МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ  
РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

ПЕРМСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ  
ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Кафедра информационные технологии и автоматизированные системы

**Дисциплина Информатика**

**Библиотека STL**

Выполнил студент ИВТ-22-2б:

Токмаков Герман Максимович

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Проверил доцент кафедры ИТАС:

Полякова Ольга Андреевна

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Пермь, 2023

# Постановка задачи

# Задача 1.

# 1. Создать последовательный контейнер.

# 2. Заполнить его элементами пользовательского типа (тип указан в варианте). Для пользовательского типа перегрузить необходимые операции.

# 3. Заменить элементы в соответствии с заданием (использовать алгоритмы replace\_if(), replace\_copy(), replace\_copy\_if(), fill()).

# 4. Удалить элементы в соответствии с заданием (использовать алгоритмы remove(),remove\_if(), remove\_copy\_if(),remove\_copy())

# 5. Отсортировать контейнер по убыванию и по возрастанию ключевого поля (использовать алгоритм sort()).

# 6. Найти в контейнере заданный элемент (использовать алгоритмы find(), find\_if(), count(), count\_if()).

# 7. Выполнить задание варианта для полученного контейнера (использовать алгоритм for\_each()) .

# 8. Для выполнения всех заданий использовать стандартные алгоритмы библиотеки STL.

# Задача 2.

# 1. Создать адаптер контейнера.

# 2. Заполнить его элементами пользовательского типа (тип указан в варианте). Для пользовательского типа перегрузить необходимые операции.

# 3. Заменить элементы в соответствии с заданием (использовать алгоритмы replace\_if(), replace\_copy(), replace\_copy\_if(), fill()).

# 4. Удалить элементы в соответствии с заданием (использовать алгоритмы remove(),remove\_if(), remove\_copy\_if(),remove\_copy())

# 5. Отсортировать контейнер по убыванию и по возрастанию ключевого поля (использовать алгоритм sort()).

# 6. Найти в контейнере элемент с заданным ключевым полем (использовать алгоритмы find(), find\_if(), count(), count\_if()).

# 7. Выполнить задание варианта для полученного контейнера (использовать алгоритм for\_each()) .

# 8. Для выполнения всех заданий использовать стандартные алгоритмы библиотеки STL.

# Задача 3

# 1. Создать ассоциативный контейнер.

# 2. Заполнить его элементами пользовательского типа (тип указан в варианте). Для пользовательского типа перегрузить необходимые операции.

# 3. Заменить элементы в соответствии с заданием (использовать алгоритмы replace\_if(), replace\_copy(), replace\_copy\_if(), fill()).

# 4. Удалить элементы в соответствии с заданием (использовать алгоритмы remove(),remove\_if(), remove\_copy\_if(),remove\_copy())

# 5. Отсортировать контейнер по убыванию и по возрастанию ключевого поля (использовать алгоритм sort()).

# 6. Найти в контейнере элемент с заданным ключевым полем (использовать алгоритмы find(), find\_if(), count(), count\_if()).

# 7. Выполнить задание варианта для полученного контейнера (использовать алгоритм for\_each()) .

# 8. Для выполнения всех заданий использовать стандартные алгоритмы библиотеки STL.

ВАРИАНТ 15:

Задача 1

1. Контейнер - список

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

Задача 2 Адаптер контейнера – очередь с приоритетами.

Задача 3 Ассоциативный контейнер - словарь

Задание 3

Найти среднее арифметическое и добавить его в конец контейнера

Задание 4

Найти элементы ключами из заданного диапазона и удалить их из контейнера

Задание 5

К каждому элементу добавить сумму минимального и максимального элементов контейнера

main 13.1:

#include <iostream>

#include <list>

#include "Pair.h"

#include <algorithm>

#include <functional>

#include <numeric>

using namespace std;

template <typename T>

void print(const list<T>&);

double leftBound = 3,

rightBound = 6;

bool removePred(Pair t) {

return t.getFirst() >= leftBound && t.getFirst() <= rightBound && t.getSecond() >= leftBound && t.getSecond() <= rightBound;

}

struct comparator {

public:

static bool cd(Pair pair1, Pair pair) {

return pair1.getFirst() + pair1.getSecond() > pair.getFirst() + pair.getSecond();

}

};

int main() {

//1

list<Pair> pairList;

pairList.push\_back(Pair(1, 2.5));

pairList.push\_back(Pair(4, 5.5));

pairList.push\_back(Pair(5, 7.5));

Pair a = accumulate(pairList.begin(), pairList.end(), Pair(0, 0), [](Pair t1, Pair t2) {return t1 + t2.getFirst() + t2.getSecond(); });

int arithmeticMeanInt = a.getFirst() / pairList.size();

double arithmeticMeanDouble = a.getSecond() / pairList.size();

pairList.push\_back(Pair(arithmeticMeanInt, arithmeticMeanDouble));

print(pairList);

//2

list<Pair> pairListSecond;

remove\_copy\_if(pairList.begin(), pairList.end(), inserter(pairListSecond, pairListSecond.begin()), removePred);

print(pairListSecond);

//3

list<Pair> pairListThird = pairList;

pairListThird.sort();

print(pairListThird);

return 0;

}

template <typename T>

void print(const list<T>& p\_list) {

for (auto itBegin = p\_list.begin(); itBegin != p\_list.end(); itBegin++) {

cout << \*itBegin << " ";

}

cout << endl;

}

main 13.2:

#include "Pair.h"

#include <iostream>

#include <list>

#include <queue>

#include <numeric>

#include <algorithm>

#include <map>

using namespace std;

bool operator<(const Pair& pr1, const Pair& pr2)

{

return ((pr1.a < pr2.a) && (pr1.b < pr2.b));

}

int main() {

setlocale(0, "");

priority\_queue <Pair> lst;

priority\_queue <Pair> tmp;

lst.push(Pair(1, 1.5));

lst.push(Pair(-2, 2.7));

lst.push(Pair(3, -3.2));

cout << "task 1:\n";

//1

Pair sr(0, 0);

while (!lst.empty())

{

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

sr = sr + lst.top();

tmp.push(lst.top());

lst.pop();

}

sr.a /= tmp.size();

sr.b /= tmp.size();

while (!tmp.empty())

{

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

lst.push(tmp.top());

tmp.pop();

}

lst.push(sr);

cout << sr << endl;

//2

int k = 1;

int i, j;

Pair res;

i = 2, j = 5;

cout << "key? "; cin >> res;

while (!lst.empty())

{

if ((k < i) || (k > j) || (res != lst.top()))

{

tmp.push(lst.top());

}

lst.pop();

k++;

}

cout << "task 2:\n";

Pair max = tmp.top();

Pair min = tmp.top();

while (!tmp.empty())

{

if (max < tmp.top()) max = tmp.top();

if (tmp.top() < min) min = tmp.top();

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

lst.push(tmp.top());

tmp.pop();

}

//3

cout << "task 3:\n";

while (!lst.empty())

{

tmp.push(max + min + lst.top());

lst.pop();

}

while (!tmp.empty())

{

lst.push(tmp.top());

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

tmp.pop();

}

return 0;

}

main 13.3:

#include "Pair.h"

#include <iostream>

#include <list>

#include <queue>

#include <numeric>

#include <algorithm>

#include <map>

using namespace std;

int main()

{

setlocale(0, "");

map <int, int> myMap = { {1, 2}, {2, 4}, {3, 6} };

cout << "task 1:\n";

int sum = 0;

for (auto& p : myMap) {

sum += p.second;

}

int avg = sum / myMap.size();

myMap[0] = avg;

for (auto& p : myMap) {

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

}

cout << "task 2:\n";

myMap.erase(2); // удалить элемент с ключом 2

cout << "task 3:\n";

auto maxIt = std::max\_element(myMap.begin(), myMap.end(),

[](const auto& p1, const auto& p2) { return p1.second < p2.second; });

auto minIt = std::min\_element(myMap.begin(), myMap.end(),

[](const auto& p1, const auto& p2) { return p1.second < p2.second; });

int maxVal = maxIt->second;

int minVal = minIt->second;

// добавить максимальное и минимальное значение ко всем элементам

for (auto& p : myMap) {

p.second += maxVal + minVal;

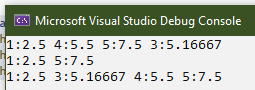
cout << p.first << " : " << p.second << endl;

}

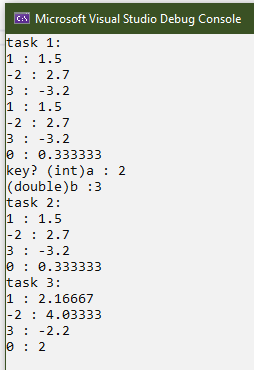
}

**Вывод программы**

main 13.1:



main 13.2:



main 13.3:

