Министерство высшего образования и науки Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

**«Пермский национальный исследовательский политехнический университет» (ПНИПУ)**

Электротехнический факультет

Кафедра «Информационные технологии и автоматизированные системы»

ОТЧЁТ

Лабораторная работа №12

Ассоциативные контейнеры библиотеки STL.

Выполнил

Студент группы РИС-22-2б

Вековшинин Д. А.

Проверила

Доцент кафедры ИТАС

Полякова О. А.

Пермь 2023

Постановка задачи

Задача 1.

1. Создать ассоциативный контейнер.

2. Заполнить его элементами стандартного типа (тип указан в варианте).

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.

Задача 2.

1. Создать ассоциативный контейнер.

2. Заполнить его элементами пользовательского типа (тип указан в варианте). 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.

Задача 3

1. Создать параметризированный класс, используя в качестве контейнера

ассоциативный контейнер.

2. Заполнить его элементами.

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 методов параметризированного класса.

Вариант 10:

Задача 1

1. Контейнер - multimap

2. Тип элементов - float

Задача 2

Тип элементов Money (см. лабораторную работу №3).

Задача 3

Параметризированный класс – Вектор (см. лабораторную работу №7)

Задание 3

Найти минимальный элемент и добавить его на заданную позицию контейнера

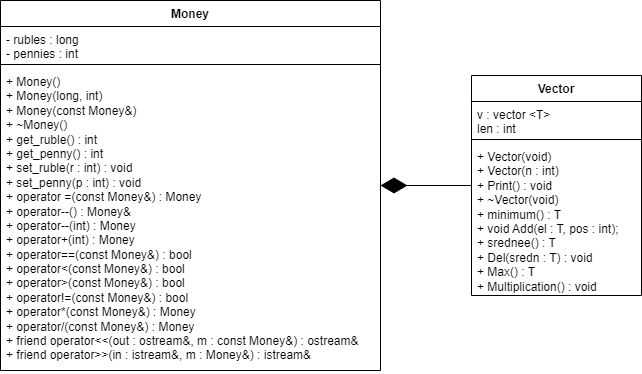
Задание 4

Найти элементы большие среднего арифметического и удалить их из контейнера

Задание 5

Каждый элемент домножить на максимальный элемент контейнера

UML-диаграмма



Код программы  
Задача 1

#include <iostream>

#include <map>

using namespace std;

typedef map<int, float>TMap;//определяем тип для работы со словарем

typedef TMap::iterator it;//итератор

//функция для формирования словаря

TMap make\_map(int n)

{

TMap m;//пустой словарь

float a;

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

{

cout << "?";

cin >> a;

//создаем пару и добавляем ее в словарь

m.insert(make\_pair(i, a));

}

return m;//возвращаем словарь как результат работы функции

}

//функция для печати словаря

void print\_map(TMap m)

{

for (int i = 0; i < m.size(); i++)

cout << i << " : " << m[i] << " " << endl;

}

//поиск минимального элемента

int Min(TMap v)

{

it i = v.begin();

int nom = 0,//номер максимального

k = 0;//счетчик элементов

float m = (\*i).second;//значение первого элемента

while (i != v.end())

{

if (m > (\*i).second)

{

m = (\*i).second;

nom = k;

}

i++;//итератор

k++;//счетчик элементов

}

return nom;//номер max

}

//вычисление среднего арифметического

float srednee(TMap m)

{

float s = 0;

//перебор словаря

for (int i = 0; i < m.size(); i++)

s += m[i];

int n = m.size();//количество элементов в словаре

return s / n;

}

TMap deleteSr(TMap m, float sr)

{

for (int i = 0; i < m.size(); i++)

{

if (m[i] > sr)

{

m.erase(i);

i--;

}

}

return m;

}

float Max(TMap v)

{

it i = v.begin();

float m = (\*i).second;//значение первого элемента

while (i != v.end())

{

if (m < (\*i).second)

{

m = (\*i).second;

}

i++;//итератор

}

return m;//max

}

void multi(TMap& v)

{

float m = Max(v);//значение максимального элемента

for (int i = 0; i < v.size(); i++)

v[i] = v[i] \* m;

}

//основная функция

void main()

{

int n, pos;

cout << "N? "; cin >> n;//количество элементов

TMap m = make\_map(n);//создать словарь

print\_map(m);//напечатать словарь

int min = Min(m);

cout << "min= " << m[min] << " nom=" << min << endl;

cout << "pos? "; cin >> pos;

m[pos] = m[min];

print\_map(m);

float sr = srednee(m);

cout << endl << "srednee: " << sr << endl;

m = deleteSr(m, sr);

print\_map(m);

float max = Max(m);

cout << "max= " << max << endl;

multi(m);

print\_map(m);

}

Задача 2

#include <iostream>

#include <map>

#include "Money.h"

using namespace std;

typedef map<int, Money>TMap;//определяем тип для работы со словарем

typedef TMap::iterator it;//итератор

//функция для формирования словаря

TMap make\_map(int n)

{

TMap m;//пустой словарь

Money a;

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

{

cin >> a;

//создаем пару и добавляем ее в словарь

m.insert(make\_pair(i, a));

}

return m;//возвращаем словарь как результат работы функции

}

//функция для печати словаря

void print\_map(TMap m)

{

for (int i = 0; i < m.size(); i++)

cout << i << " : " << m[i] << " " << endl;

}

//поиск минимального элемента

int Min(TMap v)

{

it i = v.begin();

int nom = 0,//номер максимального

k = 0;//счетчик элементов

Money m = (\*i).second;//значение первого элемента

while (i != v.end())

{

if (m > (\*i).second)

{

m = (\*i).second;

nom = k;

}

i++;//итератор

k++;//счетчик элементов

}

return nom;//номер max

}

//вычисление среднего арифметического

Money srednee(TMap m)

{

Money s = { 0,0 };

//перебор словаря

for (int i = 0; i < m.size(); i++)

s = s + m[i];

int n = m.size();//количество элементов в словаре

return s / n;

}

TMap deleteSr(TMap m, Money sr)

{

for (int i = 0; i < m.size(); i++)

{

if (m[i] > sr)

{

m.erase(i);

i--;

}

}

return m;

}

Money Max(TMap v)

{

it i = v.begin();

Money m = (\*i).second;//значение первого элемента

while (i != v.end())

{

if (m < (\*i).second)

{

m = (\*i).second;

}

i++;//итератор

}

return m;//max

}

void multi(TMap& v)

{

Money m = Max(v);//значение максимального элемента

for (int i = 0; i < v.size(); i++)

v[i] = v[i] \* m;

}

//основная функция

void main()

{

int n, pos;

cout << "N? "; cin >> n;//количество элементов

TMap m = make\_map(n);//создать словарь

print\_map(m);//напечатать словарь

int min = Min(m);

cout << "min= " << m[min] << " nom=" << min << endl;

cout << "pos? "; cin >> pos;

m[pos] = m[min];

print\_map(m);

Money sr = srednee(m);

cout << endl << "srednee: " << sr << endl;

m = deleteSr(m, sr);

print\_map(m);

Money max = Max(m);

cout << "max= " << max << endl;

multi(m);

print\_map(m);

}

Money.h

#pragma once

#include <iostream>

using namespace std;

class Money

{

long rubles;

int pennies;

public:

Money() { rubles = 0; pennies = 0; };

Money(int r, int p) { rubles = r; pennies = p; }

Money(const Money& m) { rubles = m.rubles; pennies = m.pennies; }

~Money() {}

int get\_ruble() { return rubles; }

int get\_penny() { return pennies; }

void set\_ruble(int r) { rubles = r; }

void set\_penny(int p) { pennies = p; }

//перегруженные операции

Money& operator=(const Money&);

Money& operator--();

Money operator--(int);//постфиксная операция

bool operator==(const Money&);

bool operator!=(const Money&);

Money operator+(const Money&);

Money operator\*(const Money&);

Money operator/(const int&);

bool operator >(const Money&);

bool operator <(const Money&);

//глобальные функции ввода-вывода

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

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

};

Money.cpp

#include "Money.h"

#include <iostream>

using namespace std;

//перегрузка операции присваивания

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

{

//проверка на самоприсваивание

if (&m == this) return \*this;

rubles = m.rubles;

pennies = m.pennies;

return \*this;

}

//перегрузка префиксной операции декремент

Money& Money::operator--()

{

int temp = rubles \* 100 + pennies;

temp--;

rubles = temp / 100;

pennies = temp % 100;

return \*this;

}

//перегрузка постфиксной операции декремент

Money Money::operator --(int)

{

int temp = rubles \* 100 + pennies;

temp--;

Money t(rubles, pennies);

rubles = temp / 100;

pennies = temp % 100;

return t;

}

//перегрузка бинарной операции сравнения

bool Money::operator==(const Money& t)

{

int temp1 = rubles \* 100 + pennies;

int temp2 = t.rubles \* 100 + t.pennies;

bool compare = (temp1 == temp2);

return compare;

}

bool Money::operator!=(const Money& t)

{

int temp1 = rubles \* 100 + pennies;

int temp2 = t.rubles \* 100 + t.pennies;

bool compare = (temp1 != temp2);

return compare;

}

//перегрузка глобальной функции-операции ввода

istream& operator>>(istream& in, Money& t)

{

cout << "rubles? "; in >> t.rubles;

cout << "pennies? "; in >> t.pennies;

return in;

}

//перегрузка глобальной функции-операции вывода

ostream& operator<<(ostream& out, const Money& t)

{

return (out << t.rubles << "," << t.pennies);

}

bool Money::operator <(const Money& t)

{

if (rubles < t.rubles)return true;

if (rubles == t.rubles && pennies < t.pennies)return true;

return false;

}

bool Money::operator >(const Money& t)

{

if (rubles > t.rubles)return true;

if (rubles == t.rubles && pennies > t.pennies)return true;

return false;

}

Money Money::operator+(const Money& t)

{

int temp1 = rubles \* 100 + pennies;

int temp2 = t.rubles \* 100 + t.pennies;

Money p;

p.rubles = (temp1 + temp2) / 100;

p.pennies = (temp1 + temp2) % 100;

return p;

}

//перегрузка бинарной операции умножения

Money Money::operator\*(const Money& t)

{

int temp1 = rubles \* 100 + pennies;

int temp2 = t.rubles \* 100 + t.pennies;

Money p;

p.rubles = (temp1 \* temp2) / 10000;

p.pennies = (temp1 \* temp2) % 10000;

return p;

}

Money Money::operator/(const int& t)

{

int temp1 = rubles \* 100 + pennies;

Money p;

p.rubles = (temp1 / t) / 100;

p.pennies = (temp1 / t) % 100;

return p;

}

Задача 3

#include "Money.h"

#include "Vector.h"

#include <map>

#include <iostream>

using namespace std;

void main()

{

int n, pos;

cout << "N? "; cin >> n;//количество элементов

Vector<Money> v(n);

v.Print();

Money min = v.minimum();

cout << "min= " << min << endl;

cout << "pos? "; cin >> pos;

v.Add(min, pos);

v.Print();

Money sr = v.srednee();

cout << endl << "srednee: " << sr << endl;

v.Del(sr);

v.Print();

Money max = v.Max();

cout << "max= " << max << endl;

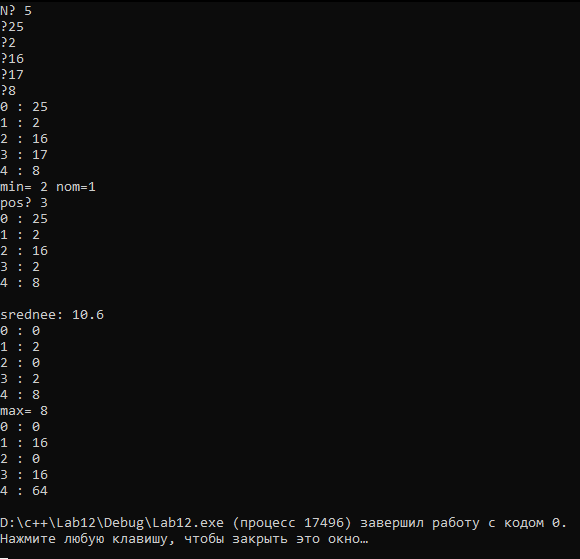
v.Multiplication();

v.Print();

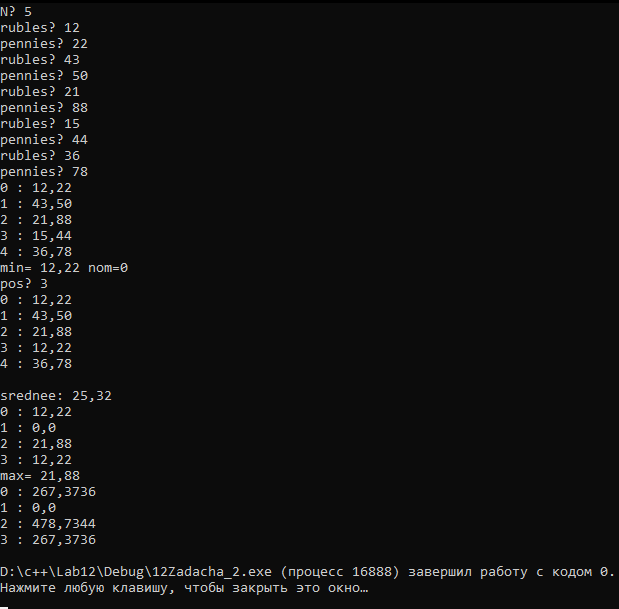
}

Результаты работы программы

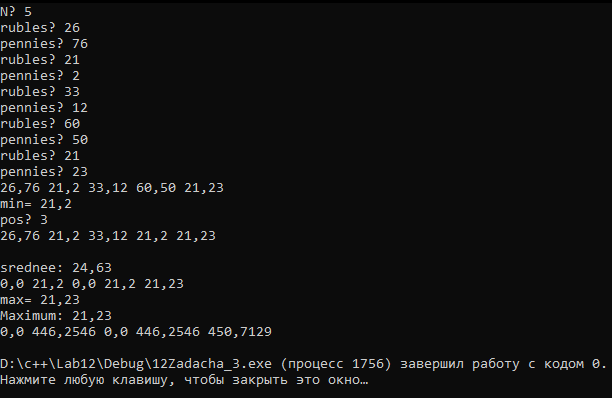
Задача 1



Задача 2



Задача 3



Ответы на контрольные вопросы

1. Что представляет собой ассоциативный контейнер?

Ассоциативный контейнер - это контейнер, который содержит упорядоченный набор элементов, представленных в форме пар ключ-значение. Элементы располагаются в контейнере в соответствии с заданным отношением порядка, определенным в компараторе. Ключи уникальны в контейнере, т.е. в контейнере не может быть двух элементов с одинаковым ключом. Основным преимуществом ассоциативных контейнеров является возможность быстрого поиска элемента по ключу благодаря использованию бинарного дерева поиска (например, красно-черного дерева) для хранения элементов. Примерами ассоциативных контейнеров в STL являются map и set.

1. Перечислить ассоциативные контейнеры в библиотеке STL.

* set - контейнер, хранящий уникальные элементы в отсортированном порядке. Все элементы в set уникальны, то есть не могут дублироваться.
* map - контейнер, хранящий пары ключ-значение, отсортированные по ключу. Ключи в map должны быть уникальными.
* multiset - контейнер, хранящий уникальные элементы в отсортированном порядке, но допускающий дублирование элементов.
* multimap - контейнер, хранящий пары ключ-значение, отсортированные по

1. Каким образом можно получить доступ к элементам ассоциативного контейнера?

Для доступа к элементам ассоциативного контейнера в STL используются итераторы. В отличие от последовательных контейнеров, в ассоциативных контейнерах элементы хранятся не в порядке их добавления, а в отсортированном порядке на основе ключа. Поэтому для доступа к элементам по индексу, как в последовательных контейнерах, в ассоциативных контейнерах используют итераторы. С помощью итераторов можно получить доступ к ключу и соответствующему ему значению элемента контейнера.

1. Привести примеры методов, используемых в ассоциативных контейнерах.

* insert(): добавляет элемент в контейнер.
* erase(): удаляет элемент из контейнера.
* find(): ищет элемент по заданному ключу.
* count(): возвращает количество элементов с заданным ключом в контейнере.
* size(): возвращает количество элементов в контейнере.
* empty(): возвращает значение true, если контейнер пуст, и false в противном случае.
* begin(): возвращает итератор, указывающий на первый элемент в контейнере.
* end(): возвращает итератор, указывающий на элемент следующий за последним \* элементом контейнера.
* lower\_bound(): возвращает итератор на первый элемент в контейнере, не меньший \* заданного ключа.
* upper\_bound(): возвращает итератор на первый элемент в контейнере, больший заданного ключа.
* equal\_range(): возвращает диапазон элементов в контейнере, соответствующих заданному ключу.

1. Каким образом можно создать контейнер map? Привести примеры.

* С помощью конструктора по умолчанию:

map<string, int> myMap;

В данном случае создается пустой контейнер map с ключами типа string и значениями типа int.

* С помощью списка инициализации:

map<string, int> myMap = {{"apple", 1}, {"banana", 2}, {"cherry", 3}};

В данном случае создается контейнер map с начальными значениями ключей и значений, которые передаются в список инициализации.

* С помощью пары итераторов:

map<std::string, int> myMap(anotherMap.begin(), anotherMap.end());

В данном случае создается контейнер map, который инициализируется парами ключ-значение из другого контейнера, заданного итераторами begin() и end().

* С помощью списка пар ключ-значение:

map<std::string, int> myMap = {make\_pair("apple", 1), make\_pair("banana", 2),make\_pair("cherry", 3)};

В данном случае создается контейнер map, который инициализируется парами ключ-значение из списка пар, созданных с помощью функции make\_pair().

1. Каким образом упорядочены элементы в контейнере map по умолчанию? Как изменить порядок на обратный?

Элементы отсортированы в порядке возрастания ключей.

Чтобы изменить порядок, можно определить пользовательскую функцию сравнения, которая будет сравнивать ключи в обратном порядке. Например:

bool compare(int a, int b) {

return a > b;

}

Затем, мы можем создать map следующим образом:

std::map<int, std::string, decltype(compare)\*> myMap(compare);

1. Какие операции определены для контейнера map?

Контейнер map поддерживает операции добавления и удаления элементов, поиска и доступа к элементам по ключу, а также проверки наличия элементов в контейнере. Кроме того, контейнер map поддерживает итераторы для обхода содержимого.

1. Написать функцию для добавления элеентов в контейнер map с помощью функции make\_pair().

template <class T1, class T2>

void addElements(std::map<T1, T2>& m, int n) {

T1 temp1 = 0;

T2 temp2;

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

{

cin >> temp2;

m.insert(make\_pair(temp1++, temp2));

}

}

1. Написать функцию для добавления элементов в контейнер map с помощь функции операци прямого доступпа [].

template <class T1, class T2>

void addElements(std::map<T1, T2>& m, T1 key, T2 value) {

m[key] = value;

}

1. Написать функцию для для печати контейнера map с помощью итератора.

template <class T1, class T2>

void printMap(map<T1, T2>& m) {

for (auto it = m.begin(); it != m.end(); ++it) {

cout << it->first << " : " << it->second << endl;

}

}

1. Написать функцию для для печати контейнера map с помощью функции операции прямого доступа [].

template <class T1, class T2>

void printMap(map<T1, T2>& m) {

for (const auto& p : m)

cout << p.first << " : " << m[p.first] << endl;

}

1. Чем отличаются контейнеры map и multimap?

Отличие между map и multimap заключается в том, что map хранит только уникальные ключи и соответствующие значения, тогда как multimap может хранить несколько значений для одного и того же ключа. Другими словами, map — это контейнер с уникальными ключами, а multimap - контейнер с неуникальными ключами.

1. Что представляет собой контейнер set?

Контейнер set - упорядоченное множество уникальных элементов. Он реализован в виде бинарного дерева поиска и обеспечивает быстрый доступ, вставку и удаление элементов в отсортированном порядке.

1. Чем отличаются контейнеры map и set?

Контейнер map предназначен для хранения пары "ключ-значение", где каждый ключ уникален, а контейнер set используется для хранения уникальных элементов, без пары "ключ-значение".

Таким образом, map используется для хранения и доступа к значению по ключу, а set используется для хранения элементов в отсортированном порядке и быстрого поиска элементов по значению.

1. Каким ообразом можно создать контейнер set? Привести примеры.

* Создание пустого контейнера с помощью конструктора по умолчанию:

set<int> mySet;

* Создание с заданными начальными значениями с помощью списка инициализации:

set<int> mySet = {1, 2, 3, 4};

* Создание с помощью диапазона значений другого контейнера:

vector<int> myVec = {1, 2, 3, 4};

set<int> mySet(myVec.begin(), myVec.end());

* Создание пустого контейнера с заданным компаратором:

struct Compare {

bool operator()(int a, int b) const {

return a > b;

}

};

set<int, Compare> mySet(Compare());

* Создание с заданными начальными значениями и компаратором:

struct Compare {

bool operator()(int a, int b) const {

return a > b;

}

};

set<int, Compare> mySet = {1, 2, 3, 4};

1. Каким образом упорядочены элементы в контейнере set по умолчанию? Как изменить порядок на обратный?

Элементы в контейнере set упорядочены по возрастанию. Чтобы изменить порядок на убывание, можно задать компаратор при создании контейнера, который будет сравнивать элементы в обратном порядке. Например:

#include <functional> // для std::greater

int main() {

set<int,greater<int>> s {5, 2, 7, 1, 8};

// элементы будут упорядочены в порядке убывания

return 0;

}

Здесь std::greater<int> - это функциональный объект, который сравнивает элементы в порядке убывания. Он передается вторым параметром шаблона контейнера set.

1. Какие методы определены для контейнера set?

* insert() - добавляет элемент в контейнер
* erase() - удаляет элемент из контейнера по значению или по итератору
* find() - ищет элемент в контейнере и возвращает итератор на найденный элемент, либо итератор на конец контейнера, если элемент не найден
* size() - возвращает количество элементов в контейнере
* empty() - возвращает true, если контейнер пуст, иначе – false
* clear() - удаляет все элементы из контейнера

1. Написать функцию для добавления элементов в контейнер set.

template <class T1>

void addElements(set<T1>& st, int n) {

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

st.insert(T1(rand()));

}

1. Написать функцию для печати контейнера set.

template <class T1>

void printSet(set<T1>& st) {

for (const auto& i : st)

cout << i << endl;

}

1. Чем отличается контейнер set и multiset?

Отличие между ними заключается в том, что set может хранить только уникальные элементы, а multiset может хранить несколько одинаковых элементов.