Министерство образования и науки Российской Федерации  
Федеральное‌ ‌государственное‌ ‌бюджетное‌ ‌образовательное‌ ‌учреждение‌

высшего‌ ‌образования‌

**«Пермский национальный исследовательский  
политехнический университет»**

Кафедра «Информационные технологии и автоматизированные системы»

**О Т Ч Ё Т**

**по лабораторной работе №18.13**

Дисциплина: «Основы теории алгоритмов и структуры данных»  
Тема: Стандартные обобщенные алгоритмы библиотеки STL

Вариант 12

Выполнил:

студент группы РИС-20-2б

Морозова Е.М.

Проверила:

доцент кафедры ИТАС

Полякова Ольга Андреевна

Пермь, 2021

**Цель работы**

1. Создание консольного приложения, состоящего из нескольких файлов.

2.Использование стандартных обобщенных алгоритмов из библиотеки STL в ОО программе.

**Постановка задачи**

1. Создать последовательный контейнер.
2. Заполнить его элементами пользовательского типа. Для пользовательского типа перегрузить необходимые операции.
3. Заменить элементы в соответствии с заданием.
4. Удалить элементы в соответствии с заданием.
5. Отсортировать контейнер по убыванию и по возрастанию ключевого поля.
6. Найти в контейнере заданный элемент.
7. Выполнить задание варианта для полученного контейнера.
8. Для выполнения всех заданий использовать стандартные алгоритмы STL.

**Задание варианта:**

Задача 1: контейнер список, тип элементов - Pair;   
Задача 2: Адаптер контейнера-очередь с приоритетами;  
Задача 3: Ассоциативный контейнер словарь с дубликатами;   
Задание 3:Найти среднее арифметическое и добавить его на заданную позицию контейнера .  
Задание 4: Найти элементы,с помощью ключей из заданного диапазона и удалить их из контейнера.  
Задание 5: Из каждый элемента вычесть среднее арифметическое контейнера.

**Анализ задачи**

1. Описание классов:

class Pair

{

public:

int first;

double second;

public:

Pair() { first = 0; second = 0.0; }

Pair(int F, double S) { first = F; second = S; }

Pair(const Pair& p) { first = p.first; second = p.second; }

~Pair() {};

int get\_first() { return first; }

double get\_second() { return second; }

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

void set\_second(double S) { second = S; }

Pair operator+ (const Pair&);

Pair operator/ (const int&);

Pair operator- (const Pair&);

Pair& operator= (const Pair&);

friend bool operator< (const Pair&, const Pair&);

friend bool operator> (const Pair&, const Pair&);

friend bool operator== (const Pair&, const Pair&);

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

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

};

class Pair

{

public:

int first;

double second;

public:

Pair() { first = 0; second = 0.0; }

Pair(int F, double S) { first = F; second = S; }

Pair(const Pair& p) { first = p.first; second = p.second; }

~Pair() {};

int get\_first() { return first; }

double get\_second() { return second; }

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

void set\_second(double S) { second = S; }

Pair operator+ (const Pair&);

Pair operator/ (const int&);

Pair operator- (const Pair&);

Pair& operator= (const Pair&);

friend bool operator< (const Pair&, const Pair&);

friend bool operator> (const Pair&, const Pair&);

friend bool operator== (const Pair&, const Pair&);

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

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

};

class Pair

{

int first;

double second;

public:

Pair() { first = 0; second = 0.0; }

Pair(int F, double S) { first = F; second = S; }

Pair(const Pair& p) { first = p.first; second = p.second; }

~Pair() {};

int get\_first() { return first; }

double get\_second() { return second; }

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

void set\_second(double S) { second = S; }

Pair operator+ (const Pair&);

Pair operator/ (const int&);

Pair operator- (const Pair&);

Pair& operator= (const Pair&);

friend bool operator< (const Pair&, const Pair&);

friend bool operator> (const Pair&, const Pair&);

friend bool operator== (const Pair&, const Pair&);

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

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

};

2. Решение первой задачи:  
mylist make(int n)

{

mylist l;

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

{

cout << " ch1: ";

int a; cin >> a;

cout << " ch2: ";

double b; cin >> b;

Pair c(a, b);

l.push\_back(c);

}

return l;

}

void print(mylist l)

{

mylist::iterator it = l.begin();

for (; it != l.end(); it++)

cout << \*it << endl;

cout << endl;

}

Pair srednee(mylist l)

{

int size = l.size();

int a = 0; double b = 0;

list<Pair>::iterator i;

for (i = l.begin(); i != l.end(); i++)

{

a = a + (\*i).first;

b = b + (\*i).second;

}

a = a / size;

b = b / size;

Pair sr(a, b);

return sr;

}

struct Greater\_s

{

bool operator()(Pair p)

{

if (p > s) return true; else return false;

}

};

struct Comp\_less

{

public:

bool operator()(Pair p1, Pair p2)

{

if (p1 > p2) return true;

else return false;

}

};

struct Equal\_s

{

bool operator()(const Pair&p)

{

return p == s;

}

};

void delsrednee(Pair& k)

{

k = k - s;

}

int main()

{

setlocale(0, "");

mylist l;

int size;

cout << "size:";

cin >> size;

l = make(size);

print(l);

mylist::iterator i=l.begin();

Pair k = srednee(l);

cout << "srednee=" << k << endl;

cout << "Position=";

int pos;

cin >> pos;

advance(i, pos-1);

l.insert(i, k);

print(l);

cout << "Сортировка по убыванию:" << endl;

l.sort(Comp\_less());

print(l);

cout << "Сортировка по возрастанию:" << endl;

l.sort();

print(l);

cout << "Какой элемент вы хотите удалить?" << endl;

cin >> s;

l.remove\_if(Equal\_s());

print(l);

cout << "Удаление среднего" << endl;

s = srednee(l);

for\_each(l.begin(), l.end(), delsrednee);

print(l);

}

3. Решение второй задачи:  
typedef queue<Pair> mylist;

typedef list<Pair> List;

Pair s;

mylist make(int n)

{

mylist l;

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

{

cout << " ch1: ";

int a; cin >> a;

cout << " ch2: ";

double b; cin >> b;

Pair c(a, b);

l.push(c);

}

return l;

}

void print(mylist l)

{

while (!l.empty())

{

cout << l.front() << ' ';

l.pop();

}

cout << '\n';

}

List copy\_queue\_to\_list(mylist&s)

{

List a;

while (!s.empty())//пока очередь не пустая

{

a.push\_back(s.front());//добавление элемента из стека в список

s.pop();//убираем элемент с вершины стека

}

return a;

}

mylist copy\_list\_to\_queue(List&a)

{

mylist s;

for (List::iterator iter = a.begin(); iter != a.end(); iter++)

{

s.push((\*iter));

}

return s;

}

Pair srednee(mylist l)

{

List s = copy\_queue\_to\_list(l);

int a = 0; double b = 0;

list<Pair>::iterator i;

for (i = s.begin(); i != s.end(); i++)

{

a = a + (\*i).first;

b = b + (\*i).second;

}

int size = s.size();

a = a / size;

b = b / size;

Pair p(a, b);

return p;

}

void add(mylist&l, Pair&k)

{

l.push(k);

}

struct Greater\_s

{

bool operator()(Pair p)

{

if (p > s) return true; else return false;

}

};

struct Comp\_less

{

public:

bool operator()(Pair p1, Pair p2)

{

if (p1 > p2) return true;

else return false;

}

};

struct Equal\_s

{

bool operator()(const Pair& p)

{

return p == s;

}

};

void delsrednee(Pair& k)

{

k = k - s;

}

void sort\_high(mylist&l)

{

List lis = copy\_queue\_to\_list(l);

List::iterator i = lis.begin();

lis.sort();

l = copy\_list\_to\_queue(lis);

}

void sort\_done(mylist&l)

{

List lis = copy\_queue\_to\_list(l);

List::iterator i = lis.begin();

lis.sort(Comp\_less());

l = copy\_list\_to\_queue(lis);

}

void del(mylist&l, Pair&s)

{

List lis = copy\_queue\_to\_list(l);

lis.remove\_if(Equal\_s());

l = copy\_list\_to\_queue(lis);

}

void delete\_srednee(mylist&l, Pair&sr)

{

List lis = copy\_queue\_to\_list(l);

for\_each(lis.begin(), lis.end(), delsrednee);

l = copy\_list\_to\_queue(lis);

}

int main()

{

setlocale(0, "");

mylist l;

int size;

cout << "size:";

cin >> size;

l = make(size);

print(l);

Pair k = srednee(l);

cout << "srednee=" << k << endl;

add(l, k);

print(l);

cout << "Сортировка по убыванию:" << endl;

sort\_done(l);

print(l);

cout << "Сортировка по возрастанию:" << endl;

sort\_high(l);

print(l);

cout << "Какой элемент вы хотите удалить?" << endl;

cin >> s;

del(l, s);

print(l);

cout << "Удаление среднего" << endl;

s = srednee(l);

delete\_srednee(l, s);

print(l);

}

4. Решение третей задачи:

typedef map<int, Pair> TMap;

typedef list<Pair> List;

typedef TMap::iterator it;

Pair s;

TMap make\_map(int n)

{

TMap m;

Pair a;

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

{

cin >> a;

m.insert(make\_pair(i, a));

}

return m;

}

void print\_map(TMap m)

{

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

cout << i << "=" << m[i] << " " << endl;

}

Pair srednee(TMap m)

{

Pair s = m[0];

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

s = s + m[i];

int n = m.size();

return s / n;

}

void delete\_srednee(TMap& m,Pair&sr)

{

int n = m.size();

Pair el = sr;

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

{

m[i] = m[i] - el;

}

}

struct Greater\_s

{

bool operator()(Pair p)

{

if (p > s) return true; else return false;

}

};

struct Comp\_less

{

public:

bool operator()(Pair p1, Pair p2)

{

if (p1 > p2) return true;

else return false;

}

};

struct Equal\_s

{

bool operator()(const Pair& p)

{

return p == s;

}

};

TMap sort\_done(TMap m)

{

TMap p;

cout << "сортировка по убыванию:" << endl;

vector<Pair> vec2;

for (int a = 0; a < m.size(); a++)

{

Pair b = m[a];

vec2.push\_back(b);

}

sort(vec2.begin(), vec2.end(), Comp\_less());

for (int o = 0; o < vec2.size(); o++)

{

Pair z = vec2[o];

p.insert(make\_pair(o, z));

}

return p;

}

TMap sort\_high(TMap p)

{

cout << "сортировка по возрастанию" << endl;

vector<Pair> vec;

for (int a = 0; a < p.size(); a++)

{

Pair b = p[a];

vec.push\_back(b);

}

sort(vec.begin(), vec.end());

TMap d;

for (int o = 0; o < vec.size(); o++)

{

Pair z = vec[o];

d.insert(make\_pair(o, z));

}

return d;

}

void delsrednee(Pair& k)

{

k = k - s;

}

int main()

{

setlocale(0, "");

int n;

cout << "N:"; cin >> n;

map<int, Pair> m = make\_map(n);

print\_map(m);

cout << endl;

Pair sr = srednee(m);

cout << "среднее=" << sr << endl;

m.insert(make\_pair(n, sr));

print\_map(m);

TMap p=sort\_done(m);

print\_map(p);

TMap d = sort\_high(m);

print\_map(d);

cout << "какой элемент вы хотите удалить?" << endl;

cin >> s;

TMap ma;

list <Pair> vec;

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

{

Pair b = d[a];

vec.push\_back(b);

}

vec.remove\_if(Equal\_s());

int ir = 0;

for (List::iterator iter = vec.begin(); iter != vec.end(); iter++)

{

ma.insert(make\_pair(ir, (\*iter)));

ir++;

}

print\_map(ma);

s = sr;

TMap mapp;

list <Pair> vec2;

for (int a = 0; a < ma.size(); a++)

{

Pair b = d[a];

vec2.push\_back(b);

}

for\_each(vec2.begin(), vec2.end(), delsrednee);

int ir2 = 0;

for (List::iterator iter = vec2.begin(); iter != vec2.end(); iter++)

{

mapp.insert(make\_pair(ir2, (\*iter)));

ir2++;

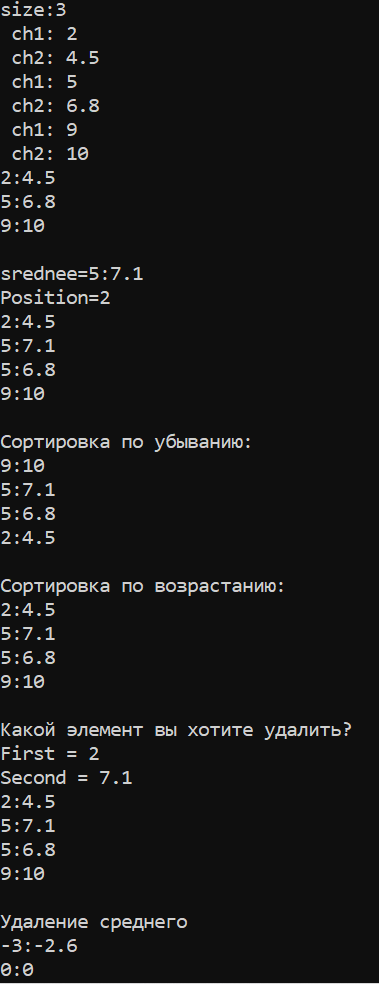
}

print\_map(mapp);

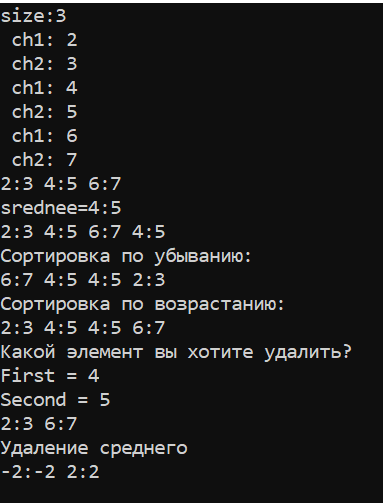
return 0;

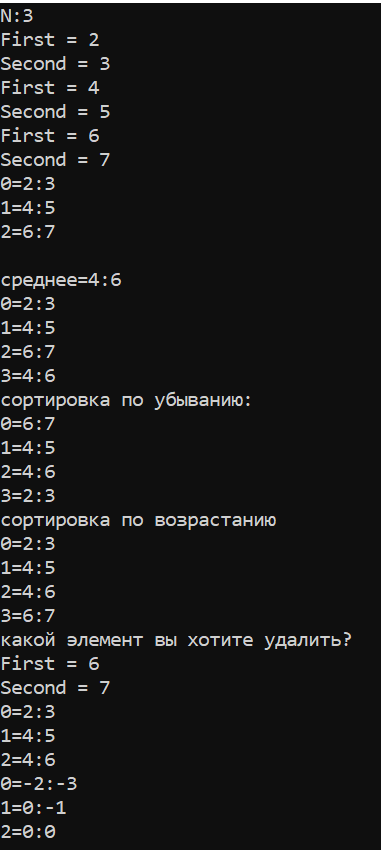
}

**Результаты работы программы**

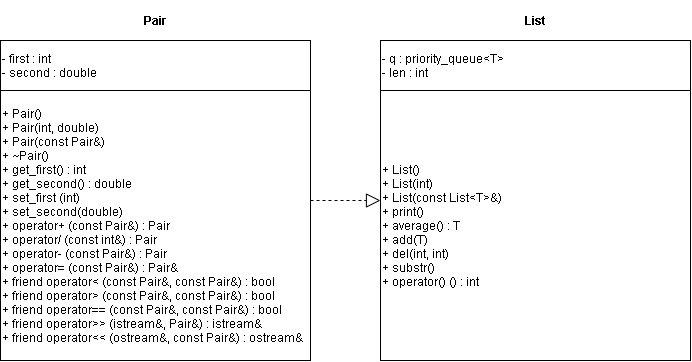
****

****

****

****

**Диаграмма классов**

****