Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение

высшего образования

**«Пермский национальный исследовательский**

**политехнический университет»**

Электротехнический факультет

Кафедра «Информационные технологии и автоматизированные системы»

направление подготовки: 09.03.01 - «Информатика и вычислительная техника»

**О Т Ч Е Т**

**по лабораторной работе “Классы” №13**

**по дисциплине**

**«Основы алгоритмизации и программирования» семестр 2**

Выполнил студент гр. ИВТ-21-1б

Ипатов Дмитрий Сергеевич

Проверил:

Ст. Преподаватель кафедры ИТАС

Яруллин Д.В.

(оценка) (подпись)

г. Пермь-2022

**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()).

* + - 1. Удалить элементы в соответствии с заданием (использовать алгоритмы

remove(),remove\_if(), remove\_copy\_if(),remove\_copy())

* + - 1. Отсортировать контейнер по убыванию и по возрастанию ключевого поля (использовать алгоритм sort()).
      2. Найти в контейнере заданный элемент (использовать алгоритмы find(), find\_if(), count(), count\_if()).
      3. Выполнить задание варианта для полученного контейнера (использовать алгоритм

for\_each()) .

* + - 1. 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()).

1. Удалить элементы в соответствии с заданием (использовать алгоритмы

remove(),remove\_if(), remove\_copy\_if(),remove\_copy())

1. Отсортировать контейнер по убыванию и по возрастанию ключевого поля (использовать алгоритм sort()).
2. Найти в контейнере элемент с заданным ключевым полем (использовать

алгоритмы find(), find\_if(), count(), count\_if()).

1. Выполнить задание варианта для полученного контейнера (использовать алгоритм

for\_each()) .

1. 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()).

1. Удалить элементы в соответствии с заданием (использовать алгоритмы

remove(),remove\_if(), remove\_copy\_if(),remove\_copy())

1. Отсортировать контейнер по убыванию и по возрастанию ключевого поля (использовать алгоритм sort()).
2. Найти в контейнере элемент с заданным ключевым полем (использовать

алгоритмы find(), find\_if(), count(), count\_if()).

1. Выполнить задание варианта для полученного контейнера (использовать алгоритм

for\_each()) .

1. Для выполнения всех заданий использовать стандартные алгоритмы библиотеки STL.

**Анализ задачи:**

**Программа 1:**

1. Создаем класс Pair, в котором есть два поля int first, double second, прописываем все методы, геттеры, сеттеры, перегружаем операторы ввода и вывода;
2. Создаем очередь, в которую будем записывать элементы типа Pair;
3. Создаем функции создания очереди и добавления элементов в очередь;
4. Ищем максимальный элемент в очереди и добавляем его в конец;
5. Вводим ключ, по которому найдем элемент для удаления;
6. Считаем среднее значение элементов в очереди и прибавляем его к каждому элементу очереди.

**Программа 2:**

Всё делаем аналогично первой программе, но теперь кроме двунаправленной очереди у нас появляется её адаптер в виде стека.

**Программа 3:**

Делаем всё аналогично первой программе, но теперь вместо двунаправленной очереди в роли контейнера будет множество.

**Код программы 1:**

#include <iostream>

#include <deque>

#include <functional>

#include <algorithm>

using namespace std;

class Pair

{

private:

int first;

double second;

public:

Pair(double m, int n);

Pair();

Pair(const Pair& t);

~Pair();

int get\_first() { return first; }

double get\_second() { return second; }

void set\_first(int n) { first = n; }

void set\_second(int m) { second = m; }

friend istream& operator>>(istream& in, Pair& p);

friend ostream& operator<<(ostream& out, const Pair& p);

Pair& operator--();

Pair operator --(int);

bool operator>(Pair& t);

bool operator<(Pair& t);

Pair& operator=(const Pair& t);

bool operator==(const Pair& t);

Pair operator+(const Pair& t);

Pair operator/(int k);

};

Pair Pair::operator/(int k)

{

first = first / k;

second = second / k;

return \*this;

}

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

{

first += t.first;

second += t.second;

return \*this;

}

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

{

if (first == t.first && second == t.second) return true;

return false;

}

Pair::Pair(double m, int n)

{

first = n;

second = m;

}

Pair::Pair()

{

first = 0;

second = 0;

}

Pair::Pair(const Pair& t)

{

first = t.first;

second = t.second;

}

Pair::~Pair() = default;

istream& operator>>(istream & in, Pair & p)

{

cout << "first:"; in >> p.first;

cout << "second:"; in >> p.second;

return in;

}

ostream& operator<<(ostream & out, const Pair & p)

{

return (out << p.first << ":" << p.second << endl);

}

Pair Pair::operator --(int)

{

first--;

return \*this;

}

Pair& Pair::operator--()

{

second--;

return \*this;

}

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

{

if (&t == this) return \*this;

first = t.first;

second = t.second;

return \*this;

}

bool Pair::operator<(Pair & t)

{

if (first < t.first && second < t.second) return true;

else return false;

}

bool Pair::operator>(Pair & t)

{

if (first > t.first && second > t.second) return true;

else return false;

}

typedef deque<Pair> Tdeque;

Pair s;

Pair p;

Tdeque make\_deque(int n)

{

Pair a;

Tdeque d;

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

{

cin >> a;

d.push\_back(a);

}

return d;

}

Tdeque make\_empty\_deque(int n)

{

Pair a(0, 0);

Tdeque k;

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

{

k.push\_back(a);

}

return k;

}

void print\_deque(Tdeque d)

{

if (!d.empty())

{

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

cout << d[i];

}

else

{

cout << "Очередь пуста!";

}

cout << '\n';

}

struct Equal\_s

{

bool operator() (Pair t)

{

return t == s;

}

};

Pair srednee(Tdeque d)

{

Pair s = d[0];

int size = d.size();

for (int i = 1; i < size; i++)

s = s + d[i];

return s / d.size();

}

void Foo(Pair & b)

{

b = b + p;

}

int main()

{

setlocale(LC\_ALL, "rus");

int n = 0;

cout << "Введите размер очереди: ";

cin >> n;

Tdeque d = make\_deque(n);

cout << "\nВведенная очередь:\n";

print\_deque(d);

Tdeque::iterator i;

cout << "Ищем максимальный элемент и добавляем его в конец очереди:\n";

i = max\_element(d.begin(), d.end());

Pair max = \*i;

d.push\_back(max);

print\_deque(d);

cout << "Создадим ключ поиска типа Pair:\n";

cin >> s;

cout << "Очередь после удаления элемента, совпадающего с ключом:\n";

d.erase(remove\_if(d.begin(), d.end(), Equal\_s()), d.end());

print\_deque(d);

cout << "\nСреднее значение очереди: " << srednee(d) << '\n';

p = srednee(d);

for\_each(d.begin(), d.end(), Foo);

print\_deque(d);

return 0;

}

**Код программы 2:**

#include <iostream>

#include <deque>

#include <stack>

#include <functional>

#include <algorithm>

using namespace std;

class Pair

{

private:

int first;

double second;

public:

Pair(double m, int n);

Pair();

Pair(const Pair& t);

~Pair();

int get\_first() { return first; }

double get\_second() { return second; }

void set\_first(int n) { first = n; }

void set\_second(int m) { second = m; }

friend istream& operator>>(istream& in, Pair& p);

friend ostream& operator<<(ostream& out, const Pair& p);

Pair& operator--();

Pair operator --(int);

bool operator>(Pair& t);

bool operator<(Pair& t);

Pair& operator=(const Pair& t);

bool operator==(const Pair& t);

Pair operator+(const Pair& t);

Pair operator/(int k);

};

Pair Pair::operator/(int k)

{

first = first / k;

second = second / k;

return \*this;

}

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

{

first += t.first;

second += t.second;

return \*this;

}

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

{

if (first == t.first && second == t.second) return true;

return false;

}

Pair::Pair(double m, int n)

{

first = n;

second = m;

}

Pair::Pair()

{

first = 0;

second = 0;

}

Pair::Pair(const Pair& t)

{

first = t.first;

second = t.second;

}

Pair::~Pair() = default;

istream& operator>>(istream & in, Pair & p)

{

cout << "first:"; in >> p.first;

cout << "second:"; in >> p.second;

return in;

}

ostream& operator<<(ostream & out, const Pair & p)

{

return (out << p.first << ":" << p.second << endl);

}

Pair Pair::operator --(int)

{

first--;

return \*this;

}

Pair& Pair::operator--()

{

second--;

return \*this;

}

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

{

if (&t == this) return \*this;

first = t.first;

second = t.second;

return \*this;

}

bool Pair::operator<(Pair & t)

{

if (first < t.first && second < t.second) return true;

else return false;

}

bool Pair::operator>(Pair & t)

{

if (first > t.first && second > t.second) return true;

else return false;

}

typedef stack<Pair> St;

typedef deque<Pair> Tdeque;

Pair p;

Pair sravn;

St make\_stack(int n)

{

St s;

Pair pr;

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

{

cin >> pr;

s.push(pr);

}

return s;

}

Tdeque copy\_stack\_to\_deque(St s)

{

Tdeque d;

while (!s.empty())

{

d.push\_front(s.top());

s.pop();

}

return d;

}

St copy\_deque\_to\_stack(Tdeque d)

{

St s;

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

s.push(d[i]);

return s;

}

void print\_stack(St s)

{

Tdeque temp;

temp = copy\_stack\_to\_deque(s);

while (!s.empty())

{

cout << s.top();

s.pop();

}

s = copy\_deque\_to\_stack(temp);

cout << '\n';

}

struct Equal\_s

{

bool operator() (Pair t)

{

return t == sravn;

}

};

Pair srednee(Tdeque d)

{

Pair s = d[0];

int size = d.size();

for (int i = 1; i < size; i++)

s = s + d[i];

return s / d.size();

}

void Foo(Pair & b)

{

b = b + p;

}

int main()

{

setlocale(LC\_ALL, "rus");

int n = 0;

cout << "Введите размер стека: ";

cin >> n;

St s = make\_stack(n);

cout << "\nВведенный стек:\n";

print\_stack(s);

Tdeque d = copy\_stack\_to\_deque(s);

Tdeque::iterator i;

cout << "Ищем максимальный элемент и добавляем его в конец стека:\n";

i = max\_element(d.begin(), d.end());

Pair max = \*i;

d.push\_back(max);

s = copy\_deque\_to\_stack(d);

cout << "Стек после изменения:\n";

print\_stack(s);

cout << "Создадим ключ поиска типа Pair:\n";

cin >> sravn;

d = copy\_stack\_to\_deque(s);

cout << "Стек после удаления элементов, совпадающих с ключом:\n";

d.erase(remove\_if(d.begin(), d.end(), Equal\_s()), d.end());

s = copy\_deque\_to\_stack(d);

print\_stack(s);

cout << "\nСреднее значение элементов стека: " << srednee(d) << '\n';

p = srednee(d);

d = copy\_stack\_to\_deque(s);

for\_each(d.begin(), d.end(), Foo);

s = copy\_deque\_to\_stack(d);

cout << "Стек после прибавления к каждому его элементу среднего значения:\n";

print\_stack(s);

return 0;

}

**Код программы 3:**

#include <iostream>

#include <set>

#include <algorithm>

#include "Pair.h"

using namespace std;

typedef set<Pair> Tset;

typedef Tset::iterator it;

Pair s;

Tset make\_set(int n)//ф-ия для создания мн-ва

{

Tset a;

Pair p;

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

{

cin >> p;

a.insert(p);

}

return a;

}

void print\_set(Tset& a)//вывод мн-ва

{

it i = a.begin();

while (i != a.end())

{

cout << \*i << "\n";

i++;

}

}

void add\_max\_elem(Tset& a)//добавить макс элемент

{

const Pair b = \*max\_element(a.begin(), a.end());

a.insert(b);

}

void del\_elem(Tset& a, int n1, int n2)//удалить элементы из промежутка [n1; n2]

{

if (n1 > a.size() || n2 > a.size() || n1 > n2) { cout << "Error!\n"; return; }

it i = a.begin();

int k = 1;

while (k != n1)//найдём итератор в левой гнранице

{

i++;

k++;

}

it beg = i;

while (k != n2 + 1)//найдём итератор в правой границе

{

i++;

k++;

}

it end = i;

a.erase(beg, end);//remove использовать не получится, т.к. значение хранится как константа, а при удалении она меняется и возникает ошибка

}

Pair avereage(Tset& a)//поиск среднего

{

it i = a.begin();

Pair sum;

while (i != a.end())

{

sum = sum + \*i;

i++;

}

return sum / a.size();

}

void add\_elem(Tset& a, Pair& p)//добавить к каждому элементу новый элемент

{

Tset b;

it i = a.begin();

while (i != a.end())

{

Pair buf = \*i;

b.insert(buf + p);

i++;

}

a = b;

}

bool operator< (const Pair& a, const Pair& b)//перегрузка оператора <, необходимо для праильной коомпиляции программы

{

return (a.first < b.first) || (a.second < b.second);

}

int main()

{

int n;

cout << "Enter n >> "; cin >> n;

Tset a = make\_set(n);

cout << "\nYour set:\n";

print\_set(a);

add\_max\_elem(a);//т.к у нас множество без дубликатов, то ничего не добавится

cout << "\nAdd max elem:\n";

print\_set(a);

int n1, n2;

cout << "\nEnter n1, n2 >> "; cin >> n1 >> n2;

del\_elem(a, n1, n2);

print\_set(a);

s = avereage(a);

add\_elem(a, s);//т.к. значения хранятся как const, то операции с ними невозможны, т.е for\_each использовать невозможно

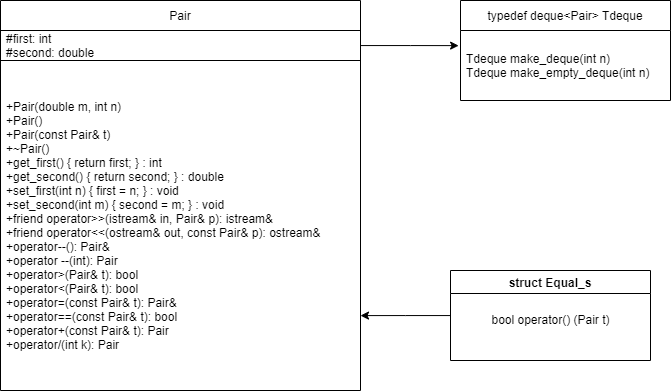
cout << "\nAvereage: " << s << "\nAdd avereage : \n";

print\_set(a);

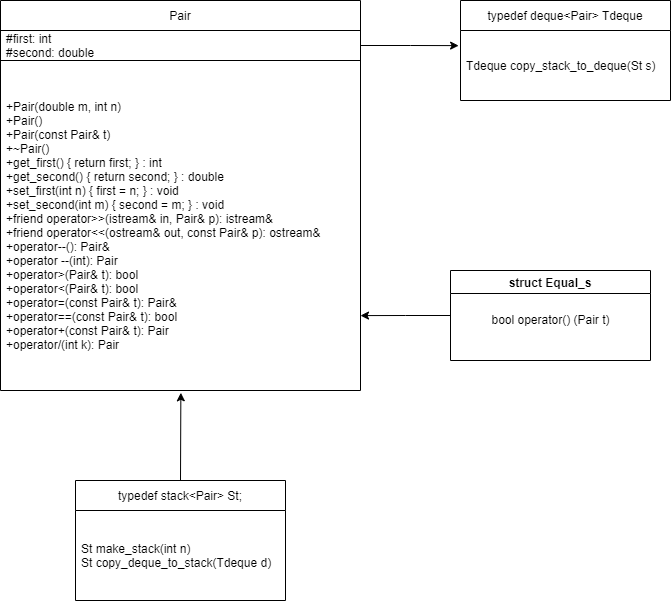
return 0;

}

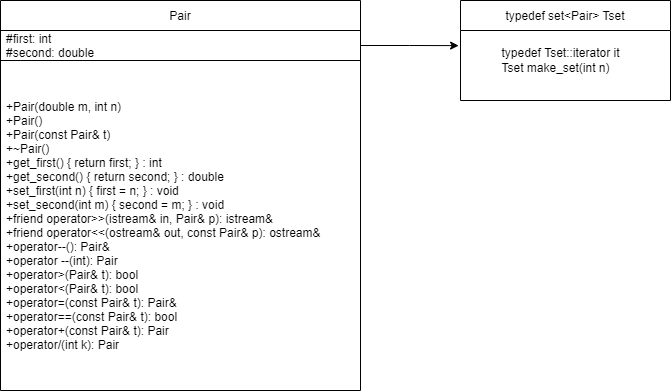
**UML 1 программа:**

****

**UML 2 программа:**

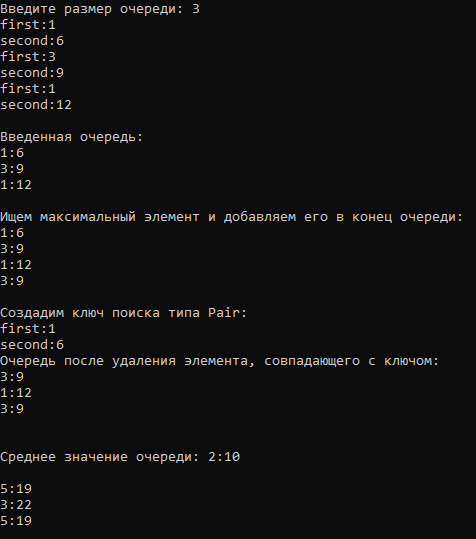
****

**UML 3 программа:**

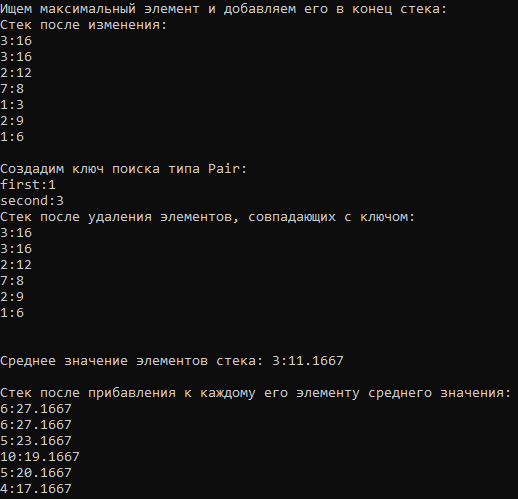
****

**Результаты программ:**

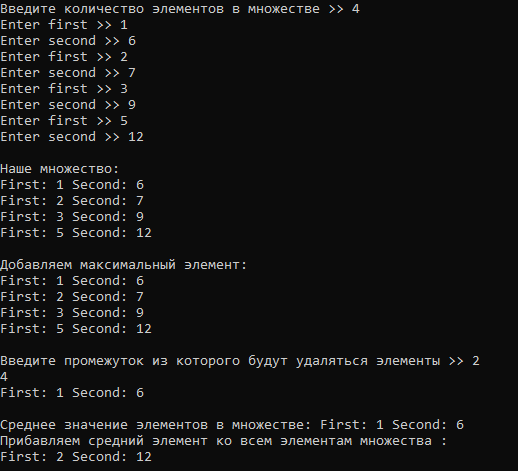
**Программа 1:**

****

**Программа 2:**

****

**Программа 3:**

****