Министерство образования и науки Российской Федерации

Федеральное государственное образовательное автономное учреждение высшего образования

"Пермский национальный исследовательский политехнический университет"

ОТЧЕТ

ПО ЛАБОРАТОРНОЙ РАБОТЕ №12

Дисциплина: Основы алгоритмизации и программирования

Тема: Ассоциативные контейнеры библиотеки STL.

|  |  |
| --- | --- |
| Выполнил работу | |
| Студент группы РИС-22-1б | |
| Хайбиев К.А. | |
|  | |
| Проверил работу | |
| Доцент кафедры ИТАС | |
| Полякова О.А. | |
|  | |

Пермь – 2023

**Анализ предметной области**

**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:

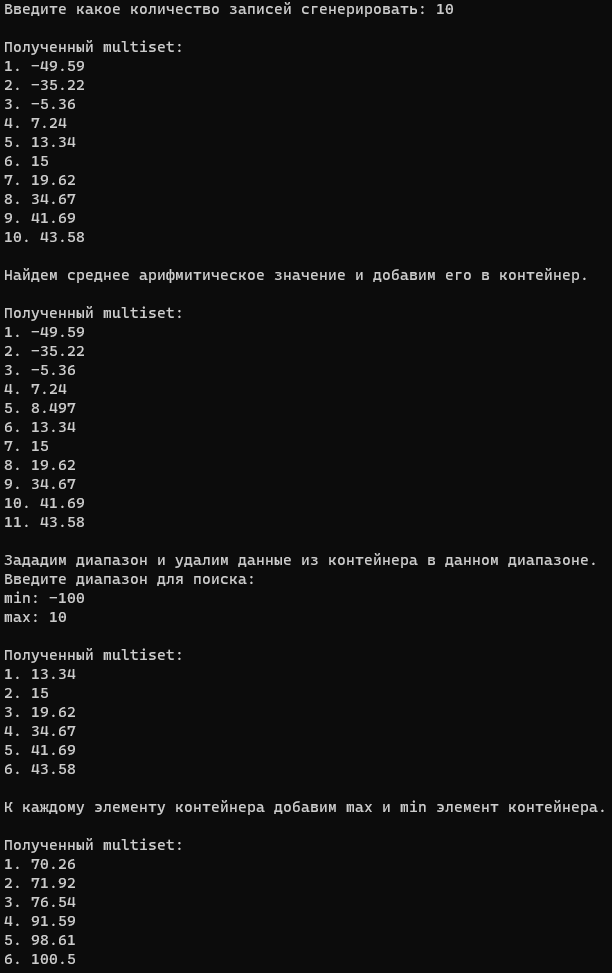
1. Создать параметризированный класс, используя в качестве контейнера ассоциативный контейнер.
2. Заполнить его элементами.
3. Добавить элементы в соответствии с заданием.
4. Удалить элементы в соответствии с заданием.
5. Выполнить задание варианта для полученного контейнера.
6. Выполнение всех заданий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

Вариант 15:

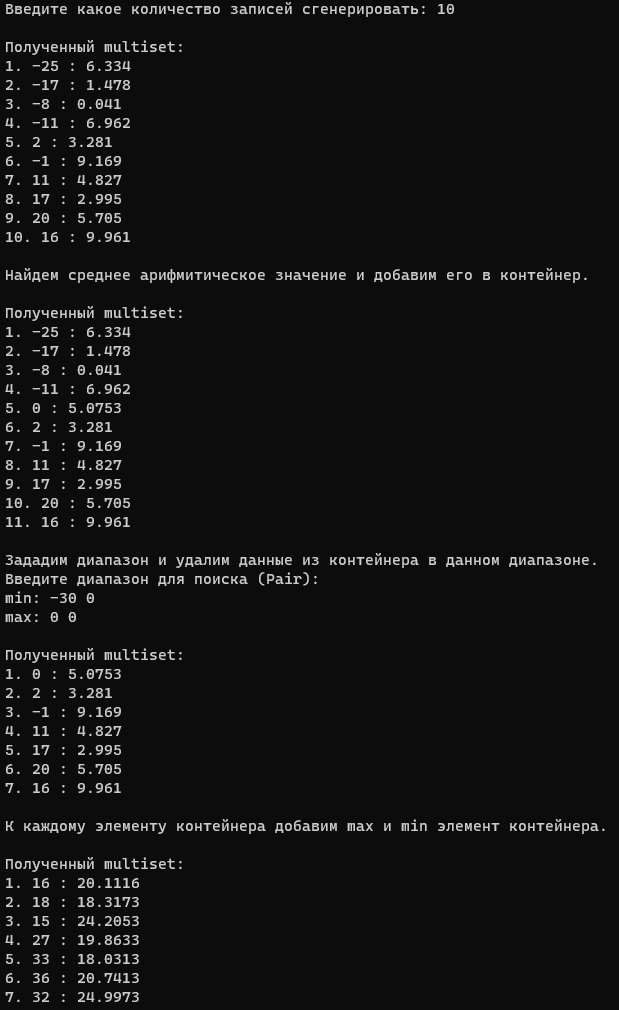
|  |  |  |
| --- | --- | --- |
| **Задача 1:**   * Контейнер – multiset. * Тип элементов – double.   **Задача 2:**   * Тип элементов Pair.   **Задача 3:**   * Параметризированный класс список. | | |
| **Задание 1.** | **Задание 2.** | **Задание 3.** |
| Найти среднее арифметическое и добавить его в конец контейнера. | Найти элементы с ключами из заданного диапазона и удалить их из контейнера. | К каждому элементу добавить сумму минимального и максимального элементов контейнера. |

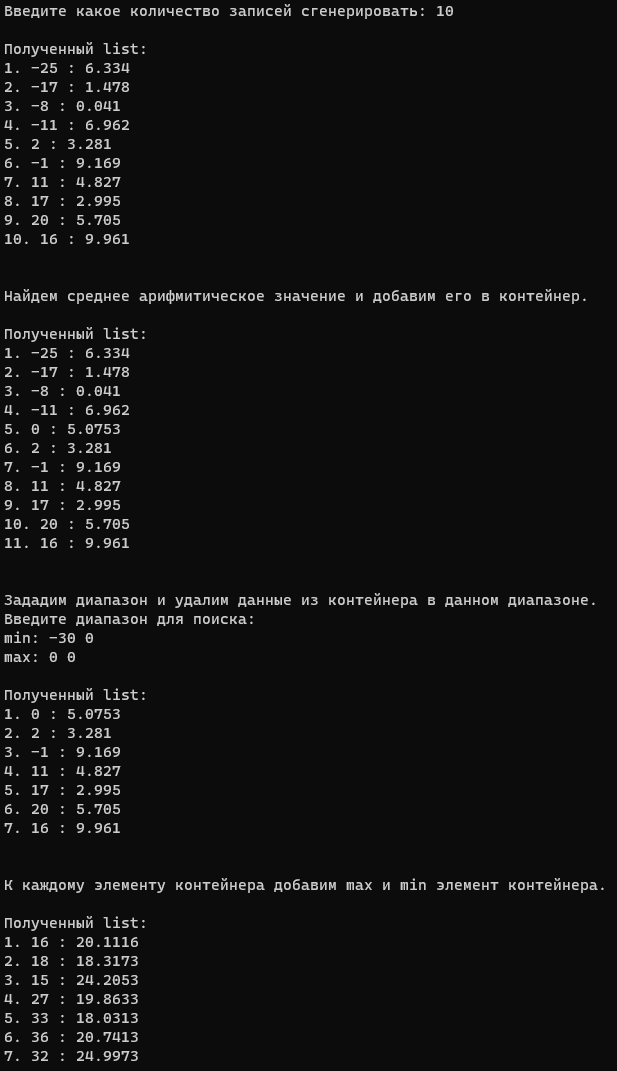
**Анализ задачи**

В данной задаче необходимо использовать контейнер multiset, который хранит данные в упорядоченном порядке, так как представляет из себя бинарное дерево. Поэтому выполнение задания 1 полностью невозможно, так как полученное число добавиться в соответствии с логикой бинарного дерева. Остальные задачи выполняются корректно с помощью встроенных методов контейнеров.

****Тестирование программы**

*Рис. 1 – Задача 1.*

*Рис. 2 – Задача 2.*

*Рис. 3 – Задача 3.*

**Заключение**

Была разработана программа, которая взаимодействует с ассоциативными контейнером STL библиотеки и использует его для хранения, взаимодействия и упорядочивания данных.

# Ответы на 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;

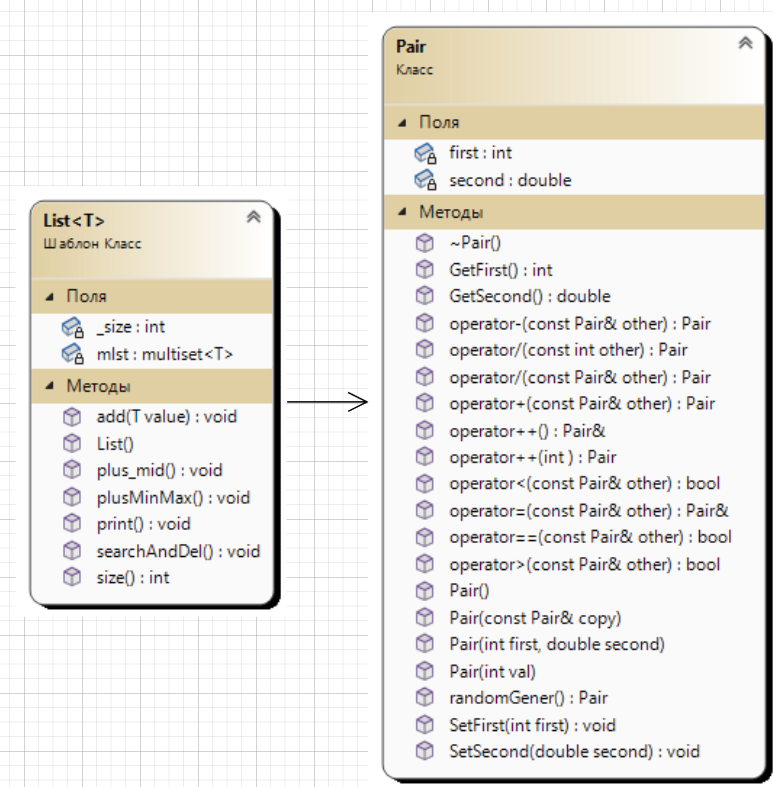
}

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.

**Приложения**

UML-диаграмма:



Приложение Б – код программы

Main.cpp:

#include"header.h"

#include <map>

int main()

{

system("chcp 1251 >> null");

//number1();

//number2();

number3();

return 0;

}

pair.h:

#pragma once

#include <iostream>

using namespace std;

class Pair

{

public:

Pair();

Pair(int first, double second);

Pair(int val);

Pair(const Pair& copy);

~Pair() {}

int GetFirst() { return this->first; }

double GetSecond() { return this->second; }

void SetFirst(int first) { this->first = first; }

void SetSecond(double second) { this->second = second; }

Pair& operator =(const Pair& other);

Pair operator +(const Pair& other) const;

Pair& operator ++();

Pair operator ++(int);

Pair operator -(const Pair& other);

bool operator <(const Pair& other) const;

bool operator >(const Pair& other);

bool operator ==(const Pair& other);

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& other);

Pair operator /(const int other);

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

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

Pair randomGener();

private:

int first;

double second;

};

pair.cpp:

#include "Pair.h"

ostream& operator <<(ostream& os, const Pair& val)

{

os << val.first << " : " << val.second;

return os;

}

istream& operator>>(istream& is, Pair& val)

{

is >> val.first >> val.second;

return is;

}

Pair::Pair()

{

first = 0;

second = 0;

}

Pair::Pair(const Pair& copy)

{

this->first = copy.first;

this->second = copy.second;

}

Pair::Pair(int first, double second)

{

this->first = first;

this->second = second;

}

Pair::Pair(int val)

{

this->first = val;

this->second = val;

}

Pair Pair::randomGener()

{

return Pair(rand() % 50 - 25, (double((rand() % 10000)) / 1000));

}

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

{

this->first = other.first;

this->second = other.second;

return \*this;

}

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

{

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

return temp;

}

Pair& Pair::operator++()

{

++this->first;

++this->second;

return \*this;

}

Pair Pair::operator++(int)

{

Pair temp(\*this);

++this->first;

++this->second;

return temp;

}

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

{

return Pair(this->first - other.first, this->second - other.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;

}

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

{

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

}

Pair Pair::operator/(const int other)

{

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

}

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

{

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

if (thiss < otherr) return true;

else return false;

}

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

{

return !(\*this < other);

}

bool Pair::operator==(const Pair& other)

{

if (this->first == other.first && this->second == other.second)

return true;

else

return false;

}

header.h:

#pragma once

#include<iostream>

#include<set>

using namespace std;

#include"number1.h"

#include"Pair.h"

#include"number2.h"

#include"List.h"

#include"number3.h"

1.h:

#pragma once

#include"header.h"

//тут все коментарии аналогичны файлу number2.h

void plus\_mid(multiset<double>& mlst)

{

double midl = 0;

for (auto& i : mlst)

midl += i;

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

}

void searchAndDel(multiset<double>& mlst)

{

double min, max;

cout << "Введите диапазон для поиска:\n" << "min: "; cin >> min;

cout << "max: "; cin >> max;

auto left = mlst.lower\_bound(min);

auto right = mlst.upper\_bound(max);

mlst.erase(left, right);

}

void plusMinMax(multiset<double>& mlst)

{

multiset<double> temp;

double min = \*mlst.begin(), max = \*mlst.begin();

for (auto& i : mlst)

{

if (i < min) min = i;

if (i > max) max = i;

}

for (auto& i : mlst)

temp.insert(i + min + max);

mlst.swap(temp);

}

void number1()

{

multiset<double> mlst;

int n;

cout << "Введите какое количество записей сгенерировать: "; cin >> n;

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

mlst.insert((double((rand() % 10000)))/100 - 50);

cout << endl << "Полученный multiset:" << endl; n = 0;

for (auto& i : mlst)

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

cout << endl << "Найдем среднее арифмитическое значение и добавим его в контейнер." << endl;

plus\_mid(mlst);

cout << endl << "Полученный multiset:" << endl; n = 0;

for (auto& i : mlst)

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

cout << endl << "Зададим диапазон и удалим данные из контейнера в данном диапазоне." << endl;

searchAndDel(mlst);

cout << endl << "Полученный multiset:" << endl; n = 0;

for (auto& i : mlst)

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

cout << endl << "К каждому элементу контейнера добавим max и min элемент контейнера." << endl;

plusMinMax(mlst);

cout << endl << "Полученный multiset:" << endl; n = 0;

for (auto& i : mlst)

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

}

2.h:

#pragma once

#include"header.h"

void plus\_mid2(multiset<Pair>& mlst) //добавление в контейнер элемента со средним арифмитическим значением

{

Pair midl = 0;

for (auto& i : mlst)

midl = midl + i;

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

}

void searchAndDel2(multiset<Pair>& mlst) //поиск и удаление данных в заданном диапазоне

{

Pair min, max;

cout << "Введите диапазон для поиска (Pair):\n" << "min: "; cin >> min;

cout << "max: "; cin >> max;

auto left = mlst.lower\_bound(min);

auto right = mlst.upper\_bound(max);

mlst.erase(left, right);

}

void plusMinMax2(multiset<Pair>& mlst) //добавлениек каждому элементу максимального и минимального элементов

{

multiset<Pair> temp;

Pair min = \*mlst.begin(), max = \*mlst.begin();

for (auto& i : mlst) //два раза проходим циклом

{ //первый раз ищем минимум и максимум

if (i < min) min = i;

if (!(i < max)) max = i;

}

for (auto& i : mlst) //второй раз составляем новый multiset

temp.insert(i + min + max);

mlst.swap(temp); //меняем данные в multisetах

}

void number2()

{

Pair pr;

multiset<Pair> mlst;

int n;

cout << "Введите какое количество записей сгенерировать: "; cin >> n;

for (int i = 0; i < n; i++) //заполнение контейнера рандомнаыми данными

mlst.insert(pr.randomGener());

cout << endl << "Полученный multiset:" << endl; n = 0;

for (auto& i : mlst) //вывод

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

cout << endl << "Найдем среднее арифмитическое значение и добавим его в контейнер." << endl;

plus\_mid2(mlst); //добавление элемента со средним арифметическим

cout << endl << "Полученный multiset:" << endl; n = 0;

for (auto& i : mlst) //вывод

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

cout << endl << "Зададим диапазон и удалим данные из контейнера в данном диапазоне." << endl;

searchAndDel2(mlst); // поиск и удаление элементов в заданном диапазоне

cout << endl << "Полученный multiset:" << endl; n = 0;

for (auto& i : mlst) //вывод

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

cout << endl << "К каждому элементу контейнера добавим max и min элемент контейнера." << endl;

plusMinMax2(mlst); //поиск максимального и минимального и добавление к каждому элементу

cout << endl << "Полученный multiset:" << endl; n = 0;

for (auto& i : mlst) //вывод

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

}

3.h:

#pragma once

#include"header.h"

void number3()

{

Pair pr;

List<Pair> mlst;

int n;

cout << "Введите какое количество записей сгенерировать: "; cin >> n; //заполнение списка рандомнаыми данными

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

mlst.add(pr.randomGener());

cout << endl << "Полученный list:" << endl;

mlst.print(); //вывод

cout << endl << "Найдем среднее арифмитическое значение и добавим его в контейнер." << endl;

mlst.plus\_mid(); //добавление элемента со средним арифметическим

cout << endl << "Полученный list:" << endl;

mlst.print(); //вывод

cout << endl << "Зададим диапазон и удалим данные из контейнера в данном диапазоне." << endl;

mlst.searchAndDel(); //поиск и удаление элементов в диапазоне

cout << endl << "Полученный list:" << endl;

mlst.print(); //вывод

cout << endl << "К каждому элементу контейнера добавим max и min элемент контейнера." << endl;

mlst.plusMinMax(); //поиск максимального и минимального, для добавление к каждому элементу

cout << endl << "Полученный list:" << endl;

mlst.print(); //вывод

}

List.h:

#pragma once

#include"header.h"

template<typename T>

class List {

public:

List() { \_size = mlst.size(); } //конструктор

void add(T value); //метод добавления элементов

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

void plus\_mid(); //добавить в список элемент со средним арифмитическим значением

void searchAndDel(); //поиск и удаление данных в заданном диапазоне

void plusMinMax(); //добавлениек каждому элементу максимального и минимального элементов

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

private:

int \_size;

multiset<T> mlst;

};

template<typename T>

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

mlst.insert(value);

++\_size;

}

template<typename T>

void List<T>::print() {

int n = 0;

for (const auto& elem : mlst) {

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

}

cout << endl;

}

template<typename T>

inline void List<T>::plus\_mid()

{

T midl = 0;

for (auto& i : mlst)

midl = midl + i;

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

++\_size;

}

template<typename T>

inline void List<T>::searchAndDel()

{

T min, max;

cout << "Введите диапазон для поиска:\n" << "min: "; cin >> min;

cout << "max: "; cin >> max;

auto left = mlst.lower\_bound(min);

auto right = mlst.upper\_bound(max);

mlst.erase(left, right);

\_size = mlst.size();

}

template<typename T>

inline void List<T>::plusMinMax()

{

multiset<T> temp;

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

for (auto& i : mlst)

{

if (i < min) min = i;

if (!(i < max)) max = i;

}

for (auto& i : mlst)

temp.insert(i + min + max);

mlst.swap(temp);

}