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 №18.12**

Дисциплина: «Основы теории алгоритмов и структуры данных»  
Тема: Ассоциативные контейнеры библиотеки STL

Вариант 14

Выполнил:

студент группы РИС-20-2б

Вичугов Алексей Дмитриевич

Проверила:

доцент кафедры ИТАС

Полякова Ольга Андреевна

Пермь, 2021

**Цель работы**

1. Создание консольного приложения, состоящего из нескольких файлов.

2. Использование ассоциативных контейнеров библиотеки STL в ОО программе.

**Постановка задачи**

1. Создать контейнеры типа, указанного в варианте.
2. Заполнить контейнеры элементами.
3. Добавить элементы в контейнеры.
4. Удалить элементы из контейнеров.
5. Выполнить задание варианта для каждого контейнера.
6. Задание варианта: Задача 1: контейнер multimap, тип элементов - float;   
   Задача 2: тип элементов Pair;   
   Задача 3: параметризованный класс список;   
   Задание 3: найти минимальный элемент и добавить его на заданную позицию контейнера.  
   Задание 4: Найти элементы, значение которых меньше среднего арифметического и удалить их из контейнера.  
   Задание 5: Каждый элемент разделить на максимальный элемент контейнера.

**Анализ задачи**

1. Описание классов:

class Para{

public:

int a;

double b;

void Init(int i, double d) {a=i;b=d;}

void Show(){cout << a << ':' << b << endl;}

bool operator <(Para &p){if (a<p.a) return 1; else return 0;}

Para& operator /=(Para &p){a/=p.a;b/=p.b;}

Para& operator =(Para &p){a=p.a;b=p.b;}

};

template <class T> class Lister{

public:

T min,max,sred;

typedef multiset <T> setter;

setter s;

typename setter::iterator i;

Lister(){for (int e=0;e<10;e++) {s.insert(rand()\*0.111);}}

void Show(){cout << "Set: " << endl; for (i=s.begin();i!=s.end();i++) cout << \*i << endl;}

void Search(){

i=s.begin();

min=\*i;

max=\*i;

sred=0;

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

if (\*i<min) min=\*i;

if (\*i>max) max=\*i;

sred+=\*i;

}

cout << "Minimum: " << min << endl;

cout << "Maximum: " << max << endl;

}

void Pushmin(){s.insert(min);}

void Lowsred(){

sred/=s.size();

cout << "Middle: " << sred << endl;

i=s.begin();

while(\*i<sred){

s.erase(i);

i=s.begin();

}

}

void Divide(){

T temp;

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

temp=\*i/max;

s.erase(i);

s.insert(temp);

i=s.begin();

i=s.find(temp);

}

}

};

2. Решение первой задачи:  
void One(){

typedef multimap<int,float> maper;

maper m;

maper::iterator i;

float min,max,sred;

int e=1;

for (int z=1;z<11;z++) m.insert(make\_pair(z,rand()\*0.36F));

i=m.begin();

min=i->second;

max=i->second;

for (i=m.begin();i!=m.end();i++){

if(i->second<min){

min=i->second;

e=i->first;

}

if(i->second>max) max=i->second;

sred+=i->second;

cout << i->second << endl;

}

cout << "Minimum: " << min << endl << "Maximum: " << max << endl;

m.insert(make\_pair(e,min));

cout << "Multimap: " << endl;

for (i=m.begin();i!=m.end();i++) cout << i->first << ' ' << i->second << endl;

sred/=m.size();

cout << "Middle: " << sred << endl;

for (i=m.begin();i!=m.end();i++){

if ((i->second)<sred){

m.erase(i->first);

i=m.begin();

}

}

cout << "Multimap: " << endl;

for (i=m.begin();i!=m.end();i++) cout << i->first << ' ' << i->second << endl;

for (i=m.begin();i!=m.end();i++){i->second/=max;}

cout << "Multimap: " << endl;

for (i=m.begin();i!=m.end();i++) cout << i->first << ' ' << i->second << endl;

}

3. Решение второй задачи:  
void Two(){

typedef map<int,Para> maper;

maper m;

maper::iterator i;

Para min,max,p;

float sred;

int position;

for (int z=0;z<10;z++){

p.Init(rand(),rand()\*0.121);

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

m[z].Show();

}

i=m.begin();

min=i->second;

max=i->second;

for (i=m.begin();i!=m.end();i++){

if(i->second<min){

min=i->second;

}

if(max<i->second) max=i->second;

sred+=i->second.a;

}

cout << "Minimum: "; min.Show(); cout << "Maximum: "; max.Show();

cout << "Position (>size): "; cin >> position;

while(cin.fail()||position<m.size()){

cin.clear();

cin.ignore(5,'\n');

cout << "Position must be more than size of map. Repeat: "; cin >> position;

}

m.insert(make\_pair(position,min));

cout << "Map: " << endl;

for (i=m.begin();i!=m.end();i++){

cout << i->first << ' ';

i->second.Show();

}

sred/=m.size();

cout << "Middle: " << sred << endl;

for (i=m.begin();i!=m.end();i++){

if ((i->second.a)<sred){

m.erase(i->first);

i=m.begin();

}

}

cout << "Map: " << endl;

for (i=m.begin();i!=m.end();i++){

cout << i->first << ' ';

i->second.Show();

}

for (i=m.begin();i!=m.end();i++){i->second/=max;}

cout << "Map: " << endl;

for (i=m.begin();i!=m.end();i++){

cout << i->first << ' ';

i->second.Show();

}

}

4. Решение третей задачи:

void Three(){

Lister <double> l;

l.Show();

l.Search();

l.Pushmin();

l.Show();

l.Lowsred();

l.Show();

l.Divide();

l.Show();

}

5. Функия main():  
main(){

int s=1;

while(s){

cout << "1. Multimap; \n2. Map; \n3. Multiset with parameterized list. " << endl << "S: "; cin >> s;

switch(s){

case 1: One(); break;

case 2: Two(); break;

case 3: Three(); break;

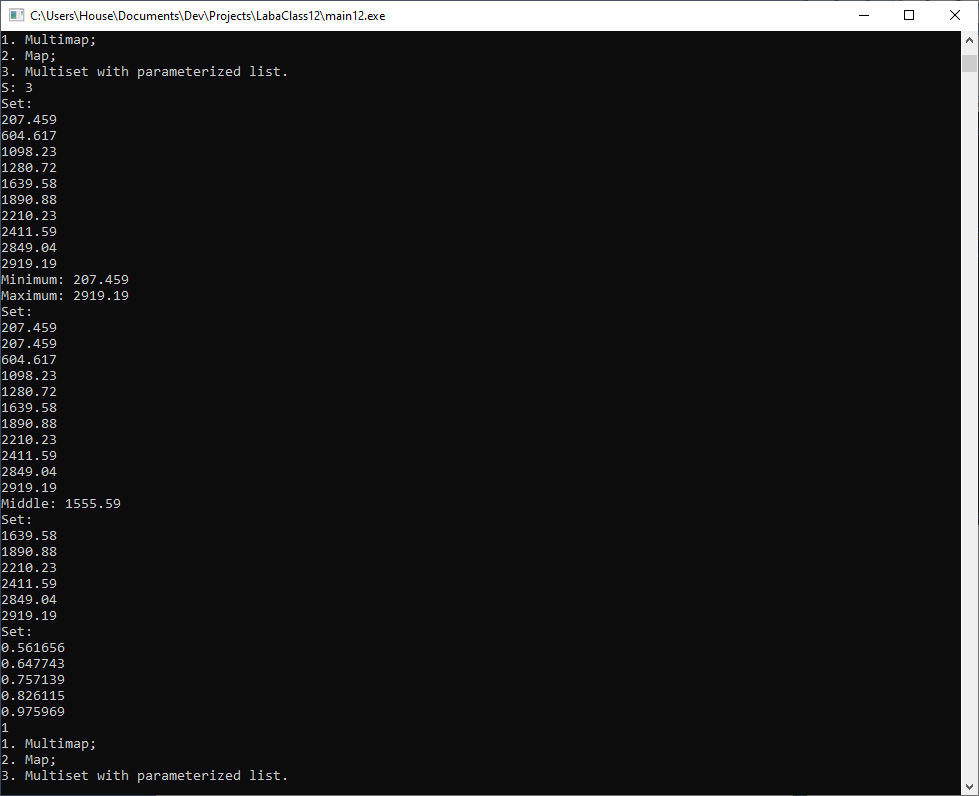
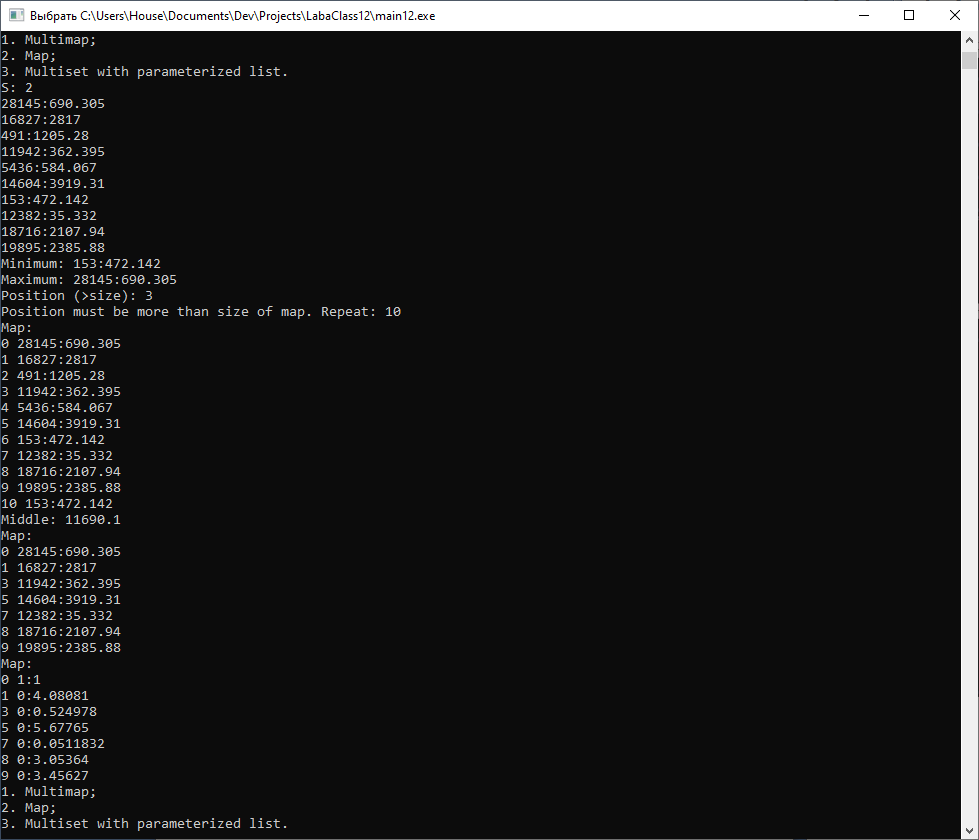
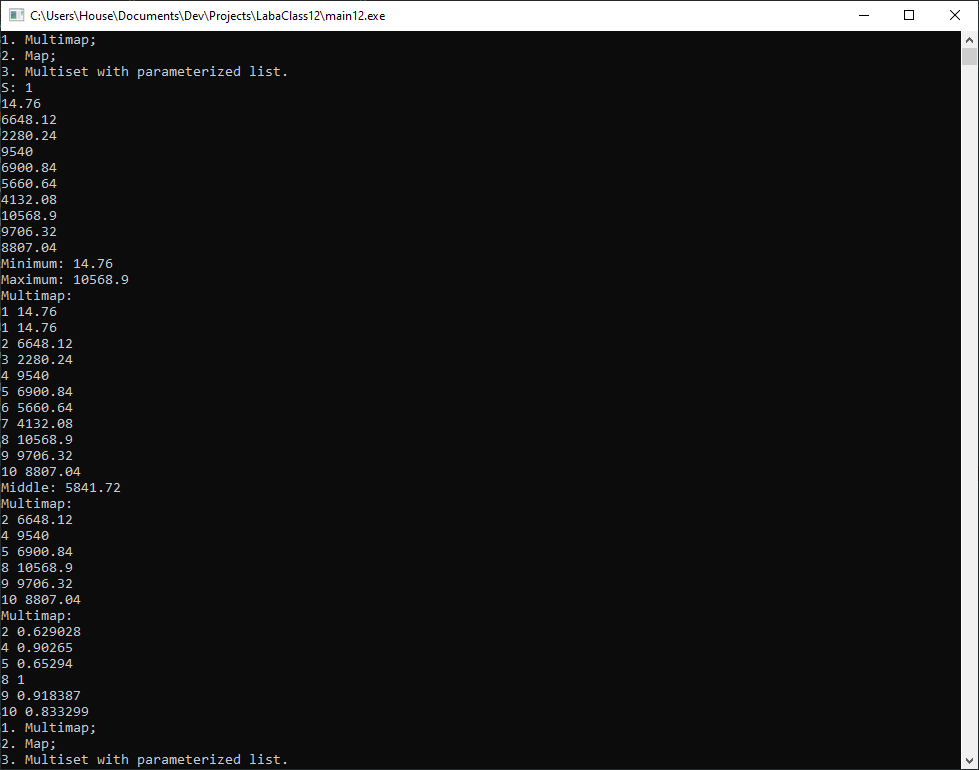
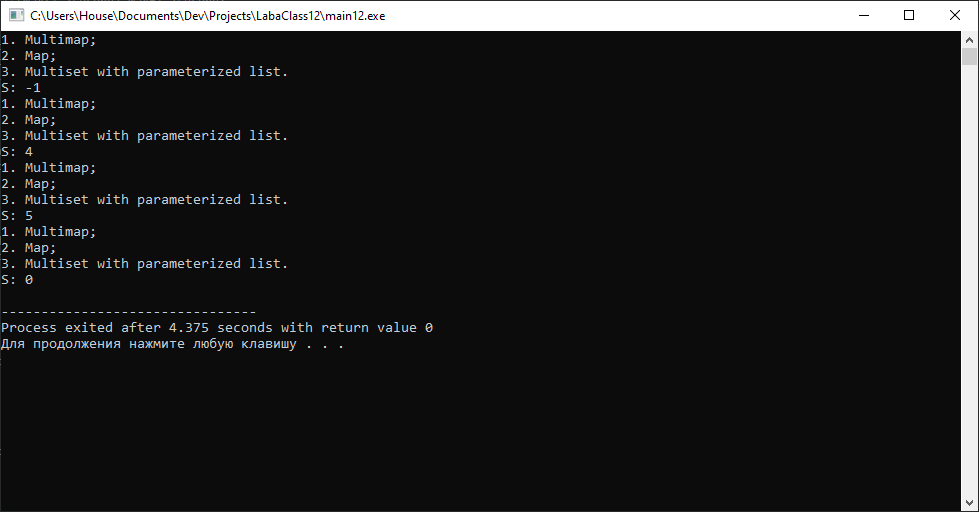
}

}

return 0;

}

**Результаты работы программы**

****

**Ответы на вопросы**

1. Массив, содержащий пару значений – ключ и отображаемое значение.
2. Map, multimap, set, multiset.
3. По ключу.
4. Insert – вставить элемент, erase – удалить элемент, clear – удалить все элементы.
5. Typedef map<тип\_данных\_ключа, тип\_данных\_значения> название\_контейнера. Например: typedef map<int, double> m.
6. В порядке добавления элементов. Очистить контейнер и добавить элементы в обратном порядке.
7. [] – доступ по индексу.
8. Void Add(){m.make\_pair(1,0.5);}
9. Void Add(int I, double k){m[i]=k;}
10. Void Show() {for (it=m.begin();it!=m.end();it++) cout << \*(it->second);}
11. Void Show() {for (int i=0;i<m.size();i++) cout << m[i];}
12. Контейнер multimap допускает хранение элементов с одинаковыми ключами (эти элементы хранятся в контейнере в порядке занесения).
13. Множество. Ассоциативный массив, в котором отслеживаются ключи, а не значения.
14. Элементы множества (set) отсортированы, в отличие от элементов словаря (map).
15. Typedef set<тип\_данных\_ключа> имя\_контейнера. Например: typedef set<int> s.
16. По возрастанию значений ключа. При объявлении указать порядок убывания (set<int, less<int>> s).
17. Функции insert, erase и count, а также функции, общие для всех контейнеров.
18. Void Add(int k){s.insert(k);};
19. Void Show(){for (it=s.begint;it!=s.end();it++) cout << \*it;}
20. В контейнере multiset ключи могут повторяться (они хранятся в порядке занесения).