Министерство науки и высшего образования Российской Федерации Федеральное государственное бюджетное образовательное учреждение высшего образования

Пермский национальный исследовательский политехнический университет

Электротехнический факультет

Кафедра информационных технологий и автоматизированных систем

**ОТЧЕТ**

**о работе по информатике**

Семестр: 2

На тему: «ООП. Ассоциативные контейнеры библиотеки STL»

Выполнил студент ИВТ-22-2б:

Казанцев Антон Васильевич

(дата, подпись)

Проверила:

Полякова Ольга Андреевна

(дата, подпись)

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.

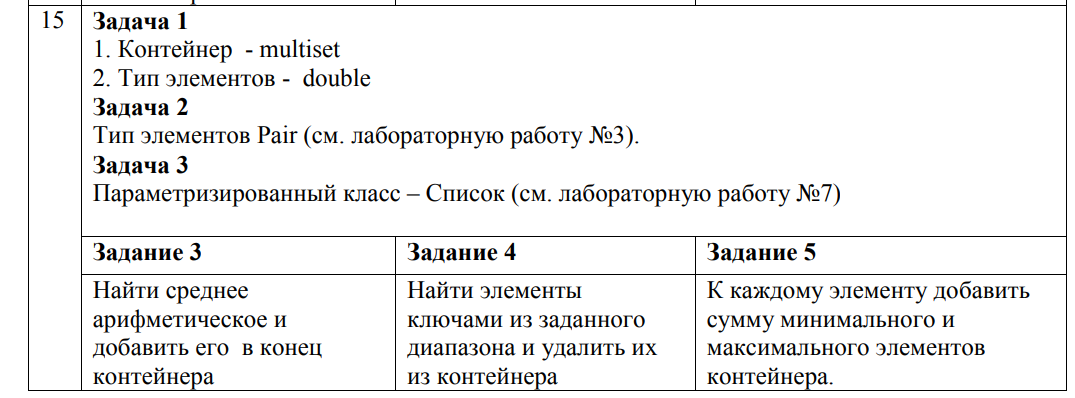
*Задача 2.*

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 15:*



**Программное решение**

#include <iostream>

#include <set>

using namespace std;

void PushArithmeticMiddle(multiset<double>& m\_multiset) {

if (m\_multiset.size() > 0) {

double Sum = 0;

for (auto i = m\_multiset.begin(); i != m\_multiset.end(); i++) {

Sum += \*i;

}

m\_multiset.insert(Sum / m\_multiset.size());

}

else {

cout << "Множество пусто!\n";

}

}

void EraseIfInRange(multiset<double>& m\_multiset, const double BeginValue, const double EndValue) {

auto i = m\_multiset.begin();

while (i != m\_multiset.end()) {

if (\*i >= BeginValue && \*i <= EndValue) {

i = m\_multiset.erase(i);

}

else {

i++;

}

}

}

void PlusMinAndMax(multiset<double>& m\_multiset) {

if (m\_multiset.size() > 0) {

multiset<double> tmp\_multiset = m\_multiset;

double MaxValue, MinValue;

MaxValue = MinValue = \*m\_multiset.begin();

for (auto i = m\_multiset.begin(); i != m\_multiset.end(); i++) {

if (\*i > MaxValue) {

MaxValue = \*i;

}

else if (\*i < MinValue) {

MinValue = \*i;

}

}

m\_multiset = {};

for (auto i = tmp\_multiset.begin(); i != tmp\_multiset.end(); i++) {

double NewValue = \*i + MaxValue + MinValue;

m\_multiset.insert(NewValue);

}

tmp\_multiset = {};

}

}

void ShowMultiset(const multiset<double>& m\_multiset) {

if (m\_multiset.size() > 0) {

for (auto i = m\_multiset.begin(); i != m\_multiset.end(); i++) {

cout << \*i << ' ';

}

}

else {

cout << "Множество пусто!";

}

cout << '\n';

}

int main() {

system("chcp 1251 > NULL");

multiset<double> a = { 1, 2, 3, 4, 5, 6, 7};

cout << "Множество с дубликатами а:\n";

ShowMultiset(a);

cout << "Добавляем среднее арифмитическое множестава а в контейнер\n";

PushArithmeticMiddle(a);

cout << "Множество а:\n";

ShowMultiset(a);

cout << "Удаляем из множества а элементы, значения которых входят в диапазон от 2 до 3\n";

EraseIfInRange(a, 2, 3);

cout << "Множество а:\n";

ShowMultiset(a);

cout << "К каждому элементу добавляем сумму минимального и максимального элементов контейнера\n";

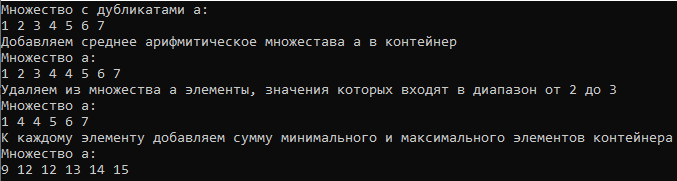
PlusMinAndMax(a);

cout << "Множество а:\n";

ShowMultiset(a);

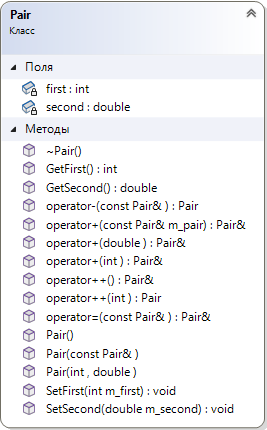
return 0;

}



**Задача 2:**

**UML-Диаграмма**



**Программное решение**

**Pair.h**

#pragma once

#include <iostream>

using namespace std;

class Pair {

int first;

double second;

public:

friend istream& operator>>(istream& in, Pair& p);

friend ostream& operator<<(ostream& out, const Pair& p);

Pair(int, double);

Pair();

Pair(const Pair&);

~Pair();

Pair& operator=(const Pair&);

Pair operator-(const Pair&);

Pair& operator+(int);

Pair& operator+(double);

Pair& operator++();

Pair operator++(int);

int GetFirst() const;

double GetSecond() const;

friend bool operator<(const Pair&, const Pair&);

void SetFirst(int m\_first);

void SetSecond(double m\_second);

Pair& operator + (const Pair& m\_pair);

};

**Pair.cpp**

#include "Pair.h"

Pair::Pair(int first, double second) {//конструктор с параметрами

this->first = first;

this->second = second;

}

istream& operator>>(istream& in, Pair& m\_pair) {//перегрузка оператора ввода

in >> m\_pair.first;

in >> m\_pair.second;

return in;

}

ostream& operator<<(ostream& out, const Pair& m\_pair) {//перегрузка оператора вывода

return (out << m\_pair.first << " : " << m\_pair.second);

}

Pair::Pair() {//конструктор без параметров

first = 0;

second = 0;

}

Pair::~Pair() {//деструктор

}

Pair::Pair(const Pair& m\_pair) {//конструктор копирования

first = m\_pair.first;

second = m\_pair.second;

}

Pair& Pair::operator=(const Pair& m\_pair) {//перегрузка оператора присваивания

if (&m\_pair != this) {

first = m\_pair.first;

second = m\_pair.second;

}

return \*this;

}

Pair Pair::operator-(const Pair& m\_pair) {//перегрузка оператора вычитания

Pair result(first - m\_pair.first, second - m\_pair.second);

return result;

}

Pair& Pair::operator+(int first) {//перегрузка оператора сложения - целый аргумент

this->first += first;

return \*this;

}

Pair& Pair::operator+(double second) {//перегрузка оператора сложения - дробный аргумент

this->second += second;

return \*this;

}

Pair& Pair::operator++() {//перегрузка оператора ++ префикс

++first;

++second;

return \*this;

}

Pair Pair::operator ++(int) {//перегрузка оператора ++ постфикс

Pair temp = \*this;

this->first++;

this->second++;

return temp;//сначала нужно вывести без изменений

}

int Pair::GetFirst() const{

return first;

}

double Pair::GetSecond() const{

return second;

}

bool operator < (const Pair& pair1, const Pair& pair) {

return (pair1.GetFirst() + pair1.GetSecond() < pair.GetFirst() + pair.GetSecond());

}

void Pair::SetFirst(int m\_first) {

first = m\_first;

}

void Pair::SetSecond(double m\_second) {

second = m\_second;

}

Pair& Pair::operator + (const Pair& m\_pair) {

first += m\_pair.first;

second += m\_pair.second;

return \*this;

}

**task2.cpp**

#include <iostream>

#include "Pair.h"

using namespace std;

#include <set>

void PushArithmeticMiddle(multiset<Pair>& m\_multiset) {

if (m\_multiset.size() > 0) {

int SumFirst = 0;

double SumSecond = 0;

for (auto i = m\_multiset.begin(); i != m\_multiset.end(); i++) {

SumFirst += (\*i).GetFirst();

SumSecond += (\*i).GetSecond();

}

m\_multiset.insert(Pair(SumFirst / m\_multiset.size(), SumSecond / m\_multiset.size()));

}

else {

cout << "Множество пусто!\n";

}

}

void EraseIfInRange(multiset<Pair>& m\_multiset, const double BeginValue, const double EndValue) {

auto i = m\_multiset.begin();

while (i != m\_multiset.end()) {

if (((\*i).GetFirst() >= BeginValue && (\*i).GetFirst() <= EndValue) || ((\*i).GetSecond() >= BeginValue && (\*i).GetSecond() <= EndValue)) {

i = m\_multiset.erase(i);

}

else {

i++;

}

}

}

void PlusMinAndMax(multiset<Pair>& m\_multiset) {

if (m\_multiset.size() > 0) {

multiset<Pair> tmp\_multiset;

Pair MaxValue = \*(m\_multiset.begin());

Pair MinValue = \*(m\_multiset.begin());

for (auto i = m\_multiset.begin(); i != m\_multiset.end(); i++) {

if (MaxValue < \*i) {

MaxValue = \*i;

}

else if (\*i < MinValue) {

MinValue = \*i;

}

}

for (auto i = m\_multiset.begin(); i != m\_multiset.end(); i++) {

Pair c = (\*i);

c + MaxValue + MinValue;

tmp\_multiset.insert(c);

}

m\_multiset = tmp\_multiset;

}

}

void ShowMultiset(const multiset<Pair>& m\_multiset) {

if (m\_multiset.size() > 0) {

for (auto i = m\_multiset.begin(); i != m\_multiset.end(); i++) {

cout << \*i << '\n';

}

}

else {

cout << "Множество пусто!";

}

cout << '\n';

}

int main() {

system("chcp 1251 > NULL");

multiset<Pair> a = { Pair(6, 4), Pair(7, 9), Pair(2, 3.5) };

cout << "Множество с дубликатами а:\n";

ShowMultiset(a);

cout << "Добавляем среднее арифмитическое множества а в контейнер\n";

PushArithmeticMiddle(a);

cout << "Множество а:\n";

ShowMultiset(a);

cout << "Удаляем из множества а элементы, значения которых входят в диапазон от 2 до 3\n";

EraseIfInRange(a, 2, 3);

cout << "Множество а:\n";

ShowMultiset(a);

cout << "К каждому элементу добавляем сумму минимального и максимального элементов контейнера\n";

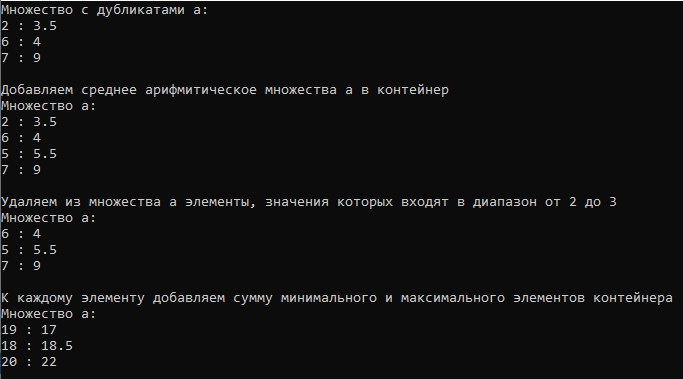
PlusMinAndMax(a);

cout << "Множество а:\n";

ShowMultiset(a);

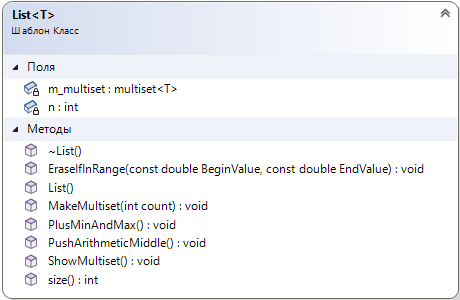
return 0;

}



**Задача 3:**

**UML-Диаграмма**



**Программное решение**

**List.h**

#pragma once

#include <iostream>

#include <set>

using namespace std;

template <typename T>

class List {

private:

multiset<T> m\_multiset;

int n = 0;

public:

List() {};

~List() {};

void MakeMultiset(int count) {

T a;

for (size\_t i = 0; i < count; i++) {

cout << "Введите " << i + 1 << " элемент: ";

cin >> a;

m\_multiset.insert(a);

}

n = count;

};

void ShowMultiset() {

if (m\_multiset.size() > 0) {

for (auto i = m\_multiset.begin(); i != m\_multiset.end(); i++) {

cout << \*i << ' ';

}

}

else {

cout << "Множество пусто!";

}

cout << '\n';

};

void PushArithmeticMiddle() {

if (m\_multiset.size() > 0) {

T Sum = 0;

for (auto i = m\_multiset.begin(); i != m\_multiset.end(); i++) {

Sum += \*i;

}

m\_multiset.insert(Sum / m\_multiset.size());

}

else {

cout << "Множество пусто!\n";

}

};

void EraseIfInRange(const double BeginValue, const double EndValue) {

auto i = m\_multiset.begin();

while (i != m\_multiset.end()) {

if (\*i >= BeginValue && \*i <= EndValue) {

i = m\_multiset.erase(i);

}

else {

i++;

}

}

};

int size() {

return n;

};

void PlusMinAndMax() {

if (m\_multiset.size() > 0) {

multiset<T> tmp\_multiset = m\_multiset;

T MaxValue, MinValue;

MaxValue = MinValue = \*m\_multiset.begin();

for (auto i = m\_multiset.begin(); i != m\_multiset.end(); i++) {

if (\*i > MaxValue) {

MaxValue = \*i;

}

else if (\*i < MinValue) {

MinValue = \*i;

}

}

m\_multiset = {};

for (auto i = tmp\_multiset.begin(); i != tmp\_multiset.end(); i++) {

T NewValue = \*i + MaxValue + MinValue;

m\_multiset.insert(NewValue);

}

tmp\_multiset = {};

}

}

};

**task3.cpp**

#include <iostream>

#include "List.h"

using namespace std;

int main() {

system("chcp 1251 > NULL");

List<double> a;

a.MakeMultiset(7);

cout << "Множество с дубликатами а:\n";

a.ShowMultiset();

cout << "Добавляем среднее арифмитическое множества a в контейнер\n";

a.PushArithmeticMiddle();

cout << "Множество а:\n";

a.ShowMultiset();

cout << "Удаляем из множества а элементы, значения которых входят в диапазон от 2 до 3\n";

a.EraseIfInRange(2, 3);

cout << "Множество а:\n";

a.ShowMultiset();

cout << "К каждому элементу добавляем сумму минимального и максимального элементов контейнера\n";

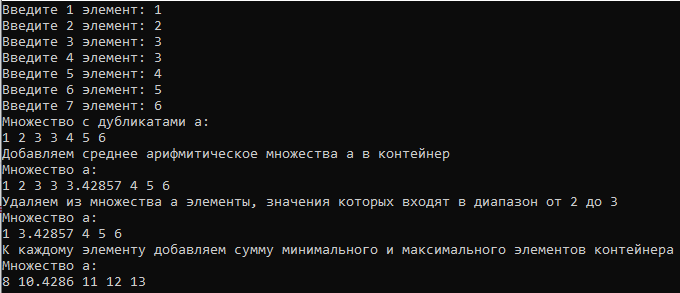
a.PlusMinAndMax();

cout << "Множество а:\n";

a.ShowMultiset();

return 0;

}



**Контрольные вопросы**

1. *Что представляет собой ассоциативный контейнер?*

Ассоциативный массив содержит пары значений. Зная одно значение,

называемое ключом (key), мы можем получить доступ к другому, называемому отображаемым значением (mapped value).

Ассоциативный массив можно представить как массив, для которого индекс не обязательно должен иметь целочисленный тип

1. *Перечислить ассоциативные контейнеры библиотеки STL.*

map - ассоциативный массив, по ключу в контейнере хранится одно значение

multimap - ассоциативный массив с повторяющимися ключами

set - массив уникальных ключей без значений

multiset - массив с повторяющимися ключами без значений

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

V& operator[](const K&) возвращает ссылку на элемент V, соответствующий значению K

1. *Привести примеры методов, используемых в ассоциативных контейнерах.*

bool empty() const, size\_type size() const, size\_type max\_size(), insert(), erase(), clear(), swap(), key\_comp(), value\_comp(), find(), count(), lower\_bound(), upper\_bound()

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

#include <iostream>

#include <map>

using namespace std;

int main() {

map<int, float> m;//словарь\

int n;//количество элементов

cout << "n: ";

cin >> n;

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 0;

}

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

Ассоциативный контейнер map требует, чтобы для типов ключа существовала операция “<”, то есть, элементы хранятся в порядке возрастания. Он хранит свои элементы отсортированными по ключу так, что перебор происходит по порядку.

Спецификация шаблона для класса map:

template <class Key, class T, class Comp = less <Key>, class Allocator = allocator <pair> >

class Comp = less <Key> - параметр, определяющий критерий

упорядочения, по умолчанию less (по возрастанию ключа).

Чтобы изменить критерий упорядочивания, нужно изменить параметр Сomp

1. *Какие операции определены для контейнера map?*

Определена операция присваивания: map& operator=(const map&);

Определены следующие операции: ==, <, <=, !=, >, >= и операция индексации ([])

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

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

cout <<" ?";

cin >> a;

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

}

1. *Написать функцию для добавления элементов в контейнер map с помощью функции операции прямого доступа [].*

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

cout <<" ?";

cin >> a;

m[i] = a;

}

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

for (map<int, float>::iterator i = m.begin(); i != m.end(); ++i) {

cout << (\*i).first << " " << (\*i).second << endl;

}

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

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

cout << m[i] << endl;

}

1. *Чем отличаются контейнеры map и multimap?*

Словари с дубликатами (multimap) допускают хранение элементов с одинаковыми ключами. Поэтому для них не определена операция доступа по индексу. Элементы с одинаковыми ключами хранятся в словаре в порядке их занесения. При удалении по ключу функция erase возвращает количество удаленных элементов. В остальном они аналогичны обычным словарям.

1. *Что представляет собой контейнер set?*

set — это контейнер, который автоматически сортирует добавляемые элементы в порядке возрастания. Но при добавлении одинаковых значений, set будет хранить только один его экземпляр. По-другому его еще называют множеством.

1. *Чем отличаются контейнеры map и set?*

Тип set позволяет хранить уникальные объекты различных типов, эффективно добавлять, удалять объекты и выполнять поиск. Тип map позволяет хранить пары ключ-значение, причем ключи должны быть уникальными.

1. *Каким образом можно создать контейнер set? Привести примеры.*

set<int> set1; // создается пустое множество

int а[5] = { 1. 2. 3. 4, 5};

set<int> set2(a, а + 5);// инициализация копированием

set<int> set3(set2); // инициализация другим множеством

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

Множество, как и словарь, требует, чтобы для типа T существовала операция “меньше” (<). Оно хранит свои элементы отсортированными, так что перебор происходит по порядку.

В множестве хранятся объекты, упорядоченные по некоторому ключу, являющемуся атрибутом самого объекта. Например, множество может хранить объекты класса Person, упорядоченные в алфавитном порядке по значению ключевого поля name. Если в множестве хранятся значения одного из встроенных типов, например int, то ключом является сам элемент.

set<int, greater<int>> set - изменение критерия упорядочения путем спецификации параметра компаратора

1. *Какие операции определены для контейнера set?*

Для вставки элементов в множество можно использовать метод insert(), для

удаления — метод erase(). Также к множествам применимы общие для всех контейнеров методы. Во всех ассоциативных контейнерах есть метод count(), возвращающий количество объектов с заданным ключом.

1. *Написать функцию для добавления элементов в контейнер set.*

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

cout <<" ? ";

cin >> a;

s.insert(a);

}

1. *Написать функцию для печати контейнера set.*

for(i = s.begin(); i != s.end(); ++i) cout<<\*i<<” ”;

1. *Чем отличаются контейнеры set и multiset?*

В множествах с дубликатами ключи могут повторяться. Элементы с одинаковыми ключами хранятся в множестве в порядке их занесения. Функция find() возвращает значение первого найденного элемента или end(), если ни одного элемента с заданным ключом не найдено.