Министерство высшего образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

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

Электротехнический факультет

Кафедра «Информационные технологии и автоматизированные системы»

ОТЧЁТ

по лабораторной работе №12

Тема: «Ассоциативные контейнеры библиотеки STL.»

Выполнили:

Студенты группы РИС-22-1б

Анистратов А.Д.

Проверил доц. 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 15:

|  |  |  |
| --- | --- | --- |
| **Задача 1**   1. Контейнер - multiset 2. Тип элементов - double   **Задача 2**  Тип элементов Pair (см. лабораторную работу №3).  **Задача 3**  Параметризированный класс – Список (см. лабораторную работу №7) | | |
| **Задание 3** | **Задание 4** | **Задание 5** |
| Найти среднее арифметическое и добавить его в конец  контейнера | Найти элементы ключами из заданного диапазона и удалить их  из контейнера | К каждому элементу добавить сумму минимального и максимального элементов  контейнера. |

**Контрольные вопросы**

1. Что представляет собой ассоциативный контейнер?

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

1. Перечислить ассоциативные контейнеры в библиотеке STL.

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

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

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

1. Привести примеры методов, используемых в ассоциативных контейнерах.

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

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

* С помощью конструктора по умолчанию:

map<string, int> myMap;

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

* С помощью списка инициализации:

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

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

* С помощью пары итераторов:

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

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

* С помощью списка пар ключ-значение:

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

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

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

Элементы отсортированы в порядке возрастания ключей.

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

bool compare(int a, int b) {

return a > b;

}

Затем, мы можем создать map следующим образом:

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

1. Какие операции определены для контейнера map?

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

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

template <class T1, class T2>

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

T1 temp1 = 0;

T2 temp2;

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

{

cin >> temp2;

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

}

}

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

template <class T1, class T2>

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

m[key] = value;

}

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

template <class T1, class T2>

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

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

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

}

}

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

template <class T1, class T2>

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

for (const auto& p : m)

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

}

1. Чем отличаются контейнеры map и multimap?

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

1. Что представляет собой контейнер set?

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

1. Чем отличаются контейнеры map и set?

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

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

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

* Создание пустого контейнера с помощью конструктора по умолчанию:

set<int> mySet;

* Создание с заданными начальными значениями с помощью списка инициализации:

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

* Создание с помощью диапазона значений другого контейнера:

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

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

* Создание пустого контейнера с заданным компаратором:

struct Compare {

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

return a > b;

}

};

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

* Создание с заданными начальными значениями и компаратором:

struct Compare {

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

return a > b;

}

};

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

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

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

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

int main() {

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

// элементы будут упорядочены в порядке убывания

return 0;

}

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

1. Какие методы определены для контейнера set?

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

1. Написать функцию для добавления элементов в контейнер set.

template <class T1>

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

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

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

}

1. Написать функцию для печати контейнера set.

template <class T1>

void printSet(set<T1>& st) {

for (const auto& i : st)

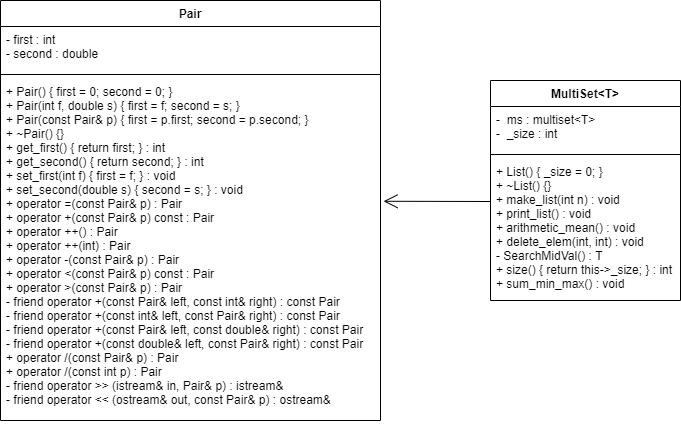
cout << i << endl;

}

1. Чем отличается контейнер set и multiset?

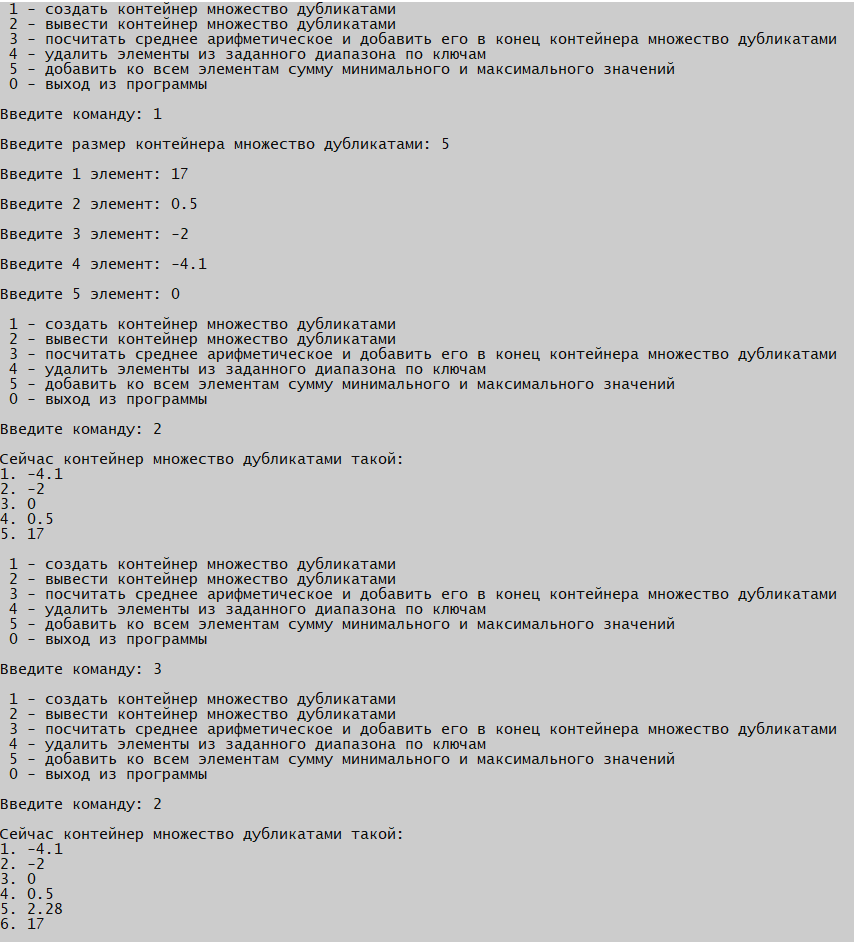
Отличие между ними заключается в том, что set может хранить только уникальные элементы, а multiset может хранить несколько одинаковых элементов.

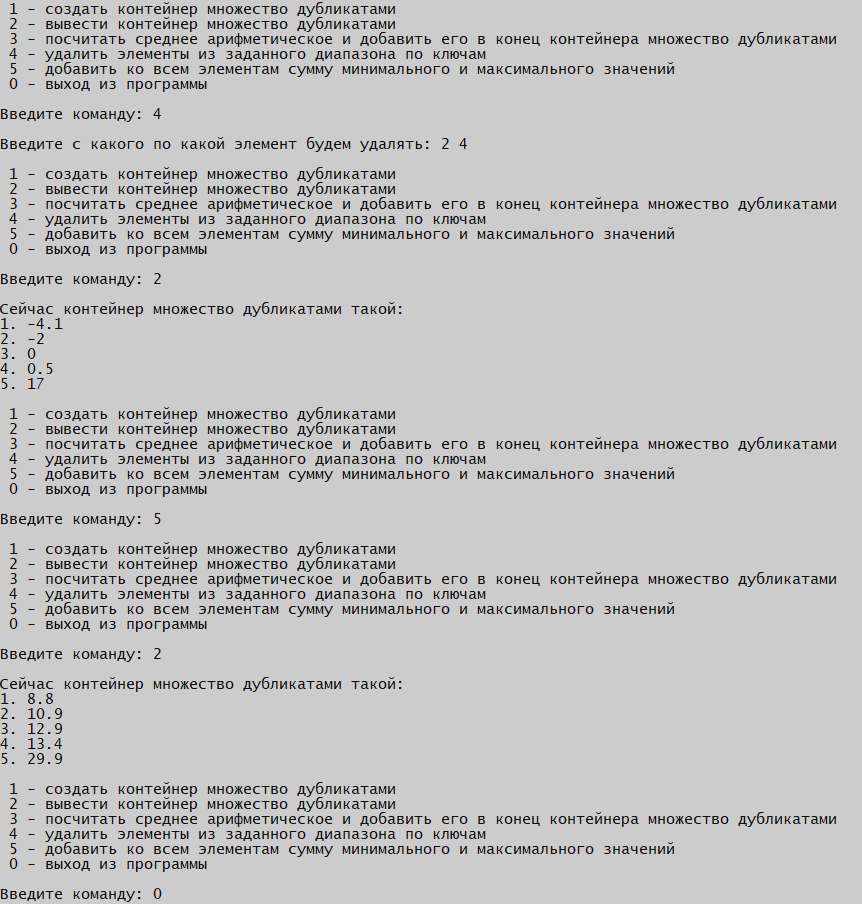
**UML таблица**

****

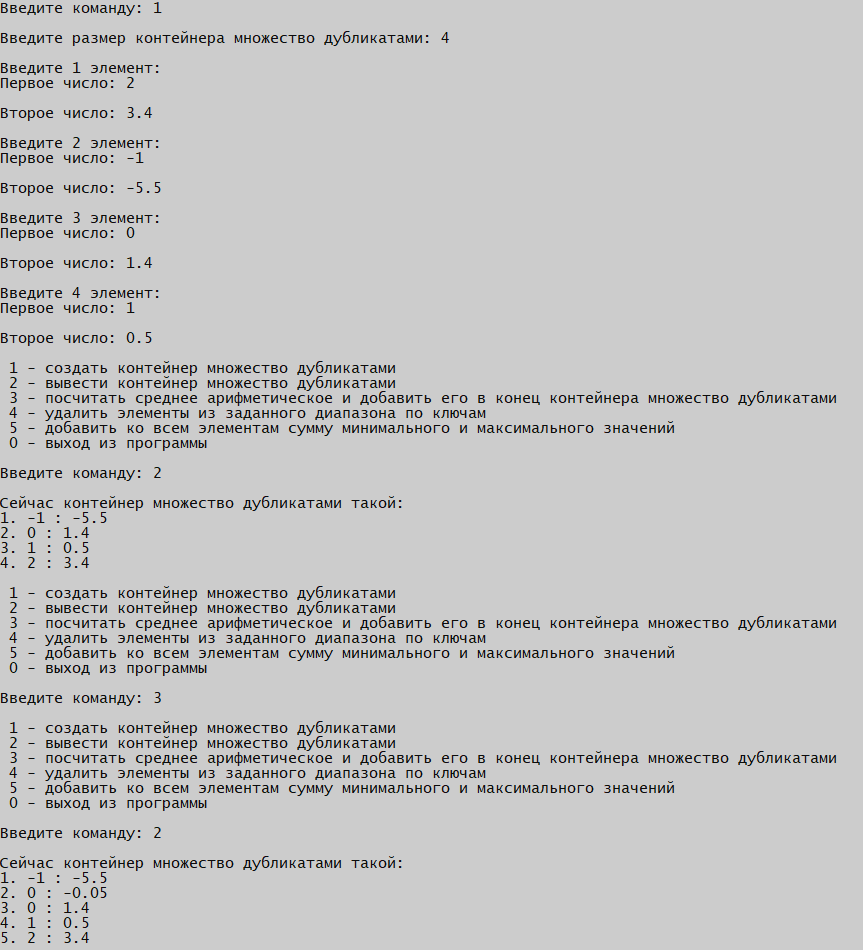
**Скриншоты работы программы**

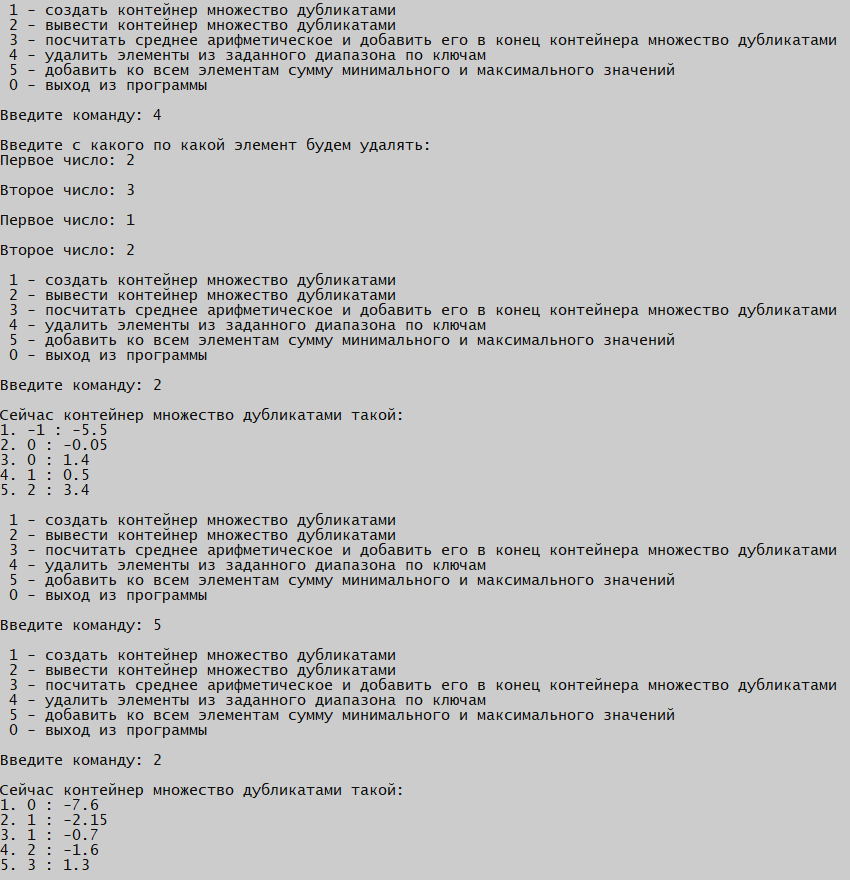
**Задача 1.**

****

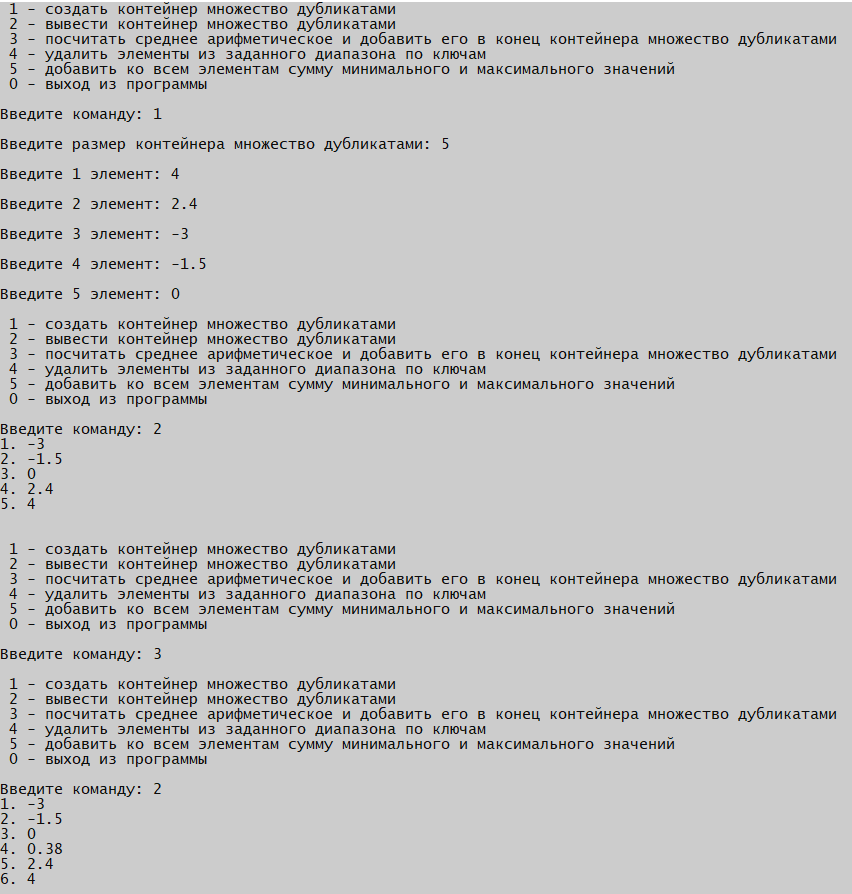
****

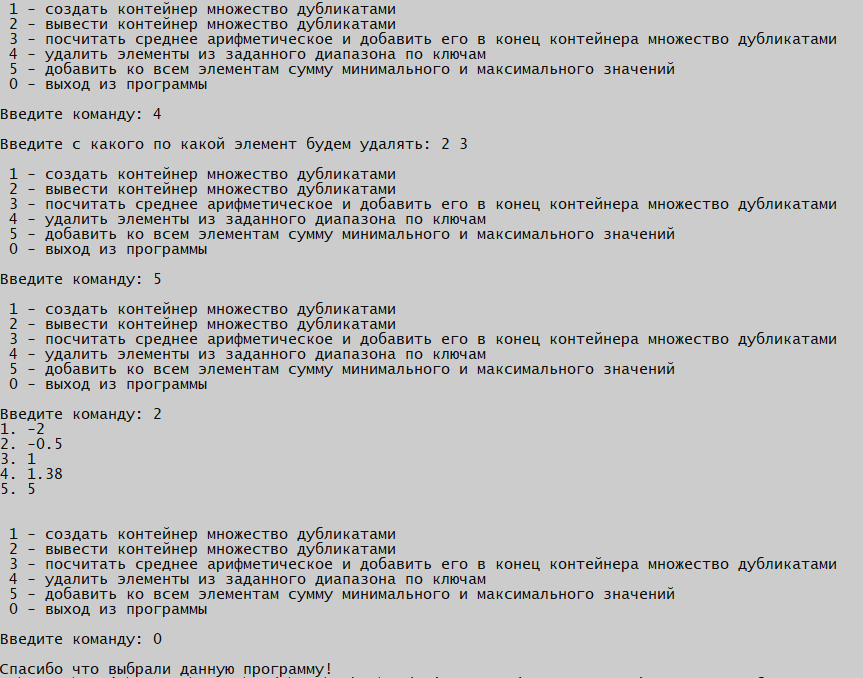
**Задача 2.**

****

****

**Задача 3.**

****

****

**Код программы**

**Задача 1.**

**Zadacha1\_main.cpp:**

#include <iostream>

#include <set>

#include <cstdlib>

#include <Windows.h>

using namespace std;

typedef multiset<double>MS; //определяем тип для работы с контейнером множество дубликатами

//функция для формирования контейнера множество дубликатами

MS make\_multiset(MS ms, int n)

{

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

{

double a;

cout << "\nВведите " << i + 1 << " элемент: ";

cin >> a;

ms.insert(a); //добавление a в конец контейнера множество дубликатами

}

return ms;

}

//функция для печати контейнера множество дубликатами

void print\_multiset(MS ms)

{

cout << "\nСейчас контейнер множество дубликатами такой: \n";

int k = 1;

for (auto& i : ms)

{

cout << k << ". " << i << endl;

k++;

}

}

//функция для подсчета среднего арифметического и добавление его в конец контейнера множество дубликатами

MS arithmetic\_mean(MS ms)

{

double a = 0;

for (auto& i : ms)

a += i;

ms.insert(a / ms.size());

return ms;

}

//функция удаления элементов из заданного диапазона по ключам

MS delete\_elem(MS ms, int start, int finish)

{

auto left = ms.lower\_bound(start);

auto right = ms.upper\_bound(finish);

ms.erase(left, right);

return ms;

}

//функция добавления суммы минимального и максимального значений

MS sum\_min\_max(MS ms)

{

multiset<double> temp; //создание промежуточного пустого контейнера

double max = 0;//здесь будет максимальное значение в контейнере множество дубликатами

//цикл нахождения максимального значения

for (auto& i : ms)

{

if (max < i)

max = i;

}

double min = 0;; //здесь будет минимальное значение в контейнере множество дубликатами

//цикл нахождения минимального значения

for (auto& i : ms)

{

if (min > i)

min = i;

}

double s = min + max; //сумма минимального и максимального элементов

//добавление к элементам s

for (auto& i : ms)

temp.insert(i + min + max);

ms.swap(temp);

return ms;

}

const char menu[] = " 1 - создать контейнер множество дубликатами\n"

" 2 - вывести контейнер множество дубликатами\n"

" 3 - посчитать среднее арифметическое и добавить его в конец контейнера множество дубликатами\n"

" 4 - удалить элементы из заданного диапазона по ключам\n"

" 5 - добавить ко всем элементам сумму минимального и максимального значений\n"

" 0 - выход из программы\n";

//основная функция

int main()

{

setlocale(LC\_ALL, "Russian");

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

try

{

multiset<double> ms; //список

int n = 0; //размер списка

for (;;)

{

cout << menu;

int start, finish, k;

int cmd; //команда

cout << "\nВведите команду: ";

cin >> cmd;

switch (cmd)

{

case 1: //создать новый контейнер множество дубликатами

ms.clear();

cout << "\nВведите размер контейнера множество дубликатами: ";

cin >> n;

ms = make\_multiset(ms, n);

break;

case 2: //вывести контейнер множество дубликатами

if (n > 0)

print\_multiset(ms);

else

cout << "\nСейчас список пуст";

break;

case 3: //посчитать среднее арифметическое и добавить его в конец контейнера множество дубликатами

ms = arithmetic\_mean(ms);

break;

case 4: //удалить элементы из заданного диапазона по ключам

cout << "\nВведите с какого по какой элемент будем удалять: ";

cin >> start >> finish;

ms = delete\_elem(ms, start, finish);

break;

case 5: //добавить ко всем элементам сумму минимального и максимального значений

ms = sum\_min\_max(ms);

break;

case 0: //выход из программы

cout << "\nСпасибо что выбрали данную программу!";

return 0;

default: //неправильно выбранная команда

cout << "\nНеправильная команда, попробуйте другую";

}

cout << endl;

system("pause");

system("cls");

}

}

catch (double) //блок обработки ошибок

{

cout << "Ошибка!";

}

return 0;

}

**Задача 2.**

**Pair.h:**

#pragma once

#include <iostream>

using namespace std;

class Pair

{

private:

int first;

double second;

public:

Pair() { first = 0; second = 0; };

Pair(int f, double s) { first = f; second = s; }

Pair(const Pair& p) { first = p.first; second = p.second; }

~Pair() {};

int get\_first() { return first; }

int get\_second() { return second; }

void set\_first(int f) { first = f; }

void set\_second(double s) { second = s; }

//перегруженные операции

Pair& operator =(const Pair& p);

Pair operator +(const Pair& p) const;

Pair& operator ++();

Pair operator ++(int);

Pair operator -(const Pair& p);

bool operator <(const Pair& p) const;

bool operator >(const Pair& p);

friend const Pair operator +(const Pair& left, const int& right);

friend const Pair operator +(const int& left, const Pair& right);

friend const Pair operator +(const Pair& left, const double& right);

friend const Pair operator +(const double& left, const Pair& right);

Pair operator /(const Pair& p);

Pair operator /(const int p);

//глобальные функции ввода-вывода

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

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

};

**Pair.cpp:**

#include "Pair.h"

#include <iostream>

using namespace std;

//перегрузка операции присваивания

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

{

//Проверка на самоприсваивание

if (&p == this)

return \*this;

else

first = p.first;

second = p.second;

return \*this;

}

//перегрузка операции сложение

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

{

Pair temp(this->first + p.first, this->second + p.second);

return temp;

}

//перегрузка инкремента

Pair& Pair::operator++()

{

++this->first;

++this->second;

return \*this;

}

Pair Pair::operator++(int)

{

Pair temp(\*this);

++this->first;

++this->second;

return temp;

}

//перегрузка операции вычитание

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

{

return Pair(this->first - p.first, this->second - p.second);

}

const Pair operator+(const Pair& left, const int& right)

{

return Pair(left.first + right, left.second);

}

const Pair operator+(const int& left, const Pair& right)

{

return right + left;

}

const Pair operator+(const Pair& left, const double& right)

{

return Pair(left.first, left.second + right);

}

const Pair operator+(const double& left, const Pair& right)

{

return right + left;

}

//перегрузка операции деление

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

{

return Pair(this->first / p.first, this->second / p.second);

}

Pair Pair::operator/(const int p)

{

return Pair(this->first / p, this->second / p);

}

//перегрузка операции сравнивания

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

{

double thiss = this->first + this->second, otherr = p.first + p.second;

if (thiss < otherr) return true;

else return false;

}

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

{

return !(\*this < p);

}

//перегрузка глобальной функции-операции ввода

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

{

cout << "\nПервое число: ";

in >> p.first;

cout << "\nВторое число: ";

in >> p.second;

return in;

}

//перегрузка глобальной функции-операции вывода

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

{

return (out << p.first << " : " << p.second);

}

**Zadacha2\_main.cpp:**

#include <iostream>

#include <set>

#include <cstdlib>

#include <Windows.h>

#include "Pair.h"

using namespace std;

typedef multiset<Pair>MS; //определяем тип для работы с контейнером множество дубликатами

//функция для формирования контейнера множество дубликатами

MS make\_multiset(MS ms, int n)

{

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

{

Pair a;

cout << "\nВведите " << i + 1 << " элемент: ";

cin >> a;

ms.insert(a); //добавление a в конец контейнера множество дубликатами

}

return ms;

}

//функция для печати контейнера множество дубликатами

void print\_multiset(MS ms)

{

cout << "\nСейчас контейнер множество дубликатами такой: \n";

int k = 1;

for (auto& i : ms)

{

cout << k << ". " << i << endl;

k++;

}

}

//функция для подсчета среднего арифметического и добавление его в конец контейнера множество дубликатами

MS arithmetic\_mean(MS ms)

{

Pair a;

for (auto& i : ms)

a = a + i;

ms.insert(a / ms.size());

return ms;

}

//функция удаления элементов из заданного диапазона по ключам

MS delete\_elem(MS ms, Pair start, Pair finish)

{

auto left = ms.lower\_bound(start);

auto right = ms.upper\_bound(finish);

ms.erase(left, right);

return ms;

}

//функция добавления суммы минимального и максимального значений

MS sum\_min\_max(MS ms)

{

multiset<Pair> temp; //создание промежуточного пустого контейнера

Pair max;//здесь будет максимальное значение в контейнере множество дубликатами

//цикл нахождения максимального значения

for (auto& i : ms)

{

if (max < i)

max = i;

}

Pair min; //здесь будет минимальное значение в контейнере множество дубликатами

//цикл нахождения минимального значения

for (auto& i : ms)

{

if (min > i)

min = i;

}

Pair s = min + max; //сумма минимального и максимального элементов

//добавление к элементам s

for (auto& i : ms)

temp.insert(i + min + max);

ms.swap(temp);

return ms;

}

const char menu[] = " 1 - создать контейнер множество дубликатами\n"

" 2 - вывести контейнер множество дубликатами\n"

" 3 - посчитать среднее арифметическое и добавить его в конец контейнера множество дубликатами\n"

" 4 - удалить элементы из заданного диапазона по ключам\n"

" 5 - добавить ко всем элементам сумму минимального и максимального значений\n"

" 0 - выход из программы\n";

//основная функция

int main()

{

setlocale(LC\_ALL, "Russian");

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

try

{

multiset<Pair> ms; //контейнер множество дубликатами

Pair p;

int n = 0; //размер контейнера множество дубликатами

for (;;)

{

cout << menu;

Pair start, finish;

int cmd; //команда

cout << "\nВведите команду: ";

cin >> cmd;

switch (cmd)

{

case 1: //создать новый контейнер множество дубликатами

ms.clear();

cout << "\nВведите размер контейнера множество дубликатами: ";

cin >> n;

ms = make\_multiset(ms, n);

break;

case 2: //вывести контейнер множество дубликатами

if (n > 0)

print\_multiset(ms);

else

cout << "\nСейчас список пуст";

break;

case 3: //посчитать среднее арифметическое и добавить его в конец контейнера множество дубликатами

ms = arithmetic\_mean(ms);

break;

case 4: //удалить элементы из заданного диапазона по ключам

cout << "\nВведите с какого по какой элемент будем удалять: ";

cin >> start >> finish;

ms = delete\_elem(ms, start, finish);

break;

case 5: //добавить ко всем элементам сумму минимального и максимального значений

ms = sum\_min\_max(ms);

break;

case 0: //выход из программы

cout << "\nСпасибо что выбрали данную программу!";

return 0;

default: //неправильно выбранная команда

cout << "\nНеправильная команда, попробуйте другую";

}

cout << endl;

system("pause");

system("cls");

}

}

catch (double) //блок обработки ошибок

{

cout << "Ошибка!";

}

return 0;

}

**Задача 3.**

**LinkedList.h:**

#pragma once

#include <iostream>

#include <set>

using namespace std;

template<typename T>

class MultiSet

{

private:

int \_size;

multiset<T> ms;

public:

MultiSet() { \_size = ms.size(); } //конструктор

void make\_multiset(T value); //метод добавления элементов

void print\_multiset(); //метод вывода списка в консоль

void arithmetic\_mean(); //добавить в список элемент со средним арифмитическим значением

void delete\_elem(); //поиск и удаление данных в заданном диапазоне

void sum\_min\_max(); //добавлениек каждому элементу максимального и минимального элементов

int size() { return \_size; } //размер

};

template<typename T>

void MultiSet<T>::make\_multiset(T value) {

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

{

double a;

cout << "\nВведите " << i + 1 << " элемент: ";

cin >> a;

ms.insert(a); //добавление a в конец контейнера множество дубликатами

}

}

template<typename T>

void MultiSet<T>::print\_multiset() {

int n = 0;

for (const auto& elem : ms) {

cout << ++n << ". " << elem << endl;

}

cout << endl;

}

template<typename T>

inline void MultiSet<T>::arithmetic\_mean()

{

T midl = 0;

for (auto& i : ms)

midl = midl + i;

ms.insert(midl / ms.size());

++\_size;

}

template<typename T>

void MultiSet<T>::delete\_elem()

{

double start, finish;

cout << "\nВведите с какого по какой элемент будем удалять: ";

cin >> start >> finish;

auto left = ms.lower\_bound(start);

auto right = ms.upper\_bound(finish);

ms.erase(left, right);

\_size = ms.size();

}

template<typename T>

inline void MultiSet<T>::sum\_min\_max()

{

multiset<T> temp;

T min = \*ms.begin(), max = \*ms.begin();

for (auto& i : ms)

{

if (i < min)

min = i;

if (!(i < max))

max = i;

}

for (auto& i : ms)

temp.insert(i + min + max);

ms.swap(temp);

}

**Zadacha3\_main.cpp:**

#include <iostream>

#include <set>

#include <cstdlib>

#include <Windows.h>

#include "LinkedList.h"

using namespace std;

MultiSet<double>MS;

const char menu[] = " 1 - создать контейнер множество дубликатами\n"

" 2 - вывести контейнер множество дубликатами\n"

" 3 - посчитать среднее арифметическое и добавить его в конец контейнера множество дубликатами\n"

" 4 - удалить элементы из заданного диапазона по ключам\n"

" 5 - добавить ко всем элементам сумму минимального и максимального значений\n"

" 0 - выход из программы\n";

//основная функция

int main()

{

setlocale(LC\_ALL, "Russian");

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

try

{

multiset<double> ms; //контейнер множество дубликатами

int n = 0; //размер контейнера множество дубликатами

for (;;)

{

cout << menu;

int cmd; //команда

cout << "\nВведите команду: ";

cin >> cmd;

switch (cmd)

{

case 1: //создать новый контейнер множество дубликатами

ms.clear();

cout << "\nВведите размер контейнера множество дубликатами: ";

cin >> n;

MS.make\_multiset(n);

break;

case 2: //вывести контейнер множество дубликатами

if (n > 0)

MS.print\_multiset();

else

cout << "\nСейчас список пуст";

break;

case 3: //посчитать среднее арифметическое и добавить его в конец контейнера множество дубликатами

MS.arithmetic\_mean();

break;

case 4: //удалить элементы из заданного диапазона по ключам

MS.delete\_elem();

break;

case 5: //добавить ко всем элементам сумму минимального и максимального значений

MS.sum\_min\_max();

break;

case 0: //выход из программы

cout << "\nСпасибо что выбрали данную программу!";

return 0;

default: //неправильно выбранная команда

cout << "\nНеправильная команда, попробуйте другую";

}

cout << endl;

/\*system("pause");

system("cls");\*/

}

}

catch (double) //блок обработки ошибок

{

cout << "Ошибка!";

}

return 0;

}