МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ государственное БЮДЖЕТНОЕ

образовательное учреждение

высшего образования

«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кафедра защиты информации



**ОТЧЁТ**

**по лабораторной работе № 5**

**«**Многофайловые проекты. Стандартная библиотека шаблонов STL**»**

**по дисциплине: «***Программирование***»**

Выполнил:Проверил:

Студент гр. «АБ-121», «АВТФ» *доцент кафедры ЗИ*

*Втюрин Александр Романович Архипова А. Б.*

«7» июня 2022г«\_\_\_» \_\_\_\_\_\_ 2022 г.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись) (подпись)

Новосибирск 2022

**Цели и задачи работы:** изучение стандартной библиотеки шаблонов (STL) языка С++, программирование и отладка программ формирования и обработки контейнеров, комбинации контейнеров.

**Задание к работе:** Написать программу решения задачи в соответствии с индивидуальным вариантом.

**Методика выполнения работы:**

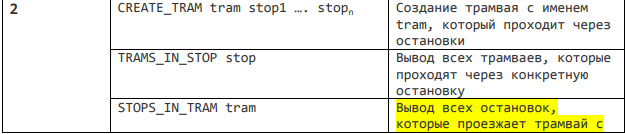
1. Разработать алгоритм решения задачи по индивидуальному заданию.

2. Написать и отладить программу решения задачи.

3. Протестировать работу программы на различных исходных данных.

ЗАДАНИЕ 5.1 «Многофайловый проект ENUM»

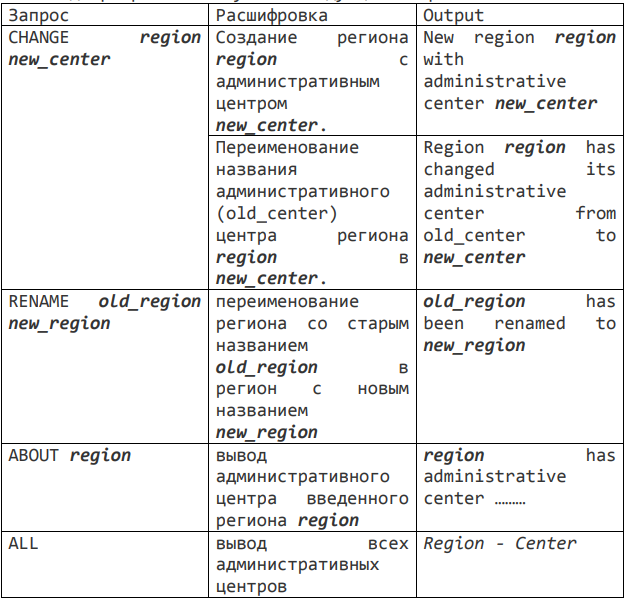
Вариант2: График движения трамваев г. Новосибирска





ЗАДАНИЕ 5.2 «Комбинация контейнеров»

Вариант 2 Необходимо реализовать справочник регионов России. На вход программе поступают следующие запросы



C++

#include <iostream>

#include <map>

#include <string>

#include <vector>

using namespace std;

ostream& operator<< (ostream& os, const map<string, string>& rusreg) {

for (auto i : rusreg) {

os << i.first << " " << i.second << endl;

}

return os;

}

map <string, vector<string>> set\_name(string text, map <string, vector<string>> trains) {

string s, n;

vector<string> v;

int i = 0;

while (text[i] != ' ') { // берем остановки и записываем трамвай

n += text[i];

i++;

}

for (i; i < text.size() + 1; i++) {

if ((text[i] == ' ') || (text[i] == '\0')) { // записываем остановки

v.push\_back(s);

s.clear();

continue;

}

s += text[i];

}

v.push\_back(s);

s.clear();

trains[n] = v;

return trains;

}

void printTown(map <string, vector<string>> b, string town) {

string trains;

int k = 0;

for (auto i : b) {

for (auto j : i.second) {

if ((town == j)) { //проверяем какой трамвай проезжает эту остановку

if (trains == i.first) continue;

trains = i.first;

k = 1;

cout << i.first << " \n";

}

}

}

if (k == 0) cout << "No one\n";

}

void printTrains(map <string, vector<string>> b, string trains) {

string town;

int k;

k = 0;

for (auto i : b) {

if (trains == i.first) {

k = 1; // проверка сушествет ли данный трамвай

for (auto j : i.second) { // для проверки какие еще трамваи проезжают эту остановку

if (j != "") cout << j << "( ";

town = j;

for (auto h : b) {

for (auto z : h.second) {

if (z == "") continue;

if ((town == z) && (h != i)) { // пробегаем по мапу чтобы вывести трамваи

cout << h.first << " ";

}

}

}

if (j != "") cout << ") ";

}

cout << endl;

}

}

if (k == 0) cout << "Tram is absent\n";

}

map <string, vector<string>> get\_name(map <string, vector<string>> a) {

for (auto i : a) {

cout << i.first << " "; // выводим трамваи

for (auto j : i.second) {

cout << j << " ";// выводим остановки

}

cout << endl;

}

return a;

}

void Task1() {

map <string, vector<string>> mtram;

while (1) {

int s = 0;

string text, stop, tram;

bool f = true;

while (f == true)

{

cout << "1.CREATE\_TRAM" << endl << "2.TRAMS\_IN\_STOP" << endl << "3.STOPS\_IN\_TRAM" << endl << "4.TRAMS" << endl << "5.EXIT." << endl;

cin >> s;

switch (s)

{

case 1:

{

cin.ignore();

getline(cin, text);

mtram = set\_name(text, mtram);

}

break;

case 2:

{

cout << "What stop? ";

cin >> stop;

printTown(mtram, stop);

}

break;

case 3:

{

cout << "What tram? ";

cin >> tram;

printTrains(mtram, tram);

}

break;

case 4:

{

get\_name(mtram);

}

break;

case 5:

{

f = false;

}

break;

default:

{

cout << "Invalid comand" << endl;

}

break;

}

}

}

}

void Task2()

{

map <string, string> rusreg;

map <string, string> :: iterator it;

pair <string, string> p;

bool f = true;

while (f == true)

{

cout << "1.Change" << endl << "2.Rename" << endl << "3.About" << endl << "4.All" << endl << "5.Exit" << endl;

int n;

cin >> n;

switch (n)

{

case 1:

{

cout << "Enter the name of the region: ";

cin.ignore();

getline(cin, p.first);

cout << "Enter the name of the administrative center: ";

getline(cin, p.second);

it = rusreg.find(p.first);

if (it != rusreg.end())

{

rusreg.erase(it);

}

rusreg.insert(p);

}

break;

case 2:

{

cout << "Enter the name of the region: ";

cin.ignore();

getline(cin, p.first);

it = rusreg.find(p.first);

if (it == rusreg.end())

{

cout << "Incorrect" << endl;

break;

}

if (it != rusreg.end())

{

p.second = it->second;

rusreg.erase(it);

cout << "Enter the new name of the region: ";

getline(cin, p.first);

rusreg.insert(p);

}

}

break;

case 3:

{

cout << "Enter the name of the region: ";

cin.ignore();

getline(cin, p.first);

it = rusreg.find(p.first);

if (it == rusreg.end())

{

cout << "Incorrect" << endl;

break;

}

if (it != rusreg.end())

{

cout << it->second;

}

}

break;

case 4:

{

cout << rusreg;

}

break;

case 5:

{

f = false;

}

break;

default:

break;

}

}

}

int main()

{

cout << "task 1:" << endl;

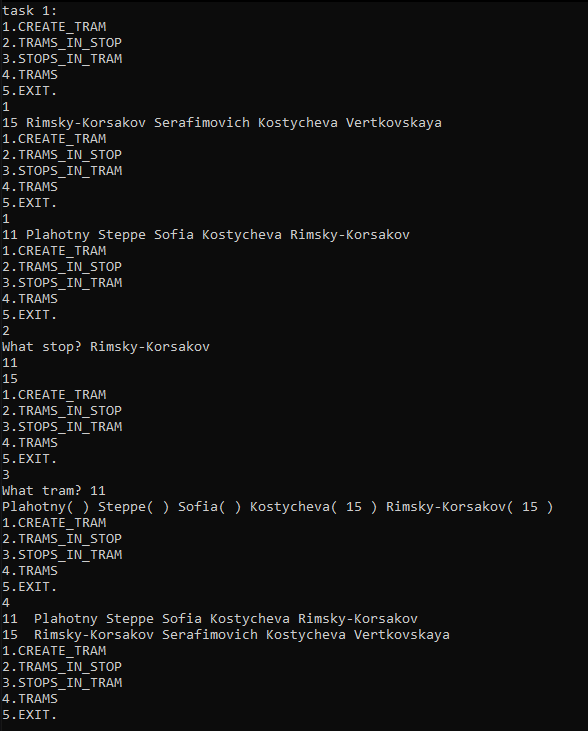
Task1();

cout << "task 2:" << endl;

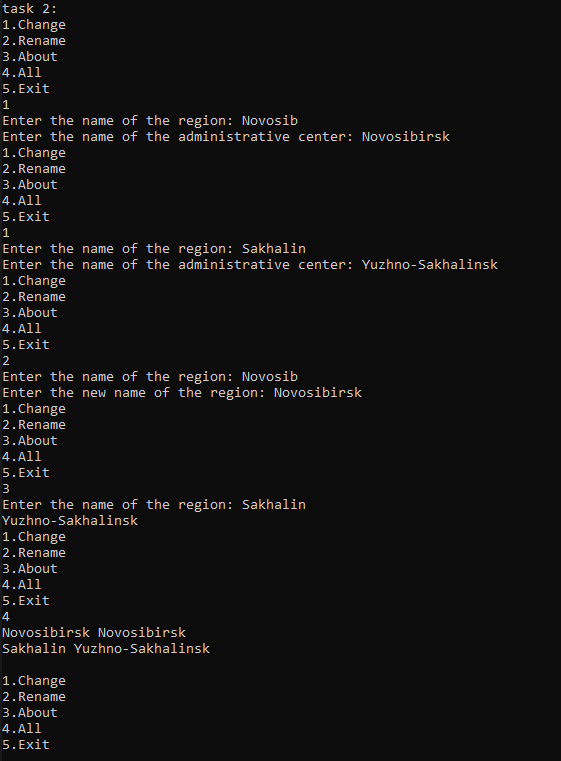
Task2();

}

Task 1:



Task 2:



C#

using System;

using System.Collections.Generic;

public static class Lab5

{

public static SortedDictionary<string, List<string>> set\_name(string text, SortedDictionary<string, List<string>> trains)

{

string s;

string n;

List<string> v = new List<string>();

int i = 0;

while (text[i] != ' ')

{

n += text[i];

i++;

}

for (i; i < text.Length + 1; i++)

{

if ((text[i] == ' ') || (text[i] == '\0'))

{

v.Add(s);

s = "";

continue;

}

s += text[i];

}

v.Add(s);

s = "";

trains[n] = v;

return new SortedDictionary<string, List<string>>(trains);

}

public static void printstop(SortedDictionary<string, List<string>> b, string town)

{

string trains;

int k = 0;

foreach (var i in b)

{

foreach (var j in i.second)

{

if ((town == j))

{

if (trains == i.first)

{

continue;

}

trains = i.first;

k = 1;

Console.Write(i.first);

Console.Write(" \n");

}

}

}

if (k == 0)

{

Console.Write("No one\n");

}

}

public static void printTram(SortedDictionary<string, List<string>> b, string trains)

{

string town;

int k;

k = 0;

foreach (var i in b)

{

if (trains == i.first)

{

k = 1;

foreach (var j in i.second)

{

if (j != "")

{

Console.Write(j);

Console.Write("( ");

}

town = j;

foreach (var h in b)

{

foreach (var z in h.second)

{

if (z == "")

{

continue;

}

if ((town == z) && (h != i))

{

Console.Write(h.first);

Console.Write(" ");

}

}

}

if (j != "")

{

Console.Write(") ");

}

}

Console.Write("\n");

}

}

if (k == 0)

{

Console.Write("Tram is absent\n");

}

}

public static SortedDictionary<string, List<string>> get\_name(SortedDictionary<string, List<string>> a)

{

foreach (var i in a)

{

Console.Write(i.first);

Console.Write(" ");

foreach (var j in i.second)

{

Console.Write(j);

Console.Write(" ");

}

Console.Write("\n");

}

return new SortedDictionary<string, List<string>>(a);

}

public static void Task1()

{

SortedDictionary<string, List<string>> mtram = new SortedDictionary<string, List<string>>();

while (true)

{

int s = 0;

string text;

string stop;

string tram;

bool f = true;

while (f == true)

{

Console.Write("1.CREATE\_TRAM");

Console.Write("\n");

Console.Write("2.TRAMS\_IN\_STOP");

Console.Write("\n");

Console.Write("3.STOPS\_IN\_TRAM");

Console.Write("\n");

Console.Write("4.TRAMS");

Console.Write("\n");

Console.Write("5.EXIT.");

Console.Write("\n");

s = int.Parse(ConsoleInput.ReadToWhiteSpace(true));

switch (s)

{

case 1:

{

cin.ignore();

text = Console.ReadLine();

mtram = new SortedDictionary<string, List<string>>(set\_name(text, new SortedDictionary<string, List<string>>(mtram)));

}

break;

case 2:

{

Console.Write("What stop? ");

stop = ConsoleInput.ReadToWhiteSpace(true);

printstop(new SortedDictionary<string, List<string>>(mtram), stop);

}

break;

case 3:

{

Console.Write("What tram? ");

tram = ConsoleInput.ReadToWhiteSpace(true);

printTram(new SortedDictionary<string, List<string>>(mtram), tram);

}

break;

case 4:

{

get\_name(new SortedDictionary<string, List<string>>(mtram));

}

break;

case 5:

{

f = false;

}

default:

{

Console.Write("Invalid comand");

Console.Write("\n");

}

break;

}

}

}

}

public static void Task2()

{

SortedDictionary<string, string> rusreg = new SortedDictionary<string, string>();

SortedDictionary<string, string> iterator it;

pair<string, string> p = new pair<string, string>();

bool f = true;

while (f == true)

{

Console.Write("1.Change");

Console.Write("\n");

Console.Write("2.Rename");

Console.Write("\n");

Console.Write("3.About");

Console.Write("\n");

Console.Write("4.All");

Console.Write("\n");

Console.Write("5.Exit");

Console.Write("\n");

int n;

n = int.Parse(ConsoleInput.ReadToWhiteSpace(true));

switch (n)

{

case 1:

{

Console.Write("Enter the name of the region: ");

cin.ignore();

p.first = Console.ReadLine();

Console.Write("Enter the name of the administrative center: ");

p.second = Console.ReadLine();

it = rusreg.find(p.first);

if (it != rusreg.end())

{

rusreg.Remove(it);

}

rusreg.insert(p);

}

break;

case 2:

{

Console.Write("Enter the name of the region: ");

cin.ignore();

p.first = Console.ReadLine();

it = rusreg.find(p.first);

if (it == rusreg.end())

{

Console.Write("Incorrect");

Console.Write("\n");

break;

}

if (it != rusreg.end())

{

p.second = it.second;

rusreg.Remove(it);

Console.Write("Enter the new name of the region: ");

cin.ignore();

p.first = Console.ReadLine();

rusreg.insert(p);

}

}

break;

case 3:

{

Console.Write("Enter the name of the region: ");

cin.ignore();

p.first = Console.ReadLine();

it = rusreg.find(p.first);

if (it == rusreg.end())

{

Console.Write("Incorrect");

Console.Write("\n");

break;

}

if (it != rusreg.end())

{

Console.Write(it.second);

}

}

break;

case 4:

{

foreach (var i in rusreg)

{

Console.Write(i.first);

Console.Write(" ");

foreach (var j in i.second)

{

Console.Write(j);

Console.Write(" ");

}

Console.Write("\n");

}

}

break;

case 5:

{

f = false;

}

break;

default:

break;

}

}

}

static int Main()

{

Console.Write("task 1:");

Console.Write("\n");

Task1();

Console.Write("task 2:");

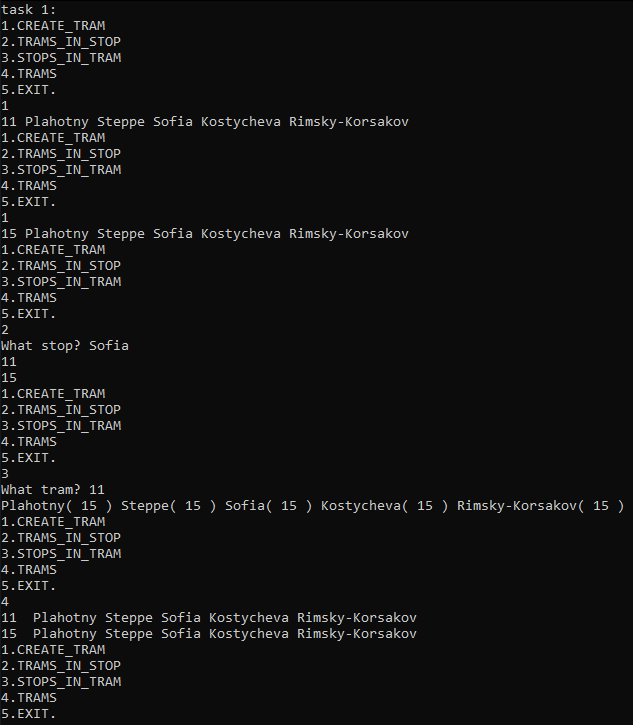
Console.Write("\n");

Task2();

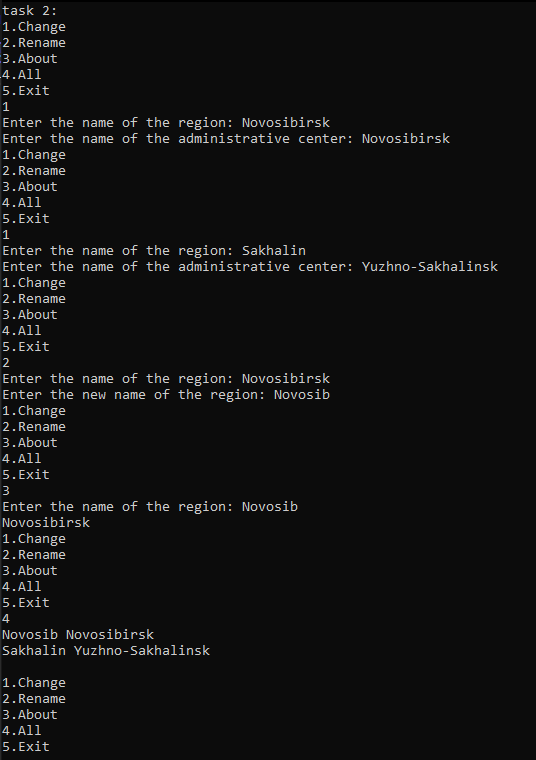
}

}

Task 1:



Task 2:



**Вывод:**

В ходе проделанной работы изучена стандартная библиотека шаблонов (STL). Реализована программа формирования и обработки контейнеров, комбинации контейнеров.. Задачи решены в соответствии с индивидуальным вариантом. Поставленные задачи реализованы на двух языках программирования: C++ и C#.