

Федеральное государственное автономное образовательное учреждение высшего  
образования

**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ  
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»**

Факультет компьютерных наук

**САМОСТОЯТЕЛЬНАЯ РАБОТА**

**Разработка многопоточных приложений с использованием OpenMP  
(Вариант 27)**

по направлению подготовки Архитектура вычислительных систем  
образовательная программа «Программная инженерия»

Выполнил:

Студент группы БПИ196

Хожаахмедов Бобурбек

---

Преподаватель:

Легалов Александр Иванович

---

Москва 2020

## Задание

27. *Пляшущие человечки*. На тайном собрании глав преступного мира города Лондона председатель собрания профессор Мориарти постановил: отныне вся переписка между преступниками должна вестись тайнописью. В качестве стандарта были выбраны «пляшущие человечки», шифр, в котором каждой букве латинского алфавита соответствует хитроумный значок. Реализовать многопоточное приложение, шифрующее исходный текст (в качестве ключа используется кодовая таблица, устанавливающая однозначное соответствие между каждой буквой и каким-нибудь числом). Каждый поток шифрует свои кусочки текста. При решении использовать парадигму портфеля задач.

## Составление программы

Алгоритм решения задачи заключается в том, что сначала считываются входные параметры в переменную `input`, а затем вызове метода `split` для того, чтобы удалить пробелы. Следующим шагом является проверка входных параметров на латиницу, для этого вызывается метод `check_for_latin`. Далее через цикл обращаемся к каждой введенной слове и через еще один, внутренний цикл обращаемся к каждому символу этого слова и вызываем метод `toSymb`, где происходит декодирование символа. Так как по условию задачи, сказано, использовать парадигму портфеля задач. Поэтому в своей программе использовал модель многопоточных приложений “Взаимодействующие равные”. (Лекция 7. [Архитектура ВС. Параллельные ВС. Многопоточность](#)). Суть этой парадигмы заключается в том, что используется динамическое распределение задач. И потому, я в своей программе использую новый поток к каждому символу (динамически распределяя задачи, но новый поток не сработает пока не завершится предыдущий поток). Для получения результата используется лямбда выражение.

Как работает метод `toSymb`: Есть два массива строк, один содержит обычные латинские символы, другой соответствующий символ или число по кодовой таблице, которая взята из этого источника: ([https://scask.ru/h\\_book\\_crypt.php?id=45](https://scask.ru/h_book_crypt.php?id=45)). На вход принимается символ. Находится индекс этого символа в первом массиве, а затем результат выводится из второго массива с тем же индексом.

## Текст программы:

```
#include <iostream>
#include <chrono>
#include <iostream>
#include <string>
#include <vector>
#include <algorithm>
#include <sstream>
#include <omp.h>
using namespace std;

void toSymb(char s, string& result)
{
#pragma omp critical
{
    string alph[26] = { "a","b","c","d","e","f", "g","h", "i","j", "k","l", "m","n",
"o","p", "q","r","s","t","u","v","w","x","y","z" };
    string decode_alph[26] = { "5","2","=", "+", "8","1", "3","4", "6","7", "{","0",
"9","*", "#","●", "}", "(,")", ";", "?", "□", "]", ".",":", ",", " " };
    int index;
    for (int i = 0; i < 26; i++)
    {
        string new_s(1, s);
        if (alph[i] == new_s)
        {
            index = i;
        }
    }

    cout << "ID of thread " << omp_get_thread_num() << " " << s << " to " <<
decode_alph[index] << "\n";
    result += decode_alph[index];
}
}

char asciitolower(char in) {
    if (in <= 'Z' && in >= 'A')
        return in - ('Z' - 'z');
    return in;
}

vector<string> split(const string& s, char delim) {
    vector<string> result;
    stringstream ss(s);
    string item;

    while (getline(ss, item, delim)) {
        result.push_back(item);
    }

    return result;
}

bool chech_for_latin(vector<string> v)
{
    for (int i = 0; i < v.size(); i++)
    {
        for (int j = 0; j < v[i].size(); j++)
        {
            if (!((v[i][j] >= 'a' && v[i][j] <= 'z') || (v[i][j] >= 'A' && v[i][j] <=
'Z'))))
            {
                throw exception();
            }
        }
    }
    return true;
}
```

```

int main()
{
    string input;
    cout << "Input some text:\n";
    getline(std::cin, input);

    vector<string> v = split(input, ' ');

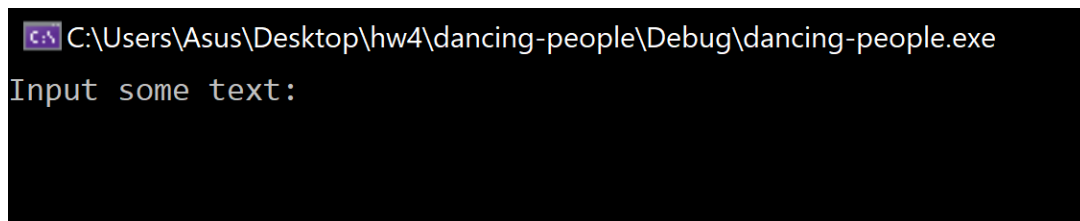
    try
    {
        chech_for_latin(v);
    }
    catch (const std::exception&)
    {
        cout << "Only Latin letters are allowed!";
        return 0;
    }

    string result = "";
    for (int i = 0; i < v.size(); i++)
    {
        std::transform(v[i].begin(), v[i].end(), v[i].begin(), asciitolower);
        char const* ca = v[i].c_str();
        int n = strlen(ca);
        if (i > 0)
            result += " ";
        omp_set_num_threads(n);
#pragma omp parallel
        {
#pragma omp for
            for (int i = 0; i < n; i++)
            {
                toSymb(ca[i], result);
            }
        }
    }
    cout << "\n\" << input << "\" << " encoded to " << "\" << result << "\"\n";
}


```

## Тестирование программы

Поскольку программа в качестве входного параметра принимает строку из латинских символов, для этого программа запрашивает ввод с клавиатуры какой-то текст (предложение, слово).



Также предусмотрен ввод неверных значений, при вводе которых программа выводит информацию об ошибке.

 Консоль отладки Microsoft Visual Studio

Input some text:

Привет

Only Latin letters are allowed!

В результате работы, программа выведет:

```
Консоль отладки Microsoft Visual Studio
Input some text:
Hello world
ID of thread 0 h to 4
ID of thread 0 e to 8
ID of thread 0 l to 0
ID of thread 0 l to 0
ID of thread 0 o to #
ID of thread 0 w to ]
ID of thread 0 o to #
ID of thread 0 r to (
ID of thread 0 l to 0
ID of thread 0 d to +
"Hello world" encoded to "4800# ]#(0+"
C:\Users\Asus\Desktop\abc\hw-4\task4\Debug\task4.exe (процесс 7784) завершил работу с кодом 0.
Нажмите любую клавишу, чтобы закрыть это окно...
```