Пенза 2020

Министерство образования Российской Федерации

Пензенский государственный университет

Кафедра «Вычислительная техника»

Выполнил:

студенты группы 17ВВ1

Кокин Денис

Беленков Никита

Принял:

Дорошенко И.Н.

**ОТЧЁТ**

по лабораторной работе №10

по курсу «Технологии программирования»

на тему «STL»

**Цель работы:** Изучение стандартной библиотеки шаблонов STL (Standard Template Library).

Лабораторное задание

Лабораторное задание выполняется на базе лабораторной работы №1. Для указанного в лабораторной работе №1 базового класса реализовать структуру данных в соответствии с лабораторным заданием.

Требования к классу: – Класс из лабораторной работы №1 должен быть дополнен всеми необходимыми функциями (конструкторами) и операторами для обеспечения работы STL контейнеров и функций.

– В классе все строковые данные представить с использованием классов string (или wstring).

Требования к программе: – Программа должна реализовать контейнер объектов класса в соответствии с вариантом задания. – Для работы с контейнером должны быть реализованы циклы с операциями: ● добавление элемента в контейнер; ● удаление элемента из контейнера; ● вывод контейнера на экран

**Листинг программы**

#include <wchar.h>

#include <string>

#include <iostream>

#include <conio.h>

#include <fstream>

#include <map>

#ifdef \_MSC\_VER

#define \_CRT\_SECURE\_NO\_WARNINGS

#endif

using namespace std;

class Locality{

public:

string Name;

string Region;

Locality() :

Population(0), Budget(0), Code(0) {

};

Locality(int Population, string Region) :

Population(Population), Budget(0), Code(0) {

this->Region = Region;

};

Locality(Locality& Local) :

Population(Local.Population), Budget(Local.Budget), Code(Local.Code) {

};

Locality(string Name, string Region, int Code = 43, int Population = 100000, int Budget = 50000) :

Population(Population), Budget(Budget), Code(Code) {

this->Region = Region;

this->Name = Name;

};

Locality(string Region, int Code) :

Population(0), Budget(0), Code(Code) {

this->Region = Region;

};

Locality(int Population, int Budget) :

Population(Population), Budget(Budget), Code(0) {

};

void Input(void) {

cout << "Enter name of locality: "; cin >> Name;

cout << "Enter region of locality: "; cin >> Region;

cout << "Enter code of locality: "; cin >> Code;

cout << "Enter population of locality: "; cin >> Population;

cout << "Enter budget of locality: "; cin >> Budget;

}

void Output(void) {

cout << ("Name: " + Name) << endl;

cout << ("Region: " + Region) << endl;

cout << "Code: " << Code << endl;

cout << "Population: " << Population << endl;

cout << "Budget: " << Budget << endl;

}

void ClassInfo() {

cout << "Class size: " << sizeof(Locality) << endl;

cout << "Length of char's arrays: " << sizeof(Name) << endl;

this->AbleBodied();

this->BudgetToUSD();

cout << endl;

}

~Locality() {

Population = NULL;

Code = NULL;

Budget = NULL;

};

void Clear() {

this->Locality::~Locality();

}

protected:

int Population;

int Code;

int Budget;

void AbleBodied() {

cout << "Able-bodied population: " << (this->Population / 2) << endl;

}

private:

void BudgetToUSD() {

cout << "Budget of locality: " << (this->Budget / 65) << " USD" << endl;

}

};

class City : public Locality {

public: int isRegionCenter;

void Output() {

Locality::Output();

cout << "Count of district: " << isRegionCenter << endl;

}

void Input() {

Locality::Input();

cout << "Enter a count of district: "; cin >> isRegionCenter;

}

};

class Township : public Locality {

public:

int SchoolCount;

void Output() {

Locality::Output();

cout << "Schools: " << SchoolCount << endl;

}

void Input() {

Locality::Input();

cout << "Enter a count of schools: "; cin >> SchoolCount;

}

};

class Village : public Township {

public: int ShopCount;

void Output() {

Township::Output();

cout << "Shops: " << ShopCount << endl;

}

void Input() {

Township::Input();

cout << "Enter a count of shops: "; cin >> ShopCount;

}

};

int main() {

int mCountOfIterations;

int mPrimitiveSample;

int tDeleteCondidate;

/// PRIMITIVES ///

std::multimap<int, int> mPrimitiveMultimap;

std::multimap<int, int>::iterator mPrimitiveMultimapIterator;

std::cout << "Input a count of key's objects in multimap for primitives: ";

std::cin >> mCountOfIterations;

for (int tItr = 0; tItr < mCountOfIterations; tItr++) {

std::cout << "Input a number: "; std::cin >> mPrimitiveSample;

mPrimitiveMultimap.insert(std::pair<int, int>(tItr, mPrimitiveSample));

}

std::cout << "\nAll values in multimap with primitives" << std::endl;

for (mPrimitiveMultimapIterator = mPrimitiveMultimap.begin(); mPrimitiveMultimapIterator != mPrimitiveMultimap.end(); ++mPrimitiveMultimapIterator)

std::cout << (\*mPrimitiveMultimapIterator).first << " => " << (\*mPrimitiveMultimapIterator).second << '\n';

cout << "Input a number of region which could be deleted: "; cin >> tDeleteCondidate;

mPrimitiveMultimapIterator = mPrimitiveMultimap.find(tDeleteCondidate);

if (mPrimitiveMultimapIterator != mPrimitiveMultimap.end()) {

mPrimitiveMultimap.erase(mPrimitiveMultimapIterator);

}

else {

cout << "The container does not contains inputed name" << endl;

}

std::cout << "\nAll values in multimap with primitives after deleteting" << std::endl;

for (mPrimitiveMultimapIterator = mPrimitiveMultimap.begin(); mPrimitiveMultimapIterator != mPrimitiveMultimap.end(); ++mPrimitiveMultimapIterator)

std::cout << (\*mPrimitiveMultimapIterator).first << " => " << (\*mPrimitiveMultimapIterator).second << '\n';

/// ///

/// POINTERS ///

std::cout << "Input a count of key's objects in multimap for Locality pointers: ";

std::cin >> mCountOfIterations;

std::multimap<int, Locality\*> mLocalityMultimap;

std::multimap<int, Locality\*>::iterator mLocalityMultimapIterator;

Locality\* SampleLocality;

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

std::cout << "Input a number-key: "; std::cin >> mPrimitiveSample;

SampleLocality = new Locality();

SampleLocality->Input();

mLocalityMultimap.insert(std::pair<int, Locality\*>(mPrimitiveSample, SampleLocality));

}

for (mLocalityMultimapIterator = mLocalityMultimap.begin(); mLocalityMultimapIterator != mLocalityMultimap.end(); mLocalityMultimapIterator++) {

std::cout << mLocalityMultimapIterator->first << " => ";

mLocalityMultimapIterator->second->Output();

}

cout << "Input a number of region which could be deleted: "; cin >> tDeleteCondidate;

mLocalityMultimapIterator = mLocalityMultimap.find(tDeleteCondidate);

if (mLocalityMultimapIterator != mLocalityMultimap.end()) {

mLocalityMultimap.erase(mLocalityMultimapIterator);

}

else {

cout << "The container does not contains inputed name" << endl;

}

std::cout << "\nAll values in multimap with primitives after deleteting" << std::endl;

for (mLocalityMultimapIterator = mLocalityMultimap.begin(); mLocalityMultimapIterator != mLocalityMultimap.end(); mLocalityMultimapIterator++) {

std::cout << mLocalityMultimapIterator->first << " => ";

mLocalityMultimapIterator->second->Output();

}

/// ///

/// OBJECTS ///

std::cout << "Input a count of key's objects in multimap for Locality objects: ";

std::cin >> mCountOfIterations;

std::multimap<int, Locality> mLocalityMultimapObjects;

std::multimap<int, Locality>::iterator mLocalityMultimapIteratorObjects;

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

std::cout << "Input a number-key: "; std::cin >> mPrimitiveSample;

Locality LocalityObject;

LocalityObject.Input();

mLocalityMultimapObjects.emplace(mPrimitiveSample, LocalityObject);

}

for (mLocalityMultimapIteratorObjects = mLocalityMultimapObjects.begin();

mLocalityMultimapIteratorObjects != mLocalityMultimapObjects.end();

mLocalityMultimapIteratorObjects++) {

std::cout << mLocalityMultimapIteratorObjects->first << " => ";

mLocalityMultimapIteratorObjects->second.Output();

}

cout << "Input a number of region which could be deleted: "; cin >> tDeleteCondidate;

mLocalityMultimapIteratorObjects = mLocalityMultimapObjects.find(tDeleteCondidate);

if (mLocalityMultimapIteratorObjects != mLocalityMultimapObjects.end()) {

mLocalityMultimapObjects.erase(mLocalityMultimapIteratorObjects);

}

else {

cout << "The container does not contains inputed name" << endl;

}

std::cout << "\nAll values in multimap with objects after deleteting" << std::endl;

for (mLocalityMultimapIteratorObjects = mLocalityMultimapObjects.begin();

mLocalityMultimapIteratorObjects != mLocalityMultimapObjects.end();

mLocalityMultimapIteratorObjects++) {

std::cout << mLocalityMultimapIteratorObjects->first << " => ";

mLocalityMultimapIteratorObjects->second.Output();

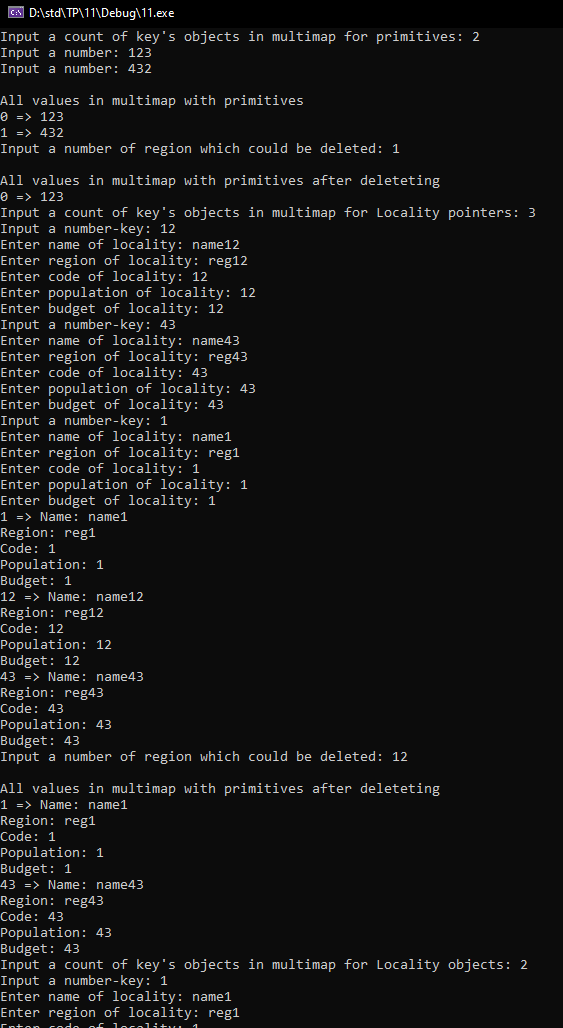
}

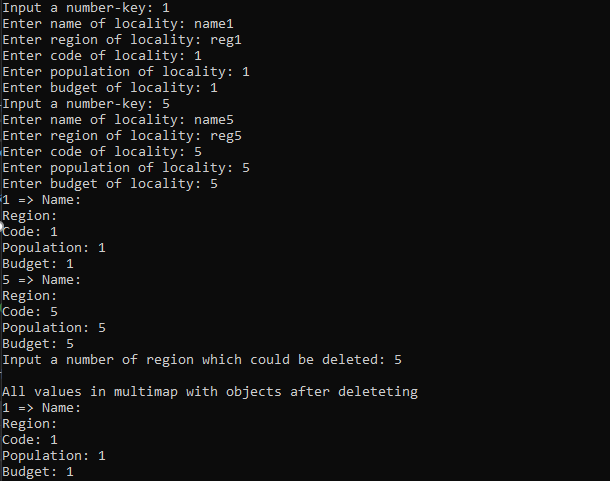
\_getch();

return 1;

}

**Результаты работы программы**





**Вывод:** студенты изучили механизмы сохранения и восстановления объектов данных. Получили навыки разработки самовосстанавливающихся структур данных.