Министерство образования и науки Российской Федерации  
Федеральное‌ ‌государственное‌ ‌бюджетное‌ ‌образовательное‌ ‌учреждение‌

высшего‌ ‌образования‌

**«Пермский национальный исследовательский  
политехнический университет»**

Кафедра «Информационные технологии и автоматизированные системы»

**ОТЧЁТ**

**по лабораторной работе № 18.12**

Дисциплина: «Информатика»

Тема: Объектно-ориентированное программирование. Ассоциативные контейнеры библиотеки STL

Вариант 21

Выполнил:

Студент группы РИС-20-1б

Ознобихин Елисей Андреевич

Проверила:

Доцент кафедры ИТАС

Полякова О. А.

**Пермь, 2021**

**Цель работы**

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

**Постановка задачи**

Задача 1.

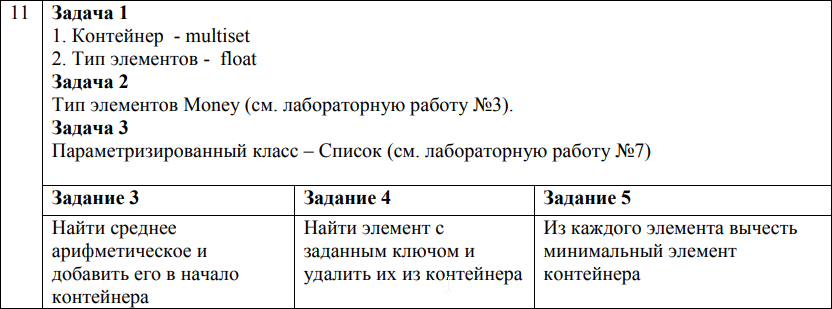
1. Создать ассоциативный контейнер.
2. Заполнить его элементами стандартного типа (тип указан в варианте).
3. Добавить элементы в соответствии с заданием
4. Удалить элементы в соответствии с заданием.
5. Выполнить задание варианта для полученного контейнера.
6. Выполнение всех заданий оформить в виде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

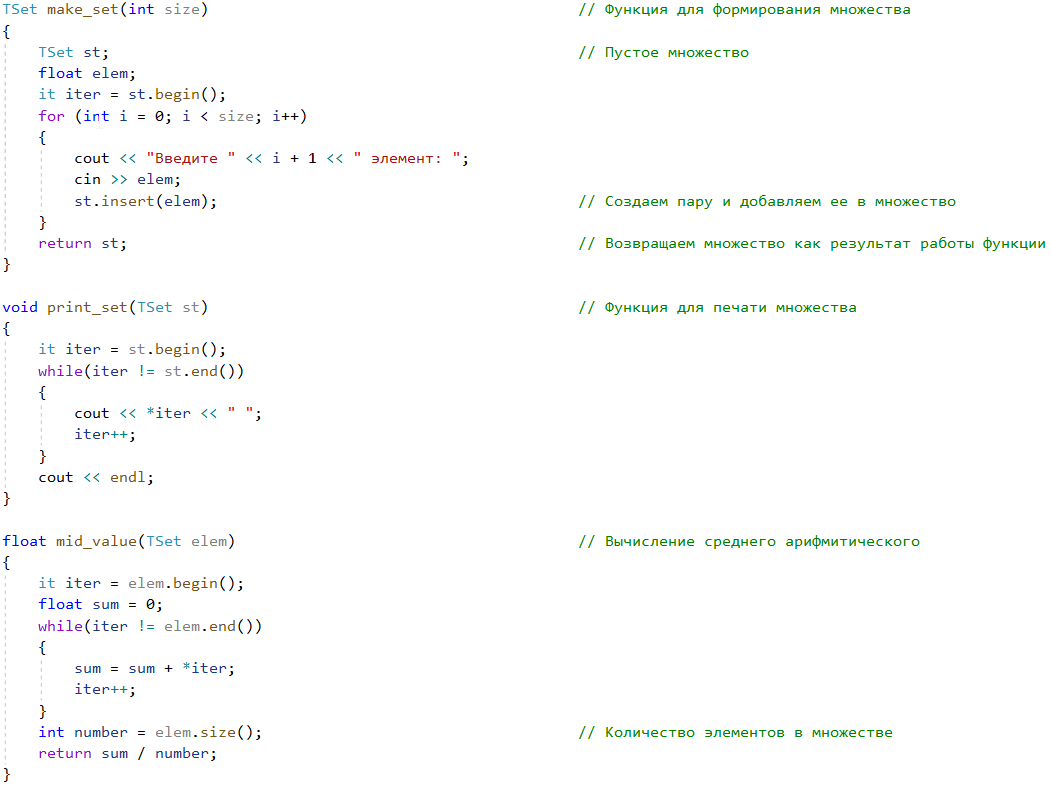
Задача 3

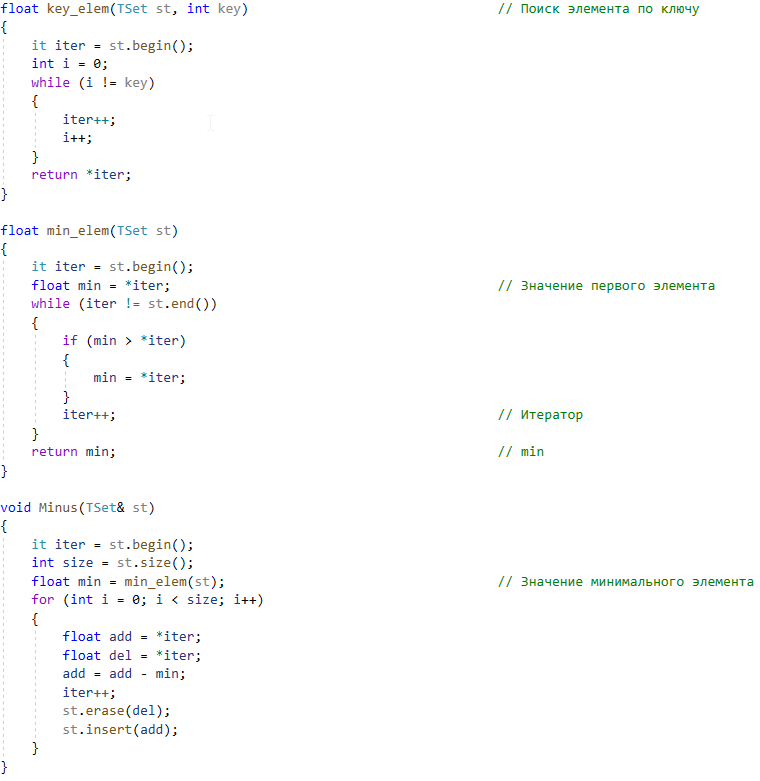
1. Создать параметризированный класс, используя в качестве контейнера ассоциативный контейнер.
2. Заполнить его элементами.
3. Добавить элементы в соответствии с заданием
4. Удалить элементы в соответствии с заданием.
5. Выполнить задание варианта для полученного контейнера.
6. Выполнение всех заданий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



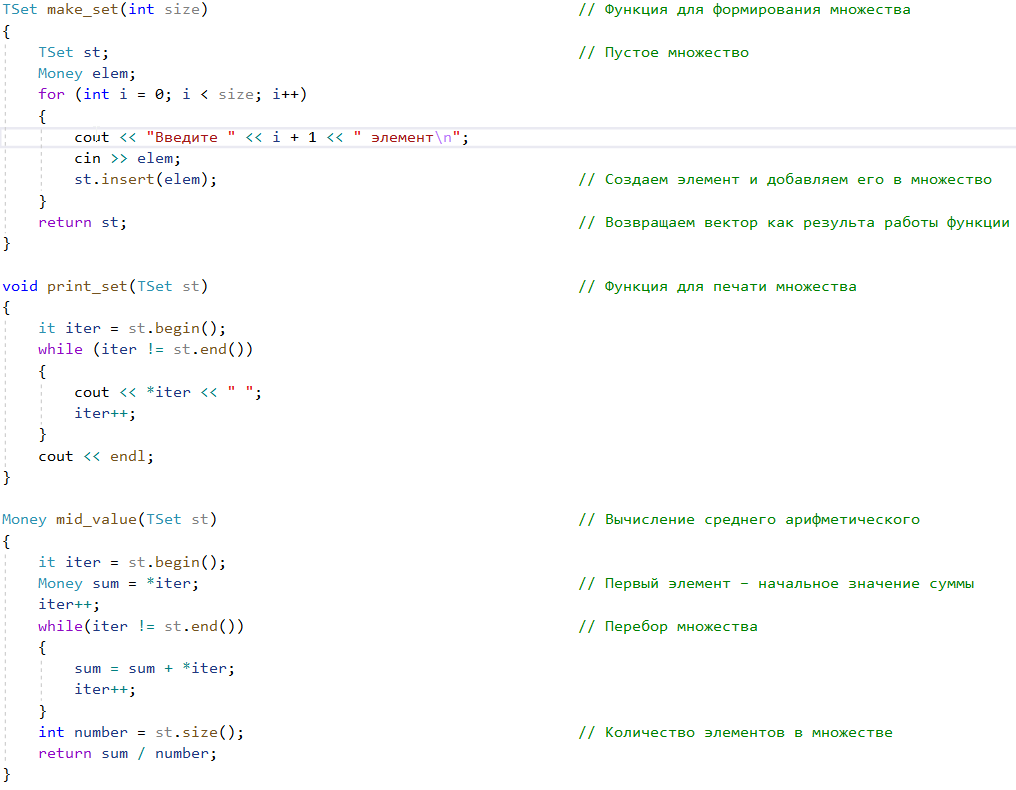
**Анализ задачи**

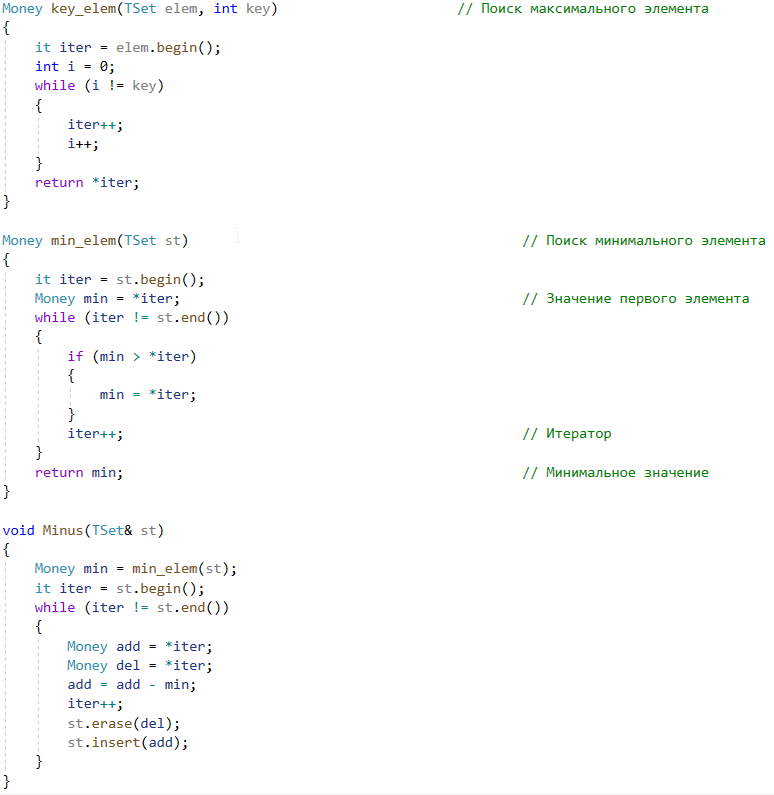
1. Для решения задачи необходимо:
   1. Реализовать определение функций для работы с множество с дубликатами типа float: создание множества, вычисление среднего арифметического, вычитание минимального элемента множества, добавление среднего арифметического во множество, вывод множества.



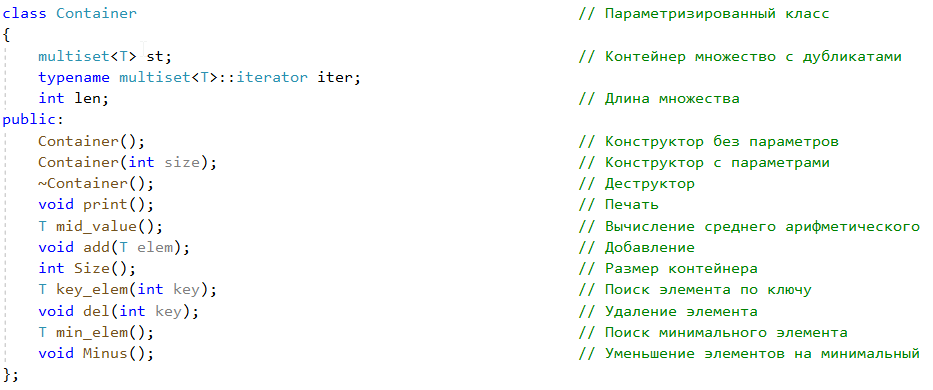


* 1. Реализовать определение функций для работы с множество с дубликатами типа Money: создание множества, вычисление среднего арифметического, вычитание минимального элемента множества, добавление среднего арифметического во множество, вывод множества.

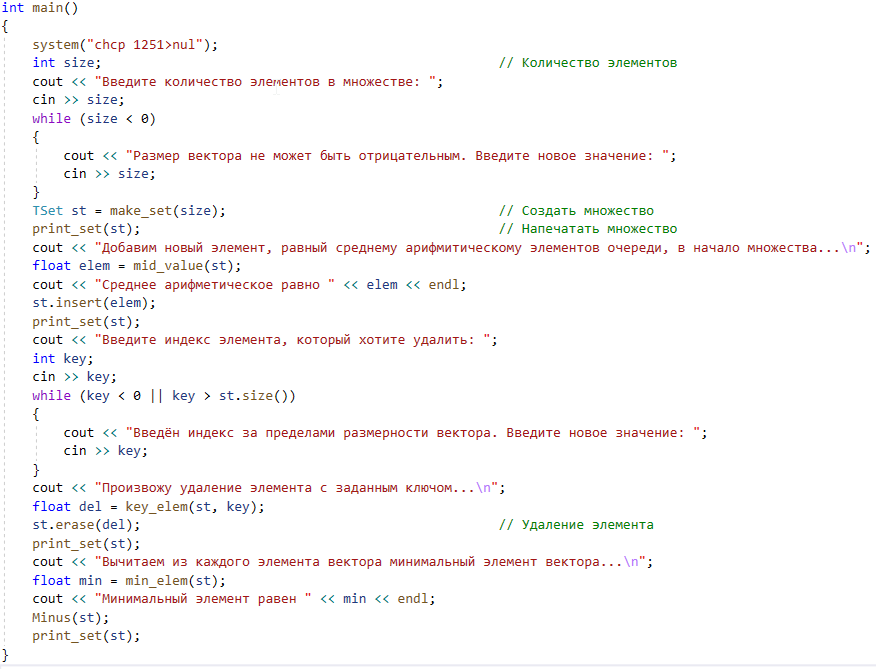


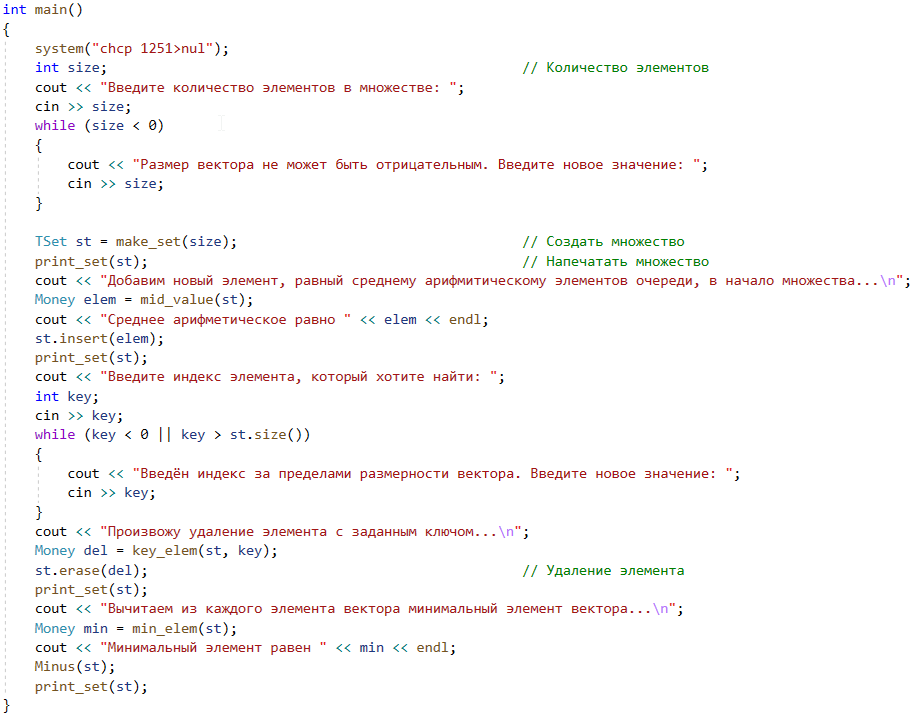


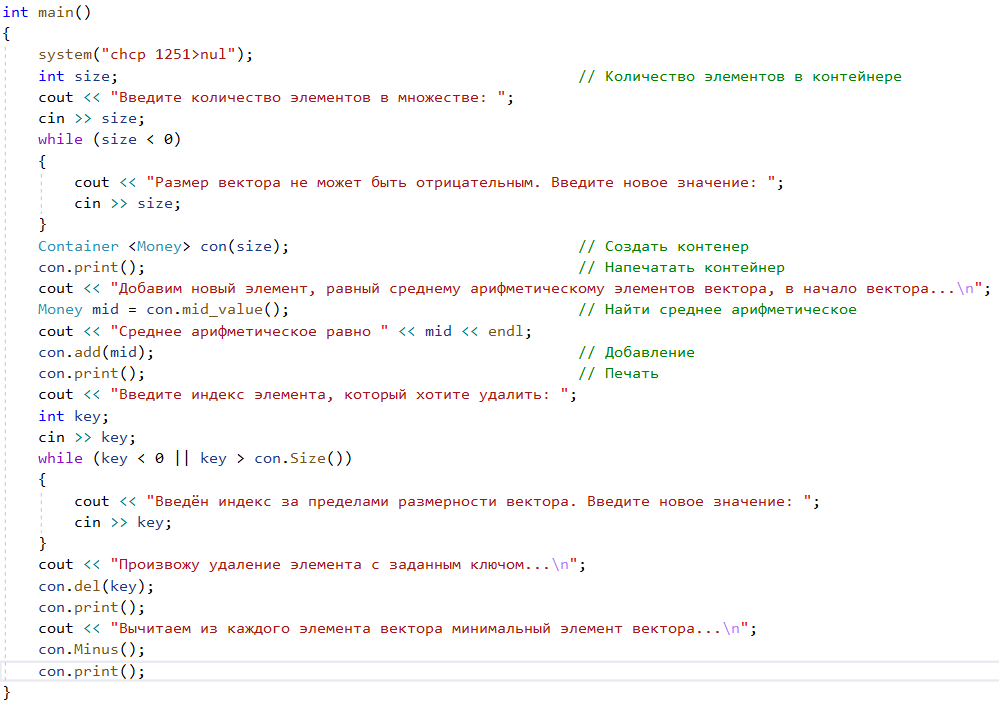
* 1. Реализовать определение параметризированного класса Container и методов данного класса.



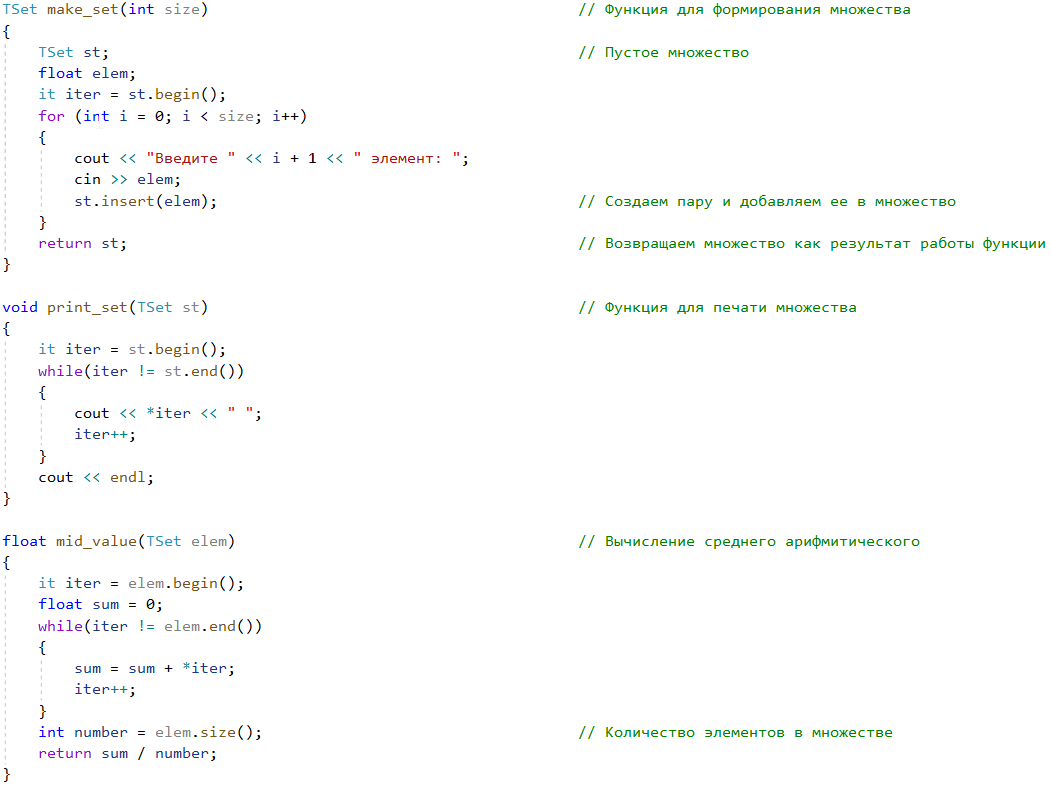
* 1. Реализовать применение этих функций в главной функции.

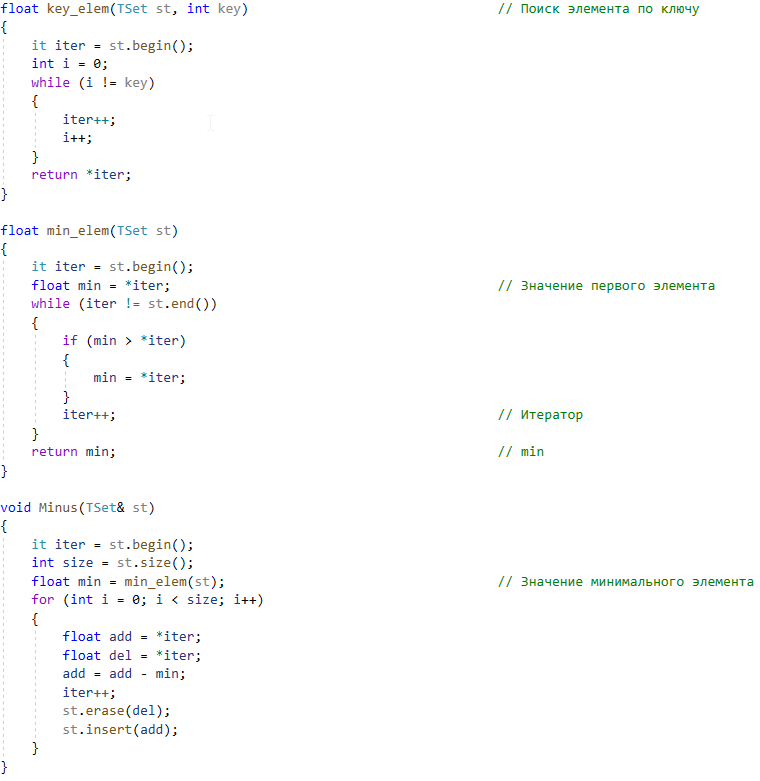




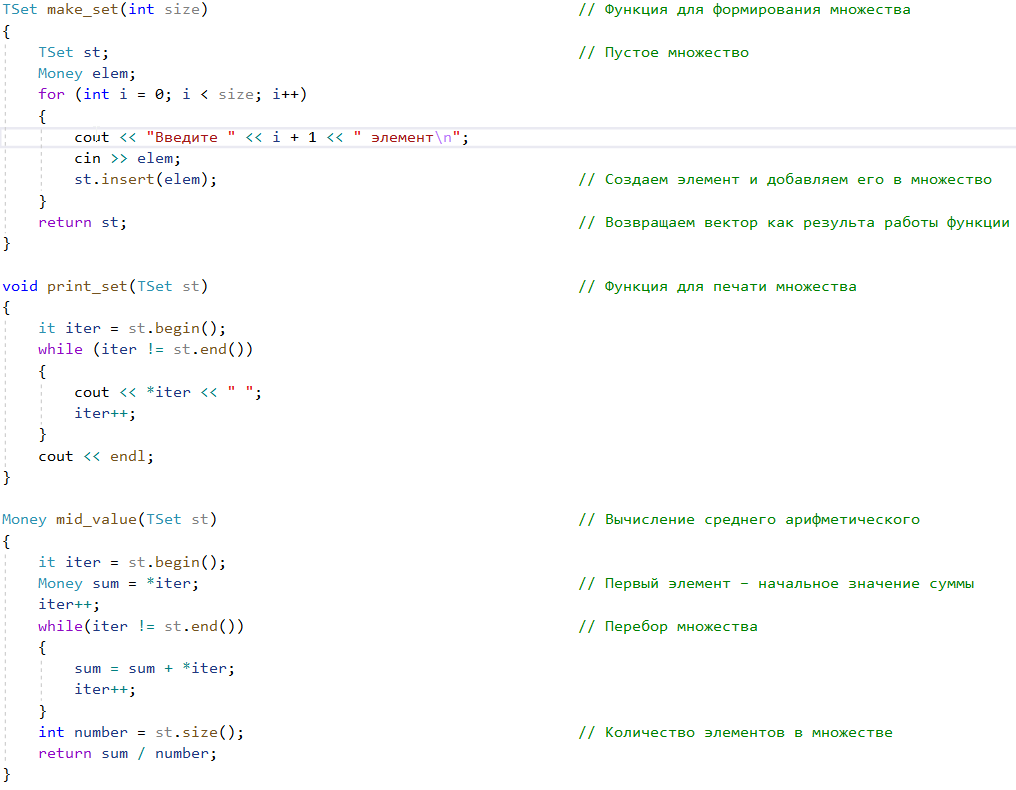


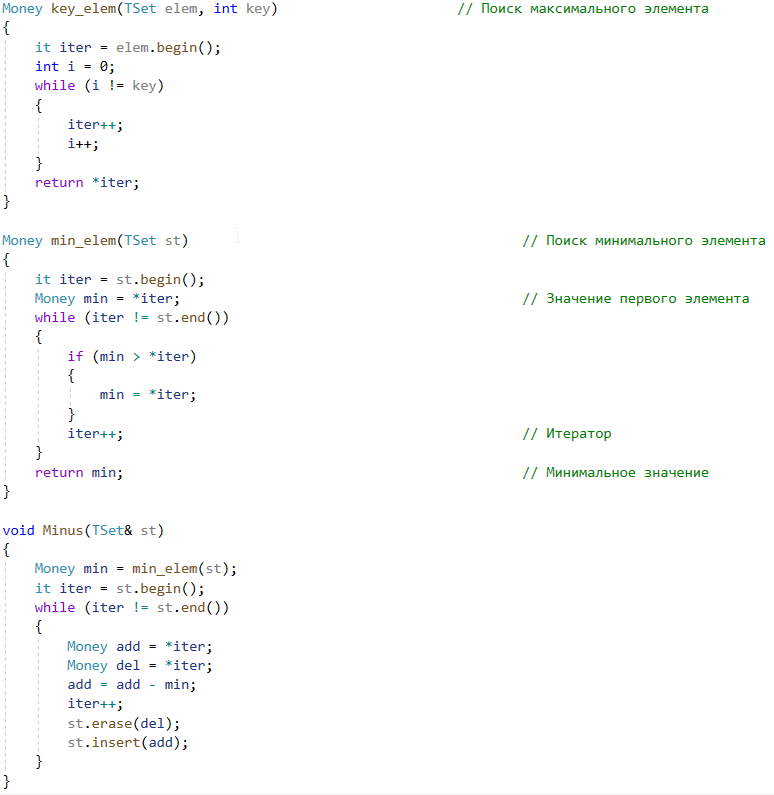
1. Поставленные задачи решены следующими действиями:
   1. В задании 1 реализованы функции для работы с множество с дубликатами типа Money: создание множества, вычисление среднего арифметического, вычитание минимального элемента множества, добавление среднего арифметического во множество, вывод множества.



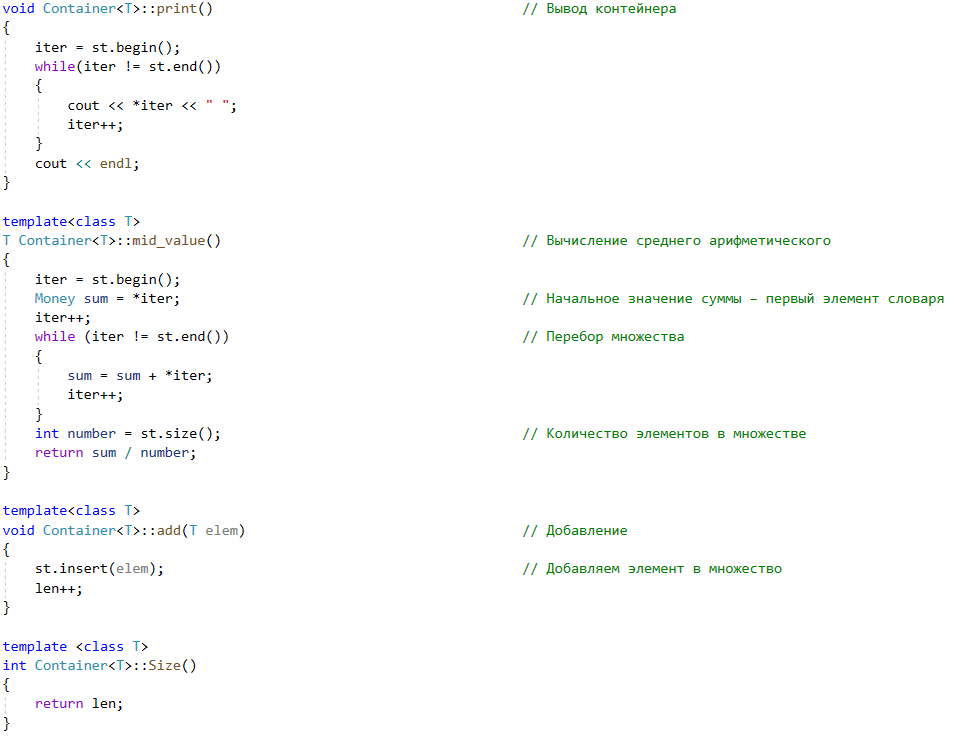


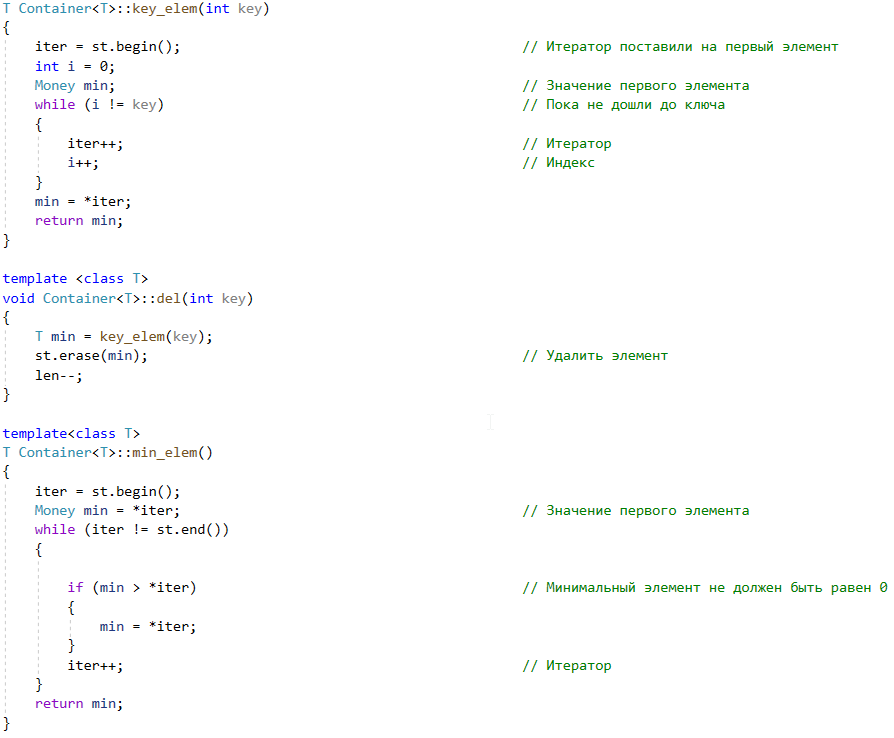
* 1. В задании реализованы функции для работы с множество с дубликатами типа Money: создание множества, вычисление среднего арифметического, вычитание минимального элемента множества, добавление среднего арифметического во множество, вывод множества.

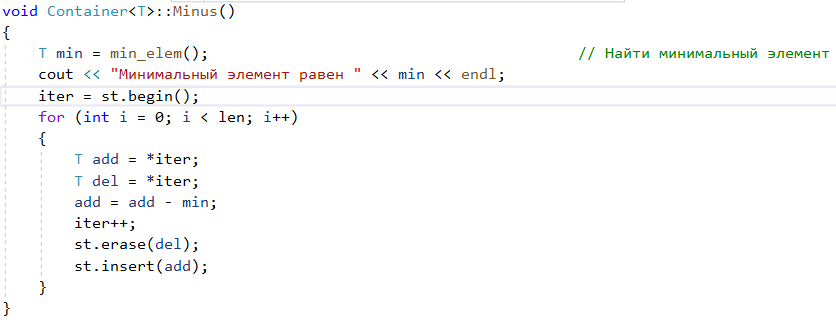




* 1. В задании находится реализация параметризированного списка и методов: создание списка, вычисление среднего арифметического, вычитание минимального элемента списка, добавление среднего арифметического в начало списка, вывод списка.







**Реализация задачи на языке С++**

**(1)**

**Файл с главной программой Лабораторная работа № 18.12(1).cpp**

#include <iostream>

#include <set>

using namespace std;

typedef multiset<float>TSet;

typedef TSet::iterator it;

TSet make\_set(int size)

{

TSet st;

float elem;

it iter = st.begin();

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

{

cout << "Введите " << i + 1 << " элемент: ";

cin >> elem;

st.insert(elem);

}

return st;

}

void print\_set(TSet st)

{

it iter = st.begin();

while(iter != st.end())

{

cout << \*iter << " ";

iter++;

}

cout << endl;

}

float mid\_value(TSet elem)

{

it iter = elem.begin();

float sum = 0;

while(iter != elem.end())

{

sum = sum + \*iter;

iter++;

}

int number = elem.size();

return sum / number;

}

float key\_elem(TSet st, int key)

{

it iter = st.begin();

int i = 0;

while (i != key)

{

iter++;

i++;

}

return \*iter;

}

float min\_elem(TSet st)

{

it iter = st.begin();

float min = \*iter;

while (iter != st.end())

{

if (min > \*iter)

{

min = \*iter;

}

iter++;

}

return min;

}

void Minus(TSet& st)

{

it iter = st.begin();

int size = st.size();

float min = min\_elem(st);

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

{

float add = \*iter;

float del = \*iter;

add = add - min;

iter++;

st.erase(del);

st.insert(add);

}

}

int main()

{

system("chcp 1251>nul");

int size;

cout << "Введите количество элементов в множестве: ";

cin >> size;

while (size < 0)

{

cout << "Размер вектора не может быть отрицательным. Введите новое значение: ";

cin >> size;

}

TSet st = make\_set(size);

print\_set(st);

cout << "Добавим новый элемент, равный среднему арифмитическому элементов очереди, в начало множества...\n";

float elem = mid\_value(st);

cout << "Среднее арифметическое равно " << elem << endl;

st.insert(elem);

print\_set(st);

cout << "Введите индекс элемента, который хотите удалить: ";

int key;

cin >> key;

while (key < 0 || key > st.size())

{

cout << "Введён индекс за пределами размерности вектора. Введите новое значение: ";

cin >> key;

}

cout << "Произвожу удаление элемента с заданным ключом...\n";

float del = key\_elem(st, key);

st.erase(del);

print\_set(st);

cout << "Вычитаем из каждого элемента вектора минимальный элемент вектора...\n";

float min = min\_elem(st);

cout << "Минимальный элемент равен " << min << endl;

Minus(st);

print\_set(st);

}

**(2)**

**Файл с главной программой Лабораторная работа № 18.12(2).cpp**

#include "C:\Users\Асус\Source\Repos\Лабораторная работа № 18.11 (2)\Лабораторная работа № 18.11 (2)\Money.h"

#include "C:\Users\Асус\Source\Repos\Лабораторная работа № 18.11 (2)\Лабораторная работа № 18.11 (2)\Money.cpp"

#include <iostream>

#include <set>

using namespace std;

typedef multiset<Money>TSet;

typedef TSet::iterator it;

TSet make\_set(int size)

{

TSet st;

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

{

cout << "Введите " << i + 1 << " элемент\n";

cin >> elem;

st.insert(elem);

}

return st;

}

void print\_set(TSet st)

{

it iter = st.begin();

while (iter != st.end())

{

cout << \*iter << " ";

iter++;

}

cout << endl;

}

Money mid\_value(TSet st)

{

it iter = st.begin();

Money sum = \*iter;

iter++;

while(iter != st.end())

{

sum = sum + \*iter;

iter++;

}

int number = st.size();

return sum / number;

}

Money key\_elem(TSet elem, int key)

{

it iter = elem.begin();

int i = 0;

while (i != key)

{

iter++;

i++;

}

return \*iter;

}

Money min\_elem(TSet st)

{

it iter = st.begin();

Money min = \*iter;

while (iter != st.end())

{

if (min > \*iter)

{

min = \*iter;

}

iter++;

}

return min;

}

void Minus(TSet& st)

{

Money min = min\_elem(st);

it iter = st.begin();

while (iter != st.end())

{

Money add = \*iter;

Money del = \*iter;

add = add - min;

iter++;

st.erase(del);

st.insert(add);

}

}

int main()

{

system("chcp 1251>nul");

int size;

cout << "Введите количество элементов в множестве: ";

cin >> size;

while (size < 0)

{

cout << "Размер вектора не может быть отрицательным. Введите новое значение: ";

cin >> size;

}

TSet st = make\_set(size);

print\_set(st);

cout << "Добавим новый элемент, равный среднему арифмитическому элементов очереди, в начало множества...\n";

Money elem = mid\_value(st);

cout << "Среднее арифметическое равно " << elem << endl;

st.insert(elem);

print\_set(st);

cout << "Введите индекс элемента, который хотите найти: ";

int key;

cin >> key;

while (key < 0 || key > st.size())

{

cout << "Введён индекс за пределами размерности вектора. Введите новое значение: ";

cin >> key;

}

cout << "Произвожу удаление элемента с заданным ключом...\n";

Money del = key\_elem(st, key);

st.erase(del);

print\_set(st);

cout << "Вычитаем из каждого элемента вектора минимальный элемент вектора...\n";

Money min = min\_elem(st);

cout << "Минимальный элемент равен " << min << endl;

Minus(st);

print\_set(st);

}

**(3)**

**Заголовочный файл Container.h**

#pragma once

#include <iostream>

#include <set>

using namespace std;

template<class T>

class Container

{

multiset<T> st;

typename multiset<T>::iterator iter;

int len;

public:

Container();

Container(int size);

~Container();

void print();

T mid\_value();

void add(T elem);

int Size();

T key\_elem(int key);

void del(int key);

T min\_elem();

void Minus();

};

template <class T>

Container<T>::Container()

{

len = 0;

}

template <class T>

Container<T>::~Container()

{

}

template <class T>

Container<T>::Container(int size)

{

Money elem;

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

{

cout << "Введите " << i + 1 << " элемент\n";

cin >> elem;

st.insert(elem);

}

len = st.size();

}

template <class T>

void Container<T>::print()

{

iter = st.begin();

while(iter != st.end())

{

cout << \*iter << " ";

iter++;

}

cout << endl;

}

template<class T>

T Container<T>::mid\_value()

{

iter = st.begin();

Money sum = \*iter;

iter++;

while (iter != st.end())

{

sum = sum + \*iter;

iter++;

}

int number = st.size();

return sum / number;

}

template<class T>

void Container<T>::add(T elem)

{

st.insert(elem);

len++;

}

template <class T>

int Container<T>::Size()

{

return len;

}

template <class T>

T Container<T>::key\_elem(int key)

{

iter = st.begin();

int i = 0;

Money min;

while (i != key)

{

iter++;

i++;

}

min = \*iter;

return min;

}

template <class T>

void Container<T>::del(int key)

{

T min = key\_elem(key);

st.erase(min);

len--;

}

template<class T>

T Container<T>::min\_elem()

{

iter = st.begin();

Money min = \*iter;

while (iter != st.end())

{

if (min > \*iter)

{

min = \*iter;

}

iter++;

}

return min;

}

template<class T>

void Container<T>::Minus()

{

T min = min\_elem();

cout << "Минимальный элемент равен " << min << endl;

iter = st.begin();

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

{

T add = \*iter;

T del = \*iter;

add = add - min;

iter++;

st.erase(del);

st.insert(add);

}

}

**Файл с главной программой Лабораторная работа № 18.12(3).cpp**

#include "C:\Users\Асус\Source\Repos\Лабораторная работа № 18.11 (2)\Лабораторная работа № 18.11 (2)\Money.h"

#include "C:\Users\Асус\Source\Repos\Лабораторная работа № 18.11 (2)\Лабораторная работа № 18.11 (2)\Money.cpp"

#include "Container.h"

#include <iostream>

using namespace std;

int main()

{

system("chcp 1251>nul");

int size;

cout << "Введите количество элементов в множестве: ";

cin >> size;

while (size < 0)

{

cout << "Размер вектора не может быть отрицательным. Введите новое значение: ";

cin >> size;

}

Container <Money> con(size);

con.print();

cout << "Добавим новый элемент, равный среднему арифметическому элементов вектора, в начало вектора...\n";

Money mid = con.mid\_value();

cout << "Среднее арифметическое равно " << mid << endl;

con.add(mid);

con.print();

cout << "Введите индекс элемента, который хотите удалить: ";

int key;

cin >> key;

while (key < 0 || key > con.Size())

{

cout << "Введён индекс за пределами размерности вектора. Введите новое значение: ";

cin >> key;

}

cout << "Произвожу удаление элемента с заданным ключом...\n";

con.del(key);

con.print();

cout << "Вычитаем из каждого элемента вектора минимальный элемент вектора...\n";

con.Minus();

con.print();

}

**Ответы на вопросы**

1. С нечисловыми индексами
2. Map set multiset multimap
3. Через итератор и “->first” “->second”
4. Size() begin() end()
5. Map M<string,int>;
6. Доступ по итератору, вычисление размера, добавление элемента, удаление по итератору, удаление по ключу
7. По возрастанию
8. M.insert(make\_pair(“A”,1);
9. M[“A”]=1;
10. For (auto iter=A.begin();iter!=A.end();iter++)cout<<iter->first<<iter->second;
11. For (string i=0;i<10;i++) cout << keys[i]<<M[keys[i]]<<endl;
12. Последний допускает хранение дубликатов
13. Множество
14. Set хранит только ключи
15. Set <int> set1;
16. По возрастанию ключа
17. Определение размера, доступ по итератору, добавление элемента
18. Set1.insert(4);
19. For (auto iter=set1.begin();iter!=set1.end();iter++)cout<<\*iter;
20. Последний допускает наличие дубликатов и хранит их в порядке добавления