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 №13

Стандартные обобщенные алгоритмы 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

1.1. Создать последовательный контейнер

1.2. Создать адаптер контейнера

1.3. Создать ассоциативный контейнер

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

3. Заменить элементы в соответствии с заданием (replace\_if, replace\_copy, replace\_copy\_if, fill)

4. Удалить элементы в соответствии с заданием (remove, remove\_if, remove\_copy\_if, remove\_copy)

5. Отсортировать контейнер по убыванию и по возрастанию ключевого поля (sort)

6. Найти в контейнере элемент с заданным ключевым полем (find, find\_if, count, count\_if)

7. Выполнить задание варианта для полученного контейнера (for\_each)

8. Для выполнения всех заданий использовать стандартные алгоритмы библиотеки STL

Задание варианта

Задача 1

1. Контейнер – вектор

2. Тип элементов Money

Задача 2

Адаптер контейнера – очередь

Задача 3

Ассоциативный контейнер – словарь

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 класса

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);

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)

{

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 <vector>

#include <functional>

#include <algorithm>

#include "money.h"

using namespace std;

vector<money> first\_make() {

int n;

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

if (n < 1)

throw exception("неверное количество");

vector<money> temp;

money val;

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

cin >> val;

temp.push\_back(val);

}

cout << "Успешно" << endl;

return temp;

}

void first\_show(string text, vector<money>& v) {

// for\_each

cout << text;

if (v.size() == 0)

cout << "Пусто" << endl;

for\_each(v.begin(), v.end(), [] (const money&m) { cout << m << " "; });

cout << endl;

}

void first\_insert(vector<money>& v) {

// min\_element

if (v.size() == 0)

throw exception("пустой вектор");

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

v.push\_back(\*minIter);

cout << "Успешно" << endl;

}

void first\_delete(vector<money>& v) {

// remove

if (v.size() == 0)

throw exception("пустой вектор");

money key;

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

auto resIter = remove(v.begin(), v.end(), key);

if (resIter == v.end())

cout << "Нет данных значений" << endl;

else

cout << "Успешно" << endl;

}

void first\_update(vector<money>& v) {

// min\_element, max\_element, transform

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;

transform(v.begin(), v.end(), v.begin(), [dif](money& m) { return m + dif; });

cout << " Успешно" << endl;

}

void first\_sort(vector<money>& v) {

// sort

if (v.size() == 0)

throw exception("пустой вектор");

sort(v.begin(), v.end());

cout << "Успешно" << endl;

}

void first\_exec() {

vector<money> 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);

cout << "Сортировка по возрастанию: ";

first\_sort(v);

first\_show("После сортировки: ", v);

}

2 часть)

#pragma once

#include <iostream>

#include <queue>

#include <functional>

#include <algorithm>

#include "money.h"

using namespace std;

void second\_exec() {

cout << "Стандартные алгоритмы не работают с адаптерами контейнеров, так как они";

cout << " не имеют итераторов, значит их нельзя просто перебрать\n";

cout << "Один из вариантов решения: сформировать контейнеры из элементов адаптера";

cout << "и уже там выполнять алгоритмы, что аналогично 1 части\n";

}

3 часть)

#pragma once

#include <iostream>

#include <map>

#include <functional>

#include <algorithm>

#include "money.h"

using namespace std;

map<int, money> third\_make() {

int n;

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

if (n < 1)

throw exception("неверное количество");

map<int, money> temp;

money val;

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

cin >> val;

temp.insert(pair<int, money>(i + 1, val));

}

cout << "Успешно" << endl;

return temp;

}

void third\_show(string text, map<int, money>& v) {

// for\_each

cout << text;

if (v.size() == 0)

cout << "Пусто" << endl;

for\_each(v.begin(), v.end(), [](const pair<int, money>& m) { cout << m.second << " "; });

cout << endl;

}

void third\_insert(map<int, money>& v) {

// min\_element

if (v.size() == 0)

throw exception("пустой вектор");

auto minIter = min\_element(v.begin(), v.end(), [](const pair<int, money>& m1, const pair<int, money>& m2) {return m1.second < m2.second; });

v.insert(pair<int, money>(100, (\*minIter).second));

cout << "Успешно" << endl;

}

void third\_delete(map<int, money>& v) {

// remove

if (v.size() == 0)

throw exception("пустой вектор");

money key;

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

auto resIter = find\_if(v.begin(), v.end(), [&key] (const pair<int, money>& m) { return m.second == key; });

if (resIter == v.end())

cout << "Нет данных значений" << endl;

else {

v.erase(resIter);

cout << "Успешно" << endl;

}

}

void third\_update(map<int, money>& v) {

// min\_element, max\_element, transform

if (v.size() == 0)

throw exception("пустой вектор");

auto minIter = min\_element(v.begin(), v.end(), [](const pair<int, money>& m1, const pair<int, money>& m2) {return m1.second < m2.second; });

cout << " Минимальный = " << (\*minIter).second;

auto maxIter = max\_element(v.begin(), v.end(), [](const pair<int, money>& m1, const pair<int, money>& m2) {return m1.second < m2.second; });

cout << " Максимальный = " << (\*maxIter).second;

money dif = (\*minIter).second + (\*maxIter).second;

for\_each(v.begin(), v.end(), [&v, &dif](pair<int, money> m) { v[m.first] = m.second + dif; });

cout << " Успешно" << endl;

}

void third\_exec() {

map<int, money> v = third\_make();

third\_show("Исходный: ", v);

cout << "Добавить минимальный в конец: ";

third\_insert(v);

third\_show("После добавления: ", v);

cout << "Удалить с заданным ключом: ";

third\_delete(v);

third\_show("После удаления: ", v);

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

third\_update(v);

third\_show("После изменения: ", v);

cout << "Сортировка по возрастанию: ";

cout << "Нельзя словарь сортировать по значению, изначально сортируется по ключу\n";

}

Определение главного файла

#include <iostream>

#include "first.h"

#include "second.h"

#include "third.h"

int main()

{

system("chcp 1251");

cout << "\n1 часть\n"; first\_exec();

cout << "\n2 часть\n"; second\_exec();

cout << "\n3 часть\n"; third\_exec();

}

Результаты работы программы





