

Федеральное государственное автономное образовательное учреждение  
высшего образования «Пермский национальный исследовательский  
политехнический университет»

Лабораторная работа №13  
«Стандартные обобщенные алгоритмы библиотеки STL»

Выполнил:

студент первого курса

ЭТФ группы РИС-23-36

Акбашева Софья Руслановна

Проверила:

Доцент кафедры ИТАС О. А. Полякова

Пермь, 2024

## Стандартные обобщенные алгоритмы библиотеки STL

### Цель задания

- 1) Создание консольного приложения, состоящего из нескольких файлов в системе программирования Visual Studio.
- 2) Использование стандартных обобщенных алгоритмов из библиотеки STL в ОО программе.

### 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.

### **Анализ задачи 1**

1) Необходимо реализовать класс `Pair`. В классе надо прописать конструкторы: по умолчанию, с параметрами и копирования; а также деструктор, для предотвращения утечки памяти.

2) Перегруженные операторы `+`, `-`, `=` и `<<` позволяют выполнять различные операции с объектами класса `Pair`. Например, оператор `+` позволяет складывать целое число с первым полем объекта класса и вещественное число со вторым полем соответственно. Оператор `<<` позволяет выводить объекты класса `Pair` в поток вывода.

3) Для использования списка необходимо подключить библиотеку `#include <list>`.

4) При создании списка необходимо указать какие объекты будут в нем храниться. В данном случае – объекты класса `Pair`: `list<Pair> list_1(N)`

5) Для использования лямда функций и других необходимо подключить: `#include <functional>`, `#include <algorithm>`.

6) При поиске максимального объекта будем считать, что необходимо сравнивать сначала по первому числу, а затем по второму числу в паре. Аналогично для поиска минимального элемента, и сортировки списка.

7) В главной функции показаны операции, которые можно выполнить с контейнером.

### **UML диаграмма**

Pair
- first: int - second: double
+ Pair() + Pair(int, double) + Pair(const Pair&) + ~Pair() + Print(): void + operator+(Pair& p): Pair + operator-(const Pair& para): Pair + setFirst(const int& x): void + setSecond(const double& y): void + getFirst(): int + getSecond(): double + operator=(const Pair& para): void + operator <<(ostream& stream, const Pair& p): friend ostream& + operator >>(istream& stream, Pair& p): friend istream& + operator >(const Pair& p): bool + operator <(const Pair& p): bool

## Код

### Файл Pair.h

```
#pragma once //предотвращает повторную загрузку заголовочного файла, если он уже был включен
#include <iostream> //стандартные потоки ввода и вывода
using namespace std;

class Pair {
    int first;
    double second;

public:
    Pair();//конструктор класса по умолчанию
    Pair(int, double);//конструктор класса с параметрами
    Pair(const Pair&);//конструктор копирования класса
    ~Pair();//деструктор класса
    void Print();

    Pair operator+(Pair& p);
    Pair operator-(const Pair& para);
    void setFirst(const int& x);
    void setSecond(const double& y);
    int getFirst();
    double getSecond();
    void operator=(const Pair& para);
    friend ostream& operator <<(ostream& stream, const Pair& p);
    friend istream& operator >>(istream& stream, Pair& p);
    bool operator >(const Pair& p);
    bool operator <(const Pair& p);
};
```

### Файл Pair.cpp

```
#include "Pair.h"
#include <iostream>
#include <string>
```

```

using namespace std;

Pair::Pair() {
    this->first = 0;
    this->second = 0;
}

Pair::Pair(int first, double second) {
    this->first = first;
    this->second = second;
}

Pair::Pair(const Pair& other) { //конструктор копирования
    this->first = other.first;
    this->second = other.second;
}

Pair::~Pair() { //деструктор
}

void Pair::Print() { //вывод параметров класса person
    cout << "(" << this->first << " : " << this->second << ")" << endl;
}

Pair Pair::operator+(Pair& p) {
    return (Pair(this->first + p.first, this->second + p.second));
}

Pair Pair::operator-(const Pair& para) {
    return (Pair(this->first - para.first, this->second - para.second));
}

void Pair::setFirst(const int& x) {
    this->first = x;
}

void Pair::setSecond(const double& y) {
    this->second = y;
}

int Pair::getFirst() {
    return this->first;
}

double Pair::getSecond() {
    return this->second;
}

void Pair::operator=(const Pair& para) {
    this->first = para.first;
    this->second = para.second;
}

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

istream& operator >>(istream& stream, Pair& p) {
    cout << "Введите число first и second через пробел: ";
    stream >> p.first >> p.second;
    return stream;
}

```

```

bool Pair::operator >(const Pair& p) {
    if (this->first < p.first) {
        return false;
    }
    if (this->first == p.first && this->second < p.second) {
        return false;
    }
    return true;
}
bool Pair::operator <(const Pair& p) {
    if (this->first > p.first) {
        return false;
    }
    if (this->first == p.first && this->second > p.second) {
        return false;
    }
    return true;
}

```

## Файл Class\_11-13.1.cpp

```

#include <iostream>
#include <list>
#include <functional>
#include <algorithm>
#include "Pair.h"
using namespace std;

ostream& operator<<(ostream& stream, list<Pair>& l) {
    for_each(l.begin(), l.end(), [](Pair& p) {cout << p << endl; });
    return stream;
}

void randomize(Pair& p) {
    int x = rand() % 1001;
    double y = (rand() % 100001)*0.01;
    p = Pair(x, y);
}

int main(){
    system("chcp 1251 > Null");
    srand(time(0));
    /*Задание 1 - Список из Pair*/

    int N;
    do {
        cout << "Введите размер списка ";
        cin >> N;
    } while (N < 1);

    Pair sum;
    list<Pair> list_1(N); //список пар
    //заполнение списка на основе функции, которая возвращает случайную пару чисел
    generate(list_1.begin(), list_1.end(), []() {Pair p; randomize(p); return p; });

    cout << "Сгенерированный список:" << endl << list_1 << endl; //вывод списка

    //добавляем каждый элемент к сумме пар чисел
    for_each(list_1.begin(), list_1.end(), [&sum](Pair& p) {sum = sum + p; }); //общая сумма

    //вычисляем среднее арифметическое

```

```

sum.setFirst(sum.getFirst() / list_1.size());//первое число
sum.setSecond(((sum.getSecond() * 10000) / list_1.size()) * 0.0001);//второе число
cout << "Среднее арифметическое " << sum << endl << endl;

list_1.push_back(sum);//добавляю в конец среднее арифметическое
cout << "После добавления среднего арифметического:" << endl << list_1 << endl;

Pair one_el, two_el;//пары для диапазона

cout << "Введите две пары чисел, из диапазона которых будут удалены пары из списка" << endl;
cout << "Первая пара: " << endl;
cin >> one_el;
cout << "Вторая пара: " << endl;
cin >> two_el;

if (one_el > two_el) {//если числа не по возрастанию, то меняю их местами
    swap(one_el, two_el);
}

list<Pair>::iterator s = remove_if(list_1.begin(), list_1.end(), [one_el, two_el](Pair& p) {return p > one_el && p <
two_el; });
list_1.erase(s, list_1.end());//удаляю из диапазона

cout << endl << "После удаления из диапазона" << endl << list_1 << endl;

//minmax_element возвращает пару из STL first и second это итераторы
pair<list<Pair>::iterator, list<Pair>::iterator> stl_pair = minmax_element(list_1.begin(), list_1.end());
cout << "Минимальный элемент: " << *stl_pair.first << endl;
cout << "Максимальный элемент: " << *stl_pair.second << endl << endl;

Pair sum_pair(stl_pair.first->getFirst() + stl_pair.second->getFirst(), stl_pair.first->getSecond() + stl_pair.second-
>getSecond());
cout << "Сумма максимального и минимального элементов " << sum_pair << endl;

for_each(list_1.begin(), list_1.end(), [&sum_pair](Pair& p) {p = p + sum_pair; });
cout << "После добавления суммы максимального и минимального к каждой паре чисел: " << endl;
cout << list_1 << endl;

cout << "Сортировка по Возрастанию: " << endl;
list_1.sort([](Pair& a, Pair& b) {return a < b; });
cout << list_1 << endl;

cout << "Сортировка по Убыванию: " << endl;
list_1.sort([](Pair& a, Pair& b) {return a > b; });
cout << list_1 << endl;

cout << "Введите значение first, по которому необходимо найти пару: ";
int ptr;
cin >> ptr;

auto i = find_if(list_1.begin(), list_1.end(), [&ptr](Pair& p) {return ptr == p.getFirst(); });
if (i != end(list_1)) {
    cout <<"Элемент найден: " << *i << endl;
}
else {
    cout << "Элемент не найден(" << endl;
}

return 0;
}

```

## Результаты работы



Введите размер списка 5

Сгенерированный список:

(531 : 79.71)

(510 : 27.48)

(461 : 30.03)

(965 : 11.94)

(502 : 151.73)

Среднее арифметическое (593 : 60.178)

После добавления среднего арифметического:

(531 : 79.71)

(510 : 27.48)

(461 : 30.03)

(965 : 11.94)

(502 : 151.73)

(593 : 60.178)

Введите две пары чисел, из диапазона которых будут удалены пары из списка

Первая пара:

Введите число first и second через пробел: 10 23

Вторая пара:

Введите число first и second через пробел: 45 56

После удаления из диапазона

(531 : 79.71)

(510 : 27.48)

(461 : 30.03)

(965 : 11.94)

(502 : 151.73)

(593 : 60.178)

Минимальный элемент: (461 : 30.03)

Максимальный элемент: (965 : 11.94)

Сумма максимального и минимального элементов (1426 : 41.97)

После добавления суммы максимального и минимального к каждой паре чисел:

(1957 : 121.68)

(1936 : 69.45)

(1887 : 72)

(2391 : 53.91)

(1928 : 193.7)

(2019 : 102.148)

Сортировка по Возрастанию:

(1887 : 72)

(1928 : 193.7)

(1936 : 69.45)

(1957 : 121.68)

(2019 : 102.148)

(2391 : 53.91)

Сортировка по Убыванию:

(2391 : 53.91)

(2019 : 102.148)

(1957 : 121.68)

(1936 : 69.45)

(1928 : 193.7)

(1887 : 72)

Введите значение first, по которому необходимо найти пару: 2019

Элемент найден: (2019 : 102.148)

## Анализ задачи 2

- 1) Необходимо реализовать класс `Pair`. В классе надо прописать конструкторы: по умолчанию, с параметрами и копирования; а также деструктор, для предотвращения утечки памяти.
- 2) Перегруженные операторы `+`, `-`, `=` и `<<` позволяют выполнять различные операции с объектами класса `Pair`. Например, оператор `+` позволяет складывать целое число с первым полем объекта класса и вещественное число со вторым полем соответственно. Оператор `<<` позволяет выводить объекты класса `Pair` в поток вывода.
- 3) Для использования очереди необходимо подключить библиотеку `#include <queue>`.
- 4) При создании очереди с приоритетом необходимо указать какие объекты будут в нем храниться. В данном случае – объекты класса `Pair`: `priority_queue<Pair, vector<Pair>, Greater_Than_Pair> qst`.
- 5) Для использования лямда функций и других необходимо подключить: `#include <functional>`, `#include <algorithm>`.
- 6) При поиске максимального объекта будем считать, что необходимо сравнивать сначала по первому числу, а затем по второму числу в паре.
- 7) В главной функции показаны операции, которые можно выполнить с контейнером.

## UML диаграмма

Pair
- first: int - second: double
+ Pair() + Pair(int, double) + Pair(const Pair&) + ~Pair() + Print(): void + operator+(Pair& p): Pair + operator-(const Pair& para): Pair + setFirst(const int& x): void + setSecond(const double& y): void + getFirst(): int + getSecond(): double + operator=(const Pair& para): void + operator <<(ostream& stream, const Pair& p): friend ostream& + operator >>(istream& stream, Pair& p): friend istream& + operator >(const Pair& p): bool + operator <(const Pair& p): bool

Greater_Than_Pair
+ operator()(Pair& p1, Pair& p2): bool

## Код

### Файл Pair.h

```
#pragma once //предотвращает повторную загрузку заголовочного файла, если он уже был включен
#include <iostream> //стандартные потоки ввода и вывода
using namespace std;

class Pair {
    int first;
    double second;

public:
    Pair();//конструктор класса по умолчанию
    Pair(int, double);//конструктор класса с параметрами
    Pair(const Pair&);//конструктор копирования класса
    ~Pair();//деструктор класса
    void Print();

    Pair operator+(Pair& p);
    Pair operator-(const Pair& para);
    void setFirst(const int& x);
    void setSecond(const double& y);
    int getFirst();
    double getSecond();
    void operator=(const Pair& para);
    friend ostream& operator <<(ostream& stream, const Pair& p);
    friend istream& operator >>(istream& stream, Pair& p);
    bool operator >(const Pair& p);
    bool operator <(const Pair& p);
};
```

## Файл Pair.cpp

```
#include "Pair.h"
#include <iostream>
#include <string>
using namespace std;

Pair::Pair() {
    this->first = 0;
    this->second = 0;
}

Pair::Pair(int first, double second) {
    this->first = first;
    this->second = second;
}

Pair::Pair(const Pair& other) { //конструктор копирования
    this->first = other.first;
    this->second = other.second;
}

Pair::~Pair() { //деструктор
}

void Pair::Print() { //вывод параметров класса person
    cout << "(" << this->first << " : " << this->second << ")" << endl;
}

Pair Pair::operator+(Pair& p) {
    return (Pair(this->first + p.first, this->second + p.second));
}

Pair Pair::operator-(const Pair& para) {
    return (Pair(this->first - para.first, this->second - para.second));
}

void Pair::setFirst(const int& x) {
    this->first = x;
}

void Pair::setSecond(const double& y) {
    this->second = y;
}

int Pair::getFirst() {
    return this->first;
}

double Pair::getSecond() {
    return this->second;
}

void Pair::operator=(const Pair& para) {
    this->first = para.first;
    this->second = para.second;
}

ostream& operator << (ostream& stream, const Pair& p) {
    stream << "(" << p.first << " : " << p.second << ")";
    return stream;
}
```

```

istream& operator >>(istream& stream, Pair& p) {
    cout << "Введите число first и second через пробел: ";
    stream >> p.first >> p.second;
    return stream;
}

bool Pair::operator >(const Pair& p) {
    if (this->first < p.first) {
        return false;
    }
    if (this->first == p.first && this->second < p.second) {
        return false;
    }
    return true;
}

bool Pair::operator <(const Pair& p) {
    if (this->first > p.first) {
        return false;
    }
    if (this->first == p.first && this->second > p.second) {
        return false;
    }
    return true;
}

```

## Файл Class\_11-13.2.cpp

```

#include <iostream>
#include <queue>
#include <functional>
#include <algorithm>
#include "Pair.h"
#include <random>
using namespace std;

void randomize(Pair& p) {
    int x = rand() % 1001;
    double y = (rand() % 100001) * 0.01;
    p = Pair(x, y);
}

class Greater_Than_Pair{
public:
    bool operator()(Pair& p1, Pair& p2){
        return p1 < p2;
    }
};

void generate_Q(priority_queue<Pair, vector<Pair>, Greater_Than_Pair>& my_Q, int Q_size) {
    Pair qwe(0,0);
    for (int i = 0; i < Q_size; i++) {
        randomize(qwe);
        my_Q.push(qwe);
    }
}

ostream& operator<<(ostream& stream, priority_queue<Pair, vector<Pair>, Greater_Than_Pair> my_Q) {
    while (!my_Q.empty()) {
        Pair user = my_Q.top();
        std::cout << user << endl;
        my_Q.pop();
    }
}

```

```

    return stream;
}

Pair all_sum(priority_queue<Pair, vector<Pair>, Greater_Than_Pair> my_Q) {
    Pair p(0, 0);
    while (!my_Q.empty()) {
        Pair upp = my_Q.top();
        p = p + upp;
        my_Q.pop();
    }
    return p;
}

void remove_from_range(priority_queue<Pair, vector<Pair>, Greater_Than_Pair>& my_Q, Pair& range_1, Pair&
range_2) {
    priority_queue<Pair, vector<Pair>, Greater_Than_Pair> queue_1;
    while (!my_Q.empty()) {
        Pair upp = my_Q.top();
        if (!(upp > range_1 && upp < range_2)) {
            queue_1.push(upp);
        }
        my_Q.pop();
    }
    my_Q = queue_1;
}

Pair maximum_pair(priority_queue<Pair, vector<Pair>, Greater_Than_Pair> my_Q) {
    Pair max_p = my_Q.top();
    return max_p;
}

Pair minimum_pair(priority_queue<Pair, vector<Pair>, Greater_Than_Pair> my_Q) {
    Pair min_p;
    while (!my_Q.empty()) {
        min_p = my_Q.top();
        my_Q.pop();
    }
    return min_p;
}

priority_queue<Pair, vector<Pair>, Greater_Than_Pair> add_to_all(priority_queue<Pair,
vector<Pair>, Greater_Than_Pair>& my_Q, Pair& para) {
    priority_queue<Pair, vector<Pair>, Greater_Than_Pair> queue_1;
    Pair upp = my_Q.top();
    while (!my_Q.empty()) {
        upp = my_Q.top();
        queue_1.push(upp + para);
        my_Q.pop();
    }
    return queue_1;
}

bool find_para(priority_queue<Pair, vector<Pair>, Greater_Than_Pair> my_Q, int& x, Pair& p) {
    bool flag = false;
    while (!my_Q.empty() && !flag) {
        p = my_Q.top();
        if (x == p.getFirst()) {
            flag = true;
        }
        my_Q.pop();
    }
    return flag;
}

```

```

}

int main() {
    system("chcp 1251 > Null");
    srand(time(0));
    /*Задание 2 - очередь с приоритетами из Pair*/

    int N;
    do {
        cout << "Введите размер очереди ";
        cin >> N;
    } while (N < 1);

    priority_queue<Pair, vector<Pair>, Greater_Than_Pair> qst;
    generate_Q(qst, N);

    cout << qst;

    Pair sum = all_sum(qst);

    //вычисляем среднее арифметическое
    sum.setFirst(sum.getFirst() / qst.size()); //первое число
    sum.setSecond(((sum.getSecond() * 1000) / qst.size()) * 0.001); //второе число
    cout << endl << "Среднее арифметическое " << sum << endl << endl;

    qst.push(sum); //добавляю в конец среднее арифметическое
    cout << "После добавления среднего арифметического." << endl << qst << endl;

    Pair one_el, two_el; //пары для диапазона

    cout << "Введите две пары чисел, из диапазона которых будут удалены пары из списка" << endl;
    cout << "Первая пара: " << endl;
    cin >> one_el;
    cout << "Вторая пара: " << endl;
    cin >> two_el;

    if (one_el > two_el) { //если числа не по возрастанию, то меняю их местами
        swap(one_el, two_el);
    }

    remove_from_range(qst, one_el, two_el);
    cout << endl << "После удаления из диапазона" << endl << qst << endl;

    Pair max_el = maximum_pair(qst);
    Pair min_el = minimum_pair(qst);

    cout << "Минимальный элемент: " << min_el << endl;
    cout << "Максимальный элемент: " << max_el << endl << endl;

    sum = max_el + min_el;
    cout << "Сумма максимального и минимального элементов " << sum << endl;
    qst = add_to_all(qst, sum);
    cout << "После добавления суммы максимального и минимального к каждой паре чисел: " << endl << qst;

    cout << "Введите значение first, по которому необходимо найти пару: ";
    int ptr;
    cin >> ptr;

    bool flag = find_para(qst, ptr, sum);

    if (flag) {

```

```

        cout << "Элемент найден: " << sum << endl;
    }
    else {
        cout << "Элемент не найден(" << endl;
    }

    return 0;
}

```

## Результаты работы

```

Введите размер очереди 5
(941 : 36.41)
(570 : 72.58)
(309 : 307.44)
(211 : 308.98)
(164 : 48.43)

Среднее арифметическое (439 : 154.768)

После добавления среднего арифметического:
(941 : 36.41)
(570 : 72.58)
(439 : 154.768)
(309 : 307.44)
(211 : 308.98)
(164 : 48.43)

Введите две пары чисел, из диапазона которых будут удалены пары из списка
Первая пара:
Введите число first и second через пробел: 400 52
Вторая пара:
Введите число first и second через пробел: 450 40.23

После удаления из диапазона
(941 : 36.41)
(570 : 72.58)
(309 : 307.44)
(211 : 308.98)
(164 : 48.43)

Минимальный элемент: (164 : 48.43)
Максимальный элемент: (941 : 36.41)

Сумма максимального и минимального элементов (1105 : 84.84)
После добавления суммы максимального и минимального к каждой паре чисел:
(2046 : 121.25)
(1675 : 157.42)
(1414 : 392.28)
(1316 : 393.82)
(1269 : 133.27)
Введите значение first, по которому необходимо найти пару: 456
Элемент не найден(

```

## Анализ задачи 3

1) Необходимо реализовать класс Pair. В классе надо прописать конструкторы: по умолчанию, с параметрами и копирования; а также деструктор, для предотвращения утечки памяти.



2) Перегруженные операторы `+`, `-`, `=` и `<<` позволяют выполнять различные операции с объектами класса `Pair`. Например, оператор `+` позволяет складывать целое число с первым полем объекта класса и вещественное число со вторым полем соответственно. Оператор `<<` позволяет выводить объекты класса `Pair` в поток вывода.

3) Для использования словаря необходимо подключить библиотеку `#include <map>`.

4) При создании очереди с приоритетом необходимо указать какие объекты будут в нем храниться. В данном случае – объекты класса `Pair`: `map<Pair, Pair>`.

5) Для использования лямда функций и других необходимо подключить: `#include <functional>`, `#include <algorithm>`.

6) При поиске максимального объекта будем считать, что необходимо сравнивать сначала по первому числу, а затем по второму числу в паре.

7) В главной функции показаны операции, которые можно выполнить с контейнером.

## **UML диаграмма**

Pair
- first: int - second: double
+ Pair() + Pair(int, double) + Pair(const Pair&) + ~Pair() + Print(): void + operator+(Pair& p): Pair + operator-(const Pair& para): Pair + setFirst(const int& x): void + setSecond(const double& y): void + getFirst(): int + getSecond(): double + operator=(const Pair& para): void + operator <<(ostream& stream, const Pair& p): friend ostream& + operator >>(istream& stream, Pair& p): friend istream& + operator >(const Pair& p): bool + operator <(const Pair& p): bool + operator==(const Pair& a, const Pair& b): friend bool + operator/(const int& x): void + operator >(const Pair& p): bool + operator <(const Pair& p): bool

## Код

### Файл Pair.h

```
#pragma once //предотвращает повторную загрузку заголовочного файла, если он уже был включен
#include <iostream> //стандартные потоки ввода и вывода
using namespace std;

class Pair {
    int first;
    double second;

public:
    Pair();//конструктор класса по умолчанию
    Pair(int, double);//конструктор класса с параметрами
    Pair(const Pair&);//конструктор копирования класса
    ~Pair();//деструктор класса
    void Print();

    Pair operator+(Pair& p);
    Pair operator-(const Pair& para);
    void setFirst(const int& x);
    void setSecond(const double& y);
    int getFirst();
    double getSecond();
    void operator=(const Pair& para);
    friend ostream& operator <<(ostream& stream, const Pair& p);
    friend istream& operator >>(istream& stream, Pair& p);
    bool operator >(const Pair& p);
    bool operator <(const Pair& p);
    friend bool operator==(const Pair& a, const Pair& b);
    void operator/(const int& x);
    friend bool operator<(const Pair& a, const Pair& b);
```

```
        friend bool operator>(const Pair& a, const Pair& b);
};
```

## Файл Pair.cpp

```
#include "Pair.h"
#include <iostream>
#include <string>
using namespace std;

Pair::Pair() {
    this->first = 0;
    this->second = 0;
}

Pair::Pair(int first, double second) {
    this->first = first;
    this->second = second;
}

Pair::Pair(const Pair& other) { //конструктор копирования
    this->first = other.first;
    this->second = other.second;
}

Pair::~Pair() { //деструктор
}

void Pair::Print() { //вывод параметров класса person
    cout << "(" << this->first << " : " << this->second << ")" << endl;
}

Pair Pair::operator+(Pair& p) {
    return (Pair(this->first + p.first, this->second + p.second));
}

Pair Pair::operator-(const Pair& para) {
    return (Pair(this->first - para.first, this->second - para.second));
}

void Pair::setFirst(const int& x) {
    this->first = x;
}

void Pair::setSecond(const double& y) {
    this->second = y;
}

int Pair::getFirst() {
    return this->first;
}

double Pair::getSecond() {
    return this->second;
}

void Pair::operator=(const Pair& para) {
    this->first = para.first;
    this->second = para.second;
}

ostream& operator <<(ostream& stream, const Pair& p) {
```

```

        stream << "(" << p.first << " : " << p.second << ")";
        return stream;
    }
    istream& operator >>(istream& stream, Pair& p) {
        cout << "Введите число first и second через пробел: ";
        stream >> p.first >> p.second;
        return stream;
    }

    bool Pair::operator >(const Pair& p) {
        if (this->first < p.first) {
            return false;
        }
        if (this->first == p.first && this->second < p.second) {
            return false;
        }
        return true;
    }
    bool Pair::operator <(const Pair& p) {
        if (this->first > p.first) {
            return false;
        }
        if (this->first == p.first && this->second > p.second) {
            return false;
        }
        return true;
    }

    bool operator==(const Pair& a, const Pair& b) {
        return a.first == b.first && a.second == b.second;
    }

    void Pair::operator/(const int& x) {
        this->first = this->first / x;
        this->second = ((this->second * 10000) / x) * 0.0001;
    }

    bool operator<=(const Pair& a, const Pair& b) {
        if (a.first >= b.first) {
            return false;
        }
        if (a.first == b.first && a.second >= b.second) {
            return false;
        }
        return true;
    }

    bool operator>(const Pair& a, const Pair& b) {
        if (a.first < b.first) {
            return false;
        }
        if (a.first == b.first && a.second < b.second) {
            return false;
        }
        return true;
    }
}

```

Файл Class\_11-13.3.cpp

```

#include <iostream>
#include <functional>
#include <algorithm>

```

```

#include "Pair.h"
#include <map>
using namespace std;

void randomize(Pair& p) {
    int x = rand() % 1001;
    double y = (rand() % 100001) * 0.01;
    p = Pair(x, y);
}

void generate_M(map<Pair, Pair>& my_M, int M_size) {
    Pair qwe_1(0, 0), qwe_2(0, 0);
    for (int i = 0; i < M_size; i++) {
        randomize(qwe_1);
        randomize(qwe_2);
        my_M[qwe_1] = qwe_2;
    }
}

ostream& operator<<(ostream& stream, const pair<Pair, Pair> p){
    stream <<"Ключ - " << p.first << "    Значение - " << p.second << "\n";
    return stream;
}

ostream& operator<<(ostream& stream, const map<Pair, Pair>& my_M) {
    map<Pair, Pair>::const_iterator It = my_M.begin();
    while (It != my_M.end())
        cout << *It++;
    return stream;
}

bool isInRange(Pair& key, Pair& lower, Pair& upper) {
    return key > lower && key < upper;
}

void remove_from_range(map<Pair, Pair>& my_M, Pair& range_1, Pair& range_2) {
    auto it = my_M.begin();
    Pair ptr;
    while (it != my_M.end()) { // Перебор всех пар ключ-значение
        ptr = it->first;
        if (!isInRange(ptr, range_1, range_2)) {
            ++it;
        }
        else {
            it = my_M.erase(it);
        }
    }
}

Pair minimum_pair(map<Pair, Pair>& my_M) {
    auto it = my_M.begin();
    Pair ptr;
    ptr = it->first;
    return ptr;
}

Pair maximum_pair(map<Pair, Pair>& my_M) {
    auto it = my_M.begin();
    Pair ptr;
    while (it != my_M.end()) { // Перебор всех пар ключ-значение
        ptr = it->first;
        ++it;
    }
}

```

```

    }
    return ptr;
}

void all_sum(map<Pair, Pair>& my_M, Pair& key, Pair& value) {
    for (const auto& pair : my_M) { // Перебор всех пар ключ-значение
        Pair asd = pair.first;
        key = key + asd;
        asd = pair.second;
        value = value + asd;
    }
}

map<Pair, Pair> add_to_all(map<Pair, Pair>& my_M, Pair& key, Pair& value) {
    map<Pair, Pair> ptr;
    for (const auto& pair : my_M) { // Перебор всех пар ключ-значение
        Pair asd_1 = pair.first;
        Pair asd_2 = pair.second;
        ptr[asd_1 + key] = (asd_2 + value);
    }
    return ptr;
}

bool findPair(map<Pair, Pair>& my_M, Pair& key, Pair& value) {
    auto it = my_M.find(key);
    if (it != my_M.end()) {
        value = Pair(it->second);
        return true;
    }
    else {
        return false;
    }
}

int main() {
    system("chcp 1251 > Null");
    srand(time(0));
    /*Задание 3 - Словарь из Pair*/

    int N;
    do {
        cout << "Введите размер словаря ";
        cin >> N;
    } while (N < 1);

    Pair sum_1(0,0), sum_2(0,0), sum_key(0, 0), sum_value(0, 0);

    map<Pair, Pair> adj;//словарь пар
    generate_M(adj, N);

    cout << "Сгенерированный словарь:" << endl << adj << endl; //вывод словаря

    //добавляем каждый элемент к сумме пар чисел
    all_sum(adj, sum_1, sum_2); //общая сумма
    sum_1 / adj.size();//Среднее арифметическое ключей
    sum_2 / adj.size();//Среднее арифметическое значений

    cout << "Среднее арифметическое ключей " << sum_1 << endl;
    cout << "Среднее арифметическое значений " << sum_2 << endl << endl;

    adj[sum_1] = sum_2;//добавляю в конец среднее арифметическое
    cout << "После добавления среднего арифметического:" << endl << adj << endl;
}

```

```

Pair one_el, two_el; // пары для диапазона

cout << "Введите две пары чисел (ключей), из диапазона которых будут удалены пары из списка" << endl;
cout << "Первая пара: " << endl;
cin >> one_el;
cout << "Вторая пара: " << endl;
cin >> two_el;

if (one_el > two_el) { // если числа не по возрастанию, то меняю их местами
    swap(one_el, two_el);
}

remove_from_range(adj, one_el, two_el);
cout << endl << "После удаления из диапазона" << endl << adj << endl;

Pair max_el = maximum_pair(adj);
Pair min_el = minimum_pair(adj);

findPair(adj, min_el, sum_1); // Вызываем функцию поиска
findPair(adj, max_el, sum_2); // Вызываем функцию поиска

cout << "Минимальный элемент по ключу: " << min_el << endl;
cout << "Ключ - " << min_el << " Значение - " << sum_1 << endl;
cout << "Максимальный элемент по ключу: " << max_el << endl;
cout << "Ключ - " << max_el << " Значение - " << sum_2 << endl << endl;

sum_key = min_el + max_el;
sum_value = sum_1 + sum_2;
cout << "Сумма максимального и минимального элементов:" << endl;
cout << "Ключ - " << sum_key << " Значение - " << sum_value << endl << endl;

adj = add_to_all(adj, sum_key, sum_value);
cout << "После добавления суммы максимального и минимального к каждой паре чисел: " << endl;
cout << adj << endl;

cout << "Введите ключевое значение, по которому необходимо найти пару" << endl;
cin >> one_el;

bool found = findPair(adj, one_el, two_el); // Вызываем функцию поиска

cout << endl;
if (found) {
    cout << "Элемент найден." << endl;
    cout << "Ключ - " << one_el << " Значение - " << two_el << endl;
}
else {
    cout << "Элемент не найден(" << endl;
}

return 0;
}

```

## Результаты работы

```

Введите размер словаря 5
Сгенерированный словарь:
Ключ - (270 : 221.81)      Значение - (433 : 158.42)
Ключ - (456 : 139.92)      Значение - (603 : 257.49)
Ключ - (564 : 11.98)       Значение - (892 : 304.82)
Ключ - (836 : 123.21)      Значение - (533 : 218.39)
Ключ - (915 : 185.78)      Значение - (380 : 21.11)

Среднее арифметическое ключей (608 : 136.54)
Среднее арифметическое значений (568 : 192.046)

После добавления среднего арифметического:
Ключ - (270 : 221.81)      Значение - (433 : 158.42)
Ключ - (456 : 139.92)      Значение - (603 : 257.49)
Ключ - (564 : 11.98)       Значение - (892 : 304.82)
Ключ - (608 : 136.54)      Значение - (568 : 192.046)
Ключ - (836 : 123.21)      Значение - (533 : 218.39)
Ключ - (915 : 185.78)      Значение - (380 : 21.11)

Введите две пары чисел (ключей), из диапазона которых будут удалены пары из списка
Первая пара:
Введите число first и second через пробел: 400 56.23
Вторая пара:
Введите число first и second через пробел: 570 56

После удаления из диапазона
Ключ - (270 : 221.81)      Значение - (433 : 158.42)
Ключ - (608 : 136.54)      Значение - (568 : 192.046)
Ключ - (836 : 123.21)      Значение - (533 : 218.39)
Ключ - (915 : 185.78)      Значение - (380 : 21.11)

Минимальный элемент по ключу: (270 : 221.81)
Ключ - (270 : 221.81)      Значение - (433 : 158.42)
Максимальный элемент по ключу: (915 : 185.78)
Ключ - (915 : 185.78)      Значение - (380 : 21.11)

Сумма максимального и минимального элементов:
Ключ - (1185 : 407.59)      Значение - (813 : 179.53)

После добавления суммы максимального и минимального к каждой паре чисел:
Ключ - (1455 : 629.4)      Значение - (1246 : 337.95)
Ключ - (1793 : 544.13)     Значение - (1381 : 371.576)
Ключ - (2021 : 530.8)      Значение - (1346 : 397.92)
Ключ - (2100 : 593.37)     Значение - (1193 : 200.64)

Введите ключевое значение, по которому необходимо найти пару
Введите число first и second через пробел: 2100 593.37

Элемент найден.
Ключ - (2100 : 593.37)      Значение - (1193 : 200.64)

```

## Вывод

В ходе работы я применила знания о работе с классами и о стандартных обобщенных алгоритмах библиотеки STL. В ходе работы были созданы: последовательный контейнер, адаптер контейнера, ассоциативный контейнер. Контейнеры были заполнены элементами пользовательского типа











Pair. В основной функции были продемонстрированы необходимые операции.

## GitHub

Ссылка: [https://github.com/SonyAkb/Laboratory-works-for-the-2-semester/tree/main/class\\_13](https://github.com/SonyAkb/Laboratory-works-for-the-2-semester/tree/main/class_13)

Laboratory-works-for-the-2-semester / class\_13 / 

 SonyAkb Add files via upload	
Name	Last commit message
 ..	
 class_13-1	Add files via upload
 class_13-2	Add files via upload
 class_13-3	Add files via upload
 uml_13-1.drawio.png	Обновление uml_13-1.drawio.png
 uml_13-2.drawio.png	Обновление uml_13-2.drawio.png
 uml_13-3.drawio.png	Обновление uml_13-3.drawio.png