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 10

Выполнил:

Студент группы ИВТ-20-2Б Галинов О.Ю.

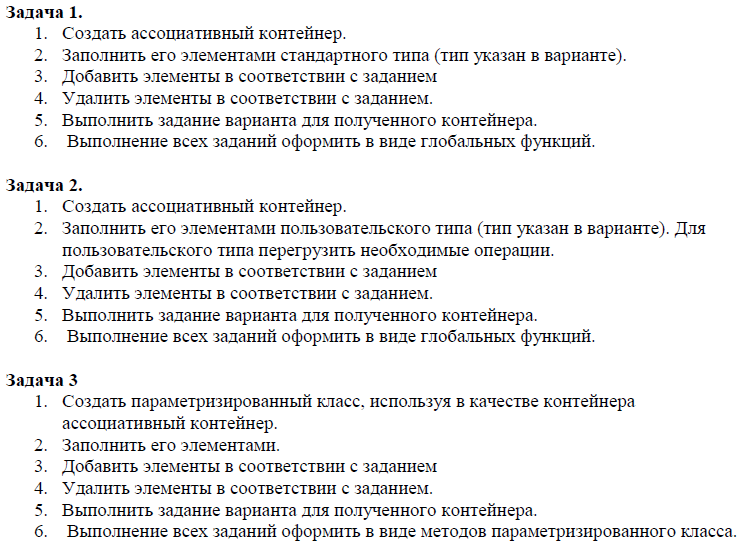
Проверила:

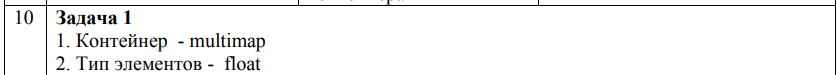
Доцент кафедры ИТАС

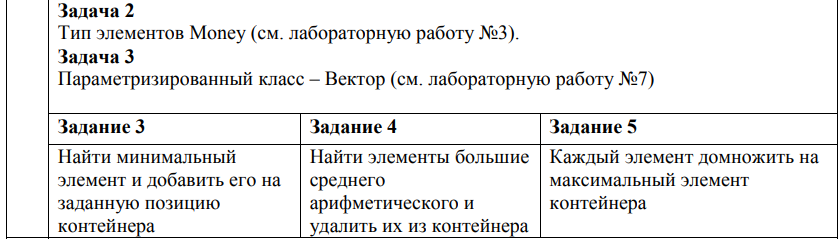
Полякова О.А.

Пермь, 2021

**Постановка задачи**







**Анализ задачи**

**1.** Для решения задач необходимо:

**1.1.** Организовать функцию Task1(), в которой будет использоваться библиотека multimap для создания ассоциативного контейнера, для решения задачи №1.

**1.2.** Организовать функцию Task2(), в которой будет использоваться библиотека multimap для создания ассоциативного контейнера типа Money, для решения задачи №2.

**1.3.** Организовать класс Money для хранения в полях для рублей и копеек(int: rub, kop) для экземпляров данного класса.

**1.4.** Организовать функцию Task3() для решения задачи №3, в которой будет использоваться объект класса Vector.

**1.5.** Организовать перегруженный класс Vector с полями под размер контейнера и ассоциативный контейнер multimap.

**2.** В ходе работы были использованы следующие типы данных:

**2.1.** Функция Task1(), в которой будет использоваться библиотека multimap для создания ассоциативного контейнера – MyMap, типа float для ключа и типа float для хранящихся данных, для решения задачи №1. Переменная size устанавливает размер контейнера.

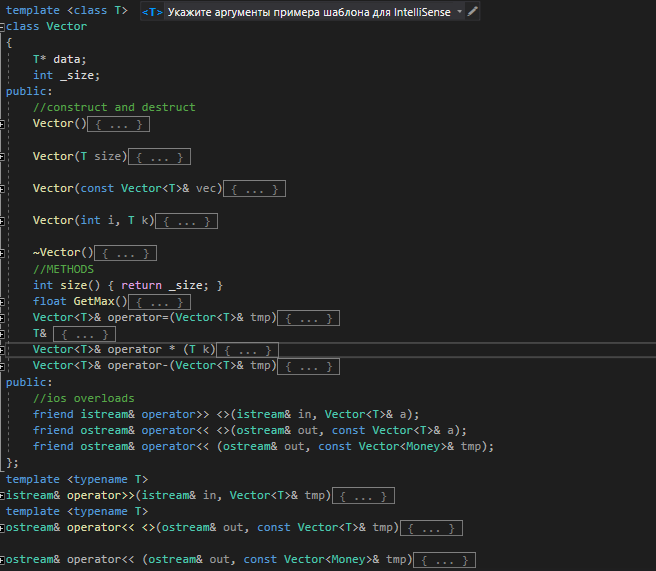


**2.2.** Функция Task2(), в которой будет использоваться библиотека multimap для создания ассоциативного контейнера – MyMap, типа float для ключа и типа Money для хранящихся данных, для решения задачи №2. Переменная size устанавливает размер контейнера.



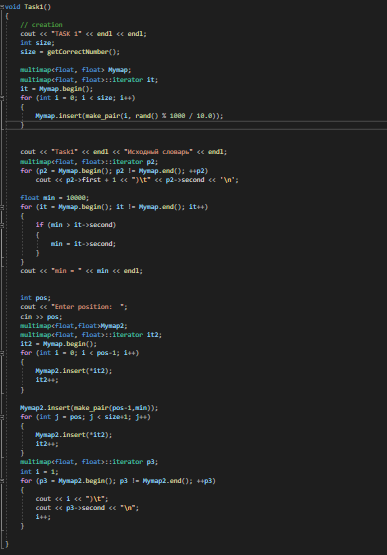
**2.3.** Функция Task3(), в которой будет использоваться, в которой будет использоваться класс Vector для создания экземпляра класса Vector, для решения задачи №3. Переменная size устанавливает размер контейнера.

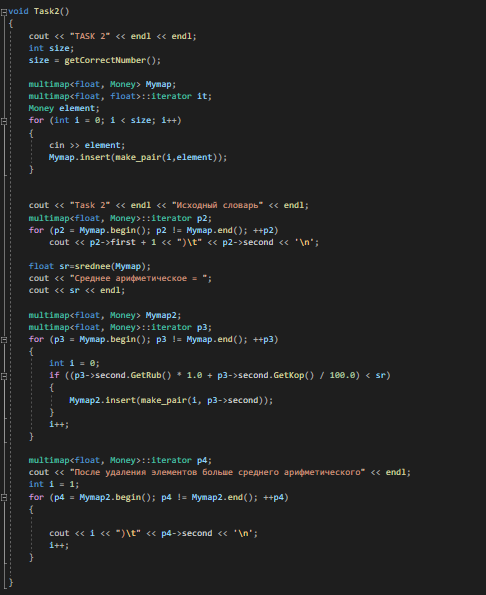


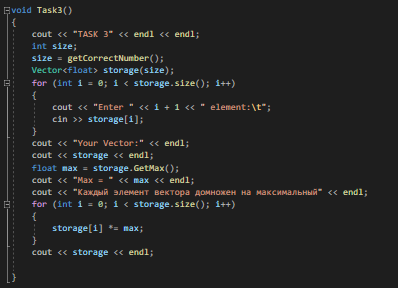


**3.** Для решения задачи данные были представлены в следующем виде:

**3.1.** Решения задач осуществляются в функциях Task1(), Task2(), Task3().

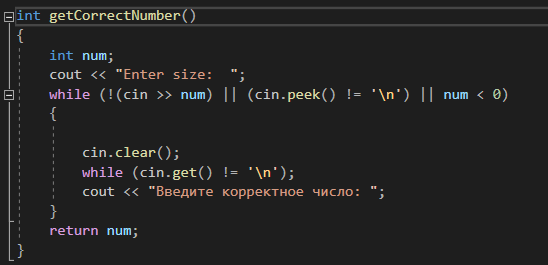




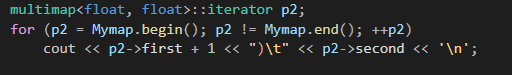


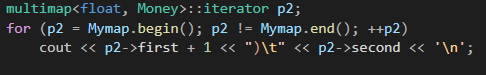
**4.** Для операций ввода и вывода использовались следующие операторы и функции:

**4.1.** Для ввода целочисленных значений используется функция GetCorrectNumber(), которая проверяет введённое данное. Таким образом будут игнорироваться данные типа char, double, float.



**4.2.** Для печати ассоциативного контейнера – MyMap(для типов float и Money) в консоль используется цикл for().





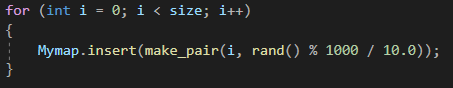
**4.3.** Для печати экземляра класса Vector в консоль используется перегрузка оператора<<



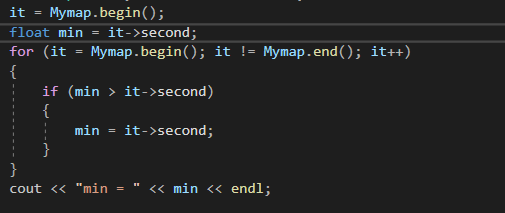


**5.** Поставленные задачи будут решены следующими действиями:

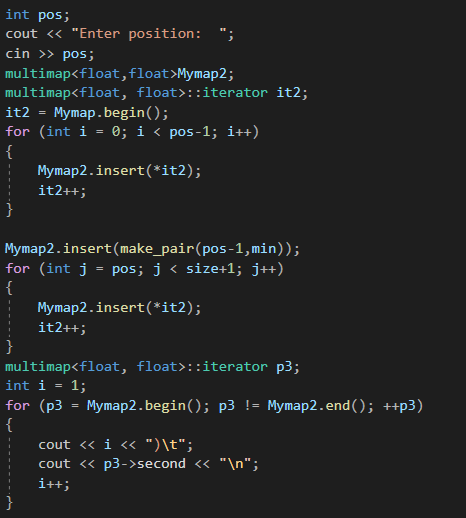
**5.1.** Для формирования ассоциативного контейнера в функции Task1()используется цикл, который итерируется по всему контейнеру и элементы добавляются через метод insert, в качестве параметра передаётся функция make\_pair, прописанная в библиотеке multimap.



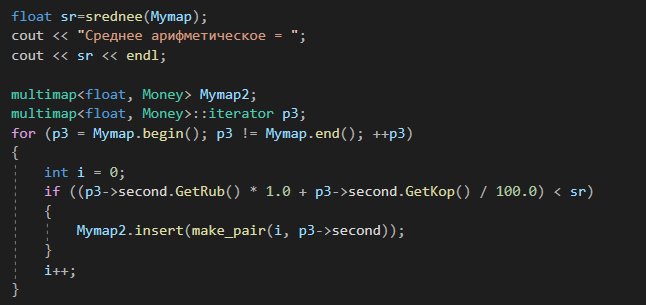
**5.2.** Для поиска минимального элемента контейнера используется итератор – it, который ставится на начало контейнера, в цикле for происходит проход по контейнеру, в котором происходит поиск путём сравнения переменной min и текущего элемента, на который указывает итератор it->second (в second хранятся данные: число).



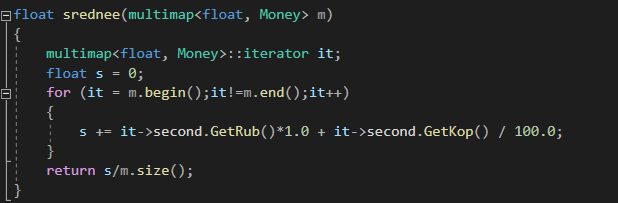
Далее найденный минимум необходимо вставить в контейнер. Для этого создается новый контейнер MyMap2, в который записываются элементы из старого и новый по заданной позиции pos при помощи 2 циклов for(). Далее создается новый итератор p3 и с его помощью выводится уже полученный контейнер.



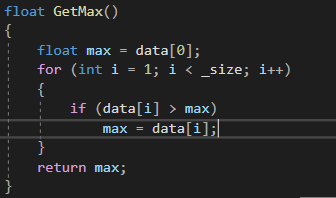
**5.3.** Для нахождения среднего арифметического в контейнере типа <float, Money> описана функция srednee. После его нахождения необходимо удалить из контейнера все элементы, большие или равные среднему арифметическому. Для этого создается новый контейнер, в который записываются элементы, удовлетворяющие условию.

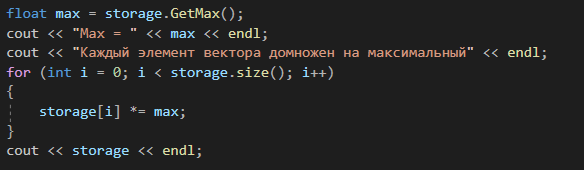


**5.4.** Для поиска среднего арифметического значения контейнера используется метод srednee(). Используется цикл for, в котором переменная s накапливает сумму всех элементов контейнера. После прохождения цикла возвращается значение s делённое на количество элементов в контейнере.

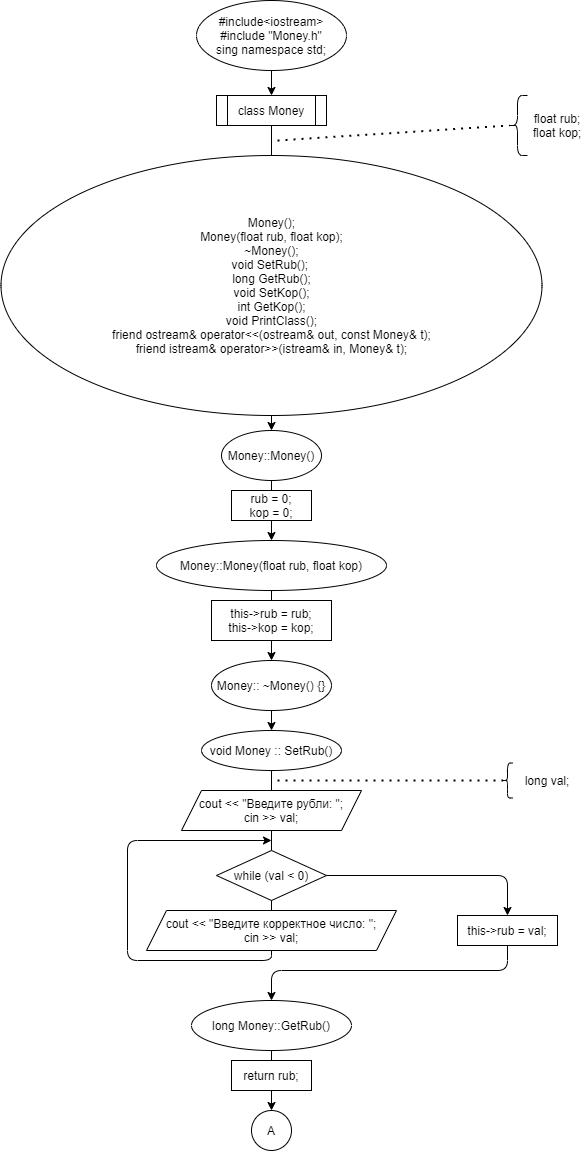


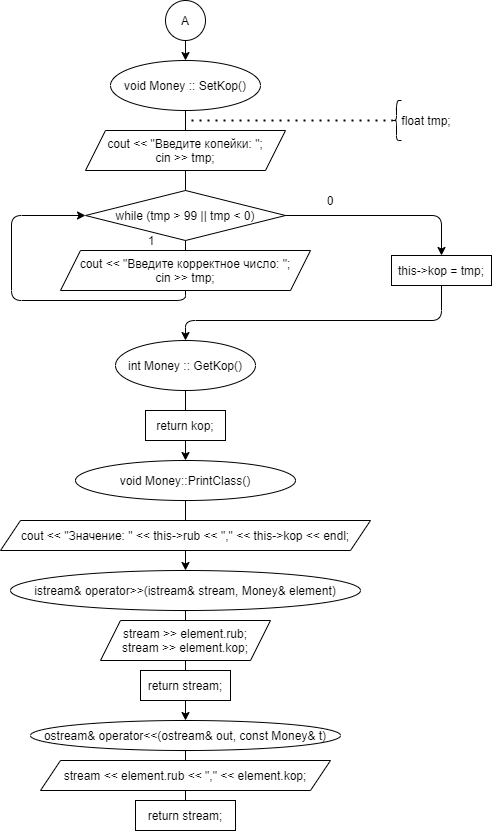
**5.5.** Для выполнения в классе Vector описана функция GetMax(), которая находит максимальный элемент вектора, далее все элементы умножаются на максимум в цикле for.

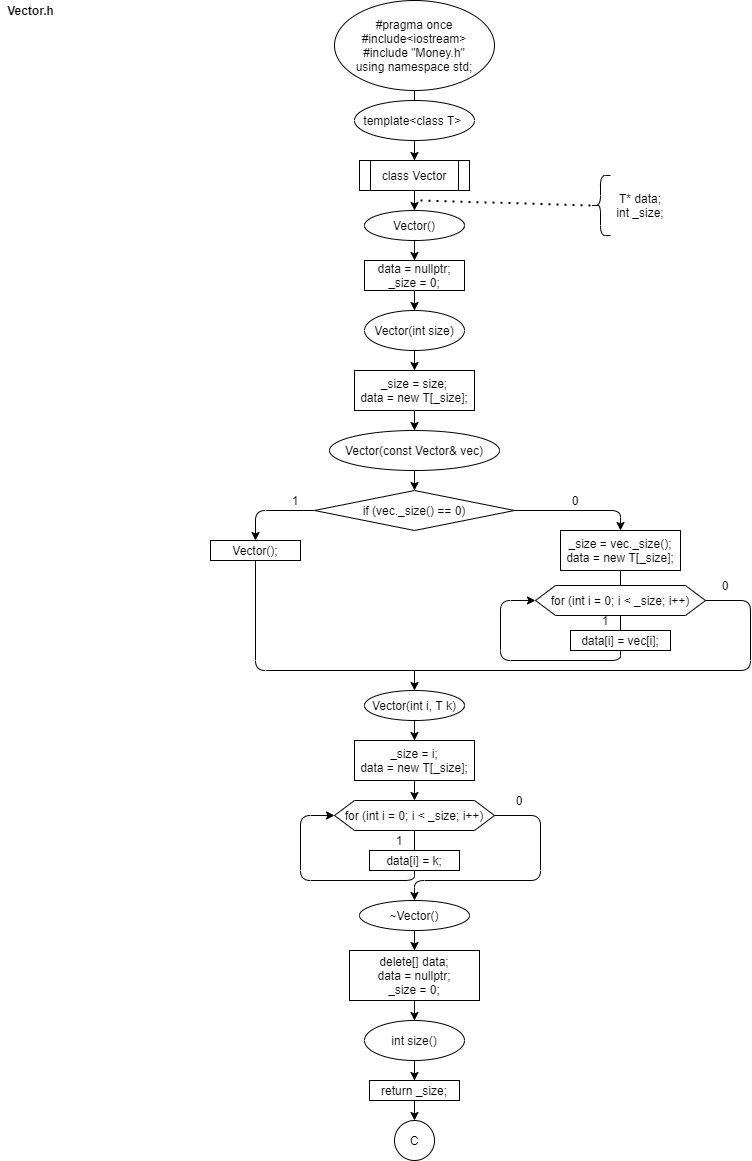


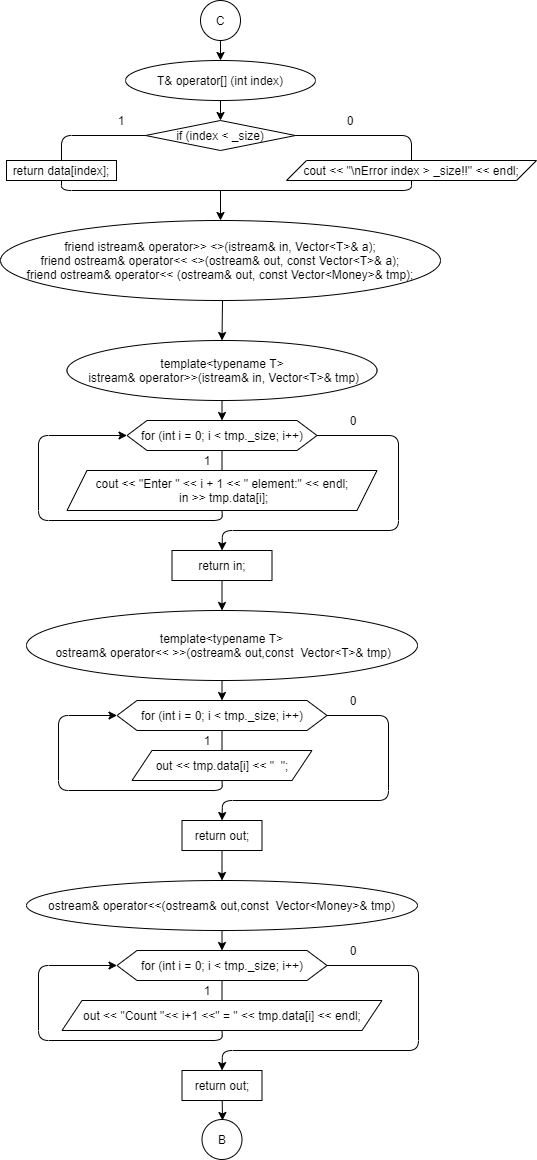


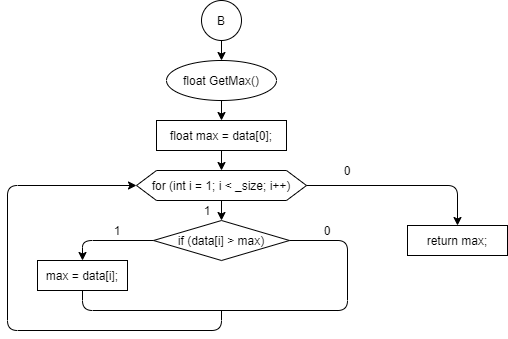
**Блок-схема**

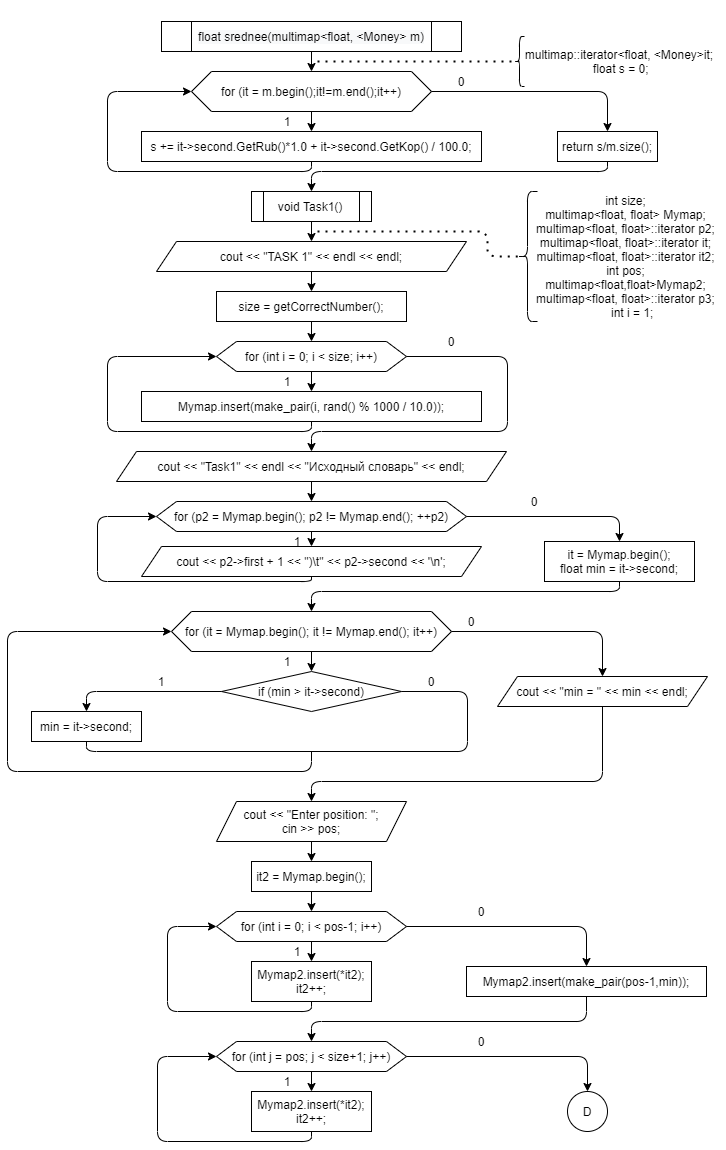
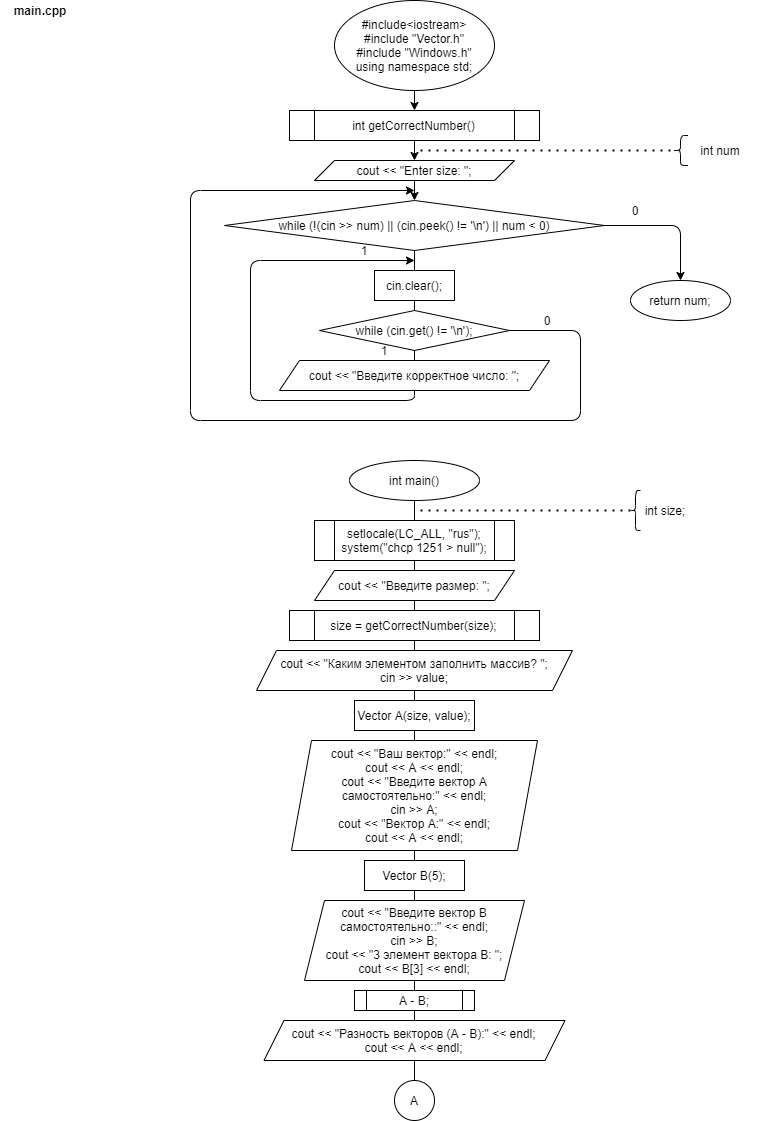
****

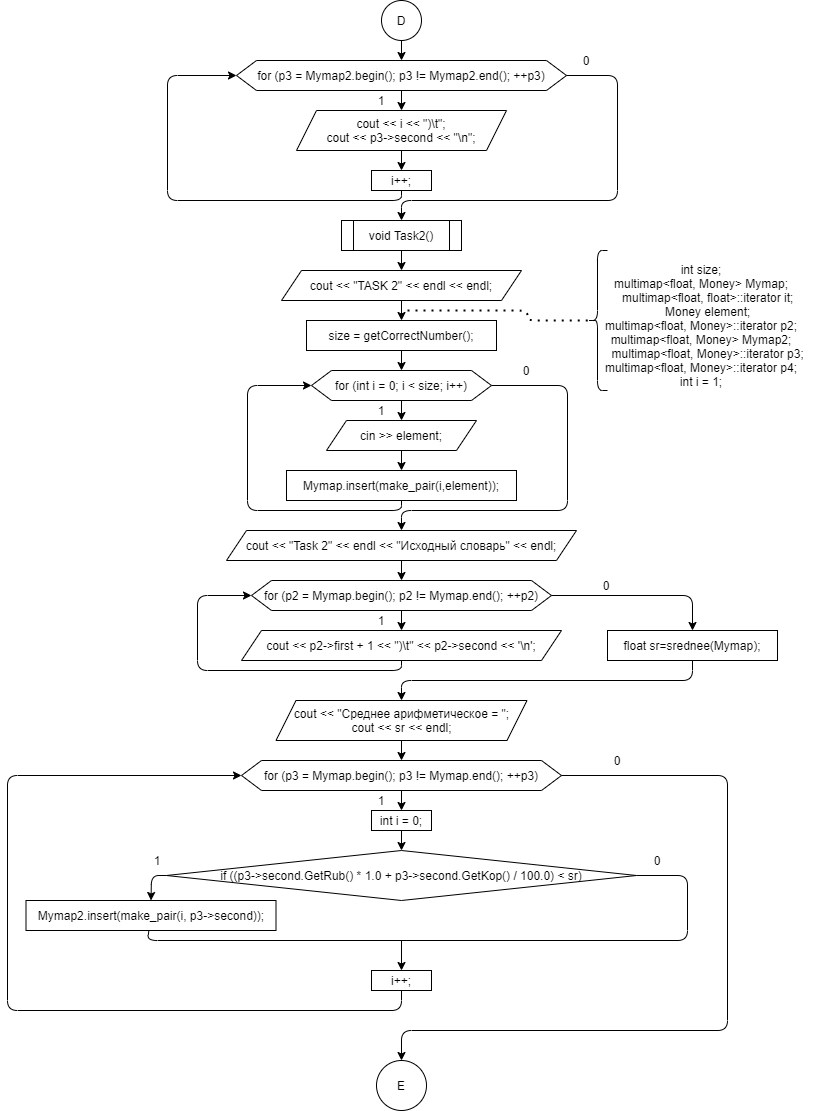
****

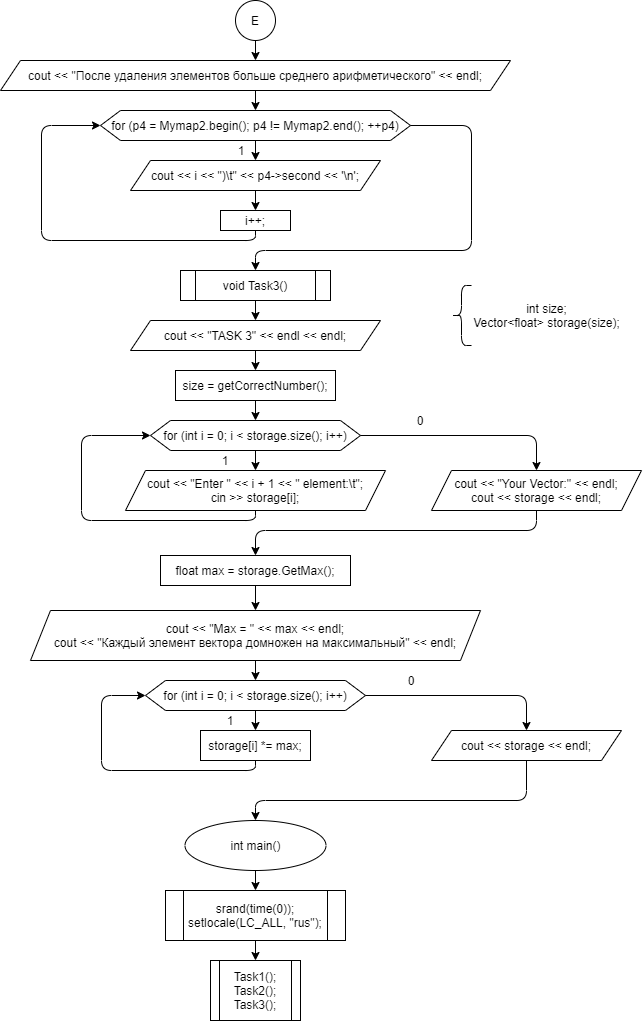
****

****

****

****

****

****

**Код**

#include <algorithm>

#include <iostream>

#include <iterator>

#include "Vector.h"

#include <map>

#include "Money.h"

#include <time.h>

using namespace std;

int getCorrectNumber()

{

int num;

cout << "Enter size: ";

while (!(cin >> num) || (cin.peek() != '\n') || num < 0)

{

cin.clear();

while (cin.get() != '\n');

cout << "Введите корректное число: ";

}

return num;

}

float srednee(multimap<float, Money> m)

{

multimap<float, Money>::iterator it;

float s = 0;

for (it = m.begin();it!=m.end();it++)

{

s += it->second.GetRub()\*1.0 + it->second.GetKop() / 100.0;

}

return s/m.size();

}

void Task1()

{

// creation

cout << "TASK 1" << endl << endl;

int size;

size = getCorrectNumber();

multimap<float, float> Mymap;

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

{

Mymap.insert(make\_pair(i, rand() % 1000 / 10.0));

}

cout << "Task1" << endl << "Исходный словарь" << endl;

multimap<float, float>::iterator p2;

for (p2 = Mymap.begin(); p2 != Mymap.end(); ++p2)

cout << p2->first + 1 << ")\t" << p2->second << '\n';

multimap<float, float>::iterator it;

it = Mymap.begin();

float min = it->second;

for (it = Mymap.begin(); it != Mymap.end(); it++)

{

if (min > it->second)

{

min = it->second;

}

}

cout << "min = " << min << endl;

int pos;

cout << "Enter position: ";

cin >> pos;

multimap<float,float>Mymap2;

multimap<float, float>::iterator it2;

it2 = Mymap.begin();

for (int i = 0; i < pos-1; i++)

{

Mymap2.insert(\*it2);

it2++;

}

Mymap2.insert(make\_pair(pos-1,min));

for (int j = pos; j < size+1; j++)

{

Mymap2.insert(\*it2);

it2++;

}

multimap<float, float>::iterator p3;

int i = 1;

for (p3 = Mymap2.begin(); p3 != Mymap2.end(); ++p3)

{

cout << i << ")\t";

cout << p3->second << "\n";

i++;

}

}

void Task2()

{

cout << "TASK 2" << endl << endl;

int size;

size = getCorrectNumber();

multimap<float, Money> Mymap;

multimap<float, float>::iterator it;

Money element;

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

{

cin >> element;

Mymap.insert(make\_pair(i,element));

}

cout << "Task 2" << endl << "Исходный словарь" << endl;

multimap<float, Money>::iterator p2;

for (p2 = Mymap.begin(); p2 != Mymap.end(); ++p2)

cout << p2->first + 1 << ")\t" << p2->second << '\n';

float sr=srednee(Mymap);

cout << "Среднее арифметическое = ";

cout << sr << endl;

multimap<float, Money> Mymap2;

multimap<float, Money>::iterator p3;

for (p3 = Mymap.begin(); p3 != Mymap.end(); ++p3)

{

int i = 0;

if ((p3->second.GetRub() \* 1.0 + p3->second.GetKop() / 100.0) < sr)

{

Mymap2.insert(make\_pair(i, p3->second));

}

i++;

}

multimap<float, Money>::iterator p4;

cout << "После удаления элементов больше среднего арифметического" << endl;

int i = 1;

for (p4 = Mymap2.begin(); p4 != Mymap2.end(); ++p4)

{

cout << i << ")\t" << p4->second << '\n';

i++;

}

}

void Task3()

{

cout << "TASK 3" << endl << endl;

int size;

size = getCorrectNumber();

Vector<float> storage(size);

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

{

cout << "Enter " << i + 1 << " element:\t";

cin >> storage[i];

}

cout << "Your Vector:" << endl;

cout << storage << endl;

float max = storage.GetMax();

cout << "Max = " << max << endl;

cout << "Каждый элемент вектора домножен на максимальный" << endl;

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

{

storage[i] \*= max;

}

cout << storage << endl;

}

int main()

{

srand(time(0));

setlocale(LC\_ALL, "rus");

Task1();

Task2();

Task3();

}

#pragma once

#include<iostream>

#include<fstream>

#include<string>

using namespace std;

class Money

{

float rub;

float kop;

public:

Money();

Money(float rub, float kop);

~Money();

void SetRub();

long GetRub();

void SetKop();

int GetKop();

void PrintClass();

friend ostream& operator<<(ostream& out, const Money& m);

friend istream& operator>>(istream& in, Money& m);

};

#include<iostream>

using namespace std;

Money::Money()

{

rub = 0;

kop = 0;

}

Money::Money(float rub, float kop)

{

this->rub = rub;

this->kop = kop;

}

Money:: ~Money() {}

void Money::SetRub()

{

float val;

cout << "Enter rub: ";

cin >> val;

while (val < 0)

{

cout << "Enter correct number: ";

cin >> val;

}

this->rub = val;

}

long Money::GetRub() { return rub; }

void Money::SetKop()

{

float tmp;

cout << "Enter kop: ";

cin >> tmp;

while (tmp > 99 || tmp < 0)

{

cout << "Enter correct number: ";

cin >> tmp;

}

this->kop = tmp;

}

int Money::GetKop() { return kop; }

void Money::PrintClass()

{

cout << "Значение: " << this->rub << "," << this->kop << endl;

}

ostream& operator<<(ostream& out, const Money& m)

{

out << m.rub;

out << ",";

out << m.kop;

return out;

}

istream& operator>>(istream& in, Money& m)

{

cout << "Enter rub: ";

in >> m.rub;

cout << "Enter kop: ";

in >> m.kop;

return in;

}

#include <iostream>

#include"Money.h"

using namespace std;

template <class T>

class Vector

{

T\* data;

int \_size;

public:

//construct and destruct

Vector()

{

data = nullptr;

\_size = 0;

}

Vector(T size)

{

\_size = size;

data = new T[\_size];

}

Vector(const Vector<T>& vec)

{

if (vec.\_size() == 0)

Vector();

else

{

\_size = vec.\_size();

data = new T[\_size];

for (int i = 0; i < \_size; i++)

data[i] = vec[i];

}

}

Vector(int i, T k)

{

\_size = i;

data = new T[\_size];

for (int i = 0; i < \_size; i++)

data[i] = k;

}

~Vector()

{

delete[] data;

data = nullptr;

\_size = 0;

}

//METHODS

int size() { return \_size; }

float GetMax()

{

float max = data[0];

for (int i = 1; i < \_size; i++)

{

if (data[i] > max)

max = data[i];

}

return max;

}

T& operator[] (int index)

{

if (index < \_size) return data[index];

else cout << "\nError index > \_size!!" << endl;

}

public:

//ios overloads

friend istream& operator>> <>(istream& in, Vector<T>& a);

friend ostream& operator<< <>(ostream& out, const Vector<T>& a);

friend ostream& operator<< (ostream& out, const Vector<Money>& tmp);

};

template <typename T>

istream& operator>>(istream& in, Vector<T>& tmp)

{

for (int i = 0; i < tmp.\_size; i++)

{

cout << "Enter " << i + 1 << " element of Vector: ";

in >> tmp.data[i];

}

return in;

}

template <typename T>

ostream& operator<< <>(ostream& out, const Vector<T>& tmp)

{

for (int i = 0; i < tmp.\_size; i++)

out << i+1 << ")\t" << tmp.data[i] << endl;

return out;

}

ostream& operator<< (ostream& out, const Vector<Money>& tmp)

{

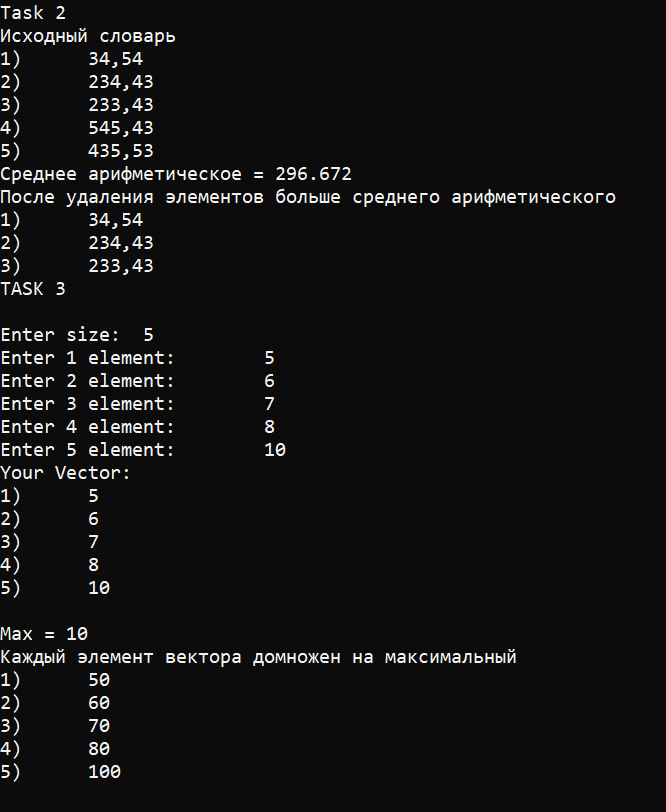
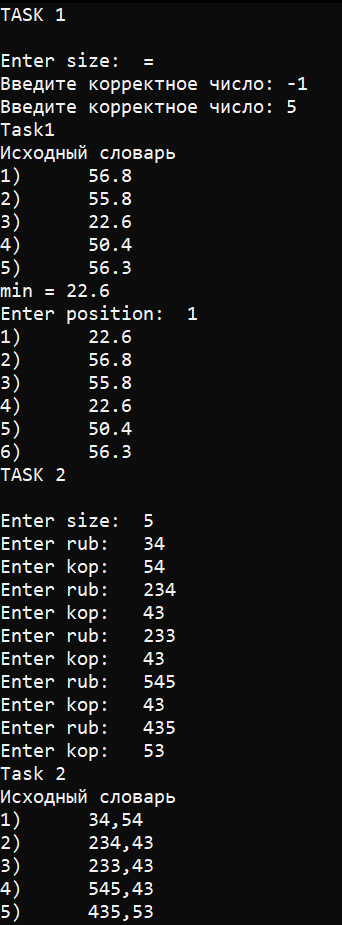
for (int i = 0; i < tmp.\_size; i++)

out << i + 1 << ")\t" << tmp.data[i] << endl;

return out;

}

**Скриншоты**



**Контрольные вопросы**



1. 

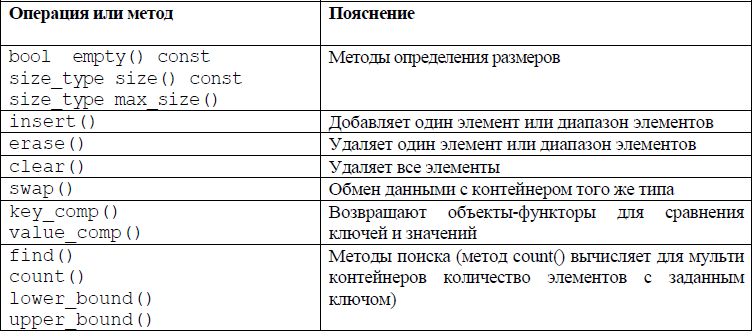


2. multimap, multimultimap, multiset, set.

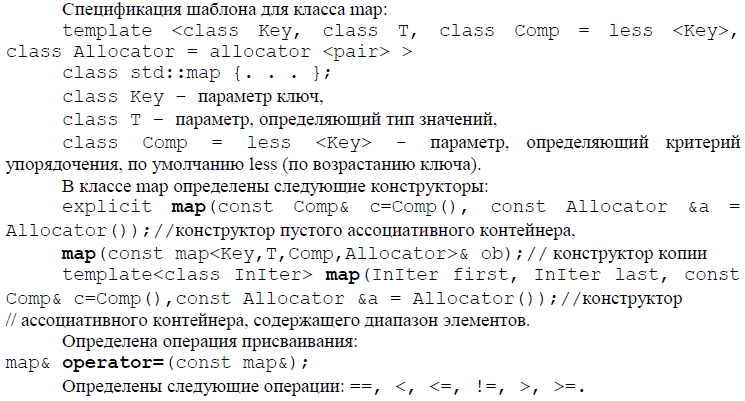


3. Использовать итератор для получения доступа к элементам ассоциативного контейнера.

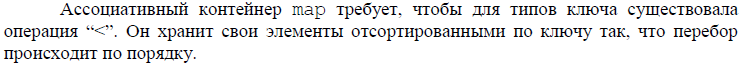


4. 



5. 



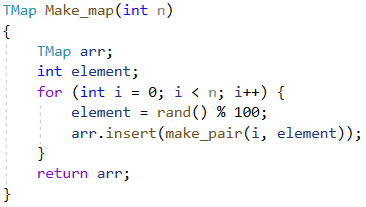
6. 



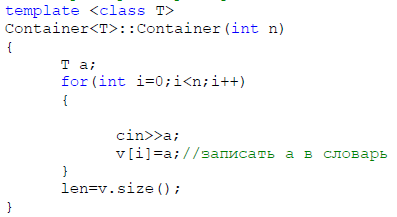
7. 



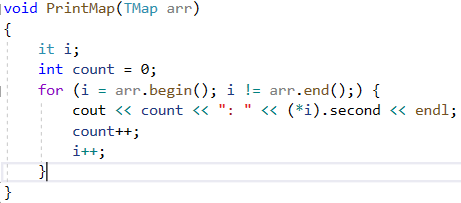


8. 

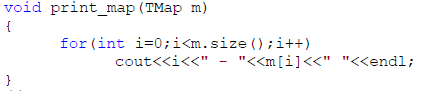


9. 

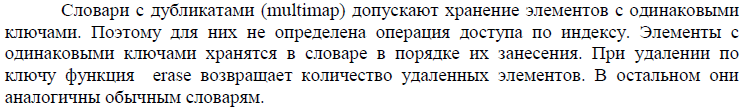


10. 



11. 



12. 

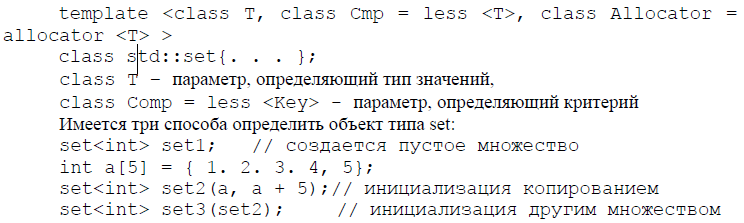


13. 

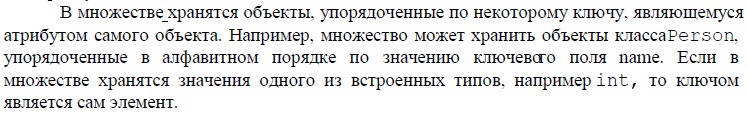


14. В множестве set значения не играют роли, так что рассматриваются только ключи.



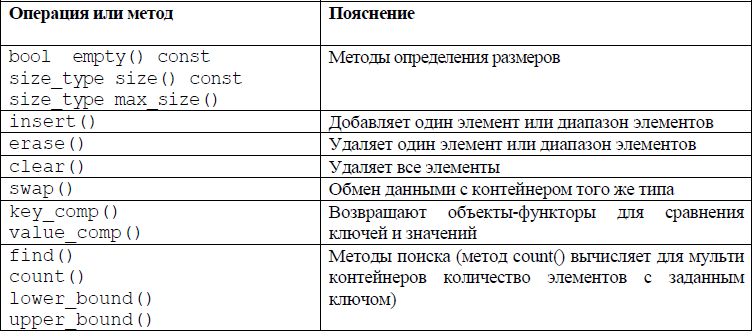
15. 



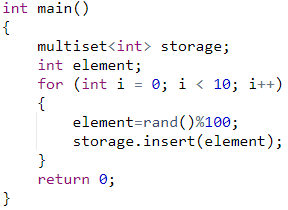
16. 



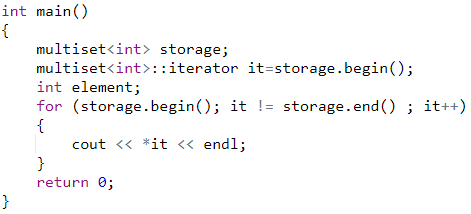
17. К множествам применимы общие для всех контейнеров методы.





18. 



19. 



20. 