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 №18.11**

Дисциплина: «Основы теории алгоритмов и структуры данных»  
Тема: Последовательные 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 последовательных 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 вектор, тип элементов float;   
   Задача 2: тип элементов Pair;   
   Задача 3: параметризованный класс список;   
   Задача 4: Адаптер контейнера очередь.   
   Задача 5: параметризованный класс список, адаптер контейнера – очередь.   
   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;

Para(){a=0;b=0;}

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

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

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

Para& operator /(int i){a/=i;b/=i;}

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

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

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

bool operator <(int i) {if (a<i) return 1; else return 0;}

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

};

template <class T> class Pole{

public:

T val;

};

2. Решение первой задачи:  
void One(){

vector <float> v (10);

vector <float>::iterator i;

for(i=v.begin();i!=v.end();i++){\*i=rand()\*0.385;cout << \*i << endl;}

float min, max,srednee=0;

i=v.begin();

min=\*i;

max=\*i;

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

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

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

srednee+=\*i;

}

cout << "The bigest element: " << max << endl;

cout << "The smallest element: " << min << endl;

int position;

cout << "Enter the position: "; cin >> position;

while (cin.fail()||position<1||position>v.size()+1){

cin.clear();

cin.ignore(10,'\n');

cout << "Position must have a value (0<p<11)! Repeat: ";

cin >> position;

}

i=v.begin();

for (int e=1;e<position;e++) i++;

v.insert(i, min);

cout << "Vector: " << endl;

for (i=v.begin();i!=v.end();i++) cout << \*i << endl;

srednee/=v.size();

cout << "Srednee: " << srednee << endl;

for (i=v.begin();i!=v.end();i++) {if(\*i<srednee) v.erase(i--);}

cout << "Vector: " << endl;

for (i=v.begin();i!=v.end();i++) cout << \*i << endl;

for (i=v.begin();i!=v.end();i++) \*i/=max;

cout << "Vector: " << endl;

for (i=v.begin();i!=v.end();i++) cout << \*i << endl;

}

3. Решение второй задачи:  
void Two(){

vector<Para> v(10);

vector<Para>::iterator i;

int r,min,max,srednee,position;

double rd;

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

r=rand();

rd=r\*0.837;

i->Init(r, rd);

}

cout << "Vector: " << endl;

for (i=v.begin();i!=v.end();i++) i->Show();

i=v.begin();

min=i->a;

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

if (i->a<min) min=i->a;

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

srednee+=i->a;

}

cout << "The bigest element: " << max << ':' << max\*0.837 << endl;

cout << "The smallest element: " << min << ':' << min\*0.837<< endl;

cout << "Enter the position: "; cin >> position;

while (cin.fail()||position<1||position>v.size()+1){

cin.clear();

cin.ignore(10,'\n');

cout << "Position must have a value (0<p<11)! Repeat: ";

cin >> position;

}

i=v.begin();

Para u(min,min\*0.837);

for (int e=1;e<position;e++) i++;

v.insert(i, u);

cout << "Vector: " << endl;

for (i=v.begin();i!=v.end();i++) i->Show();

srednee/=v.size();

cout << "Srednee: " << srednee << endl;

for (i=v.begin();i!=v.end();i++) if(i->a<srednee) v.erase(i--);

cout << "Vector: " << endl;

for (i=v.begin();i!=v.end();i++) i->Show();

for (i=v.begin();i!=v.end();i++) {i->a/=max;i->b/=max;}

cout << "Vector: " << endl;

for (i=v.begin();i!=v.end();i++) i->Show();

cout << endl;

}

4. Решение третей задачи:

void Three(){

list<Pole<double> > v(10);

list<Pole<double> >::iterator i;

double min,max,srednee;

int position;

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

i->val=rand();

}

cout << "List: " << endl;

for (i=v.begin();i!=v.end();i++) cout << i->val << endl;

i=v.begin();

min=i->val;

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

if (i->val<min) min=i->val;

}

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

if (i->val<min) min=i->val;

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

srednee+=i->val;

}

cout << "The bigest element: " << max << endl;

cout << "The smallest element: " << min << endl;

cout << "Enter the position: "; cin >> position;

while (cin.fail()||position<1||position>v.size()+1){

cin.clear();

cin.ignore(10,'\n');

cout << "Position must have a value (0<p<11)! Repeat: ";

cin >> position;

}

i=v.begin();

for (int e=1;e<position;e++) i++;

Pole <double> p;

p.val=min;

v.insert(i, p);

cout << "List: " << endl;

for (i=v.begin();i!=v.end();i++) cout << i->val << endl;

srednee/=v.size();

cout << "Srednee: " << srednee << endl;

for (i=v.begin();i!=v.end();i++) if(i->val<srednee) v.erase(i--);

cout << "List: " << endl;

for (i=v.begin();i!=v.end();i++) cout << i->val << endl;

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

cout << "List: " << endl;

for (i=v.begin();i!=v.end();i++) cout << i->val << endl;

}

5. Решение четвёртой задачи:  
void Four(){

queue<Para> q;

Para p,min,max,temp;

int sred=0,position,r;

double rd;

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

r=rand();

rd=r\*0.837;

p.Init(r,rd);

q.push(p);

}

cout << "Queue: " << endl;

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

q.front().Show();

temp=q.front();

q.pop();

q.push(temp);

}

min=q.front();

max=min;

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

if (q.front()>max) max=q.front();

if (min>q.front()) min=q.front();

sred+=q.front().a;

temp=q.front();

q.pop();

q.push(temp);

}

cout << "The bigest element: "; max.Show();

cout << "The smallest element: "; min.Show();

cout << "Enter the position: "; cin >> position;

while (cin.fail()||position<1||position>q.size()+1){

cin.clear();

cin.ignore(10,'\n');

cout << "Position must have a value (0<p<11)! Repeat: ";

cin >> position;

}

for (int i=1;i<position;i++){

temp=q.front();

q.pop();

q.push(temp);

}

q.push(min);

for (int i=0;i<(q.size()-position);i++){

temp=q.front();

q.pop();

q.push(temp);

}

cout << "Queue: " << endl;

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

q.front().Show();

temp=q.front();

q.pop();

q.push(temp);

}

sred/=10;

cout << "Srednee: " << sred << endl;

for (int i=0;i<11;i++){

if (q.front().a<sred) q.pop();

else{

temp=q.front();

q.pop();

q.push(temp);

}

}

cout << "Queue: " << endl;

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

q.front().Show();

temp=q.front();

q.pop();

q.push(temp);

}

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

temp=q.front();

temp/=max;

q.pop();

q.push(temp);

}

cout << "Queue: " << endl;

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

q.front().Show();

temp=q.front();

q.pop();

q.push(temp);

}

}

6. Решение пятой задачи:  
void Five(){

queue<Pole<double> > q;

Pole <double> p;

Pole <double> min, max, temp;

int sred=0,position,z=1;

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

p.val=rand()\*0.396;

cout << p.val << endl;

q.push(p);

}

min=q.front();

max=min;

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

if (q.front().val>max.val) max=q.front();

if (min.val>q.front().val) min=q.front();

sred+=q.front().val;

temp=q.front();

q.pop();

q.push(temp);

}

cout << "The bigest element: "; cout << max.val << endl;

cout << "The smallest element: " << min.val << endl;

cout << "Enter the position: "; cin >> position;

while (cin.fail()||position<1||position>q.size()+1){

cin.clear();

cin.ignore(10,'\n');

cout << "Position must have a value (0<p<11)! Repeat: ";

cin >> position;

}

for (int i=1;i<position;i++){

temp=q.front();

q.pop();

q.push(temp);

}

q.push(min);

for (int i=0;i<(q.size()-position);i++){

temp=q.front();

q.pop();

q.push(temp);

}

cout << "Queue: " << endl;

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

cout << q.front().val << endl;

temp=q.front();

q.pop();

q.push(temp);

}

sred/=10;

cout << "Srednee: " << sred << endl;

for (int i=0;i<11;i++){

if (q.front().val<sred) q.pop();

else{

temp=q.front();

q.pop();

q.push(temp);

}

}

cout << "Queue: " << endl;

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

cout << q.front().val << endl;

temp=q.front();

q.pop();

q.push(temp);

}

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

temp=q.front();

temp.val=temp.val/max.val;

q.pop();

q.push(temp);

}

cout << "Queue: " << endl;

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

cout << q.front().val << endl;

temp=q.front();

q.pop();

q.push(temp);

}

}

7. Функия main():  
main(){

int s=1;

while (s){

cout << "1. Vector; \n2. Vector with user`s class; \n3. List with parametrized user`s class; \n4. Queue with user`s class; \n5. Queue with parametrized user`s class. " << endl << "S: "; cin >> s;

switch(s){

case 1: One(); break;

case 2: Two(); break;

case 3: Three(); break;

case 4: Four(); break;

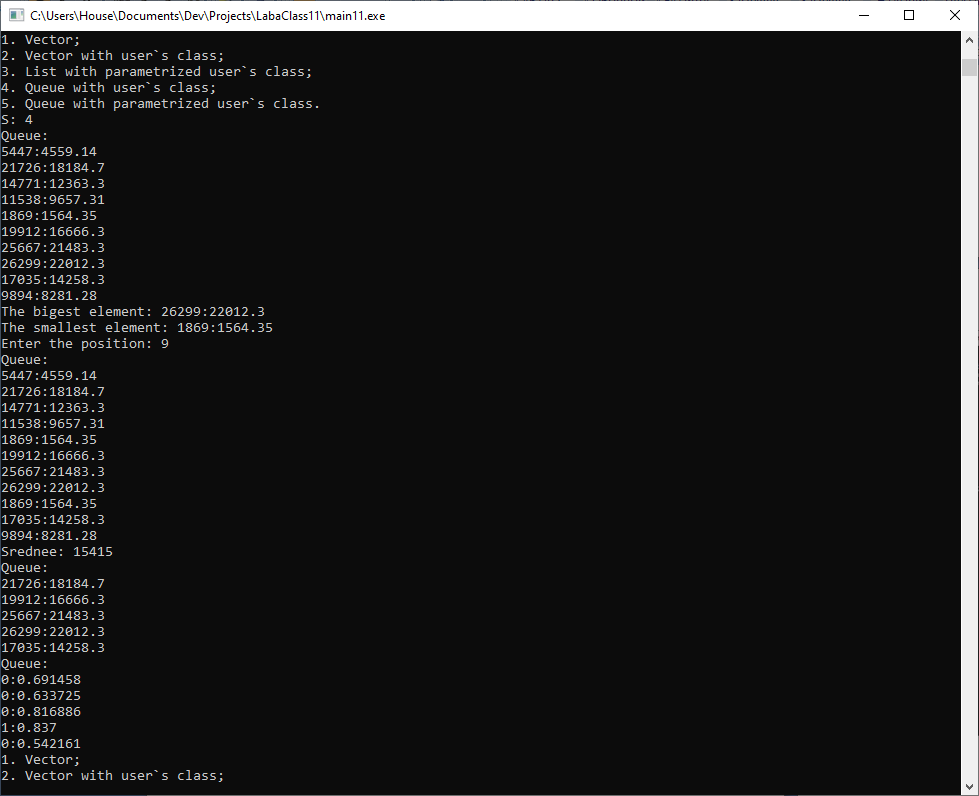
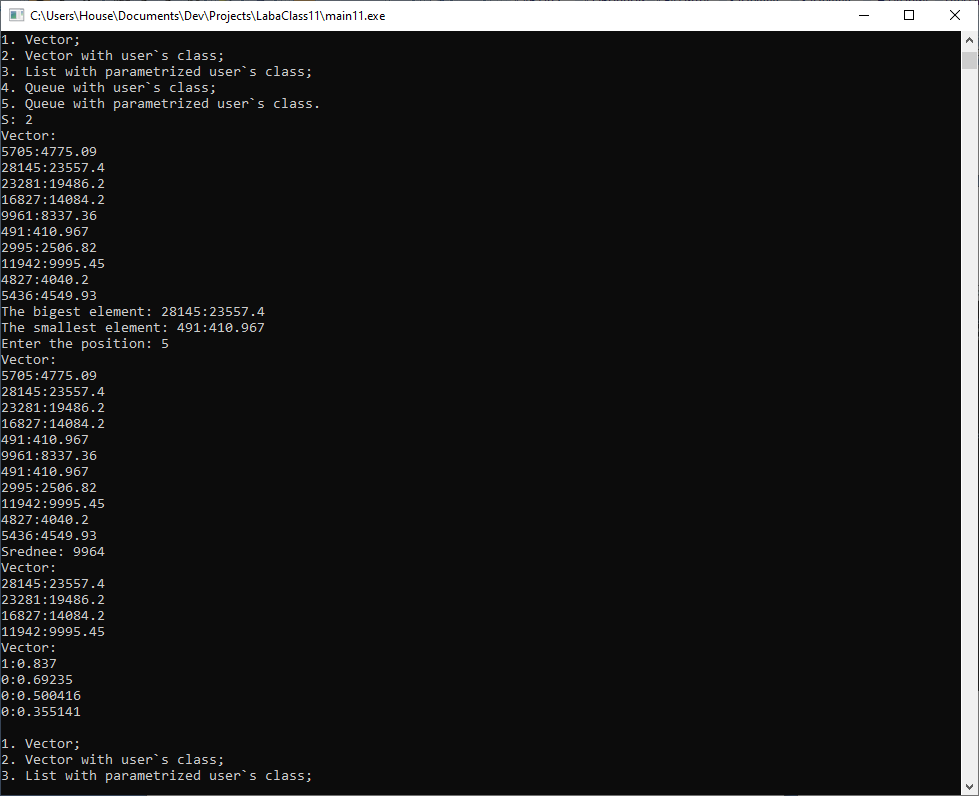
case 5: Five(); break;

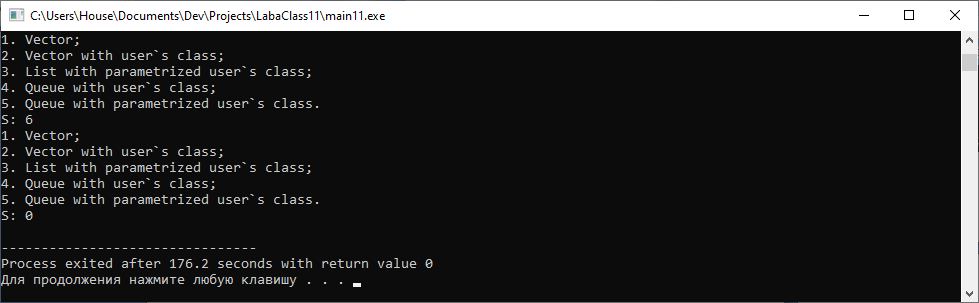
}

}

}

**Результаты работы программы**

****

****

**Ответы на вопросы**

1. Набор контейнерных классов и набор обобщённых алгоритмов.
2. Последовательные и ассоциативные.
3. Подключить библиотеку соответствующего контейнера.
4. Указатель на элемент контейнера.
5. Разыменование, присваивание, сравнение итераторов, перемещение его по всем элементам контейнера.
6. Изначально итератор указывает на первый элемент контейнера. По ходу цикла он инкрементируется, пока не станет указывать не элемент, стоящий за последним элементом контейнера.
7. Входные, выходные, прямые, двунаправленные, итераторы произвольного доступа.
8. ==, !=, clear, insert, erase, size, empty, begin, end .
9. Доступ по индексу [], потому что элемент с необходимым индексом можно быстро найти.
10. Вставка и удаление, так как при них не приходится сдвигать элементы.
11. Вставка и удаление первого и последнего элементов, так как к ним можно быстро получить доступ без значительных затрат ресурсов на сдвиг остальных элементов.
12. Push\_back, pop\_back, insert, erase, [], at, swap, clear.
13. Push\_back, pop\_back, push\_front, pop\_front, insert, erace, swap, clear, splice.
14. Push\_back, pop\_back, push\_front, pop\_front, insert, erase, [], at.
15. V.erace(it2, it5); it2 и it5 – итераторы, указывающие на элементы 2 и 5 соответственно.
16. it=v.end();it--; v.erace(it);
17. l.erace(it2, it5); it2 и it5 – итераторы, указывающие на элементы 2 и 5 соответственно.
18. it=l.end();it--; l.erace(it);
19. dq.erace(it2, it5); it2 и it5 – итераторы, указывающие на элементы 2 и 5 соответственно.
20. it=dq.end();it--; dq.erace(it);
21. void Show(){for (it=cont.begin();it!=cont.end();it++) cout << \*it;}
22. Специализированные последовательные контейнеры – стек, очередь и очередь с приоритетами. Они реализованы на обычных контейнерах.
23. В первом случае стек создаётся на базе двусторонней очереди, а во втором – на базе списка.
24. Push, pop, top, empty, size.
25. Push, pop, front, back, empty, size.
26. В очереди с приоритетами максимальный элемент всегда сдвигается в начало контейнера.
27. Скопировать (последовательно передавая во второй стек и удаляя из первого) все элементы, кроме удаляемого во второй стек и заменить содержимое первого стека содержимым второго.   
    Или скопировать (последовательно передавая во второй стек и удаляя из первого) все элементы, стоящие до удаляемого во второй стек, удалить нужный элемент и вернуть все элементы из второго стека в первый.
28. Скопировать (последовательно передавая во вторую очередь и удаляя из первой) все элементы, кроме удаляемого во вторую очередь и заменить содержимое первой очереди содержимым второй.  
    Или скопировать (последовательно передавая во вторую очередь и удаляя из первой) все элементы, стоящие до удаляемого во вторую очередь, удалить нужный элемент и вернуть все элементы из второй очереди в первую.
29. Void Show() {it=s.begin(); for (;it!=s.end();it++) cout << \*it;}
30. Void Show() {it=q.begin(); for (;it!=q.end();it++) cout << \*it;}