Лабораторная работа №12

Ассоциативные контейнеры библиотеки STL.

1. Цель задания:

1) Создание консольного приложения, состоящего из нескольких файлов в системе программирования Visual Studio.
2) Использование ассоциативных контейнеров библиотеки STL в ОО программе.

2. Теоретические сведения

2.1. Ассоциативные контейнеры (массивы)

Ассоциативный массив содержит пары значений. Зная одно значение, называемое **ключом** (key), мы можем получить доступ к другому, называемому **отображенным значением** (mapped value).

Ассоциативный массив можно представить как массив, для которого индекс не обязательно должен иметь целочисленный тип:
V& operator[](const K&) возвращает ссылку на элемент V, соответствующий значению K.

Ассоциативные контейнеры – это обобщение понятия ассоциативного массива.

В библиотеке STL реализован шаблон `pair`, для представления пары «ключ – значение». Атрибутами пары являются элементы `first` и `second`. Для пар определен полный набор попарных сравнений: ==, !=, <, >, <=,>= . Для конструирования пары определен шаблон функции `make_pair()`, которую можно использовать вместо конструктора пары. При ее использовании нет необходимости задавать тип элементов пары.

Ассоциативные контейнеры обладают интерфейсом, похожим на интерфейс последовательных контейнеров.

Конструкторы обеспечивают следующие способы создания ассоциативных контейнеров:

```
container <тип>s0;//пустой контейнер
container <тип>s1 (s0);//копирование
container <тип>s2 (begin, end);//инициализация диапазоном
```

Для всех ассоциативных контейнеров определены деструктор, опреация присваивания и операции отношения.
Ассоциативные контейнеры предоставляют стандартный доступ с помощью итераторов – прямых и обратных.

`iterator begin()`	Возвращают итератор на начало контейнера (итерации будут производиться в прямом направлении)
`iterator end()`	Возвращают итератор на конец контейнера (итерации в прямом направлении будут закончены)
`reverse_iterator begin()`	Возвращают реверсивный итератор на конец контейнера (итерации будут производиться в обратном направлении)
`reverse_iterator end()`	Возвращают реверсивный итератор на начало контейнера (итерации в обратном направлении будут закончены

Ассоциативные контейнеры обладают интерфейсом, похожим на интерфейс последовательных контейнеров.

Операция или метод	**Пояснение**

`bool empty() const`
`size_type size() const`
`size_type max_size()`
	Методы определения размеров

`insert()`	Добавляет один элемент или диапазон элементов
`erase()`	Удаляет один элемент или диапазон элементов
`clear()`	Удаляет все элементы
`swap()`	Обмен данными с контейнером того же типа
`key_comp()` `value_comp()`	Возвращают объекты-функторы для сравнения ключей и значений

`find()`
`count()`
`lower_bound()`
`upper_bound()`
	Методы поиска (метод count() вычисляет для мульти контейнеров количество элементов с заданным ключом)

2.2. Ассоциативный контейнер словарь (map)

Ассоциативный контейнер **map** – это последовательность пар (ключ, значение), которая обеспечивает быстрое получение значения по ключу. Контейнер `map` предоставляет двунаправленные итераторы.

Ассоциативный контейнер `map` требует, чтобы для типов ключа существовала операция “<”. Он хранит свои элементы отсортированными по ключу так, что перебор происходит по порядку.

Спецификация шаблона для класса map:

```
template <class Key, class T, class Comp = less <Key>, class Allocator = allocator <pair> >
class std::map {. . . };
```

`class Key` – параметр ключ,

`class T` – параметр, определяющий тип значений,

`class Comp = less <Key>` - параметр, определяющий критерий упорядочения, по умолчанию less (по возрастанию ключа).

В классе map определены следующие конструкторы:

```
explicit map(const Comp& c=Comp(), const Allocator &a = Allocator());//конструктор пустого ассоциативного контейнера,
map(const map<Key,T,Comp,Allocator>& ob);// конструктор копии
template<class InIter> map(InIter first, InIter last, const Comp& c=Comp(),const Allocator &a = Allocator());//конструктор
// ассоциативного контейнера, содержащего диапазон элементов.
```

Определена операция присваивания:

```
map& operator=(const map&);
```

Определены следующие операции: `==, <, <=, !=, >, >=.`

В `map` хранятся пары ключ/значение в виде объектов типа `pair`.

Типичная операция для ассоциативного контейнера – это ассоциативный поиск при помощи операции индексации ([]).

```
mapped_type& operator[](const key_type& K);

//создание контейнера map
#include <iostream>
```

```
#include <map>
using namespace std;
typedef map<int, float> tmap;//определение типа tmap
void main()
{
     tmap m1;//словарь
     tmap::iterator mi1;//итератор
     int n;//количество элементов
     cout<<"\nn-?";
     cin>>n;
     float a;
     for(int i=0;i<n;i++)
     {
          cout<<"?";
          cin>>a;
//создаем пару и добавляем ее в словарь
          m1.insert(make_pair(i,a));
     }
. . . . . .
}
```

2.2. Ассоциативный контейнер словарь дубликатами (multimap)

Словари с дубликатами (multimap) допускают хранение элементов с одинаковыми ключами. Поэтому для них не определена операция доступа по индексу. Элементы с одинаковыми ключами хранятся в словаре в порядке их занесения. При удалении по ключу функция erase возвращает количество удаленных элементов. В остальном они аналогичны обычным словарям.

2.3. Ассоциативный контейнер множество (set)

Множества `set` можно рассматривать как ассоциативные массивы, в которых значения не играют роли, так что мы отслеживаем только ключи.

Множество, как и словарь, требует, чтобы для типа T существовала операция “меньше” (<). Оно хранит свои элементы отсортированными, так что перебор происходит по порядку.

В множестве хранятся объекты, упорядоченные по некоторому ключу, являющемуся атрибутом самого объекта. Например, множество может хранить объекты класса`Person`, упорядоченные в алфавитном порядке по значению ключевого поля name. Если в множестве хранятся значения одного из встроенных типов, например `int`, то ключом является сам элемент.

```
     template <class T, class Cmp = less <T>, class Allocator =
allocator <T> >
     class std::set{. . . };
```

`class T` – параметр, определяющий тип значений,

`class Comp = less <Key>` - параметр, определяющий критерий

Имеется три способа определить объект типа set:

```
     set<int> set1;   // создается пустое множество
     int a[5] = { 1. 2. 3. 4, 5};
     set<int> set2(a, a + 5);// инициализация копированием
     set<int> set3(set2);     // инициализация другим множеством
```

Для вставки элементов в множество можно использовать метод insert(), для удаления — метод erase(). Также к множествам применимы общие для всех контейнеров методы. Во всех ассоциативных контейнерах есть метод count(), возвращающий количе-

ство объектов с заданным ключом. Так как и в множествах, и в словарях все ключи уникальны, то метод count() возвращает либо 0, если элемент не обнаружен, либо 1.

```
//создание контейнера set
#include <iostream>
#include <set>
using namespace std;
typedef set<int, less<int>>tset;//определение типа tset
tset ::iterator i;//итератор
void main()
{
     int a[4]={1,3,5,7};
     tset s(a,a+4);//множество создается копированием
     s.insert(10);//вставка элементов
     s.insert(6);
     for(i=s.begin();i!=s.end();i++)
          cout<<*i<<" ";
     cout<<endl;
}
```

2.4. Ассоциативный контейнер множество дубликатами (multiset)

В множествах с дубликатами ключи могут повторяться. Элементы с одинаковыми ключами хранятся в множестве в порядке их занесения. Функция find() возвращает значение первого найденного элемента или end(), если ни одного элемента с заданным ключом не найдено.

3. Постановка задачи

Задача 1.

1. Создать ассоциативный контейнер.
2. Заполнить его элементами стандартного типа (тип указан в варианте).
3. Добавить элементы в соответствии с заданием
4. Удалить элементы в соответствии с заданием.
5. Выполнить задание варианта для полученного контейнера.
6. Выполнение всех заданий оформить в виде глобальных функций.

Задача 2.

1. Создать ассоциативный контейнер.
2. Заполнить его элементами пользовательского типа (тип указан в варианте). Для пользовательского типа перегрузить необходимые операции.
3. Добавить элементы в соответствии с заданием
4. Удалить элементы в соответствии с заданием.
5. Выполнить задание варианта для полученного контейнера.
6. Выполнение всех заданий оформить в виде глобальных функций.

Задача 3

1. Создать параметризированный класс, используя в качестве контейнера ассоциативный контейнер.
2. Заполнить его элементами.
3. Добавить элементы в соответствии с заданием
4. Удалить элементы в соответствии с заданием.
5. Выполнить задание варианта для полученного контейнера.
6. Выполнение всех заданий оформить в виде методов параметризированного класса.

4. Ход работы

Задача 1

1. Контейнер – словарь (map);
2. тип элементов - int ;
3. Найти среднее арифметическое элементов словаря и добавить его в словарь.
4. Удалить максимальный элемент из вектора.
5. Каждый элемент разделить на минимальное значение вектора.

1. Создать пустой проект. Для этого требуется
   1.1. Запустить MS Visual Studio:
   1.2. Выбрать команду File/New/Project
   1.3. В окне New Project выбрать Win Console 32 Application, в поле Name указать имя проекта (Lab12), в поле Location указать место положения проекта (личную папку), нажать кнопку Ok.
   1.4. В следующем окне выбрать кнопку Next.
   1.5. В следующем окне выбрать кнопку Next.
   1.6. В диалоговом окне Additional Settings установить флажок Empty (Пустой проект) и нажать кнопку Finish.
   1.7. В результате выполненных действий получим пустой проект.
2. Добавим в проект файл Lab12_main.cpp, содержащий основную программу. Для этого нужно:
   2.1. Вызвать контекстное меню проекта в панели Обозреватель решений (Solution Explorer), выбрать в нем пункт меню Add/ New Item.
   2.2. В диалоговом окне Add New Item – Lab12 выбрать Категорию Code, шаблон – C++File (.cpp), задать имя файла Lab12_main. В результате выполненных действий получим пустой файл Lab12_main.cpp, в котором будет редактироваться текст программы.
   2.3. Ввести следующий текст программы:

```
#include <iostream>
#include <map>
using namespace std;
typedef map<int, int>TMap;//определяем тип для работы со словарем
typedef TMap::iterator it;//итератор

//функция для формирования словаря
TMap make_map(int n)
{
	TMap m;//пустой словарь
	int a;
	for(int i=0;i<n;i++)
	{
		cout<<"?";
		cin>>a;
	//создаем пару и добавляем ее в словарь
		m.insert(make_pair(i,a));

	}
	return m ;//возвращаем словарь как результат работы функции
}
//функция для печати словаря
void print_map(TMap m)
{
	for(int i=0;i<m.size();i++)
		cout<<i<<" : "<<m[i]<<" "<<endl;
}
```

```
//основная функция
void main()
void main()
{
	int n;
	cout<<"N?"; cin>>n;//количество элементов
	TMap m=make_map(n);//создать словарь
	print_map(m);//напечатать словарь
}
```

3. Запустить программу на выполнение и протестировать ее работу.
4. Добавим функции для выполнения задания 3.

```
//вычисление среднего арифметического
int srednee(TMap v)
{
	int s=0;
//перебор словаря
	for(int i=0;i<v.size();i++)
		s+=v[i];
	int n=v.size();//количество элементов в словаре
	return s/n;
}
```

5. Добавим в функцию main() вызов функций для выполнения задания 3.

```
void main()
{
	int n;
	cout<<"N?"; cin>>n;
	TMap m=make_map(n);
	print_map(m);
	//вычисление среднего
	int el=srednee(m);
	cout<<"srednee="<<el<<endl;
	//добавление в конец
	m.insert(make_pair(n,el));
	print_map(m);
}
```

6. Запустить программу на выполнение и протестировать ее работу.
7. Добавим функции для выполнения задания 4.

```
/поиск максимального элемента
int Max(TMap v)
{
	it i=v.begin();
	int nom=0,//номер максимального
	k=0;//счетчик элементов
	int m=(*i).second;//значение первого элемента
	while(i!=v.end())
	{
		if(m<(*i).second)
		{
			m=(*i).second;
			nom=k;
		}
		i++;//итератор
		k++;//счетчик элементов
	}
	return nom;//номер max
}
```

8. Добавим в функцию main() вызов функций для выполнения задания 4.

```
void main()
{
	int n;
	cout<<"N?"; cin>>n;
	TMap m=make_map(n);
	print_map(m);
```

```
	. . . . .
	int max=Max(m);
	cout<<"max="<<m[max]<<"  nom="<<max<<endl;
	m.erase(max);//удаление элемента
	print_map(m);
}
```

9. Запустить программу на выполнение и протестировать ее работу.
10. Добавим функции для выполнения задания 5.

```
int Min(TMap v)
{
	it i=v.begin();int nom=0,k=0;
	int m=(*i).second;//значение первого элемента
	while(i!=v.end())
	{
		if(m>(*i).second)
		{
			m=(*i).second;
			nom=k;
		}
		i++;//итератор
		k++;//счетчик элементов
	}
	return nom;//номер max
}
void delenie(TMap &v)
{

	int m=v[Min(v)];//значение минимального элемента
	for(int i=0;i<v.size();i++)
		v[i]=v[i]/m;
}
```

11. Добавим в функцию main() вызов функций для выполнения задания 4.

```
void main()
{
	int n;
	cout<<"N?"; cin>>n;
	TMap m=make_map(n);
	print_map(m);
	. . . . . .
	int min=Min(m);
	cout<<"min="<<m[min]<<"  nom="<<min<<endl;
	delenie(m);
	print_map(m);
}
```

12. Запустить программу на выполнение и протестировать ее работу.

Задача 2.

1. Контейнер - словарь.
2. Тип элементов Time (см. лабораторную работу №3).
3. Найти среднее арифметическое словаря и добавить его в словарь под номером k.
4. Удалить максимальный элемент из словаря.
5. Каждый элемент разделить на минимальное значение словаря.
6. Выполнение всех заданий оформить в виде глобальных функций.

1. Добавим новый проект в существующее решение. Для этого требуется вызвать контекстное меню для Решения (Solution) lab12 в Обозревателе решений (Solution Explorer).
2. В контекстном меню выбрать пункт Add./ New Project. В окне New Project выбрать Win Console 32 Application, в поле Name указать имя проекта (Zadacha_2), в поле Location указать место положения проекта (личную папку), нажать кнопку Ok.

3. В следующем окне выбрать кнопку Next.
4. В следующем окне выбрать кнопку Next.
5. В диалоговом окне Additional Settings установить флажок Empty (Пустой проект) и нажать кнопку Finish.
6. В результате выполненных действий получим пустой проект Zadacha_2, добавленный в решение Lab12.
7. Добавляем в него класс Time из лабораторной работы 11. Для этого нужно вызвать контекстное меню проекта и выбрать пункт Добавить/Существующий элемент (Add/Existing Item). В диалоговом окне выбрать нужные файлы. При использовании файла Time.h в директиве #include нужно будет указывать полный путь к этому файлу, т. к. он будет расположен в папке отличной от папки текущего проекта.
8. Добавим в проект файл zadacha2_main.cpp, содержащий основную программу. Для этого нужно:
    a. Вызвать контекстное меню проекта в панели Обозреватель решений (Solution Explorer), выбрать в нем пункт меню Add/ New Item.
    b. В диалоговом окне Add New Item – Zadacha_2 выбрать Категорию Code, шаблон – C++File (.cpp), задать имя файла zadacha2_main.cpp. В результате выполненных действий получим пустой файл zadacha2_main.cpp, в котором будет редактироваться текст программы.
9. Ввести следующий текст программы:

```
#include <iostream>
#include <map>
#include "D:\РАБОТА\ЛабыС++\ПВИ МВД\...\time.h"
using namespace std;

typedef map<int, Time>TMap;//определяем тип для работы со словаре
typedef TMap::iterator it;

//функция для формирования словаря
TMap make_map(int n)
{
TMap m;//пустой словарь
Time a;
	for(int i=0;i<n;i++)
	{
		cin>>a;
	//создаем пару и добавляем ее в словарь
		m.insert(make_pair(i,a));
	}
	return m ;//возвращаем вектор как результа работы функции
}
//функция для печати словаря
void print_map(TMap m)
{
	for(int i=0;i<m.size();i++)
		cout<<i<<" - "<<m[i]<<" "<<endl;
}
//вычисление среднего арифметического
Time srednee(TMap m)
{
	Time s=m[0];//первый элемент - начальное значение суммы
//перебор словаря
	for(int i=1;i<m.size();i++)
		s=s+m[i];
	int n=m.size();//количество элементов в словаре
	return s/n;
}
//поиск максимального элемента
```

```
int Max(TMap v)
{
	it i=v.begin();int nom=0,k=0;
	Time m=(*i).second;//значение первого элемента
	while(i!=v.end())
	{
		if(m<(*i).second)
		{
			m=(*i).second;
			nom=k;
		}
		i++;//итератор
		k++;//счетчик элементов
	}
	return nom;//номер max
}
//поиск минимального элемента
int Min(TMap v)
{
	it i=v.begin();int nom=0,k=0;
	Time m=(*i).second;//значение первого элемента
	while(i!=v.end())
	{
		if(m>(*i).second)
		{
			m=(*i).second;
			nom=k;
		}
		i++;//итератор
		k++;//счетчик элементов
	}
	return nom;//номер max
}

void delenie(TMap &v)
{

	Time m=v[Min(v)];
	for(int i=0;i<v.size();i++)
		v[i]=v[i]/m;
}
void main()
{
	int n;
	cout<<"N?"; cin>>n;
	map<int, Time> m=make_map(n);
	print_map(m);
	//вычисление среднего
	Time el=srednee(m);
	cout<<"srednee="<<el<<endl;
	//добавление в конец
	m.insert(make_pair(n,el));
	print_map(m);
	int max=Max(m);
	cout<<"max="<<m[max]<<"  nom="<<max<<endl;
	m.erase(max);
	print_map(m);
	int min=Min(m);
	cout<<"min="<<m[min]<<"  nom="<<min<<endl;
	delenie(m);
	print_map(m);
}
```

13. Запустить программу на выполнение и протестировать ее работу.

Задача 3.

1. Контейнер - словарь.
2. Тип элементов Time (см. лабораторную работу №3).
3. Найти среднее арифметическое словаря и добавить его в словарь под номером k.
4. Удалить максимальный элемент из словаря.
5. Каждый элемент разделить на минимальное значение словаря.
6. Выполнение всех заданий оформить в виде методов параметризированного класса.

1. Добавим новый проект в существующее решение. Для этого требуется вызвать контекстное меню для Решения (Solution) lab12 в Обозревателе решений (Solution Explorer).
2. В контекстном меню выбрать пункт Add./ New Project. В окне New Project выбрать Win Console 32 Application, в поле Name указать имя проекта (Zadacha_3), в поле Location указать место положения проекта (личную папку), нажать кнопку Ok.
3. В следующем окне выбрать кнопку Next.
4. В следующем окне выбрать кнопку Next.
5. В диалоговом окне Additional Settings установить флажок Empty (Пустой проект) и нажать кнопку Finish.
6. В результате выполненных действий получим пустой проект Zadacha_3, добавленный в решение Lab12.
7. Добавляем в него класс Time из лабораторной работы 11. Для этого нужно вызвать контекстное меню проекта и выбрать пункт Добавить/Существующий элемент (Add/Existing Item). В диалоговом окне выбрать нужные файлы. При использовании файла Time.h в директиве #include нужно будет указывать полный путь к этому файлу, т. к. он будет расположен в папке отличной от папки текущего проекта.
8. Добавим в проект файл Container.h, содержащий описание и определение методов параметризированнго класса. Для этого нужно:
    a. Вызвать контекстное меню проекта в панели Обозреватель решений (Solution Explorer), выбрать в нем пункт меню Add/ New Item.
    b. В диалоговом окне Add New Item – Zadacha_2 выбрать Категорию Code, шаблон – Header File(.h), задать имя файла Container.h. В результате выполненных действий получим пустой файл Container.h, в котором будет редактироваться текст программы.
9. Ввести следующий текст программы:

```
#pragma once
#include <iostream>
#include <map> //словарь
using namespace std;
//параметризированный класс
template<class T>
class Container
{
map<int, T> v; //контейнер словарь
int len; //длина словаря

public:
	Container(void); //конструктор без параметров
	Container(int n); //конструктор с параметрами
	void Print(); //печать
	~Container(void); //деструктор
};
//реализация методов
//конструктор без параметров
template <class T>
Container<T>::Container()
```

```
{
	len=0;

}
//деструктор
template <class T>
Container<T>::~Container(void)
{
}
//конструктор с параметрами
template <class T>
Container<T>::Container(int n)
{
	T a;
	for(int i=0;i<n;i++)
	{

		cin>>a;
		v[i]=a;//записать а в словарь
	}
	len=v.size();
}
//вывод контейнера
template <class T>
void Container<T>::Print()
{
	for(int i=0;i<v.size();i++)
		cout<<i<<" - "<<v[i]<<" "<<endl;
	cout<<endl;
}
```

10. Добавим в проект файл zadacha3_main.cpp, содержащий основную программу. Для этого нужно:
    a. Вызвать контекстное меню проекта в панели Обозреватель решений (Solution Explorer), выбрать в нем пункт меню Add/ New Item.
    b. В диалоговом окне Add New Item – Zadacha_3 выбрать Категорию Code, шаблон – C++File (.cpp), задать имя файла zadacha3_main.cpp. В результате выполненных действий получим пустой файл zadacha3_main.cpp, в котором будет редактироваться текст программы.
11. Ввести следующий текст программы:

```
#include "D:\РАБОТА\ЛабыС++\ПВИ МВД\лабы 2009_2010\lab11\time.h"
#include "Container.h"
#include <iostream>
using namespace std;

void main()
{
	int n; //количество элементов в контейнере
	cout<<"N?"; cin>>n;
	Container <Time> v(n);//создать контенер
	v.Print();//напечатать контейнер
}
```

12. Запустить программу на выполнение и протестировать ее работу.
13. Добавим функции для выполнения задания 3 в файл `Container.h`

```
//вычисление среднего арифметического
template<class T>
T Container<T>::Srednee()
{
	Time s=v[0]; //начальное значение суммы - первый элемент словаря
//перебор словаря
	for(int i=1;i<v.size();i++)
```

```
            s=s+v[i];
      int n=v.size();//количество элементов в словаре
      return s/n;
}
//добавление
template<class T>
void Container<T>::Add(int n, T el)
{
      v.insert(make_pair(n,el));//формируем пару и добавляем ее в словарь
}
```

14. Добавим функции для выполнения задания 3 в функцию main()

```
void main()
{
      int n;
      cout<<"N?"; cin>>n;
      Container<Time> v(n);
      v.Print();
      Time t=v.Srednee();//найти среднее арифметическое
      cout<<"Srednee="<<t<<endl;
      cout<<"Add srednee"<<endl;
      cout<<"pos?";
      int pos;
      cin>>pos;//позиция для добавления
      v.Add(pos,t);//добавление
      v.Print();//печать
}
```

15. Запустить программу на выполнение и протестировать ее работу.

16. Добавим функции для выполнения задания 4 в файл `Container.h`

```
template <class T>
int Container<T>::Max()
{
map<int, T>::iterator  i=v.begin();//итератор поставили на первый элемент
      int nom=0,k=0;
      Time m=(*i).second;//значение первого элемента
      while(i!=v.end()) //пока нет конца контейнера
      {
            if(m<(*i).second)
            {
                  m=(*i).second;
                  nom=k;
            }
            i++;//итератор
            k++;//счетчик элементов
      }
      return nom;//номер max
   }
void Container<T>::Del()
{
      int max=Max();
      cout<<"max="<<v[max]<<"  nom="<<max<<endl;
      v.erase(max);//удалить максимальный элемент
}
```

17. Добавим функции для выполнения задания 4 в функцию main():

```
      void main()
{
      . . . . .
      cout<<"Delete max:  "<<endl;
      v.Del();
      v.Print();
}
```

18. Запустить программу на выполнение и протестировать ее работу.

19. Добавим функции для выполнения задания 5 в файл Container.h

```
template<class T>
int Container<T>::Min()
{
	map<int, T>::iterator  i=v.begin();
	int nom=0,k=0;
	Time m=(*i).second;//значение первого элемента
	while(i!=v.end())
	{
//минимальный элемент не должен быть равен 0
if((*i).second.get_min()!=0&&(*i).second.get_sec()!=0)
		if(m>(*i).second)
		{
			m=(*i).second;
			nom=k;
		}
		i++;//итератор
		k++;//счетчик элементов
	}
	return nom;//номер min
}
template<class T>
void Container<T>::Delenie()
{
	T m=v[Min()];//найти минимальный элемент
	cout<<"Min= "<<m<<endl;
		for(int i=0;i<v.size();i++)
		v[i]=v[i]/m;
}
```

20. Добавим функции для выполнения задания 4 в функцию main():

```
	void main()
{
	. . . . .
	cout<<"Delenie na min:  "<<endl;
	v.Delenie();
	v.Print();}
```

21. Запустить программу на выполнение и протестировать ее работу.

5. Варианты

<table>
<tr><td>№</td><td colspan="3">Задание</td></tr>
<tr><td rowspan="3">1</td><td colspan="3">Задача 1
1. Контейнер - multimap
2. Тип элементов - double
Задача 2
Тип элементов Time (см. лабораторную работу №3).
Задача 3
Параметризированный класс – Вектор (см. лабораторную работу №7)</td></tr>
<tr><td>Задание 3</td><td>Задание 4</td><td>Задание 5</td></tr>
<tr><td>Найти максимальный элемент и добавить его в начало контейнера</td><td>Найти минимальный элемент и удалить его из контейнера</td><td>К каждому элементу добавить среднее арифметическое контейнера</td></tr>
<tr><td>2</td><td colspan="3">Задача 1
1. Контейнер - set
2. Тип элементов - float
Задача 2
Тип элементов Time (см. лабораторную работу №3).
Задача 3</td></tr>
</table>

<table>
<tr><td></td><td colspan="3">Параметризированный класс – Вектор (см. лабораторную работу №7)</td></tr>
<tr><td></td><td>Задание 3</td><td>Задание 4</td><td>Задание 5</td></tr>
<tr><td></td><td>Найти минимальный элемент и добавить его в конец контейнера</td><td>Найти элемент с заданным ключом и удалить его из контейнера</td><td>К каждому элементу добавить сумму минимального и максимального элементов контейнера</td></tr>
<tr><td>3</td><td colspan="3">Задача 1
1. Контейнер - multiset
2. Тип элементов - double
Задача 2
Тип элементов Time (см. лабораторную работу №3).
Задача 3
Параметризированный класс – Вектор (см. лабораторную работу №7)</td></tr>
<tr><td></td><td>Задание 3</td><td>Задание 4</td><td>Задание 5</td></tr>
<tr><td></td><td>Найти элемент с заданным ключом и добавить его на заданную позицию контейнера</td><td>Найти элемент с заданным ключом и удалить его из контейнера</td><td>Найти разницу между максимальным и минимальным элементами контейнера и вычесть ее из каждого элемента контейнера</td></tr>
<tr><td>4</td><td colspan="3">Задача 1
1. Контейнер - map
2. Тип элементов - int
Задача 2
Тип элементов Time (см. лабораторную работу №3).
Задача 3
Параметризированный класс – Вектор (см. лабораторную работу №7)</td></tr>
<tr><td></td><td>Задание 3</td><td>Задание 4</td><td>Задание 5</td></tr>
<tr><td></td><td>Найти максимальный элемент и добавить его в конец контейнера</td><td>Найти элемент с заданным ключом и удалить его из контейнера</td><td>К каждому элементу добавить среднее арифметическое элементов контейнера</td></tr>
<tr><td>5</td><td colspan="3">Задача 1
1. Контейнер - set
2. Тип элементов - float
Задача 2
Тип элементов Time (см. лабораторную работу №3).
Задача 3
Параметризированный класс – Вектор (см. лабораторную работу №7)</td></tr>
<tr><td></td><td>Задание 3</td><td>Задание 4</td><td>Задание 5</td></tr>
<tr><td></td><td>Найти минимальный элемент и добавить его на заданную позицию контейнера</td><td>Найти элементы большие среднего арифметического и удалить их из контейнера</td><td>Каждый элемент домножить на максимальный элемент контейнера</td></tr>
<tr><td>6</td><td colspan="3">Задача 1
1. Контейнер - multimap
2. Тип элементов - double
Задача 2</td></tr>
</table>

<table>
<tr><td></td><td colspan="3">Тип элементов Money (см. лабораторную работу №3).
Задача 3
Параметризированный класс – Множество (см. лабораторную работу №7)</td></tr>
<tr><td></td><td>Задание 3</td><td>Задание 4</td><td>Задание 5</td></tr>
<tr><td></td><td>Найти максимальный элемент и добавить его в начало контейнера</td><td>Найти минимальный элемент и удалить его из контейнера</td><td>К каждому элементу добавить среднее арифметическое контейнера</td></tr>
<tr><td>7</td><td colspan="3">Задача 1
1. Контейнер - multiset
2. Тип элементов - float
Задача 2
Тип элементов Money (см. лабораторную работу №3).
Задача 3
Параметризированный класс – Вектор (см. лабораторную работу №7)</td></tr>
<tr><td></td><td>Задание 3</td><td>Задание 4</td><td>Задание 5</td></tr>
<tr><td></td><td>Найти минимальный элемент и добавить его в конец контейнера</td><td>Найти элемент с заданным ключом и удалить его из контейнера</td><td>К каждому элементу добавить сумму минимального и максимального элементов контейнера</td></tr>
<tr><td>8</td><td colspan="3">Задача 1
1. Контейнер - map
2. Тип элементов - double
Задача 2
Тип элементов Money (см. лабораторную работу №3).
Задача 3
Параметризированный класс – Вектор (см. лабораторную работу №7)</td></tr>
<tr><td></td><td>Задание 3</td><td>Задание 4</td><td>Задание 5</td></tr>
<tr><td></td><td>Найти элемент с заданным ключом и добавить его на заданную позицию контейнера</td><td>Найти элемент с заданным ключом и удалить его из контейнера</td><td>Найти разницу между максимальным и минимальным элементами контейнера и вычесть ее из каждого элемента контейнера</td></tr>
<tr><td>9</td><td colspan="3">Задача 1
1. Контейнер - set
2. Тип элементов - int
Задача 2
Тип элементов Money (см. лабораторную работу №3).
Задача 3
Параметризированный класс – Вектор (см. лабораторную работу №7)</td></tr>
<tr><td></td><td>Задание 3</td><td>Задание 4</td><td>Задание 5</td></tr>
<tr><td></td><td>Найти максимальный элемент и добавить его в конец контейнера</td><td>Найти элемент с заданным ключом и удалить его из контейнера</td><td>К каждому элементу добавить среднее арифметическое элементов контейнера</td></tr>
<tr><td>10</td><td colspan="3">Задача 1
1. Контейнер - multimap
2. Тип элементов - float</td></tr>
</table>

<table>
<tr><td></td><td colspan="3">Задача 2
Тип элементов Money (см. лабораторную работу №3).
Задача 3
Параметризированный класс – Вектор (см. лабораторную работу №7)</td></tr>
<tr><td></td><td>Задание 3</td><td>Задание 4</td><td>Задание 5</td></tr>
<tr><td></td><td>Найти минимальный элемент и добавить его на заданную позицию контейнера</td><td>Найти элементы большие среднего арифметического и удалить их из контейнера</td><td>Каждый элемент домножить на максимальный элемент контейнера</td></tr>
<tr><td>11</td><td colspan="3">Задача 1
1. Контейнер - multiset
2. Тип элементов - float
Задача 2
Тип элементов Money (см. лабораторную работу №3).
Задача 3
Параметризированный класс – Список (см. лабораторную работу №7)</td></tr>
<tr><td></td><td>Задание 3</td><td>Задание 4</td><td>Задание 5</td></tr>
<tr><td></td><td>Найти среднее арифметическое и добавить его в начало контейнера</td><td>Найти элемент с заданным ключом и удалить их из контейнера</td><td>Из каждого элемента вычесть минимальный элемент контейнера</td></tr>
<tr><td>12</td><td colspan="3">Задача 1
1. Контейнер - map
2. Тип элементов - int
Задача 2
Тип элементов Pair (см. лабораторную работу №3).
Задача 3
Параметризированный класс – Список (см. лабораторную работу №7)</td></tr>
<tr><td></td><td>Задание 3</td><td>Задание 4</td><td>Задание 5</td></tr>
<tr><td></td><td>Найти среднее арифметическое и добавить его на заданную позицию контейнера</td><td>Найти элементы ключами из заданного диапазона и удалить их из контейнера</td><td>Из каждого элемента вычесть среднее арифметическое контейнера.</td></tr>
<tr><td>13</td><td colspan="3">Задача 1
1. Контейнер - set
2. Тип элементов - double
Задача 2
Тип элементов Pair (см. лабораторную работу №3).
Задача 3
Параметризированный класс – Список (см. лабораторную работу №7)</td></tr>
<tr><td></td><td>Задание 3</td><td>Задание 4</td><td>Задание 5</td></tr>
<tr><td></td><td>Найти максимальный элемент и добавить его в конец контейнера</td><td>Найти элементы ключами из заданного диапазона и удалить их из контейнера</td><td>К каждому элементу добавить среднее арифметическое контейнера.</td></tr>
<tr><td>14</td><td colspan="3">Задача 1
1. Контейнер - multimap</td></tr>
</table>

<table>
<tr><td></td><td colspan="3">2. Тип элементов - float
Задача 2
Тип элементов Pair (см. лабораторную работу №3).
Задача 3
Параметризированный класс – Список (см. лабораторную работу №7)</td></tr>
<tr><td></td><td>Задание 3</td><td>Задание 4</td><td>Задание 5</td></tr>
<tr><td></td><td>Найти минимальный элемент и добавить его на заданную позицию контейнера</td><td>Найти меньше среднего арифметического и удалить их из контейнера</td><td>Каждый элемент разделить на максимальный элемент контейнера.</td></tr>
<tr><td>15</td><td colspan="3">Задача 1
1. Контейнер - multiset
2. Тип элементов - double
Задача 2
Тип элементов Pair (см. лабораторную работу №3).
Задача 3
Параметризированный класс – Список (см. лабораторную работу №7)</td></tr>
<tr><td></td><td>Задание 3</td><td>Задание 4</td><td>Задание 5</td></tr>
<tr><td></td><td>Найти среднее арифметическое и добавить его в конец контейнера</td><td>Найти элементы ключами из заданного диапазона и удалить их из контейнера</td><td>К каждому элементу добавить сумму минимального и максимального элементов контейнера.</td></tr>
</table>

6. Контрольные вопросы

1. Что представляет собой ассоциативный контейнер?
2. Перечислить ассоциативные контейнеры библиотеки STL.
3. Каким образом можно получить доступ к элементам ассоциативного контейнера?
4. Привести примеры методов, используемых в ассоциативных контейнерах.
5. Каким образом можно создать контейнер map? Привести примеры.
6. Каким образом упорядочены элементы в контейнере map по умолчанию? Как изменить порядок на обратный?
7. Какие операции определены для контейнера map?
8. Написать функцию для добавления элементов в контейнер map с помощью функции make_pair().
9. Написать функцию для добавления элементов в контейнер map с помощью функции операции прямого доступа [].
10. Написать функцию для печати контейнера map с помощью итератора.
11. Написать функцию для печати контейнера map с помощью функции операции прямого доступа [].
12. Чем отличаются контейнеры map и multimap?
13. Что представляет собой контейнер set?
14. Чем отличаются контейнеры map и set?
15. Каким образом можно создать контейнер set? Привести примеры.
16. Каким образом упорядочены элементы в контейнере set по умолчанию? Как изменить порядок на обратный?
17. Какие операции определены для контейнера set?
18. Написать функцию для добавления элементов в контейнер set.

19. Написать функцию для печати контейнера set.
20. Чем отличаются контейнеры set и multiset?

7. Содержание отчета

1) Постановка задачи (общая и конкретного варианта).
2) Функции для решения задачи 1.
3) Основная программа для решения задачи 3
4) Объяснение результатов работы программы.
5) Описание пользовательского класса для решения задачи 2.
6) Определение перегруженных операций для пользовательского класса.
7) Функции для решения задачи 2 .
8) Основная программа для решения задачи 2.
9) Объяснение результатов работы программы.
10) Описание параметризированного класса для решения задачи 3.
11) Определение методов и операций для решения задачи 3.
12) Основная программа для решения задачи 3
13) Объяснение результатов работы программы.
14) Ответы на контрольные вопросы