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 № 11 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 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 последовательный 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 последовательный 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.

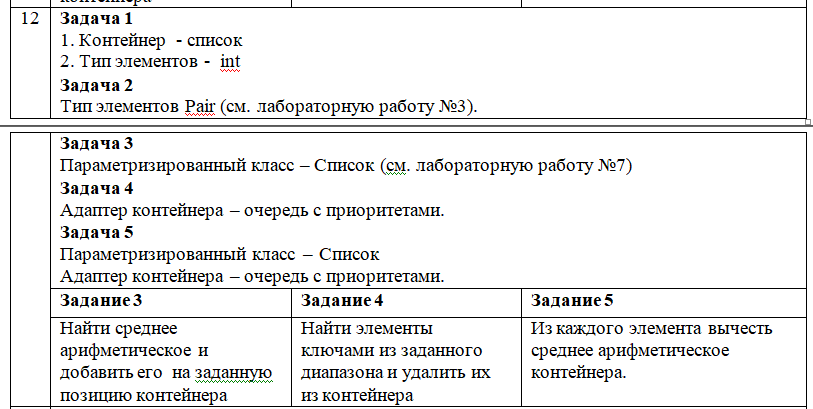
Задача 4.

1. Создать адаптер контейнера.
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 5.

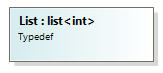
1. Создать параметризированный класс, используя в качестве контейнера адаптер контейнера.
2. Заполнить его элементами.
3. Добавить элементы в соответствии с заданием.
4. Удалить элементы в соответствии с заданием.
5. Выполнить задание варианта для полученного контейнера.
6. Выполнение всех заданий оформить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

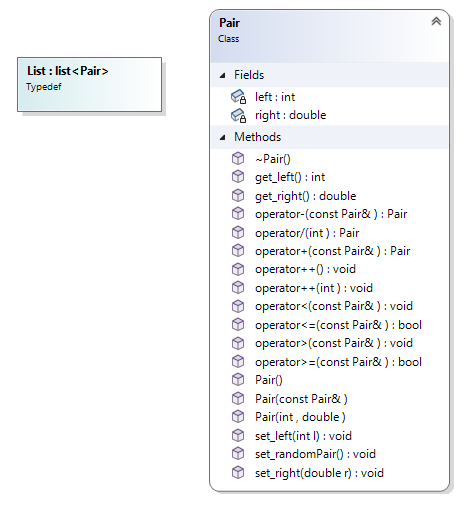
*Вариант 12*

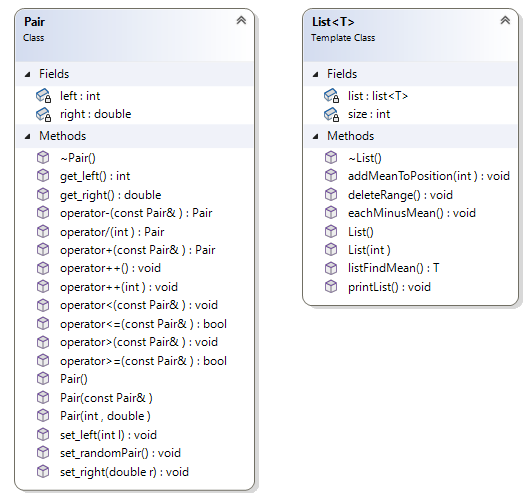


**UML-диаграммы**

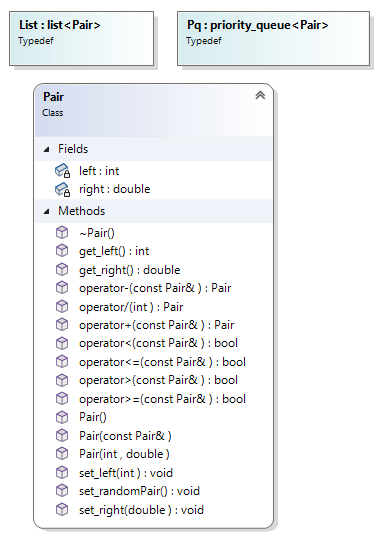
*Задача 1:*

****

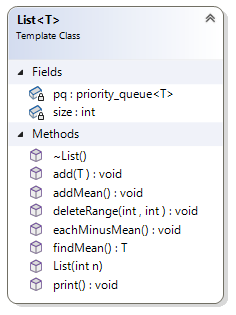
*Задача 2:*

*Задача 3:*

*Задача 4:*

**

*Задача 5:*

**

**Программное решение**

*Задача 1:*

main.cpp

/\* Вариант 12. Задача 1.

1. Контейнер - список

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

3адание 3: Из каждого элемента вычесть среднее арифметическое контейнера.

\*/

#include <iostream>

#include <list>

#include <ctime>

using namespace std;

typedef list<int> List;

List makeList(int size) {

List temp;

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

int e = (rand() % 20);

temp.push\_back(e);

}

return temp;

}

void printList(List& list) {

for (auto i = list.begin(); i != list.end(); ++i) {

cout << \*i << ' ';

}

cout << endl;

}

int listFindMean(List& list) {

int sum = 0;

for (auto i = list.begin(); i != list.end(); ++i) {

sum += \*i;

}

return sum / list.size();

}

void addMeanToPosition(List& list, int pos) {

int mean = listFindMean(list);

auto i = list.begin();

for (int j = 0; j < pos - 1; j++) {

i++;

}

list.insert(i, mean);

}

void deleteRange(List& list) {

int left, right;

cout << "Введите левую границу диапазона: ";

cin >> left;

cout << "Введите правую границу диапазона: ";

cin >> right;

for (auto i = list.begin(); i != list.end();) {

if (\*i >= left && \*i <= right)

i = list.erase(i);

else

++i;

}

}

void eachMinusMean(List& list) {

int mean = listFindMean(list);

for (auto i = list.begin(); i != list.end(); ++i) {

\*i = \*i - mean;

}

}

int main() {

setlocale(LC\_ALL, "RU");

srand(time(NULL));

List list;

int size;

cout << "Введите размер списка: ";

cin >> size;

list = makeList(size);

cout << "Cписок: ";

printList(list);

cout << "Среднее арифметическое = " << listFindMean(list) << endl;

int pos;

cout << endl << "Введите позицию для добавления среднего арифметического: ";

cin >> pos;

addMeanToPosition(list, pos);

cout << "Cписок после добавления: ";

printList(list);

cout << endl << "Удаление элементов из указанного диапазона:" << endl;

deleteRange(list);

cout << "Список после удаления элементов: ";

printList(list);

cout << "Среднее арифметическое = " << listFindMean(list) << endl;

eachMinusMean(list);

cout << endl << "Список после удаления среднего арифметического из каждого элемента: ";

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 l);

void set\_right(double r);

void set\_randomPair();

//перегрузки операций ввода-вывода

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

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

//перегрузки операторов сравнения

void operator < (const Pair&);

void operator > (const Pair&);

bool operator >= (const Pair&);

bool operator <= (const Pair&);

//перегрузка префиксного инкремента

void operator ++ ();

//перегрузка постфиксного инкремента

void operator ++ (int);

//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

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

if ((this->left < p.left) || (this->left == p.left && this->right < p.right)) {

cout << "Первая пара чисел меньше, чем вторая пара" << endl;

}

else {

cout << "Первая пара чисел НЕ меньше, чем вторая пара" << endl;

}

}

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

if ((this->left > p.left) || (this->left == p.left && this->right > p.right)) {

cout << "Первая пара чисел больше, чем вторая пара" << endl;

}

else {

cout << "Первая пара чисел НЕ больше, чем вторая пара" << endl;

}

}

void Pair::operator ++ () {

++left; }

void Pair::operator ++ (int) {

right += 1.0; }

//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;

}

//перегрузки операций сравнения

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

if (left >= p.left && right >= p.right)

return 1;

else

return 0;

}

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

if (left <= p.left && right <= p.right)

return 1;

else

return 0;

}

main.cpp

/\* Вариант 12. Задача 2.

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

3адание 3: Из каждого элемента вычесть среднее арифметическое контейнера.

\*/

#include "Pair.h"

#include <list>

#include <ctime>

typedef list<Pair> List;

List makeList(int size) {

List temp;

Pair p;

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

p.set\_randomPair();

temp.push\_back(p);

}

return temp;

}

void printList(List& list) {

for (auto i = list.begin(); i != list.end(); ++i) {

cout << \*i << ' ';

}

cout << endl;

}

Pair listFindMean(List& list) {

Pair sum;

for (auto i = list.begin(); i != list.end(); ++i) {

sum = sum + \*i;

}

return sum / list.size();

}

void addMeanToPosition(List& list, int pos) {

Pair mean = listFindMean(list);

auto i = list.begin();

for (int j = 0; j < pos - 1; j++) {

i++;

}

list.insert(i, mean);

}

void deleteRange(List& list) {

Pair left, right;

cout << "Введите левую границу диапазона: " << endl;

cin >> left;

cout << "Введите правую границу диапазона: " << endl;

cin >> right;

for (auto i = list.begin(); i != list.end();) {

if (\*i >= left && \*i <= right)

i = list.erase(i);

else

++i;

}

}

void eachMinusMean(List& list) {

Pair mean = listFindMean(list);

for (auto i = list.begin(); i != list.end(); ++i) {

\*i = \*i - mean;

}

}

int main() {

setlocale(LC\_ALL, "RU");

srand(time(NULL));

List list;

int size;

cout << "Введите размер списка: ";

cin >> size;

list = makeList(size);

cout << "Cписок: ";

printList(list);

cout << "Среднее арифметическое = " << listFindMean(list) << endl;

int pos;

cout << endl << "Введите позицию для добавления среднего арифметического: ";

cin >> pos;

addMeanToPosition(list, pos);

cout << "Cписок после добавления: ";

printList(list);

cout << endl << "Удаление элементов из указанного диапазона:" << endl;

deleteRange(list);

cout << "Список после удаления элементов: ";

printList(list);

cout << "Среднее арифметическое = " << listFindMean(list) << endl;

eachMinusMean(list);

cout << endl << "Список после удаления среднего арифметического из каждого элемента: " << endl;

printList(list);

return 0;

}

*Задача 3:*

List.h

#pragma once

#include <iostream>

#include <list>

using namespace std;

template <typename T>

class List

{

private:

list<T> list;

int size = list.size();

public:

List();

List(int);

~List();

void printList();

T listFindMean();

void addMeanToPosition(int);

void deleteRange();

void eachMinusMean();

};

List.cpp

#include "List.h"

template <typename T>

List<T>::List() {

size = 0;

}

template <typename T>

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

T elem;

size = s;

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

cin >> elem;

list.push\_back(elem);

}

}

template <typename T>

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

template <typename T>

void List<T>::printList() {

for (auto i = list.begin(); i != list.end(); i++) {

cout << \*i << ' ';

}

cout << endl;

}

template <typename T>

T List<T>::listFindMean() {

T sum;

for (auto i = list.begin(); i != list.end(); i++) {

sum = sum + \*i;

}

return sum / size;

}

template<typename T>

void List<T>::addMeanToPosition(int pos) {

T mean = listFindMean();

auto i = list.begin();

for (int j = 0; j < pos - 1; j++) {

i++;

}

list.insert(i, mean);

}

template <typename T>

void List<T>::deleteRange() {

T left, right;

cout << "Введите левую границу диапазона: " << endl;

cin >> left;

cout << "Введите правую границу диапазона: " << endl;

cin >> right;

for (auto i = list.begin(); i != list.end();) {

if (\*i >= left && \*i <= right)

i = list.erase(i);

else

i++;

}

}

template <typename T>

void List<T>::eachMinusMean() {

T mean = listFindMean();

for (auto i = list.begin(); i != list.end(); ++i) {

\*i = \*i - mean;

}

}

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 l);

void set\_right(double r);

void set\_randomPair();

//перегрузки операций ввода-вывода

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

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

//перегрузки операторов сравнения

void operator < (const Pair&);

void operator > (const Pair&);

bool operator >= (const Pair&);

bool operator <= (const Pair&);

//перегрузка префиксного инкремента

void operator ++ ();

//перегрузка постфиксного инкремента

void operator ++ (int);

//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

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

if ((this->left < p.left) || (this->left == p.left && this->right < p.right)) {

cout << "Первая пара чисел меньше, чем вторая пара" << endl;

}

else {

cout << "Первая пара чисел НЕ меньше, чем вторая пара" << endl;

}

}

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

if ((this->left > p.left) || (this->left == p.left && this->right > p.right)) {

cout << "Первая пара чисел больше, чем вторая пара" << endl;

}

else {

cout << "Первая пара чисел НЕ больше, чем вторая пара" << endl;

}

}

void Pair::operator ++ () {

++left;

}

void Pair::operator ++ (int) {

right += 1.0;

}

//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;

}

//перегрузки операций сравнения

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

if (left >= p.left && right >= p.right)

return 1;

else

return 0;

}

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

if (left <= p.left && right <= p.right)

return 1;

else

return 0;

}

main.cpp

/\* Вариант 12. Задача 3.

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

3адание 3: Из каждого элемента вычесть среднее арифметическое контейнера.

\*/

#include <iostream>

#include "Pair.h"

#include "List.h"

#include "List.cpp"

int main() {

setlocale(LC\_ALL, "rus");

int size;

cout << "Введите размер списка: ";

cin >> size;

cout << "Введите элементы списка через Enter:" << endl;

List<Pair> list(size);

cout << "Cписок: ";

list.printList();

cout << "Среднее арифметическое = " << list.listFindMean() << endl;

int pos;

cout << endl << "Введите позицию для добавления среднего арифметического: ";

cin >> pos;

list.addMeanToPosition(pos);

cout << "Cписок после добавления: ";

list.printList();

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

list.deleteRange();

cout << "Список после удаления элементов: ";

list.printList();

cout << "Среднее арифметическое = " << list.listFindMean() << endl;

list.eachMinusMean();

cout << endl << "Список после удаления среднего арифметического из каждого элемента: " << endl;

list.printList();

return 0;

}

*Задача 4:*

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 4.

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

3адание 3: Из каждого элемента вычесть среднее арифметическое контейнера.

\*/

#include "Pair.h"

#include <list>

#include <queue>

#include <ctime>

typedef priority\_queue<Pair> Pq;

typedef list<Pair> List;

Pq make\_pq(int size) {

Pq temp;

Pair p;

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

p.set\_randomPair();

temp.push(p);

}

return temp;

}

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

List list;

while (!pq.empty()) {

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

pq.pop();

}

return list;

}

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

Pq pq;

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

pq.push(\*it);

}

return pq;

}

void print\_pq(Pq pq) {

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

for (auto it = list.cbegin(); it != list.cend(); ++it) {

cout << \*it << " ";

}

cout << endl;

}

Pair findMean(Pq& pq) {

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

int count = 1;

Pair sum = pq.top();

pq.pop();

while (!pq.empty()) {

sum = sum + pq.top();

pq.pop();

count++;

}

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

return sum / count;

}

void addMeanToPosition(Pq& pq, int pos) {

Pair mean = findMean(pq);

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

auto i = list.begin();

for (int j = 0; j < pos - 1; j++) {

i++;

}

list.insert(i, mean);

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

}

void deleteRange(Pq& pq) {

Pair left, right;

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

cout << "Введите левую границу диапазона: " << endl;

cin >> left;

cout << "Введите правую границу диапазона: " << endl;

cin >> right;

for (auto i = list.begin(); i != list.end();) {

if (\*i >= left && \*i <= right)

i = list.erase(i);

else

i++;

}

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

}

void eachMinusMean(Pq& pq) {

Pair mean = findMean(pq);

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

for (auto i = list.begin(); i != list.end(); ++i) {

\*i = \*i - mean;

}

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

}

int main() {

setlocale(LC\_ALL, "RU");

srand(time(NULL));

Pq pq;

int size;

cout << "Введите размер очереди: ";

cin >> size;

pq = make\_pq(size);

cout << "Очередь: ";

print\_pq(pq);

cout << "Среднее арифметическое = " << findMean(pq) << endl;

addMeanToPosition(pq, 0); //позиция не указывается, потому что очередь с приоритетом

cout << endl << "Очередь после добавления: ";

print\_pq(pq);

cout << endl << "Удаление элементов из указанного диапазона:" << endl;

deleteRange(pq);

cout << "Очередь после удаления элементов: ";

print\_pq(pq);

cout << "Среднее арифметическое = " << findMean(pq) << endl;

eachMinusMean(pq);

cout << endl << "Очередь после удаления среднего арифметического из каждого элемента: " << endl;

print\_pq(pq);

return 0;

}

*Задача 5:*

List.h

#pragma once

#include <list>

#include <queue>

#include <iostream>

using namespace std;

template <typename T>

class List {

priority\_queue<T> pq;

int size = pq.size();

public:

List(int n);

~List() {};

void add(T);

void print();

T findMean();

void addMean();

void deleteRange(int, int);

void eachMinusMean();

};

template <typename T>

priority\_queue<T> copy\_list\_to\_pq(list<T> list) {

priority\_queue<T> pq;

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

pq.push(\*it);

}

return pq;

}

template <typename T>

list<T> copy\_pq\_to\_list(priority\_queue<T> pq) {

list<T> list;

while (!pq.empty()) {

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

pq.pop();

}

return list;

}

template <typename T>

void List<T>::add(T data) {

pq.push(data);

}

template <typename T>

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

T elem;

size = s;

cout << "Введите значения через Enter: \n";

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

cin >> elem;

this->add(elem);

}

}

template <typename T>

void List<T>::print() {

list<T> list = copy\_pq\_to\_list(pq);

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

cout << \*it << " ";

}

}

template <typename T>

T List<T>::findMean() {

T sum{};

list<T> list = copy\_pq\_to\_list(pq);

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

sum = sum + \*it;

}

T mean = sum / list.size();

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

cout << "\nСреднее арифметическое списка: " << mean << endl;

return mean;

}

template<typename T>

void List<T>::addMean() {

T mean = findMean();

this->add(mean);

}

template <typename T>

void List<T>::deleteRange(int left, int right) {

list<T> list = copy\_pq\_to\_list(pq);

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

if (\*it >= left && \*it <= right)

it = list.erase(it);

else

it++;

}

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

}

template <typename T>

void List<T>::eachMinusMean() {

T mean = findMean();

list<T> list = copy\_pq\_to\_list(pq);

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

\*it = \*it - mean;

}

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

}

main.cpp

/\* Вариант 12. Задача 5.

Параметризированный класс – Список

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

3адание 3: Из каждого элемента вычесть среднее арифметическое контейнера.

\*/

#include "List.h"

#include <iostream>

using namespace std;

int main() {

setlocale(LC\_ALL, "RU");

int size;

cout << "Введите размер списка: ";

cin >> size;

List<int> list(size);

cout << "Список: ";

list.print();

list.addMean();

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

list.print();

cout << "\n\nУдаление элементов, входящих в диапазон:";

int right, left;

cout << "\nЛевая граница: ";

cin >> left;

cout << "Правая граница: ";

cin >> right;

list.deleteRange(left, right);

cout << "Список после удаления: ";

list.print();

list.eachMinusMean();

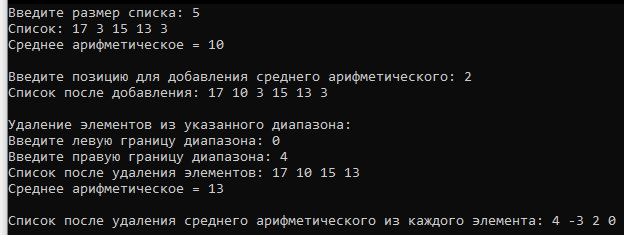
cout << "\nСписок после вычитания среднего арифметического: " << endl;

list.print();

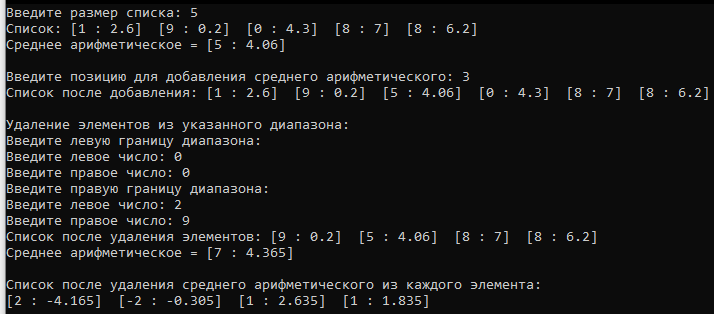
return 0;

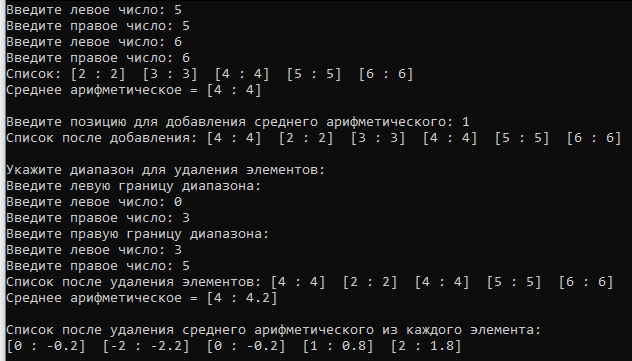
}

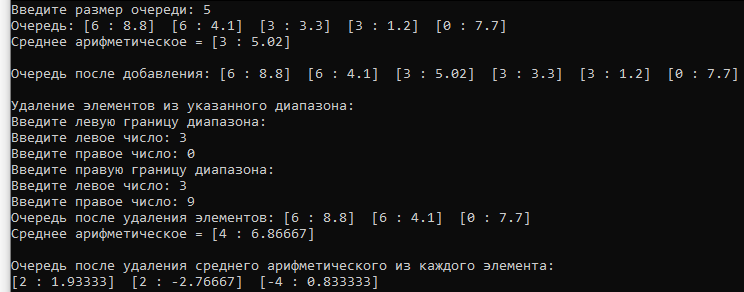
**Результат работы программы**

*Задача 1:*

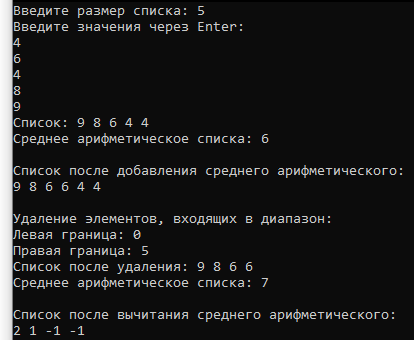
*Задача 2:*



*Задача 3:*

*Задача 4:*

*Задача 5:*

**

**Контрольные вопросы**

1. *Из каких частей состоит библиотека STL?*

STL – Standard Template Library, стандартная библиотека шаблонов состоит из двух основных частей: набора контейнерных классов и набора обобщенных алгоритмов. Контейнеры – это объекты, содержащие другие однотипные объекты. Обобщенные алгоритмы реализуют большое количество процедур, применимых к контейнерам: поиск, сортировку, слияние и т. п.

1. *Какие типы контейнеров существуют в STL?*

Последовательные контейнеры (векторы (vector), списки (list) и двусторонние очереди (deque)), ассоциативные контейнеры (словари (mар), словари с дубликатами (multimap), множества (set), множества с дубликатами (multiset) и битовые множества (bitset)), есть еще специализированные контейнеры (или адаптеры контейнеров), реализованные на основе базовых: стеки (stack), очереди (queue) и очереди с приоритетами (priority\_queue).

1. *Что нужно сделать для использования контейнера STL в своей программе?*

Для использования контейнера в программе необходимо включить в нее соответствующий заголовочный файл. Тип объектов, сохраняемых в контейнере, задается с помощью аргумента шаблона, например:

#include <vector>

#include <list>

1. *Что представляет собой итератор?*

Итераторы (iterators) – это объекты, которые по отношению к контейнеру играют роль указателей. Они позволяют получить доступ к содержимому контейнера примерно так же, как указатели используются для доступа к элементам массива.

1. *Какие операции можно выполнять над итераторами?*

С итераторами можно работать так же, как с указателями. К ним можно применить операции \*, инкремента, декремента. Присваивание одного итератора другому. Сравнение итераторов на равенство и неравенство (== и !=).

1. *Каким образом можно организовать цикл для перебора контейнера с использованием итератора?*

for (iterator it = v.begin(); it != v.end(), ++it) cout << v[it] << endl; //vector

1. *Какие типы итераторов существуют?*

* Итераторы ввода (input iterator) поддерживают операции равенства, разыменования и инкремента. ==, !=, \*i, ++i, i++, \*i++
* Итераторы вывода (output iterator) поддерживают операции разыменования, допустимые только с левой стороны присваивания, и инкремента. ++i, i++, \*i = t, \*i++ = t
* Однонаправленные итераторы (forward iterator) поддерживают все операции итераторов ввода/вывода и, кроме того, позволяют без ограничения применять присваивание. ==, !=, =, \*i, ++i, i++, \*i++
* Двунаправленные итераторы (bidirectional iterator) обладают всеми свойствами forward-итераторов, а также имеют дополнительную операцию декремента (--i, i--, \*i--), что позволяет им проходить контейнер в обоих направлениях.
* Итераторы произвольного доступа (random access iterator) обладают всеми свойствами bidirectional-итераторов, а также поддерживают операции сравнения и адресной арифметики, то есть непосредственный доступ по индексу. i += n, i + n, i -= n, i - n, i1 - i2, i[n], i1 < i2, i1 <= i2, i1 > i2, i1 >= i2

В STL также поддерживаются обратные итераторы (reverse iterators). Обратными итераторами могут быть либо двунаправленные итераторы, либо итераторы произвольного доступа, но проходящие последовательность в обратном направлении.

1. *Перечислить операции и методы общие для всех контейнеров.*

Операции равенства (==), неравенства (!=), операция присваивания (=), clear, insert, erase, size\_type size() const, size\_type max\_size() const, bool empty0 const, iterator begin(), iterator end(), reverse\_iterator begin(), reverse\_iterator end().

1. *Какие операции являются эффективными для контейнера vector? Почему?*

Контейнер вектор является аналогом обычного массива, за исключением того, что он автоматически выделяет и освобождает память по мере необходимости. Контейнер эффективно обрабатывает произвольную выборку элементов с помощью операции индексации [] или метода at. Происходит это потому, что массив - это последовательно занятая память, так что доступ к любому элементу происходит быстро.

1. *Какие операции являются эффективными для контейнера list? Почему?*

Контейнер список организует хранение объектов в виде двусвязного списка. Каждый элемент списка содержит три поля: значение элемента, указатель на предшествующий и указатель на последующий элементы списка. Вставка и удаление работают эффективно для любой позиции элемента в списке, поскольку не требуется заново перевыделять память, достаточно переобозначить связи с предыдущим и последующим элементом.

1. *Какие операции являются эффективными для контейнера deque? Почему?*

Контейнер двусторонняя очередь во многом аналогичен вектору, элементы хранятся в непрерывной области памяти. Но в отличие от вектора двусторонняя очередь эффективно поддерживает вставку и удаление первого элемента (так же, как и последнего). Дек является сочетанием вектора и списка, поэтому при добавлении элементов в начало создается новый массив, конец которого по принципа списка указывает на изначальный массив. Аналогично с добавлением элементов в конец.

1. *Перечислить методы, которые поддерживает последовательный контейнер vector.*

push\_back(), pop\_back(), insert, erase, [], at, swap, clear()

1. *Перечислить методы, которые поддерживает последовательный контейнер list.*

push\_back(), pop\_back(), push\_front(), pop\_back(), insert(), erase, swap, clear(), splice

1. *Перечислить методы, которые поддерживает последовательный контейнер deque.*

push\_back(), push\_front(), pop\_back, pop\_front, insert, erase, [] ,at

1. *Задан контейнер vector. Как удалить из него элементы со 2 по 5?*

for (int i = 2; i <= 5; ++i) v.erase(v.begin()+i);

1. *Задан контейнер vector. Как удалить из него последний элемент?*

v.erase(--v.end()); //итератор end() указывает на ячейку памяти после последнего элемента

1. *Задан контейнер list. Как удалить из него элементы со 2 по 5?*

list<int>::iterator beg = l.begin(); advance(beg, 2);

list<int>::iterator end = l.begin(); advance(end, 6); //удаляем включительно

l.erase(beg, end);

1. *Задан контейнер list. Как удалить из него последний элемент?*

l.erase(--l.end());

1. *Задан контейнер deque. Как удалить из него элементы со 2 по 5?*

auto beg = q.begin(); advance(beg, 2);

auto end = q.begin(); advance(end, 6);

q.erase(beg, end);

1. *Задан контейнер deque. Как удалить из него последний элемент?*

q.erase(--q.end());

1. *Написать функцию для печати последовательного контейнера с использованием итератора.*

void print(deque<int>q) {

for (deque<int>::iterator elem = q.begin(); elem != q.end(); ++elem)

cout << \*elem << " ";

}

1. *Что представляют собой адаптеры контейнеров?*

Специализированные последовательные контейнеры — стек, очередь и очередь с приоритетами — не являются самостоятельными контейнерными классами, а реализованы на основе рассмотренных выше классов, поэтому они называются адаптерами контейнеров.

1. *Чем отличаются друг от друга объявления stack<int> s и stack<int, list<int>> s?*

По умолчанию для стека прототипом является класс deque. Объявление stack<int> s создает стек на базе двусторонней очереди (по умолчанию). Если по каким-то причинам нас это не устраивает, и мы хотим создать стек на базе списка, то объявление будет выглядеть следующим образом: stack<int, list<int> > s;

1. *Перечислить методы, которые поддерживает контейнер stack.*

push () - добавление в конец; pop () - удаление из конца; top () - получение текущего элемента стека; empty() - проверка пустой стек или нет; size () – получение размера стека.

1. *Перечислить методы, которые поддерживает контейнер queue.*

push () – добавление в конец очереди; pop () – удаление из начала очереди; front () – получение первого элемента очереди; back()- получение последнего элемента очереди; empty () – проверка пустая очередь или нет; size() – получение размера очереди.

1. *Чем отличаются друг от друга контейнеры queue и priority\_queue?*

Шаблонный класс priority\_queue (заголовочный файл <queue>) поддерживает такие же операции, как и класс queue, но реализация класса возможна либо на основе вектора (реализация по умолчанию), либо на основе списка. Очередь с приоритетами отличается от обычной очереди тем, что для извлечения выбирается максимальный элемент из хранимых в контейнере. Поэтому после каждого изменения состояния очереди максимальный элемент из оставшихся сдвигается в начало контейнера.

1. *Задан контейнер stack. Как удалить из него элемент с заданным номером?*

stack<int> copy;

int counter = 0;

while (counter != 2) {

copy.push(s.top());

s.pop();

++counter;

}

while (counter != 6) { //удаление невключительно

s.pop();

++counter;}

while (!copy.empty()) {

s.push(copy.top());

copy.pop();}

1. *Задан контейнер queue. Как удалить из него элемент с заданным номером?*

//перемещение всех элементов до удаляемого в конец

for (int i = 1; i < num; i++) {

push(qwe, qwe->head->key);

pop(qwe); }

pop(qwe); //удаление элемента

//возвращение очереди в прежнее состояние

//первый элемент снова ставится в начало

for (int i = 0; i < (qwe->size) - (num + 2 \* k); i++) {

push(qwe, qwe->head->key);

pop(qwe); }

1. *Написать функцию для печати контейнера stack с использованием итератора.*

void print(stack<int> s) {

int size = s.size();

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

cout << s.top() << " ";

s.pop(); }

}

1. *Написать функцию для печати контейнера queue с использованием итератора.*

void print(queue<int> s) {

int size = s.size();

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

cout << s.front() << " ";

s.pop(); }

}