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

Лабораторная работа №13

" Стандартные обобщенные алгоритмы библиотеки STL "

Вариант: 12

Выполнил студент ИВТ-24-2б:
Шишкин Максим Григорьевич

_____
(дата, подпись)

Проверил доцент кафедры ИТАС:

Полякова Ольга Андреевна

_____
(дата, подпись)

Пермь 2025

Содержание

1 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:

Задача 1 1. Контейнер - список

Тип элементов Pair (см. лабораторную работу №3). **Задача 2** Адаптер контейнера – очередь с приоритетами. **Задача 3** Ассоциативный контейнер словарь с дубликатами		
Задание 3	**Задание 4**	**Задание 5**
Найти среднее арифметическое и добавить его на заданную позицию контейнера	Найти элементы ключами из заданного диапазона и удалить их из контейнера	Из каждого элемента вычесть среднее арифметическое контейнера.

2 Код на C++

Pair.h main.cpp

13лабапределека (Глобальна

```cpp
#pragma once
#include <iostream>
using namespace std;

class Pair {
    int first;
    double second;

public:
    Pair() : first(0), second(0.0) {}
    Pair(int f, double s) : first(f), second(s) {}

    int getFirst() const { return first; }
    double getSecond() const { return second; }

    void setFirst(int f) { first = f; }
    void setSecond(double s) { second = s; }

    friend ostream& operator<<(ostream& os, const Pair& p) {
        os << "(" << p.first << ", " << p.second << ")";
        return os;
    }

    bool operator==(const Pair& other) const {
        return first == other.first && second == other.second;
    }

    bool operator<(const Pair& other) const {
        return first < other.first;
    }

    bool operator>(const Pair& other) const {
        return first > other.first;
    }

    Pair operator+(const Pair& other) const {
        return Pair(first + other.first, second + other.second);
    }

    Pair operator/(int val) const {
        return Pair(first / val, second / val);
    }

    Pair operator-(const Pair& other) const {
        return Pair(first - other.first, second - other.second);
    }
};
```

Pair.h main.cpp

13лабапределека (Глобальная

```cpp
#include <iostream>
#include <list>
#include <queue>
#include <map>
#include <algorithm>
#include <string>
#include <locale>

using namespace std;

class Pair {
    int first;
    double second;

public:
    Pair() : first(0), second(0.0) {}
    Pair(int f, double s) : first(f), second(s) {}

    int getFirst() const { return first; }
    double getSecond() const { return second; }

    void setFirst(int f) { first = f; }
    void setSecond(double s) { second = s; }

    friend ostream& operator<<(ostream& os, const Pair& p) {
        os << "(" << p.first << ", " << p.second << ")";
        return os;
    }

    bool operator==(const Pair& other) const {
        return first == other.first && second == other.second;
    }

    bool operator<(const Pair& other) const {
        return first < other.first;
    }

    bool operator>(const Pair& other) const {
        return first > other.first;
    }

    Pair operator+(const Pair& other) const {
        return Pair(first + other.first, second + other.second);
    }

    Pair operator/(int val) const {
        return Pair(first / val, second / val);
    }

    Pair operator-(const Pair& other) const {
        return Pair(first - other.first, second - other.second);
    }
};

void printList(const list<Pair>& lst, const string& msg) {
    cout << msg << endl;
    for (list<Pair>::const_iterator it = lst.begin(); it != lst.end(); ++it)
        cout << *it << " ";
    cout << endl;
}

void task1_list() {
    list<Pair> lst = { Pair(1, 1.1), Pair(2, 2.2), Pair(3, 3.3), Pair(4, 4.4) };
    printList(lst, "\n--- Задача 1: Список ---\nНачальный список:");

    replace_if(lst.begin(), lst.end(), [](const Pair& p) { return p.getFirst() > 2; }, Pair(0, 0));
    printList(lst, "После replace_if (first > 2 заменены на (0,0)):");

    lst.remove_if([](const Pair& p) { return p.getSecond() == 0; });
    printList(lst, "После remove_if (удалены second == 0):");

    lst.sort();
    printList(lst, "Сортировка по возрастанию:");

    lst.sort(greater<Pair>());
    printList(lst, "Сортировка по убыванию:");

    list<Pair>::iterator it = find(lst.begin(), lst.end(), Pair(1, 1.1));
    cout << (it != lst.end() ? "Элемент найден: " : "Элемент не найден") << *it << endl;

    int count = count_if(lst.begin(), lst.end(), [](const Pair& p) { return p.getFirst() < 3; });
    cout << "Количество элементов с first < 3: " << count << endl;

    Pair sum;
    for (list<Pair>::iterator it = lst.begin(); it != lst.end(); ++it)
        sum = sum + *it;

    Pair avg = sum / static_cast<int>(lst.size());
    for (list<Pair>::iterator it = lst.begin(); it != lst.end(); ++it)
        *it = *it - avg;

    printList(lst, "После вычитания среднего:");
}

void printQueue(priority_queue<Pair> q, const string& msg) {
    cout << msg << endl;
    while (!q.empty()) {
```

```
    while (!q.empty()) {
        cout << q.top() << " ";
        q.pop();
    }
    cout << endl;
}

void task2_priority_queue() {
    priority_queue<Pair> q;
    q.push(Pair(1, 1.1));
    q.push(Pair(4, 4.4));
    q.push(Pair(3, 3.3));

    cout << "\n--- Задача 2: Очередь с приоритетом ---\n";
    printQueue(q, "Исходная очередь:");
    // Замену/удаление элементов стандартная очередь с приоритетом не поддерживает напрямую
}

void printMultimap(multimap<int, Pair> mm, const string& msg) {
    cout << msg << endl;
    for (multimap<int, Pair>::const_iterator it = mm.begin(); it != mm.end(); ++it) {
        cout << it->first << " => " << it->second << endl;
    }
}

void task3_multimap() {
    multimap<int, Pair> mm = {
        make_pair(1, Pair(1, 1.1)),
        make_pair(2, Pair(2, 2.2)),
        make_pair(3, Pair(3, 3.3)),
        make_pair(4, Pair(4, 4.4))
    };

    cout << "\n--- Задача 3: Multimap с дубликатами ---" << endl;
    printMultimap(mm, "Исходный контейнер:");

    Pair sum;
    for (multimap<int, Pair>::iterator it = mm.begin(); it != mm.end(); ++it)
        sum = sum + it->second;

    Pair avg = sum / static_cast<int>(mm.size());

    mm.insert(make_pair(-1, avg));
    printMultimap(mm, "После вставки среднего:");

    for (multimap<int, Pair>::iterator it = mm.begin(); it != mm.end(); ) {
        if (it->first >= 2 && it->first <= 3)
            it = mm.erase(it);
        else
            ++it;
    }
    printMultimap(mm, "После удаления по диапазону ключей [2,3]:");

    for (multimap<int, Pair>::iterator it = mm.begin(); it != mm.end(); ++it)
        it->second = it->second - avg;
    printMultimap(mm, "После вычитания среднего из значений:");
}

int main() {
    setlocale(LC_ALL, "Russian");
    task1_list();
    task2_priority_queue();
    task3_multimap();
    return 0;
}
```

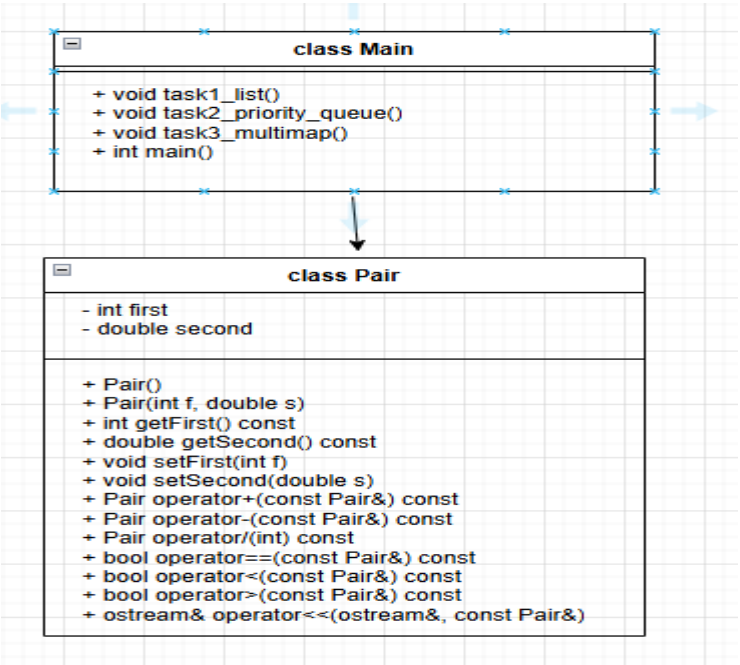
3 Результаты работы

```
--- Задача 1: Список ---
Начальный список:
(1, 1.1) (2, 2.2) (3, 3.3) (4, 4.4)
После replace_if (first > 2 заменены на (0,0)):
(1, 1.1) (2, 2.2) (0, 0) (0, 0)
После remove_if (удалены second == 0):
(1, 1.1) (2, 2.2)
Сортировка по возрастанию:
(1, 1.1) (2, 2.2)
Сортировка по убыванию:
(2, 2.2) (1, 1.1)
Элемент найден: (1, 1.1)
Количество элементов с first < 3: 2
После вычитания среднего:
(1, 0.55) (0, -0.55)

--- Задача 2: Очередь с приоритетом ---
Исходная очередь:
(4, 4.4) (3, 3.3) (1, 1.1)

--- Задача 3: Multimap с дубликатами ---
Исходный контейнер:
1 => (1, 1.1)
2 => (2, 2.2)
3 => (3, 3.3)
4 => (4, 4.4)
После вставки среднего:
-1 => (2, 2.75)
1 => (1, 1.1)
2 => (2, 2.2)
3 => (3, 3.3)
4 => (4, 4.4)
После удаления по диапазону ключей [2,3]:
-1 => (2, 2.75)
1 => (1, 1.1)
4 => (4, 4.4)
После вычитания среднего из значений:
-1 => (0, 0)
1 => (-1, -1.65)
4 => (2, 1.65)
```

4 UML-диаграмма классов



5 Ссылка на github

ссылка на github - https://github.com/MAKSPOWERO/mas1