Министерство науки и высшего образования РФ

Федеральное бюджетное государственное образовательное учреждение высшего профессионального образования

Пермский национальный исследовательский политехнический университет (ПНИПУ)

Факультет: Электротехнический (ЭТФ)

Направление: 09.03.01 «Информатика и вычислительная техника (Автоматизированные системы обработки информации и управления)»

Кафедра: «Информационные технологии и автоматизированных систем» (ИТАС)

Основы алгоритмизации

Лабораторная работа №12

Ассоциативные контейнеры библиотеки STL

Студент: Балтаев Э. А.

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

(дата, подпись)

Группа: АСУ-20-1бзу

Работу проверил:

доцент кафедры ИТАС

Полякова О. А.

Оценка: \_\_\_\_\_\_\_\_\_\_\_

Подпись:\_\_\_\_\_\_\_\_\_\_\_

Пермь 2021г.

Постановка задачи:

Часть 1)

1. Создать ассоциативный контейнер

2. Заполнить его элементами стандартного типа

3. Добавить элементы в соответствии с заданием

4. Удалить элементы в соответствии с заданием

5. Выполнить задание варианта для полученного контейнера

6. Выполнение всех заданий оформить в виде глобальных функций

Часть 2)

1. Создать ассоциативный контейнер

2. Заполнить его элементами пользовательского типа. Для пользовательского типа перегрузить необходимые операции

3. Добавить элементы в соответствии с заданием

4. Удалить элементы в соответствии с заданием

5. Выполнить задание варианта для полученного контейнера

6. Выполнение всех заданий оформить в виде глобальных функций

Часть 3)

1. Создать параметризованный класс, используя в качестве контейнера ассоциативный контейнер

2. Заполнить его элементами стандартного типа

3. Добавить элементы в соответствии с заданием

4. Удалить элементы в соответствии с заданием

5. Выполнить задание варианта для полученного контейнера

6. Выполнение всех заданий оформить в виде виде методов параметризованного класса

Задание варианта:

Задача 1

1. Контейнер – multiset

2. Тип элементов – double

Задача 2

Тип элементов Money

Задача 3

Параметризированный класс – Вектор

Задание 3: Найти минимальный элемент и добавить его в конец контейнера

Задание 4: Найти элемент с заданным ключом и удалить его из контейнера

Задание 5: К каждому элементу добавить сумму минимального и максимального элементов контейнера

Определение классов

1) Сумма

#pragma once

#include <iostream>

using namespace std;

class money

{

private:

long rub;

int cop;

public:

money();

money(long r, int c);

money(const money& m);

~money() {}

long get\_rub() { return rub; }

long get\_cop() { return cop; }

void set\_rub(long r);

void set\_cop(int c);

money& operator= (const money& m);

money operator+ (const money& m) const;

bool operator== (const money& m) const;

bool operator!= (const money& m);

bool operator< (const money& m) const;

friend istream& operator>> (istream& in, money& m);

friend ostream& operator<< (ostream& out, const money& m);

};

Определение компонентных функций

1) Сумма

#include "money.h"

money::money()

{

rub = 0;

cop = 0;

}

money::money(long r, int c)

{

set\_rub(r);

set\_cop(c);

}

money::money(const money& m)

{

rub = m.rub;

cop = m.cop;

}

void money::set\_rub(long r)

{

if (r < 0)

rub = 0;

else

rub = r;

}

void money::set\_cop(int c)

{

if (c < 0)

cop = 0;

else {

rub += c / 100;

cop = c % 100;

}

}

money& money::operator=(const money& m)

{

if (&m == this)

return \*this;

rub = m.rub;

cop = m.cop;

return \*this;

}

money money::operator+(const money& m) const

{

money temp(rub + m.rub, cop + m.cop);

return temp;

}

bool money::operator==(const money& m) const

{

if (rub == m.rub && cop == m.cop)

return true;

return false;

}

bool money::operator!=(const money& m)

{

if (rub != m.rub || cop != m.cop)

return true;

return false;

}

bool money::operator<(const money& m) const

{

if (rub < m.rub)

return true;

if (rub == m.rub && cop < m.cop)

return true;

return false;

}

istream& operator>>(istream& in, money& m)

{

long r;

int c;

cout << "Введите сумму через пробел: ";

in >> r; m.set\_rub(r);

in >> c; m.set\_cop(c);

return in;

}

ostream& operator<<(ostream& out, const money& m)

{

out << m.rub << "," << m.cop;

return out;

}

1 часть)

#pragma once

#include <iostream>

#include <set>

#include <algorithm>

using namespace std;

multiset<double> first\_make() {

int n;

cout << "Количество элементов? "; cin >> n;

if (n < 1)

throw exception("неверное количество");

cout << "Введите через пробел " << n << " элементов: ";

multiset<double> temp;

double val;

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

cin >> val;

temp.insert(val);

}

cout << "Успешно" << endl;

return temp;

}

void first\_show(string text, multiset<double>& v) {

cout << text;

if (v.size() == 0)

cout << "Пусто" << endl;

for (auto iter = v.begin(); iter != v.end(); iter++)

cout << \*iter << " ";

cout << endl;

}

void first\_insert(multiset<double>& v) {

if (v.size() == 0)

throw exception("пустой вектор");

auto minIter = min\_element(v.begin(), v.end());

v.insert(\*minIter);

cout << "Успешно" << endl;

}

void first\_delete(multiset<double>& v) {

if (v.size() == 0)

throw exception("пустой вектор");

float key;

cout << "Key? "; cin >> key;

auto findIter = find(v.begin(), v.end(), key);

if (findIter == v.end())

cout << "Нет данного значения" << endl;

else {

v.erase(findIter);

cout << "Успешно" << endl;

}

}

void first\_update(multiset<double>& v) {

if (v.size() == 0)

throw exception("пустой вектор");

auto minIter = min\_element(v.begin(), v.end());

cout << " Минимальный = " << \*minIter;

auto maxIter = max\_element(v.begin(), v.end());

cout << " Максимальный = " << \*maxIter;

double dif = \*minIter + \*maxIter;

multiset<double> temp;

for (auto iter = v.begin(); iter != v.end(); iter++) {

temp.insert(\*iter + dif);

}

v.clear();

v = temp;

cout << " Успешно" << endl;

}

void first\_exec() {

multiset<double> v = first\_make();

first\_show("Исходный: ", v);

cout << "Добавить минимальный в конец: ";

first\_insert(v);

first\_show("После добавления: ", v);

cout << "Удалить с заданным ключом: ";

first\_delete(v);

first\_show("После удаления: ", v);

cout << "Ко всем элементам добавить сумму минимального и максимального: ";

first\_update(v);

first\_show("После изменения: ", v);

}

2 часть)

#pragma once

#include <iostream>

#include <set>

#include <algorithm>

#include "money.h"

using namespace std;

multiset<money> second\_make() {

int n;

cout << "Количество сумм? "; cin >> n;

if (n < 1)

throw exception("неверное количество");

multiset<money> temp;

money val;

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

cin >> val;

temp.insert(val);

}

cout << "Успешно" << endl;

return temp;

}

void second\_show(string text, multiset<money>& v) {

cout << text;

if (v.size() == 0)

cout << "Пусто" << endl;

for (auto iter = v.begin(); iter != v.end(); iter++)

cout << \*iter << " ";

cout << endl;

}

void second\_insert(multiset<money>& v) {

if (v.size() == 0)

throw exception("пустой вектор");

auto minIter = min\_element(v.begin(), v.end());

v.insert(\*minIter);

cout << "Успешно" << endl;

}

void second\_delete(multiset<money>& v) {

if (v.size() == 0)

throw exception("пустой вектор");

money key;

cout << "Key? "; cin >> key;

auto findIter = find(v.begin(), v.end(), key);

if (findIter == v.end())

cout << "Нет данного значения" << endl;

else {

v.erase(findIter);

cout << "Успешно" << endl;

}

}

void second\_update(multiset<money>& v) {

if (v.size() == 0)

throw exception("пустой вектор");

auto minIter = min\_element(v.begin(), v.end());

cout << " Минимальный = " << \*minIter;

auto maxIter = max\_element(v.begin(), v.end());

cout << " Максимальный = " << \*maxIter;

money dif = \*minIter + \*maxIter;

multiset<money> temp;

for (auto iter = v.begin(); iter != v.end(); iter++) {

temp.insert(\*iter + dif);

}

v.clear();

v = temp;

cout << " Успешно" << endl;

}

void second\_exec() {

multiset < money> v = second\_make();

second\_show("Исходный: ", v);

cout << "Добавить минимальный в конец: ";

second\_insert(v);

second\_show("После добавления: ", v);

cout << "Удалить с заданным ключом: ";

second\_delete(v);

second\_show("После удаления: ", v);

cout << "Ко всем элементам добавить сумму минимального и максимального: ";

second\_update(v);

second\_show("После изменения: ", v);

}

3 часть)

#pragma once

#include <iostream>

#include <set>

#include <algorithm>

using namespace std;

template <class T>

class VectorThird

{

private:

multiset<T> v;

public:

VectorThird();

~VectorThird();

void make();

void show(string text);

void insert();

void del();

void upd();

};

template<class T>

inline VectorThird<T>::VectorThird()

{

}

template<class T>

inline VectorThird<T>::~VectorThird()

{

v.clear();

}

template<class T>

inline void VectorThird<T>::make()

{

int n;

cout << "Количество элементов? "; cin >> n;

if (n < 1)

throw exception("неверное количество");

cout << "Введите через пробел " << n << " элементов: ";

T val;

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

cin >> val;

v.insert(val);

}

cout << "Успешно" << endl;

}

template<class T>

inline void VectorThird<T>::show(string text)

{

cout << text;

if (v.size() == 0)

cout << "Пусто" << endl;

for (auto iter = v.begin(); iter != v.end(); iter++)

cout << \*iter << " ";

cout << endl;

}

template<class T>

inline void VectorThird<T>::insert()

{

if (v.size() == 0)

throw exception("пустой вектор");

auto minIter = min\_element(v.begin(), v.end());

v.insert(\*minIter);

cout << "Успешно" << endl;

}

template<class T>

inline void VectorThird<T>::del()

{

if (v.size() == 0)

throw exception("пустой вектор");

T key;

cout << "Key? "; cin >> key;

auto findIter = find(v.begin(), v.end(), key);

if (findIter == v.end())

cout << "Нет данного значения" << endl;

else {

v.erase(findIter);

cout << "Успешно" << endl;

}

}

template<class T>

inline void VectorThird<T>::upd()

{

if (v.size() == 0)

throw exception("пустой вектор");

auto minIter = min\_element(v.begin(), v.end());

cout << " Минимальный = " << \*minIter;

auto maxIter = max\_element(v.begin(), v.end());

cout << " Максимальный = " << \*maxIter;

T dif = \*minIter + \*maxIter;

multiset<T> temp;

for (auto iter = v.begin(); iter != v.end(); iter++) {

temp.insert((\*iter) + dif);

}

v.clear();

v = temp;

cout << " Успешно" << endl;

}

void third\_exec() {

VectorThird<int> v;

v.make();

v.show("Исходный: ");

cout << "Добавить минимальный в конец: ";

v.insert();

v.show("После добавления: ");

cout << "Удалить с заданным ключом: ";

v.del();

v.show("После удаления: ");

cout << "Ко всем элементам добавить сумму минимального и максимального: ";

v.upd();

v.show("После изменения: ");

}

Определение главного файла

#include <iostream>

#include "first.h"

#include "second.h"

#include "third.h"

int main()

{

system("chcp 1251");

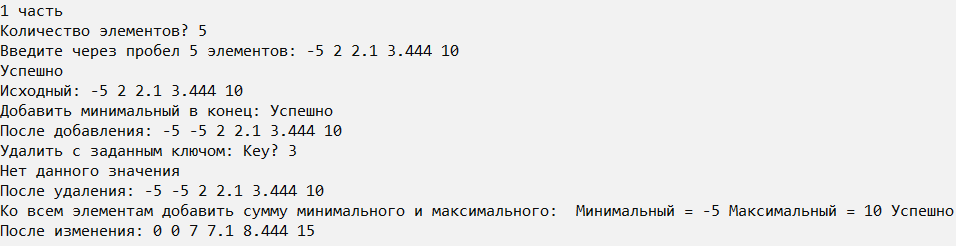
first\_exec();

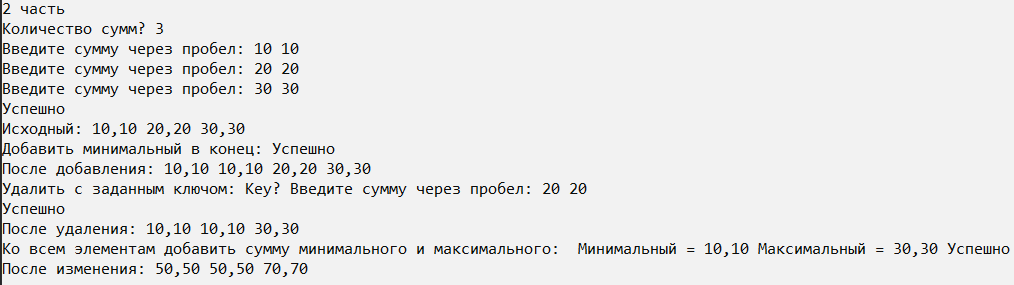
second\_exec();

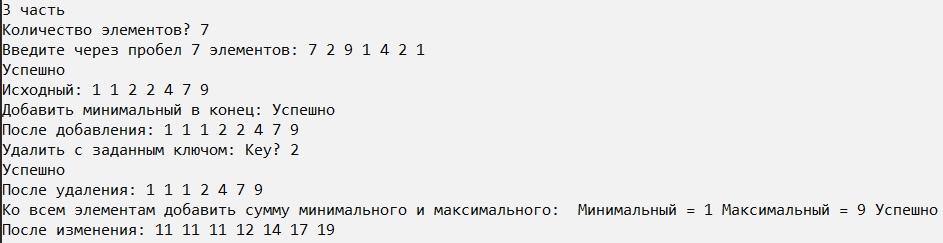
third\_exec();

}

Результаты работы программы







Ответы на контрольные вопросы:

1) Содержит пары значений «Ключ-значение»

2) map, multimap, set, multiset

3) [] – Доступ по ключу

4) begin, end, empty, insert, erase, clear, swap, find

5) map<K, V> v;

6) Отсортированы по ключу; Изменить параметр Comp в конструкторе

7) [], ==/!=, </>, >=/<=

8) void add(map<int, int>& dic, int k, int v) { dic.insert(make\_pair(k,v)); }

9) void add(map<int, int>& dic, int k, int v) { dic[k] = v; }

10) void print(map<int, int>& dic) { for(auto it = dic.begin(); it != dic.end(); it++) cout << it->first << it->second<<endl; }

11) void print(map<int, int>& dic) { for(auto it = dic.begin(); it != dic.end(); it++) cout << it->first << dic[it->first]<<endl;

12) По уникальности ключа, во втором с таким же ключом может быть несколько значений

13) Ассоциативный массив, в которых значения не играют роли, отслеживаются ключи

14) set хорошо подходит для пользовательского типа, в которых есть ключ как поле. В принципе никаких.

15) set<int> set2(set1); / int a[5] = {…}; set<int> set1(a, a+5);

16) Отсортированы по ключу. Изменить параметр конструктора Cmp

17) insert, erase, count, \*, begin, end …

18) insert(val);

19) for(auto it = s.begin(); it != s.end(); it++) cout<<\*it<<endl;

20) Во втором ключи могут повторяться