оперативной памяти.