**Министерство образования и науки Российской Федерации**

Федеральное государственное автономное образовательное учреждение высшего образования «Национальный исследовательский университет

«Высшая школа экономики»

Кафедра Компьютерной безопасности

**ОТЧЕТ**

**К ДОМАШНЕЙ РАБОТЕ №2**

**по дисциплине**

**«Языки программирования»**

Работу выполнил

студент группы СКБ182 \_\_\_\_\_\_\_\_\_\_\_\_\_ Р.Р. Модзгвришвили

подпись, дата

Работу проверил \_\_\_\_\_\_\_\_\_\_\_\_\_ С. А. Булгаков

Подпись, дата

Оглавление

[1.Постановка задачи 2](#_Toc531646608)

[2.Основная часть 4](#_Toc531646609)

[2.1 Идея Решения 4](#_Toc531646610)

[2.2 Путь решения 4](#_Toc531646611)

[3. Тестирование 5](#_Toc531646612)

[Приложение А 6](#_Toc531646613)

[Приложение Б 6](#_Toc531646614)

[Приложение В 7](#_Toc531646615)

[Приложение Г 8](#_Toc531646616)

# 1.Постановка задачи

Внесите изменения в код домашней работы №1 таким образом, чтобы функция

SybmolType TranslateCodeToSymbol( CodeType code )

выполняла преобразование входного параметра code по следующему правилу: младшие 8 бит со-

держат код клавиши, приведенный в таблице 1, а биты с 12 по 15 содержат информацию о действу-

ющих модификаторах. Модификаторы задаются следующим образом:

 бит 12 выставлен – нажата клавиша *Shift* (код нажатия/отпускания клавиши 0x32);

 бит 13 выставлен – нажата клавиша *Caps Lock* (код нажатия/отпускания клавиши 0x42).

При этом, модификация символов выполняется согласно таблице 2 и следующему правилу:

 клавиши *Caps Lock* и *Shift* не нажаты – все символы должны быть из первой группы;

 клавиши *Caps Lock* и *Shift* нажаты – все символы должны быть из первой группы;

 клавиша *Caps Lock* не нажата, а *Shift* нажата – все символы должны быть из второй группы;

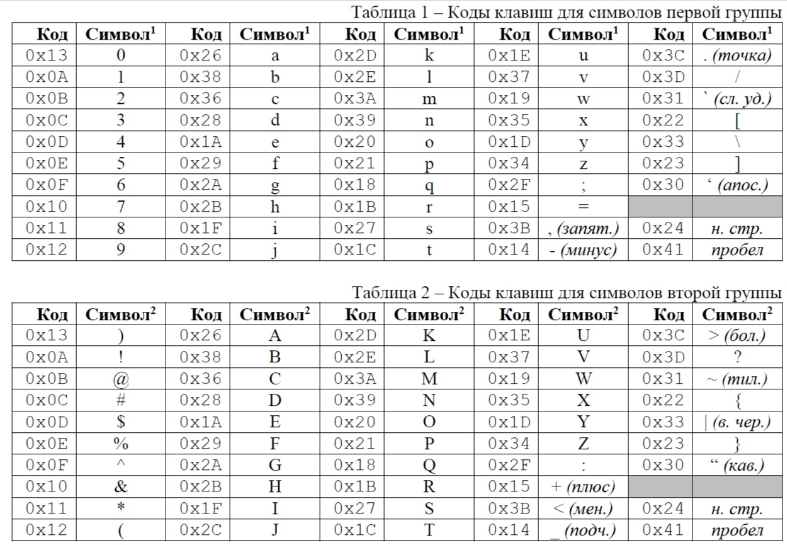
 клавиша *Caps Lock* нажата, а *Shift* не нажата – все символы должны быть из второй группы.

Для работы с битами использовать операции Си++ «И», «ИЛИ», «НЕ» и маску (по ситуации).

Функция TranslateCodeToSymbol() должна выполнять только преобразование кода

клавиши в код символа (коды символов приведены в таблице 3).

Остальные требования домашней работы 1 остаются в силе. В отчете к домашней работе 2

отразить только внесенные изменения. Индивидуальные варианты приведены в таблице 4.

Номер варианта: 6

0x320x300x260x320x390x280x410x2B0x260x270x1C0x410x1C0x2B0x200x1E0x410

x270x2E0x260x1F0x390x410x1C0x2B0x1A0x410x320x2C0x320x260x380x380x1A0x

1B0x190x200x360x2D0x320x3D0x240x360x320x200x3A0x1A0x410x1C0x200x410x3

A0x1D0x410x260x1B0x3A0x270x3B0x410x3A0x1D0x410x380x1A0x260x3A0x1F0x27

0x2B0x410x380x200x1D0x320x0A0x240x200x320x410x290x1B0x260x380x2C0x200

x1E0x270x410x280x260x1D0x320x0A0x410x360x320x200x2E0x2E0x200x200x2B0x

320x0A0x410x360x320x260x2E0x2E0x260x1D0x320x0A0x300x240x2B0x320x1A0x4

10x360x2B0x200x1B0x1C0x2E0x1A0x280x410x1F0x390x410x2B0x1F0x270x410x2C

0x200x1D0x3C

# 2.Основная часть

# 2.1 Идея Решения

Для реализации данной задачи нам необходимо реализовать программу, состоящую из нескольких файлов, которые далее будут собраны в единый проект. На вход программа должна получить набор символов и цифр из *test.txt*. Далее программа разделяет символы на группы по четыре, игнорируя первые 2 символа (0х). После чего согласно данным из *Таблицы 1* и *Таблицы 2*. Значения этих таблиц будет содержаться в *input.cpp*. Символы будут переведен при помощи функции *TranslateCodeToSymbol*. После чего результат будет выведен в окно терминала.

# 2.2 Путь решения

Для начала при помощи цикла *While* производится условие “считывать пока считывается”. После чего первые два символа “отбрасываются” так как они лишь указывают на разрядность чисел. При помощи третьего и четвертого символа вычисляем значение val (value). Так же стоит отметить, что из третьего символа вычитается 48, так как мы ожидаем на ввод лишь числа, а из четвертого 55 в том случае, если он входит в диапазон A-F.(см. Листинг 1)

(Листинг 1)

it\_sym1=int(sym1)-48;

if ((int(sym2)>=48)&&(int(sym2)<=57))

{

it\_sym2=int(sym2)-48;

}

else

{

it\_sym2=int(sym2)-55;

}

Далее проверяем условие нажатия клавиш *Caps Lock* и *Shift.* Для этого введем два флага типа *bool.* Если программа получит значения данных клавиш (*0х32* и *0х42*) , то меняет значение переменных типа *bool* на противоположное.(см. Листинг 2)

(Листинг 2)

if (val==0x32)

{

f\_shift=(!(f\_shift));

}

else if (val==0x42)

{

f\_caps=(!(f\_caps));

}

В ином случае, преобразовываем *val* при помощи побитового сдвига влево на 8 бит. Если значения флага клавиши *Shift* равно 1, то осуществляем побитовый сдвиг к *val* 13, если значения флага клавиши *Caps Lock* равно 1, то на 12. (см. Листинг 3)

(Листинг 3)

val=val<<8;

if (f\_shift)

{

val|=(it\_sym1<<13);

}

if (f\_caps)

{

val|=(it\_sym1<<12);

}

# 3. Тестирование

Введем в терминал команду *make test.* Результат показан в листинге 4.

(Листинг 4)

/main <test.txt

"And hast thou slain the Jabberwock?

Come to my arms, my beamish boy!

O frabjous day! Collooh! Callay!"

He chortled in his joy.

# Приложение А

(Код makefile)

main: main.o input.o

g++ main.o input.o -o main

main.o: main.cpp input.hpp

g++ -c main.cpp -o main.o

input.o: input.cpp input.hpp

g++ -c input.cpp -o input.o

test: main test.txt

./main <test.txt

# Приложение Б

(код файла input.hpp)

#ifndef INPUT\_HPP

#define INPUT\_HPP

typedef int CodeType;

typedef char SymbolType;

SymbolType TranslateCodetoSymbol(CodeType code);

#endif // INPUT\_HPP

# Приложение В

(код файла main.cpp)

#include <iostream>

#include "input.hpp"

int main()

{

char a,sym1,sym2;

int it\_sym1,it\_sym2,val;

bool f\_shift=0,f\_caps=0;

while ((std::cin>>a>>a>>sym1>>sym2))

{

it\_sym1=int(sym1)-48;

if ((int(sym2)>=48)&&(int(sym2)<=57))

{

it\_sym2=int(sym2)-48;

}

else

{

it\_sym2=int(sym2)-55;

}

val=it\_sym1\*16+it\_sym2;

if (val==0x32)

{

f\_shift=(!(f\_shift));

}

else if (val==0x42)

{

f\_caps=(!(f\_caps));

}

else

{

if (f\_shift)

{

val|=(it\_sym1<<13);

}

if (f\_caps)

{

val|=(it\_sym1<<12);

}

std::cout<<TranslateCodetoSymbol(val);

}

}

std::cout<<'\n';

return 0;

}

# 

# Приложение Г

(Код файла input.cpp)

#include "input.hpp"

SymbolType TranslateCodetoSymbol(CodeType code)

{

bool flag=0;

if ((code&1<<12)^(code&1<<13))

{

flag=1;

}

switch (code) {

case 0x13: return flag?')':'0';

case 0x0A: return flag?'!':'1';

case 0x0B: return flag?'@':'2';

case 0x0C: return flag?'#':'3';

case 0x0D: return flag?'$':'4';

case 0x0E: return flag?'%':'5';

case 0x0F: return flag?'^':'6';

case 0x10: return flag?'&':'7';

case 0x11: return flag?'\*':'8';

case 0x12: return flag?'(':'9';

case 0x26: return flag?'A':'a';

case 0x38: return flag?'B':'b';

case 0x36: return flag?'C':'c';

case 0x28: return flag?'D':'d';

case 0x1A: return flag?'E':'e';

case 0x29: return flag?'F':'f';

case 0x2A: return flag?'G':'g';

case 0x2B: return flag?'H':'h';

case 0x1F: return flag?'I':'i';

case 0x2C: return flag?'J':'j';

case 0x2D: return flag?'K':'k';

case 0x2E: return flag?'L':'l';

case 0x3A: return flag?'M':'m';

case 0x39: return flag?'N':'n';

case 0x20: return flag?'O':'o';

case 0x21: return flag?'P':'p';

case 0x18: return flag?'Q':'q';

case 0x1B: return flag?'R':'r';

case 0x27: return flag?'S':'s';

case 0x1C: return flag?'T':'t';

case 0x1E: return flag?'U':'u';

case 0x37: return flag?'V':'v';

case 0x19: return flag?'W':'w';

case 0x35: return flag?'X':'x';

case 0x1D: return flag?'Y':'y';

case 0x34: return flag?'Z':'z';

case 0x2F: return flag?':':';';

case 0x15: return flag?'+':'=';

case 0x3B: return flag?'<':',';

case 0x14: return flag?'\_':'-';

case 0x3C: return flag?'>':'.';

case 0x3D: return flag?'?':'/';

case 0x31: return flag?'~':'`';

case 0x22: return flag?'{':'[';

case 0x33: return flag?'|':'\\';

case 0x23: return flag?'}':']';

case 0x30: return flag?'"':'\'';

case 0x24: return '\n';

case 0x41: return ' ';

default:

return ' ';

}

}