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 № 13 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 последовательный контейнер.
2. Заполнить его элементами пользовательского типа (тип указан в варианте). Для пользовательского типа перегрузить необходимые операции.
3. Заменить элементы в соответствии с заданием (использовать алгоритмы replace\_if(), replace\_copy(), replace\_copy\_if(), fill()).
4. Удалить элементы в соответствии с заданием (использовать алгоритмы remove(),remove\_if(), remove\_copy\_if(),remove\_copy()).
5. Отсортировать контейнер по убыванию и по возрастанию ключевого поля (использовать алгоритм sort()).
6. Найти в контейнере заданный элемент (использовать алгоритмы find(), find\_if(), count(), count\_if()).
7. Выполнить задание варианта для полученного контейнера (использовать алгоритм for\_each()) .
8. Для выполнения всех заданий использовать стандартные алгоритмы библиотеки STL.

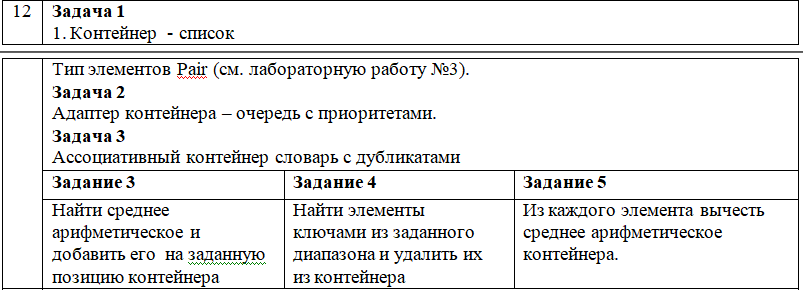
Задача 2.

1. Создать адаптер контейнера.
2. Заполнить его элементами пользовательского типа (тип указан в варианте). Для пользовательского типа перегрузить необходимые операции.
3. Заменить элементы в соответствии с заданием (использовать алгоритмы replace\_if(), replace\_copy(), replace\_copy\_if(), fill()).
4. Удалить элементы в соответствии с заданием (использовать алгоритмы remove(),remove\_if(), remove\_copy\_if(),remove\_copy()).
5. Отсортировать контейнер по убыванию и по возрастанию ключевого поля (использовать алгоритм sort()).
6. Найти в контейнере элемент с заданным ключевым полем (использовать алгоритмы find(), find\_if(), count(), count\_if()).
7. Выполнить задание варианта для полученного контейнера (использовать алгоритм for\_each()) .
8. Для выполнения всех заданий использовать стандартные алгоритмы библиотеки STL.

Задача 3

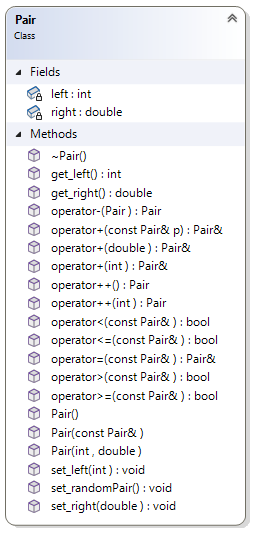
1. Создать ассоциативный контейнер.
2. Заполнить его элементами пользовательского типа (тип указан в варианте). Для пользовательского типа перегрузить необходимые операции.
3. Заменить элементы в соответствии с заданием (использовать алгоритмы replace\_if(), replace\_copy(), replace\_copy\_if(), fill()).
4. Удалить элементы в соответствии с заданием (использовать алгоритмы remove(),remove\_if(), remove\_copy\_if(),remove\_copy()).
5. Отсортировать контейнер по убыванию и по возрастанию ключевого поля (использовать алгоритм sort()).
6. Найти в контейнере элемент с заданным ключевым полем (использовать алгоритмы find(), find\_if(), count(), count\_if()).
7. Выполнить задание варианта для полученного контейнера (использовать алгоритм for\_each()).
8. Для выполнения всех заданий использовать стандартные алгоритмы библиотеки STL.

*Вариант 12*

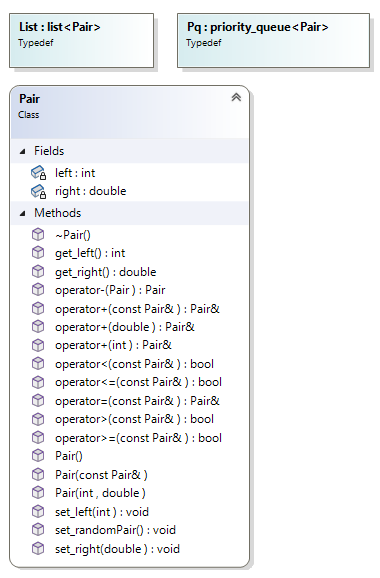
**

**UML-диаграммы**

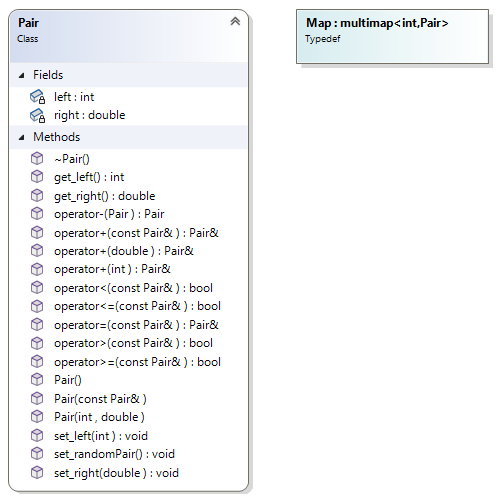
*Задача 1*



*Задача 2*



*Задача 3*

**

**Программное решение**

*Задача 1*

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&);

Pair operator - (Pair);

Pair& operator +(const Pair& p);

Pair operator ++ (int);

Pair operator ++ ();

Pair& operator+(int);

Pair& operator+(double);

Pair& operator=(const Pair&);

bool operator<(const Pair&)const;

bool operator>(const Pair&)const;

bool operator <=(const Pair&)const;

bool operator >=(const Pair&)const;

};

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 << "Левое число: ";

in >> p.left;

cout << "Правое число: ";

in >> p.right;

return in;

}

//перегрузка операции вывода

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

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

}

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

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

left = p.left;

right = p.right;

return \*this;

}

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

if (left != p.left) {

return left < p.left;

}

else {

return right < p.right;

}

}

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

if (left != p.left) {

return left > p.left;

}

else {

return right > p.right;

}

}

Pair Pair::operator++(int) {

right++;

return \*this;

}

Pair Pair::operator++() {

left++;

return \*this;

}

Pair Pair::operator-(Pair p) {

left -= p.left;

right -= p.right;

return \*this;

}

Pair& Pair::operator+(int first) {

this->left += left;

return \*this;

}

Pair& Pair::operator+(double second) {

this->right += right;

return \*this;

}

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

{

double sthis = this->left + this->right;

double sother = p.left + p.right;

if (sthis <= sother) return true;

return false;

}

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

{

double sthis = this->left + this->right;

double sother = p.left + p.right;

if (sthis >= sother) return true;

return false;

}

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

{

this->left += p.left;

this->right += p.right;

return \*this;

}

main.cpp

/\* Вариант 12. Задача 1.

1. Контейнер - список

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 <list>

#include <ctime>

#include <algorithm>

typedef list<Pair> List;

//создание списка

List makeList(int size) {

List temp;

Pair p;

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

p.set\_randomPair();

temp.push\_back(p);

}

return temp;

}

//вывод списка в консоль

void printList(List list) {

for (auto i : list) {

cout << i << endl;

}

}

//поиск среднего арифметического

Pair findMean(List list) {

Pair sum;

int count = 0;

for\_each(list.begin(), list.end(), [&sum] (Pair& p)

{

sum = sum + p;

}

);

sum.set\_left(sum.get\_left() / list.size());

sum.set\_right(sum.get\_right() / list.size());

return sum;

}

//добавление среднего арифметического в очередь

void addMean(List& list, int ind) {

Pair mean = findMean(list);

auto li = list.begin();

advance(li, ind);

list.insert(li, mean);

}

//удаление элементов из указанного диапазона

void deleteRange(List& list) {

Pair first, second;

cout << "\nВведите первую границу:\n";

cin >> first;

cout << "Введите вторую границу:\n";

cin >> second;

if (first > second) {

swap(first, second);

}

auto s = remove\_if(list.begin(), list.end(), [first, second] (Pair& p) {

return p >= first && p <= second;

}

);

list.erase(s, list.end());

}

//вычитание среднего арифметического из каждого элемента

void eachMinusMean(List& list) {

Pair mean = findMean(list);

for\_each(list.begin(), list.end(), [&mean] (Pair& p) {p = p - mean; });

}

int main() {

setlocale(LC\_ALL, "RU");

srand(time(NULL));

List list;

int size;

cout << "Введите количество элементов: ";

cin >> size;

list = makeList(size);

cout << "Cписок: " << endl;

printList(list);

cout << "Cреднее арифметическое: " << findMean(list) << endl;

int ind;

cout << "\nУкажите индекс для вставки среднего арифметического: ";

cin >> ind;

addMean(list, ind);

cout << "Cписок после вставки среднего арифметического: " << endl;

printList(list);

cout << "\nУдаление диапазона: ";

deleteRange(list);

cout << "\nCписок после удаления: " << endl;

printList(list);

cout << "Cреднее арифметическое: " << findMean(list) << endl;

eachMinusMean(list);

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

printList(list);

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);

//вспомогательные функции

void set\_randomPair();

//перегрузки операций ввода-вывода

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

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

//перегрузки арифметических операций

Pair operator - (Pair);

Pair& operator + (const Pair&);

Pair& operator + (int);

Pair& operator + (double);

//перегрузка операции присваивания

Pair& operator = (const Pair&);

//перегрузки операций сравнения

bool operator < (const Pair&)const;

bool operator > (const Pair&)const;

bool operator <= (const Pair&)const;

bool operator >= (const Pair&)const;

};

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;

}

//вспомогательные функции

void Pair::set\_randomPair() {

left = rand() % 10;

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

}

//перегрузка операции ввода

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

cout << "Левое число: ";

in >> p.left;

cout << "Правое число: ";

in >> p.right;

return in;

}

//перегрузка операции вывода

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

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

}

//перегрузки арифметических операций

Pair Pair::operator - (Pair p) {

left -= p.left;

right -= p.right;

return \*this;

}

Pair& Pair::operator + (int first) {

this->left += left;

return \*this;

}

Pair& Pair::operator + (double second) {

this->right += right;

return \*this;

}

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

{

this->left += p.left;

this->right += p.right;

return \*this;

}

//перегрузка операции присваивания

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

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

left = p.left;

right = p.right;

return \*this;

}

//перегрузки операций сравнения

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

if (left != p.left) {

return left < p.left;

}

else {

return right < p.right;

}

}

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

if (left != p.left) {

return left > p.left;

}

else {

return right > p.right;

}

}

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

{

double sthis = this->left + this->right;

double sother = p.left + p.right;

if (sthis <= sother) return true;

return false;

}

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

{

double sthis = this->left + this->right;

double sother = p.left + p.right;

if (sthis >= sother) return true;

return false;

}

main.cpp

/\* Вариант 12. Задача 2.

Контейнер - список

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

Адаптер контейнера – очередь с приоритетами.

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 <list>

#include <ctime>

#include <queue>

#include <algorithm>

typedef list<Pair> List;

typedef priority\_queue<Pair> Pq;

//создание очереди

Pq makePq(int size) {

Pq temp;

Pair p;

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

p.set\_randomPair();

temp.push(p);

}

return temp;

}

//копирование оереди в список

List copy\_pq\_to\_lst(Pq pq) {

List list;

while (!pq.empty()) {

list.push\_back(pq.top());

pq.pop();

}

return list;

}

//копирование списка в очередь

Pq copy\_lst\_to\_pq(List list) {

Pq pq;

while (!list.empty()) {

pq.push(list.back());

list.pop\_back();

}

return pq;

}

//вывод очереди в консоль

void print\_pq(Pq pq) {

while (!pq.empty()) {

cout << pq.top() << endl;

pq.pop();

}

cout << endl;

}

//поиск среднего арифметического

Pair findMean(Pq pq) {

List list = copy\_pq\_to\_lst(pq);

Pair sum;

int count = 0;

for\_each(list.begin(), list.end(), [&sum](Pair& p) {

sum = sum + p;

});

sum.set\_left(sum.get\_left() / list.size());

sum.set\_right(sum.get\_right() / list.size());

return sum;

}

//добавление среднего арифметического в очередь

void addMean(Pq& pq, int ind) {

Pair mean = findMean(pq);

List list = copy\_pq\_to\_lst(pq);

auto li = list.begin();

advance(li, ind);

list.insert(li, mean);

pq = copy\_lst\_to\_pq(list);

}

//удаление элементов из указанного диапазона

void deleteRange(Pq& pq) {

List list = copy\_pq\_to\_lst(pq);

Pair first, second;

cout << "\nВведите первую границу:\n";

cin >> first;

cout << "Введите вторую границу:\n";

cin >> second;

if (first > second) {

swap(first, second);

}

auto s = remove\_if(list.begin(), list.end(), [first, second](Pair& p) {

return p >= first && p <= second;

});

list.erase(s, list.end());

pq = copy\_lst\_to\_pq(list);

}

//вычитание среднего арифметического из каждого элемента

void eachMinusMean(Pq& pq) {

List list = copy\_pq\_to\_lst(pq);

Pair mean = findMean(pq);

for\_each(list.begin(), list.end(), [&mean](Pair& p) {p = p - mean; });

pq = copy\_lst\_to\_pq(list);

}

int main() {

setlocale(LC\_ALL, "RU");

srand(time(NULL));

Pq pq;

int size;

cout << "Введите количество элементов: ";

cin >> size;

pq = makePq(size);

cout << "Очередь: " << endl;

print\_pq(pq);

cout << "Cреднее арифметическое: " << findMean(pq) << endl;

addMean(pq, 0); //очередь с приоритетом автоматически расставляет элементы, поэтому позиция 0

cout << "\nОчередь после вставки среднего арифметического: " << endl;

print\_pq(pq);

cout << "Удаление диапазона: ";

deleteRange(pq);

cout << "\nОчередь после удаления: " << endl;

print\_pq(pq);

cout << "Cреднее арифметическое: " << findMean(pq) << endl;

eachMinusMean(pq);

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

print\_pq(pq);

return 0;

}

*Задача 3*

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);

//вспомогательные функции

void set\_randomPair();

//перегрузки операций ввода-вывода

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

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

//перегрузки арифметических операций

Pair operator - (Pair);

Pair& operator + (const Pair&);

Pair& operator + (int);

Pair& operator + (double);

//перегрузка операции присваивания

Pair& operator = (const Pair&);

//перегрузки операций сравнения

bool operator < (const Pair&)const;

bool operator > (const Pair&)const;

bool operator <= (const Pair&)const;

bool operator >= (const Pair&)const;

};

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;

}

//вспомогательные функции

void Pair::set\_randomPair() {

left = rand() % 10;

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

}

//перегрузка операции ввода

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

cout << "Левое число: ";

in >> p.left;

cout << "Правое число: ";

in >> p.right;

return in;

}

//перегрузка операции вывода

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

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

}

//перегрузки арифметических операций

Pair Pair::operator - (Pair p) {

left -= p.left;

right -= p.right;

return \*this;

}

Pair& Pair::operator + (int first) {

this->left += left;

return \*this;

}

Pair& Pair::operator + (double second) {

this->right += right;

return \*this;

}

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

{

this->left += p.left;

this->right += p.right;

return \*this;

}

//перегрузка операции присваивания

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

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

left = p.left;

right = p.right;

return \*this;

}

//перегрузки операций сравнения

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

if (left != p.left) {

return left < p.left;

}

else {

return right < p.right;

}

}

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

if (left != p.left) {

return left > p.left;

}

else {

return right > p.right;

}

}

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

{

double sthis = this->left + this->right;

double sother = p.left + p.right;

if (sthis <= sother) return true;

return false;

}

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

{

double sthis = this->left + this->right;

double sother = p.left + p.right;

if (sthis >= sother) return true;

return false;

}

main.cpp

/\* Вариант 12. Задача 3.

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

Ассоциативный контейнер словарь с дубликатами

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>

#include <algorithm>

typedef multimap<int, Pair> Map;

//создание словаря

Map makeMap(int size) {

Map temp;

Pair p;

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

p.set\_randomPair();

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

}

return temp;

}

//вывод словаря в консоль

void printMap(Map map) {

for (auto it : map) {

cout << it.first << " : " << it.second << endl;

}

}

//поиск среднего арифметического

Pair findMean(Map map) {

Pair sum;

int count = 0;

for (auto it : map) {

sum = sum + it.second;

count++;

}

Pair mean;

mean.set\_left(sum.get\_left() / count);

mean.set\_right(sum.get\_right() / count);

return mean;

}

//добавление среднего арифметического на заданную позицию

void addMean(Map& map, int pos) {

Pair mean = findMean(map);

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

}

//удаление элементов из указанного диапазона

void deleteRange(Map& map) {

int first, second;

cout << "\nВведите первую границу целого числа пары: ";

cin >> first;

cout << "Введите вторую границу целого числа пары: ";

cin >> second;

if (first > second) {

swap(first, second);

}

while (find\_if(map.begin(), map.end(), [&first, &second](decltype(\*map.begin())& it) -> bool {

return it.second.get\_left() >= first && it.second.get\_left() <= second;

}

) != map.end()) {

const auto it = find\_if(map.begin(), map.end(), [&first, &second](decltype(\*map.begin())& it) -> bool {

return it.second.get\_left() >= first && it.second.get\_left() <= second;

}

);

map.erase(it->first);

}

}

//вычитание среднего арифметического из каждого элемента

void eachMinusMean(Map& map) {

Pair mean = findMean(map);

for\_each(map.begin(), map.end(), [mean] (pair<const int, Pair>& p) { p.second = p.second - mean; });

}

int main() {

setlocale(LC\_ALL, "RU");

srand(time(NULL));

Map map;

int size;

cout << "Введите количество элементов: ";

cin >> size;

map = makeMap(size);

cout << "Словарь: " << endl;

printMap(map);

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

int ind;

cout << "\nВведите индекс для вставки среднего арифметического: ";

cin >> ind;

addMean(map, ind);

cout << "Словарь после вставки среднего арифметического: " << endl;

printMap(map);

cout << "\nУдаление диапазона: ";

deleteRange(map);

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

printMap(map);

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

eachMinusMean(map);

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

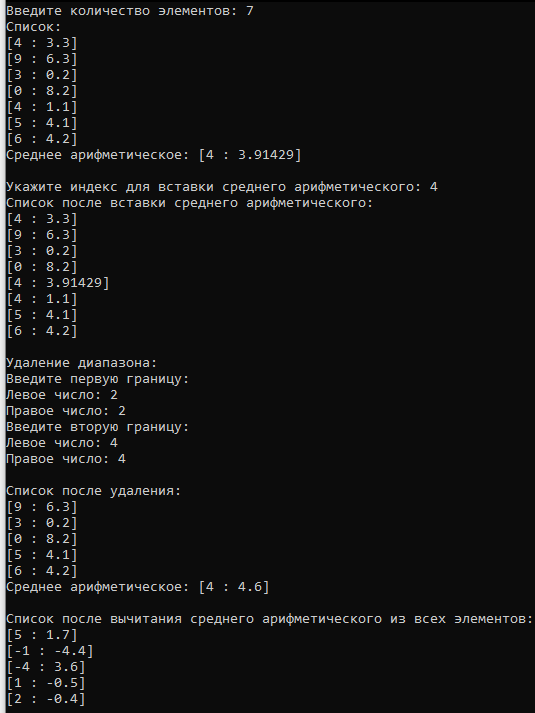
printMap(map);

return 0;

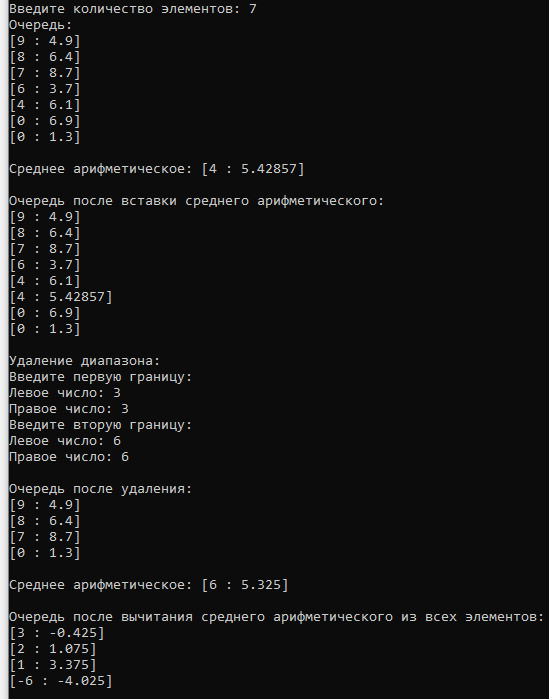
}

**Результат работы программы**

*Задача 1*

**

*Задача 2*

**

*Задача 3*

