Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«**Пермский национальный исследовательский политехнический университет»**

Кафедра «Информационные технологии и автоматизированные системы»

**ОТЧЕТ**

Дисциплина: «Информатика»

Тема: Лабораторная работа №13

" Стандартные обобщенные алгоритмы библиотеки STL"

Семестр 2

Выполнил работу

Студент группы РИС-24-3Б

Гузий С.А.

Проверил

Доцент кафедры ИТАС

Полякова О.А.

Г. Пермь-2025

**Анализ предметной области**

**Постановка задачи**

Задача 1:

1. Создать последовательный контейнер.
2. Заполнить его элементами стандартного типа (тип указан в варианте).
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. Выполнение всех заданий использовать стандартные алгоритмы библиотеки STL.

Задача 2:

1. Создать адаптер контейнер.
2. Заполнить его элементами стандартного типа (тип указан в варианте).
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. Выполнение всех заданий использовать стандартные алгоритмы библиотеки STL.

Задача 3:

1. Создать ассоциативный контейнер.
2. Заполнить его элементами стандартного типа (тип указан в варианте).
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. Выполнение всех заданий использовать стандартные алгоритмы библиотеки STL.

Вариант 15:

|  |  |  |
| --- | --- | --- |
| **Задача 1:**   * Контейнер – список. * Тип элементов – Pair.   **Задача 2:**   * Адаптер контейнера – очередь.   **Задача 3:**   * Ассоциативный контейнер – множество с дубликатами. | | |
| **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. |

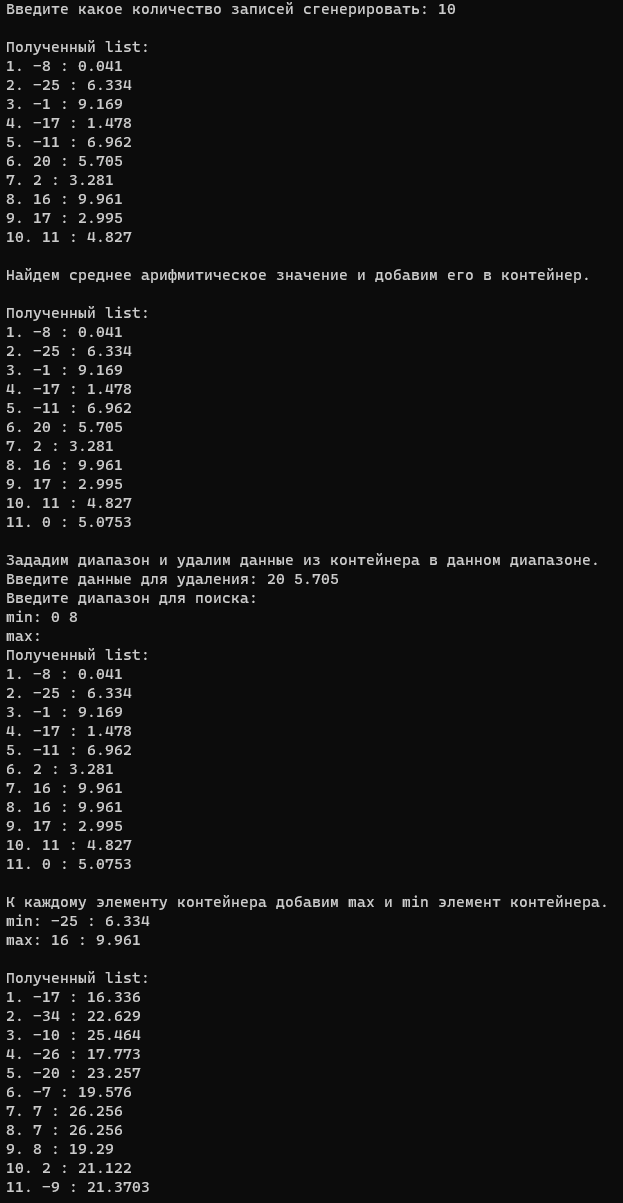
**Анализ задачи**

Задача 1: для того, чтобы реализовать функцию добавления элемента со средним значений можно воспользоваться алгоритмом for\_each с лямбда выражением, которое будет добавлять к временному элементу данные. А после поделить данный элемент на длину списка. Для задания 2 необходимо использовать функцию remove\_if с лямбда выражением (предикатом), который сверяет данные списка с заданным элементом. Для выполнения задания 3 необходимо использовать функции min\_element и max\_element, для того чтобы найти минимальный и максимальный элемент в списке.

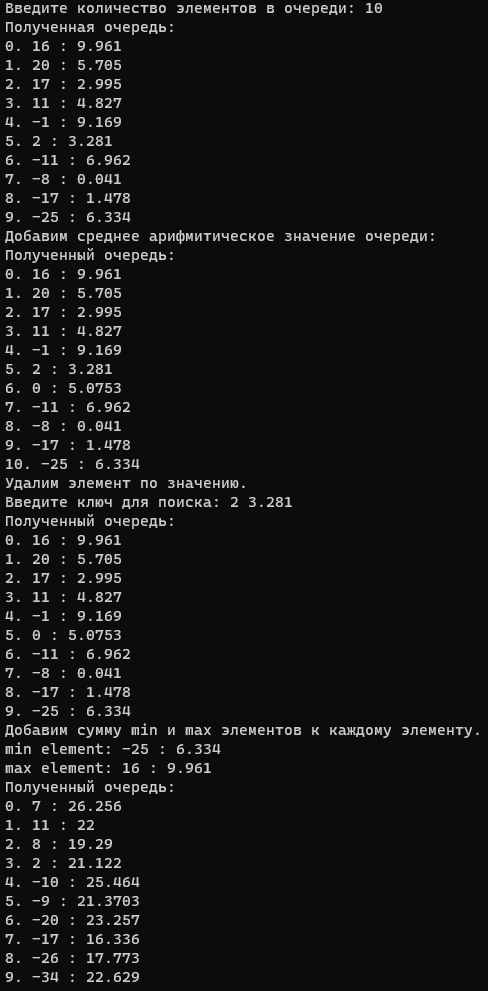
Задача 2: у очереди с приоритетами нет итераторов, а также с ней не могут взаимодействовать функции из библиотеки algorithm, потому что они основаны на том, что они работают с итераторами. Поэтому задачи, которые указаны в варианте номер 15 невозможно выполнить методами, которые предложены. Даже какими-либо методами приоритетной очереди это тоже не

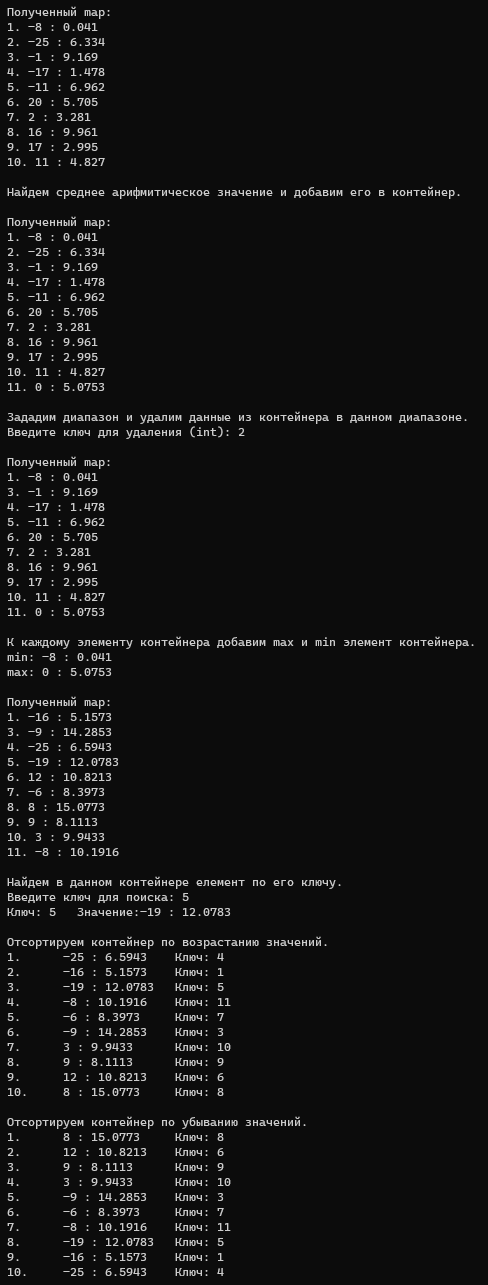
выполняются.

Задача 3: Итераторы контейнера map в большинстве функций библиотеки algorithm не поддерживаются. Поэтому в данной задаче использовано 2 функции: min\_element и max\_element, для того чтобы найти минимальный и максимальный элемент в списке соответственно. Это обусловлено тем, что данные в этом контейнере хранятся парно.

****Тестирование программы**

*Рис. 1 – Задача 1.*

*Рис. 2 – Задача 2.*

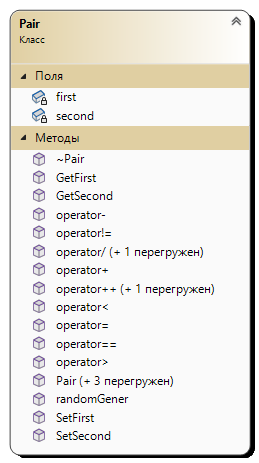
*Рис. 3 – Задача 3.*

**Заключение**

Была разработана программа, которая взаимодействует с контейнерами STL библиотеки и использует его для хранения, взаимодействия и упорядочивания данных. С данными контейнерами взаимодействуют функции из библиотеки STL algorithm.

**Приложения**

UML-диаграмма:



Приложение Б – код программы

Main.cpp:

#include"header.h"

int main()

{

system("chcp 1251 >> null");

//number1(); //вызов функций

//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"

//единственное нормальное задание во всей лабе

void plus\_mid(list<Pair>& lst) //добавление среднее в конец

{

Pair midl(0,0);

for\_each(lst.begin(), lst.end(), [&midl](Pair this\_el) //перебираем весь список с помощью функции

{midl = midl + this\_el;}); //это анонимная функция, которая все складывает в переменую midl, которую перетаскивает по ссылке из тела функции

lst.push\_back(midl / lst.size()); //просто пушим назаж

}

void searchAndDel(list<Pair>& lst) //ищем и удаляем

{

Pair y;

cout << "Введите данные для удаления: "; cin >> y;

int min, max;

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

cout << "max: "; cin >> max; //можно задать диапазон, работает с индексами

auto left = lst.begin(); auto right = lst.begin();

advance(left, min); //смещаем итераторы, чтобы установит диапазон

advance(right, max);

remove\_if(left, right, [&y](Pair& x) //удалем если введенное число совпадает с тем, на которое мы указываем

{return x == y;}); //анонимная функция

}

void plusMinMax(list<Pair>& lst) //добавление к каждому элементу мин и макс значения

{

Pair min = \*min\_element(lst.begin(), lst.end()); //находим мин и макс

Pair max = \*max\_element(lst.begin(), lst.end());

cout << "min: " << min << endl << "max: " << max << endl;

for\_each(lst.begin(), lst.end(), //проходим по всему списку

[&min, &max](Pair& this\_el) {this\_el = this\_el + max + min; }); //лямбда-выражение - которое добавляет к каждому элементу мин и макс

}

void number1() //функция для управления, здест ничего особенного, только в конце коммы

{

Pair p;

list<Pair> lst;

int n;

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

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

lst.push\_back(p.randomGener());

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

for (auto& i : lst)

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

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

plus\_mid(lst);

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

for (auto& i : lst)

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

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

searchAndDel(lst);

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

for (auto& i : lst)

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

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

plusMinMax(lst);

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

for (auto& i : lst)

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

cout << endl << "Найдем в данном контейнере елемент и его индекс." << endl;

cout << "Введите элемент для поиска: "; cin >> p;

auto it = find(lst.begin(), lst.end(), p);

if (it != lst.end())

cout << "Номер элемента: " << distance(lst.begin(), it) + 1 << "\tЭлемент: " << p;

else

cout << "Элемент не найден." << endl;

cout << endl << "Отсортируем контейнер по возрастанию." << endl;

lst.sort(); //sort не работает бля списка, но есть встроенный метод

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

for (auto& i : lst)

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

cout << endl << "Отсортируем контейнер по убыванию." << endl;

lst.sort(greater<Pair>()); //здесь мы меняем порядок сортировки на обратный с помощью объекта, который инвертирует

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

for (auto& i : lst)

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

}

2.h:

#pragma once

void Show(priority\_queue<Pair> q);

Pair SearchMidVal(priority\_queue<Pair>& q);

void DelElem(priority\_queue<Pair>& q);

void plusMinMax(priority\_queue<Pair>& q);

/\*

У очереди с приоретами нет итераторов, а также с ней не могут взаимодействовать

функции из библиотеки algorithm, потому что они основаны на том, что они работают

с итераторами. Поэтому задачи, которые указаны в варианте номер 15 невозможно выполнить

методами, которые предложены. Даже какими-либо методами приоритетной очереди это тоже не

выполняются.

ЗДЕСЬ ПОМОГУТ ТОЛЬКО КОСТЫЛИ!!!

Поэтому выполнение данного задания из лабораторной работы номер 11 - то что нужно.

\*/

void number2() //Это я даже не буду объяснять. Если очень интересно, то идем в 11 лабу там это 4 задача

{ //просто ужас, как можно было такое составить

Pair temp; //неадекватное задание

priority\_queue<Pair> pq; //у нее даже итераторв нет

int size;

cout << "Введите количество элементов в очереди: "; cin >> size;

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

pq.push(temp.randomGener());

cout << "Полученная очередь:" << endl;

Show(pq);

cout << "Добавим среднее арифмитическое значение очереди:" << endl;

pq.push(SearchMidVal(pq));

cout << "Полученный очередь:" << endl;

Show(pq);

cout << "Удалим элемент по значению." << endl;

DelElem(pq);

cout << "Полученный очередь:" << endl;

Show(pq);

cout << "Добавим сумму min и max элементов к каждому элементу." << endl;

plusMinMax(pq);

cout << "Полученный очередь:" << endl;

Show(pq);

return;

}

void Show(priority\_queue<Pair> q)

{

int j = 0;

while (!q.empty())

{

cout << j++ << ". " << q.top() << endl;

q.pop();

}

}

Pair SearchMidVal(priority\_queue<Pair>& q)

{

priority\_queue<Pair> temp(q);

Pair mid = 0;

while (!temp.empty())

{

mid = mid + temp.top();

temp.pop();

}

return mid / q.size();

}

void DelElem(priority\_queue<Pair>& q)

{

Pair key, k;

priority\_queue<Pair> temp;

cout << "Введите ключ для поиска: "; cin >> key;

while (!q.empty())

{

k = q.top();

if (key == k)

q.pop();

else

{

temp.push(k);

q.pop();

}

}

q.swap(temp);

}

void plusMinMax(priority\_queue<Pair>& q)

{

Pair k;

Pair loc\_min = q.top(), loc\_max = 0;

priority\_queue<Pair> temp(q);

while (!temp.empty())

{

k = temp.top();

if (loc\_max < k)

{

loc\_max = k;

temp.pop();

}

else if (loc\_min > k)

{

loc\_min = k;

temp.pop();

}

else temp.pop();

}

cout << "min element: " << loc\_min << endl;

cout << "max element: " << loc\_max << endl;

temp.swap(q);

while (!temp.empty())

{

k = temp.top();

q.push(k + loc\_min + loc\_max);

temp.pop();

}

}

3.h:

#pragma once

#include"header.h"

static int shet = 0; //генератор ключей

void print(map<int, Pair>& mp) //функция вывода словаря

{

cout << endl << "Полученный map:" << endl;

for (auto& i : mp)

cout << i.first << ". " << i.second << endl;

}

void plus\_mid(map<int, Pair>& mp) //функция добавляет среднее арифмитическое

{

Pair midl(0, 0); //НО!!!

for (auto& i : mp) //так как с контейнером map не работает библилтека algorithm, точнее большинство методов

midl = midl + i.second; //тут мы все делаем ручками через обычные циклы

mp.emplace(++shet, midl / mp.size()); //впринципе так в номере 3 везде

}

void searchAndDel(map<int, Pair>& mp) //удаление по ключу

{ //ничего интересного, просто вызов метода

int y;

cout << "Введите ключ для удаления (int): "; cin >> y;

mp.erase(y);

}

void plusMinMax(map<int, Pair>& mp) //добавление к каждому элементу минимального и максимального значения

{

Pair min = min\_element(mp.begin(), mp.end())->second; //находим минимальный и максимальный элемент

Pair max = max\_element(mp.begin(), mp.end())->second; //использую функцию библиотеки algorithm

cout << "min: " << min << endl << "max: " << max << endl; //а дальше просто циклом проходим и добавляем

for (auto& i : mp)

i.second = i.second + max + min; //синтаксис такой, потому что += у pair не перегружен

}

void myfind(map<int, Pair>& mp) //мой поиск. Просто поиск по ключу через метод контейнера map

{

int a;

cout << "Введите ключ для поиска: "; cin >> a;

auto it = mp.find(a);

if (it != mp.end())

cout << "Ключ: " << a << " Значение:" << it->second << endl;

else cout << "Элемента с таким ключом нет в словаре." << endl;

}

void printsortup(map<int, Pair>& mp) //Якобы сортировка, но не так.

{ //map - бинарное дерево и файлы там отсортированы, поэтому мы просто ключами делаем pair и выводим наш контейнер

int a = 0;

map<Pair, int> temp;

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

temp.emplace(i->second, i->first);

for (auto& i : temp)

cout << ++a <<".\t" << i.first << " \tКлюч: " << i.second << endl;

}

void printsortdown(map<int, Pair>& mp) //здесь тоже самое что и сверху, только выводим контейнер наоборот

{

int a = 0;

map<Pair, int> temp;

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

temp.emplace(i->second, i->first);

for (auto i = temp.crbegin(); i != temp.crend(); i++)

cout << ++a << ".\t" << i->first << " \tКлюч: " << i->second << endl;

}

void number3() //функция, которая управляет всем

{ //вызывает функции выше и пишет сообщения.

Pair p; //чтобы не спамить в main

map<int, Pair> mp;

int n;

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

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

mp.emplace(++shet, p.randomGener());

cout << "Тип вывода:\nКлюч. Значение" << endl;

print(mp);

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

plus\_mid(mp);

print(mp);

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

searchAndDel(mp);

print(mp);

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

plusMinMax(mp);

print(mp);

cout << endl << "Найдем в данном контейнере елемент по его ключу." << endl;

myfind(mp);

cout << endl << "Отсортируем контейнер по возрастанию значений." << endl;

printsortup(mp);

cout << endl << "Отсортируем контейнер по убыванию значений." << endl;

printsortdown(mp);

}

Header.h:

#pragma once

#include<iostream> //подключение всего, что есть

#include<list>

#include<queue>

#include<map>

#include<algorithm>

using namespace std;

#include"Pair.h"

#include"number1.h"

#include"number2.h"

#include"number3.h"

Ссылка на github: https://github.com/Maybeeeeebl