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**

**Тема:** Лабораторная работа № 12 по ООП

Семестр: 2

Выполнил студент ИВТ-22-2б:

Солодов Александр Андреевич

Проверил доцент кафедры ИТАС:

Полякова Ольга Андреевна

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.

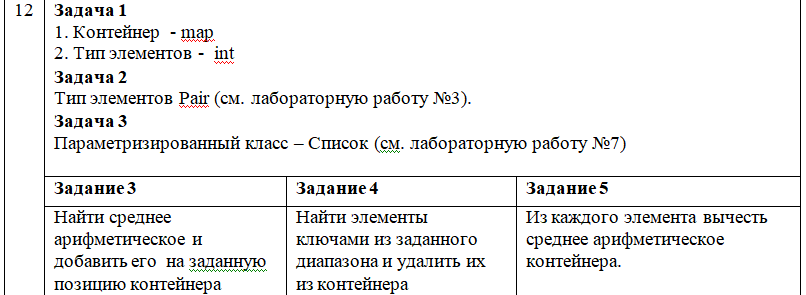
*Задача 2*

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*

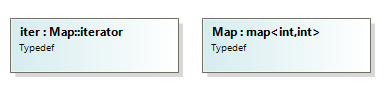
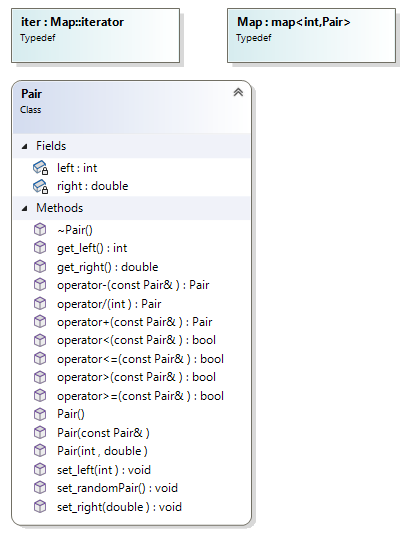
1. Создать параметризированный класс, используя в качестве контейнера ассоциативный контейнер.
2. Заполнить его элементами.
3. Добавить элементы в соответствии с заданием.
4. Удалить элементы в соответствии с заданием.
5. Выполнить задание варианта для полученного контейнера.
6. Выполнение всех заданий оформить в виде методов параметризированного класса.

*Вариант 12*

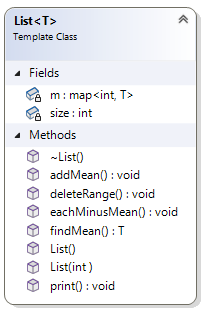
**

**UML-диаграммы**

*Задача 1*

**

*Задача 2*

*Задача 3*

**Программное решение**

*Задача 1*

main.cpp

/\* Вариант 12. Задача 1

1. Контейнер - map

2. Тип элементов - int

Задание 1: Найти среднее арифметическое и добавить его на заданную позицию контейнера

Задание 2: Найти элементы ключами из заданного диапазона и удалить их из контейнера

Задание 3: Из каждого элемента вычесть среднее арифметическое контейнера

\*/

#include <iostream>

#include <map>

#include <ctime>

using namespace std;

typedef map<int, int> Map;

typedef Map::iterator iter;

//создание словаря

Map makeMap(int size) {

Map m;

int data;

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

data = rand() % 20;

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

}

return m;

}

//вывод словаря в консоль

void printMap(Map m) {

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

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

}

}

//поиск среднего арифметического

int findMean(Map m) {

int sum = 0;

int count = 0;

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

sum = sum + it->second;

count++;

}

return sum / count;

}

//удаление элементов из указанного диапазона

void deleteRange(Map& m) {

int left, right;

cout << "Введите левую границу: ";

cin >> left;

cout << "Введите правую границу: ";

cin >> right;

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

if (left <= it->second && it->second <= right) {

m.erase(it++);

}

else {

++it;

}

}

}

//вычитание среднего арифметического из каждого элемента

void eachMinusMean(Map& m) {

int mean = findMean(m);

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

it->second = it->second - mean;

}

}

int main() {

setlocale(LC\_ALL, "RU");

srand(time(NULL));

Map map;

int size;

cout << "Введите размер словаря: ";

cin >> size;

map = makeMap(size);

cout << "Словарь: " << endl;

printMap(map);

int mean = findMean(map);

cout << "Среднее арифметическое: " << mean << endl;

map.insert(make\_pair(size, mean));

cout << "\nСловарь после добавления среднего арифметического: " << endl;

printMap(map);

cout << endl << "Укажите диапазон для удаления элементов: " << endl;

deleteRange(map);

cout << "Словарь после удаления: " << endl;

printMap(map);

cout << "Среднее арифметическое: " << findMean(map) << endl;

eachMinusMean(map);

cout << "\nСловарь после вычитания среднего арифметического из элементов: " << endl;

printMap(map);

return 0;

}

*Задача 2*

Pair.h

#pragma once

#include <iostream>

using namespace std;

class Pair {

private:

int left;

double right;

public:

Pair();

Pair(int, double);

Pair(const Pair&);

~Pair();

//геттеры

int get\_left();

double get\_right();

//сеттеры

void set\_left(int);

void set\_right(double);

void set\_randomPair();

//перегрузки операций ввода-вывода

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

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

//перегрузки операторов сравнения

bool operator < (const Pair&)const;

bool operator > (const Pair&)const;

bool operator <= (const Pair&)const;

bool operator >= (const Pair&)const;

//перегрузки арифметических действий с парами

Pair operator + (const Pair&);

Pair operator - (const Pair&);

Pair operator / (int);

};

Pair. cpp

#include "Pair.h"

//конструкторы

Pair::Pair() {

left = 0;

right = 0.0;

}

Pair::Pair(int l, double d) {

left = l;

right = d;

}

Pair::Pair(const Pair& p) {

left = p.left;

right = p.right;

}

//деструктор

Pair::~Pair() {}

//геттеры

int Pair::get\_left() {

return left;

}

double Pair::get\_right() {

return right;

}

//сеттеры

void Pair::set\_left(int l) {

left = l;

}

void Pair::set\_right(double r) {

right = r;

}

void Pair::set\_randomPair() {

left = rand() % 10;

right = (rand() % 100 - 1) / 10.0;

}

//перегрузка операции ввода

istream& operator >> (istream& in, Pair& p) {

cout << "Введите левое число: ";

in >> p.left;

cout << "Введите правое число: ";

in >> p.right;

return in;

}

//перегрузка операции вывода

ostream& operator << (ostream& out, const Pair& p) {

return (out << "[" << p.left << " : " << p.right << "]" << " ");

}

//перегрузка операций сравнения

bool Pair::operator < (const Pair& p)const {

return ((left < p.left) || (left == p.left && right < p.right));

}

bool Pair::operator > (const Pair& p)const {

return ((left > p.left) || (left == p.left && right > p.right));

}

bool Pair::operator <= (const Pair& p)const {

return (left <= p.left && right <= p.right);

}

bool Pair::operator >= (const Pair& p)const {

return (left >= p.left && right >= p.right);

}

//перегрузка операции сложения для двух пар

Pair Pair::operator + (const Pair& p) {

Pair temp;

temp.left = left + p.left;

temp.right = right + p.right;

return temp;

}

//перегрузка операции вычитания для двух пар

Pair Pair::operator - (const Pair& p) {

Pair temp;

temp.left = left - p.left;

temp.right = right - p.right;

return temp;

}

//перегрузка операции деления для пары

Pair Pair::operator / (int n) {

Pair temp;

temp.left = left / n;

temp.right = right / n;

return temp;

}

main.cpp

/\* Вариант 12. Задача 2

1. Контейнер - map

2. Тип элементов Pair (см. лабораторную работу №3).

Задание 1: Найти среднее арифметическое и добавить его на заданную позицию контейнера

Задание 2: Найти элементы ключами из заданного диапазона и удалить их из контейнера

Задание 3: Из каждого элемента вычесть среднее арифметическое контейнера

\*/

#include "Pair.h"

#include <ctime>

#include <map>

typedef map<int, Pair> Map;

typedef Map::iterator iter;

//создание словаря

Map makeMap(int size) {

Map m;

Pair p;

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

p.set\_randomPair();

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

}

return m;

}

//вывод словаря в консоль

void printMap(Map m) {

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

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

}

}

//поиск среднего арифметического

Pair findMean(Map m) {

Pair temp;

int intSum = 0;

double doubleSum = 0.0;

int count = 0;

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

intSum = intSum + it->second.get\_left();

doubleSum = doubleSum + it->second.get\_right();

count++;

}

temp.set\_left(intSum / count);

temp.set\_right(doubleSum / count);

return temp;

}

//удаление элементов из указанного диапазона

void deleteRange(Map& m) {

int left, right;

cout << "Введите левую границу целых чисел пары: ";

cin >> left;

cout << "Введите правую границу целых чисел пары: ";

cin >> right;

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

if (left <= it->second.get\_left() && it->second.get\_left() <= right) {

m.erase(it++);

}

else {

++it;

}

}

}

//вычитание среднего арифметического из каждого элемента

void eachMinusMean(Map& m) {

Pair mean = findMean(m);

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

it->second = it->second - mean;

}

}

int main() {

setlocale(LC\_ALL, "RU");

srand(time(NULL));

Map map;

int size;

cout << "Введите размер словаря: ";

cin >> size;

map = makeMap(size);

cout << "Словарь: " << endl;

printMap(map);

Pair mean = findMean(map);

cout << "Среднее арифметическое: " << mean << endl;

map.insert(make\_pair(size, mean));

cout << "\nСловарь после добавления среднего арифметического: " << endl;

printMap(map);

cout << endl << "Укажите диапазон для удаления элементов: " << endl;

deleteRange(map);

cout << "Словарь после удаления: " << endl;

printMap(map);

cout << "Среднее арифметическое: " << findMean(map) << endl;

eachMinusMean(map);

cout << "\nСловарь после вычитания среднего арифметического из элементов: " << endl;

printMap(map);

return 0;

}

*Задача 3*

List.h

#pragma once

#include <iostream>

#include <set>

#include <map>

using namespace std;

template <typename T>

class List

{

private:

map<int, T> m;

int size;

public:

List();

List(int);

~List();

void print();

T findMean();

void addMean();

void deleteRange();

void eachMinusMean();

};

//констуктор по умолчанию

template <typename T>

List<T>::List() {

size = 0;

}

//конструктор с параметрами

template <typename T>

List<T>::List(int s) {

T data;

size = s;

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

cout << "Введите значение: ";

cin >> data;

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

}

}

//деструктор

template <typename T>

List<T>::~List() {}

//вывод списка в консоль

template <typename T>

void List<T>::print() {

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

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

}

}

//поиск среднего арифметического

template <typename T>

T List<T>::findMean() {

T sum = 0;

int count = 0;

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

sum = sum + it->second;

count++;

}

T mean = sum / count;

return mean;

}

//добавление среднего арифметического в консоль

template <typename T>

void List<T>::addMean() {

T mean = findMean();

m.insert(make\_pair(size, mean));

}

//удаление элементов из указанного диапазона

template <typename T>

void List<T>::deleteRange() {

T left, right;

cout << "Введите левую границу диапазона: ";

cin >> left;

cout << "Введите правую границу диапазона: ";

cin >> right;

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

if (left <= it->second && it->second <= right) {

m.erase(it++);

}

else {

++it;

}

}

}

//вычитание среднего арифметического из каждого элемента

template <typename T>

void List<T>::eachMinusMean() {

T mean = findMean();

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

it->second = it->second - mean;

}

}

main.cpp

/\* Вариант 12. Задача 3

Контейнер - map

Тип элементов Pair (см. лабораторную работу №3)

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

Задание 1: Найти среднее арифметическое и добавить его на заданную позицию контейнера

Задание 2: Найти элементы ключами из заданного диапазона и удалить их из контейнера

Задание 3: Из каждого элемента вычесть среднее арифметическое контейнера

\*/

#include "List.h"

int main() {

setlocale(LC\_ALL, "RUS");

int size;

cout << "Введите размер: ";

cin >> size;

List<int> map(size);

cout << "Список: " << endl;

map.print();

int mean = map.findMean();

cout << "Среднее арифметическое: " << mean << endl;

map.addMean();

cout << "\nСписок после добавления среднего арифметического: " << endl;

map.print();

cout << endl << "Укажите диапазон для удаления элементов: " << endl;

map.deleteRange();

cout << "Список после удаления: " << endl;

map.print();

cout << "Среднее арифметическое: " << map.findMean() << endl;

map.eachMinusMean();

cout << "\nСловарь после вычитания среднего арифметического из элементов: " << endl;

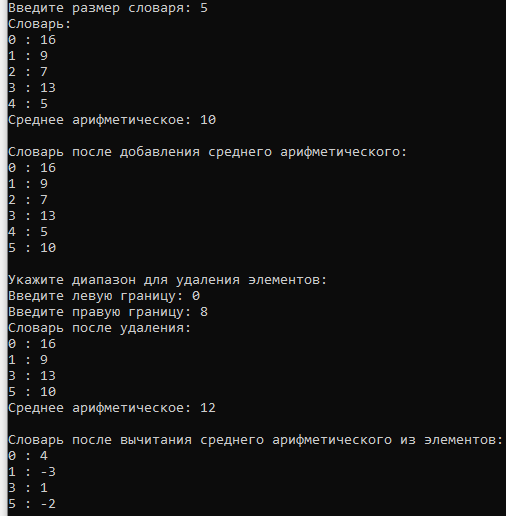
map.print();

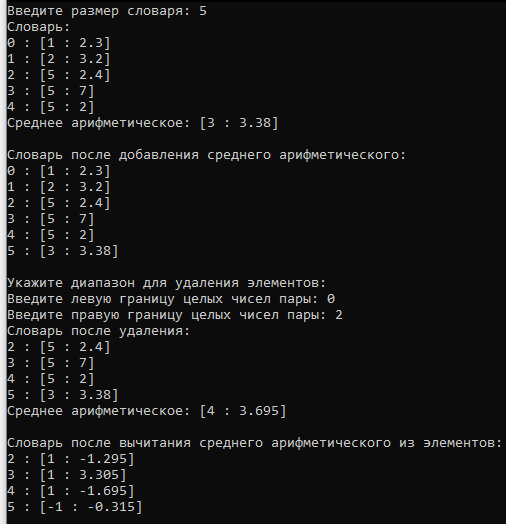
return 0;

}

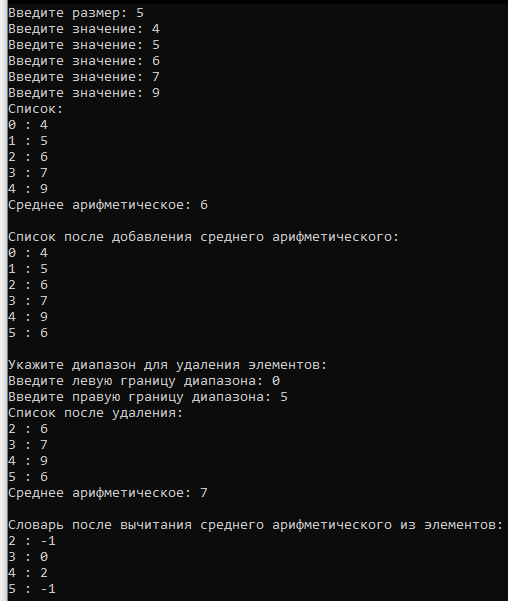
**Результат работы программы**

*Задача 1*



*Задача 2*

*Задача 3*



**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.