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 4

Выполнил:

Студент группы ИВТ-20-2Б Тедеев Алесандр Зурабович

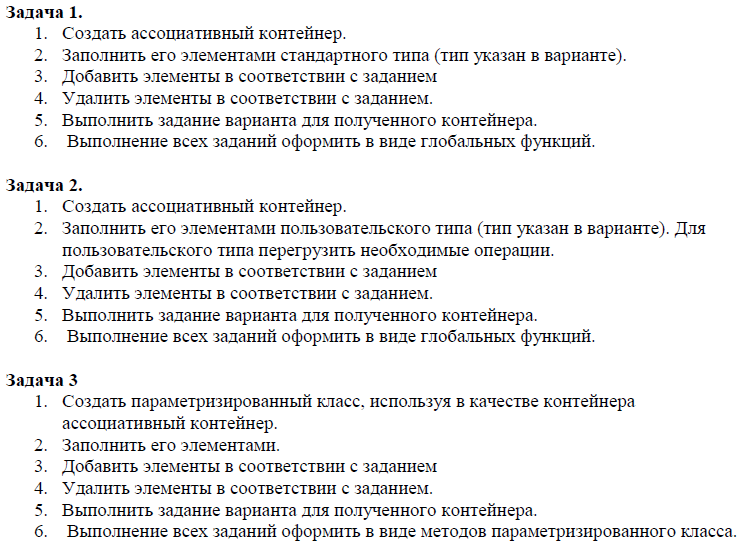
Проверила:

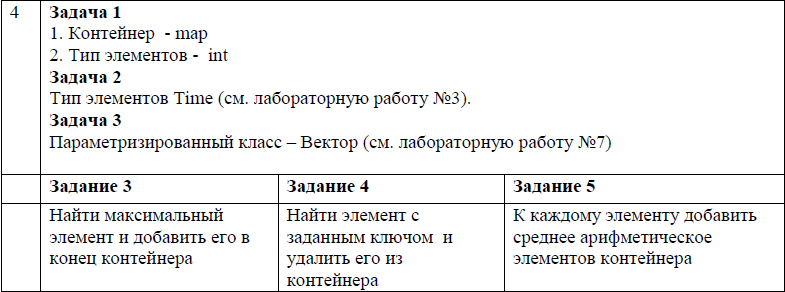
Доцент кафедры ИТАС

Полякова О.А.

Пермь, 2021

**Постановка задачи**





**Анализ задачи**

**1.** Для решения задач необходимо:

**1.1.** Организовать функцию Problem1(), в которой будет использоваться библиотека map для создания ассоциативного контейнера, для решения задачи №1.

**1.2.** Организовать функцию Problem2(), в которой будет использоваться библиотека map для создания ассоциативного контейнера типа Time, для решения задачи №2.

**1.3.** Организовать класс Time для хранения в полях для минут и секунд (int: mins, secs) для экземпляров данного класса.

**1.4.** Организовать функцию Problem3() 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 map.

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

**2.1.** В функции main() используется пременная vvod типа int для меню, реализованного через ветвление switch().

char vvod = '6';

while (vvod != '0') {

cout << "\nКакое задание?" << endl

<< "1) Задание 1;" << endl

<< "2) Задание 2;" << endl

<< "3) Задание 3;" << endl

<< "0) Выход из меню" << "\n\n" << ">";

cin >> vvod;

switch (vvod)

{

case '1':

{

Problem1();

}

break;

case '2':

{

Problem2();

}

break;

case '3':

{

Problem3();

}

break;

default:

if (vvod != '0')

cout << "ERROR!";

break;

}

}

**2.2.** Функция Problem1(), в которой будет использоваться библиотека map для создания ассоциативного контейнера – TMap, типа int для ключа и типа int для хранящихся данных, для решения задачи №1. Переменная size устанавливает размер контейнера.

typedef map<int, int>TMap;

int size = SafeInput1(true);

**2.3.** Функция Problem2(), в которой будет использоваться библиотека map для создания ассоциативного контейнера – TMap2, типа int для ключа и типа Time для хранящихся данных, для решения задачи №2. Переменная size устанавливает размер контейнера.

typedef map<int, Time>TMap2;

int size = SafeInput1(true);

**2.4.** Функция Problem3(), в которой будет использоваться, в которой будет использоваться класс Vector для создания экземпляра класса Vector, для решения задачи №3. Переменная size устанавливает размер контейнера.

int size = SafeInput1(true);

Vector<int> array(size);

template<class Type>

class Vector

{

protected:

map<int, Type> arr;//Type\* arr;

int size;

public:

Vector();

Vector(int length); //конструктор с параметрами

Vector(const Vector& object); //конструктор копирования

~Vector(); //очистка памяти

Vector& operator=(const Vector<Type>& other);

Type& operator[] (int index); //получение значения по индексу

void operator+(const Type k); //сложение элемента массива с константой

int operator()(); //получение размера

void PrintMap();

int FindAverageValue();

void insert(int pos, Type element);

friend ostream& operator<< (ostream& out, const Vector<Type>& p)

{

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

out << p.arr[i] << " ";

}

cout << endl;

return out;

}

friend istream& operator>> (istream& in, Vector<Type>& p)

{

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

in >> p.arr[i];

}

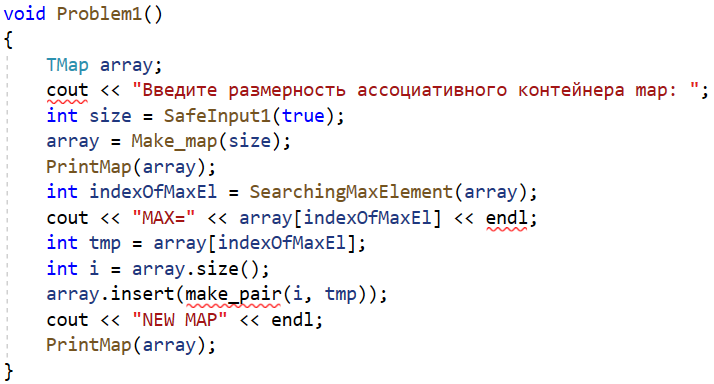
return in;

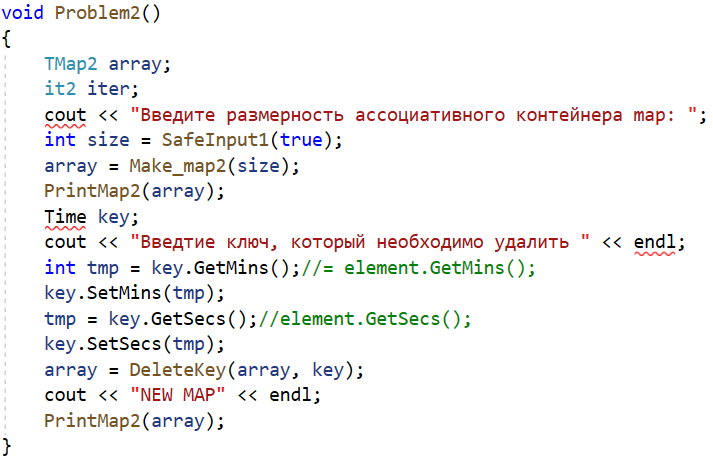
}

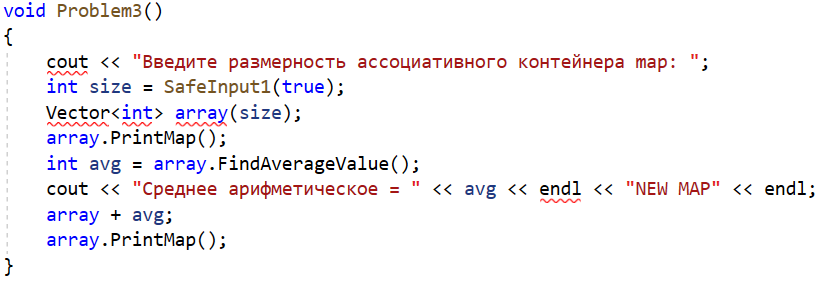
};

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

**3.1.** Решения задач осуществляются в функциях Problem1(), Problem2(), Problem3().

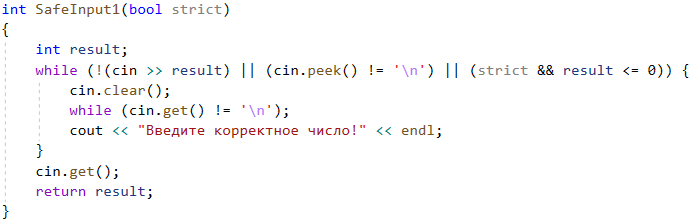




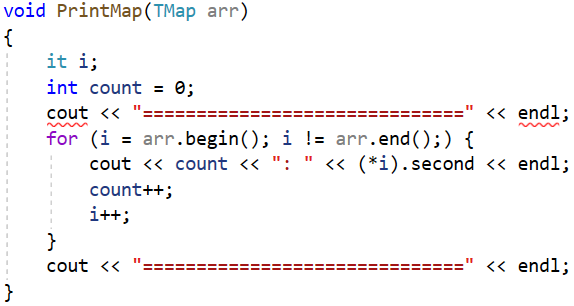


**4.** Для операций ввода и вывода использовались следующие операторы и функции:

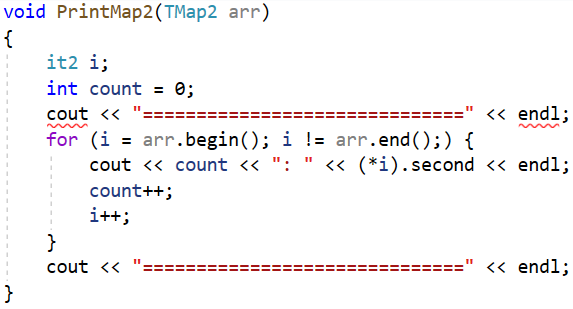
**4.1.** Для ввода целочисленных значений используется функция SafeInput1(), которая проверяет введённое данное. Таким образом будут игнорироваться данные типа char, double, float.



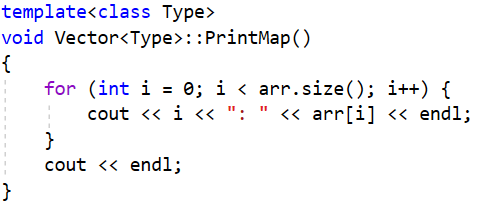
**4.2.** Для печати ассоциативного контейнера – TMap, в консоль используется функция PrintMap().



**4.3.** Для печати ассоциативного контейнера – TMap2, в консоль используется функция PrintMap2().

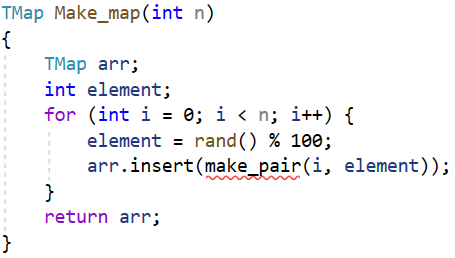


**4.4.** Для печати экземляра класса Vector в консоль используется метод данного класса - PrintMap().

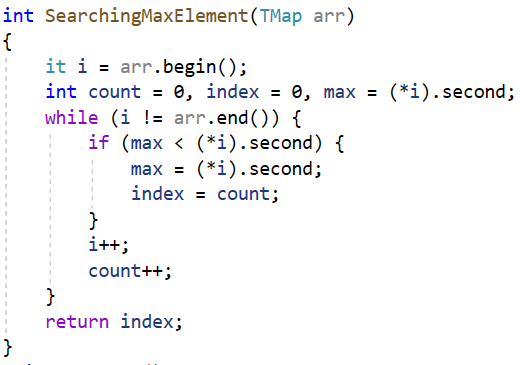


**5.** Поставленные задачи будут решены следующими действиями:

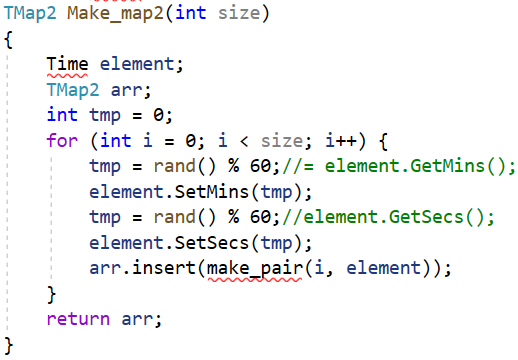
**5.1.** Для формирования ассоциативного контейнера в функции Problem1() используется функция Make\_map().В цикле for используется цикл, который итерируется по всему контейнеру и элементы добавляются через метод insert, в качестве параметра передаётся функция make\_pair, прописанная в библиотеке map.



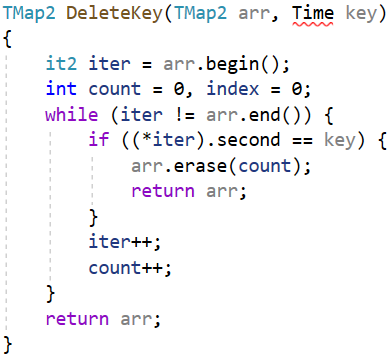
**5.2.** Для поиска максимального элемента контейнера используется итератор – i, который ставится на начало контейнера, в цикле while происходит проход по контейнеру, в котором происходит поиск путём сравнения переменной max и текущего элемента, на который указывает итератор (\*i).second (в second хранятся данные: число), также в переменную index записывается номер данного элемента в контейнере.



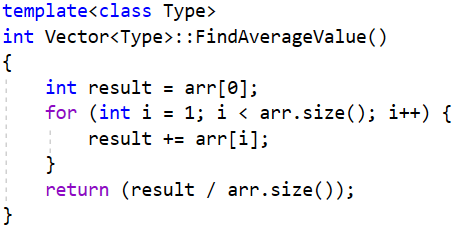
**5.3.** Для формирования ассоциативного контейнера в функции Problem2() используется функция Make\_map2(). Используется for цикл, который итерируется по всему контейнеру. Данные для объекта класса Time генерируются в переменной tmp через функцию rand(), а затем элементы добавляются через метод insert, в качестве параметра передаётся функция make\_pair, прописанная в библиотеке map.



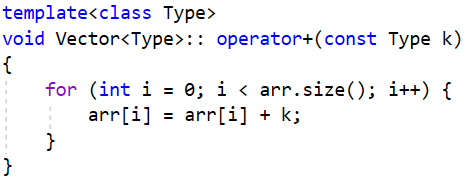
**5.4.** Для удаления ключа, введённого пользователем, из ассоциативного контейнера в функции Problem2() используется функция DeleteKey(). Используется цикл while, который итерируется по всему контейнеру с помощью итератора – iter. Если текущий элемент, на который указывает итератор равен введённому ключу, то удаление происходит через метод erase из библиотеки map и возвращение контейнера.



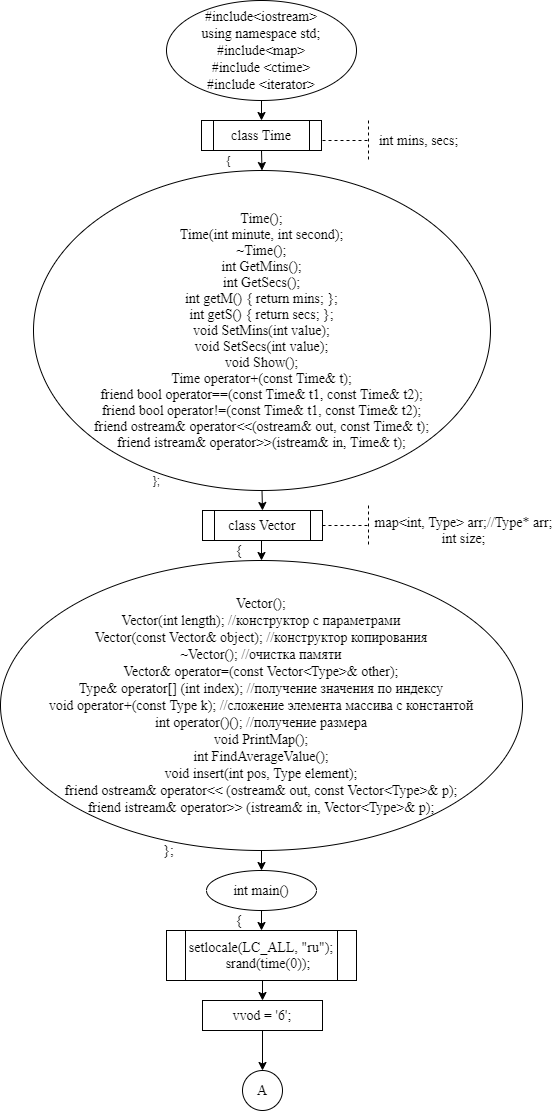
**5.5.** Для поиска среднего арифметического значения контейнера используется метод FindAverageValue(). Используется цикл for, в котором переменная result накапливает сумму всех элементов контейнера. После прохождения цикла возвращается значение result делённое на количество элементов в контейнере.

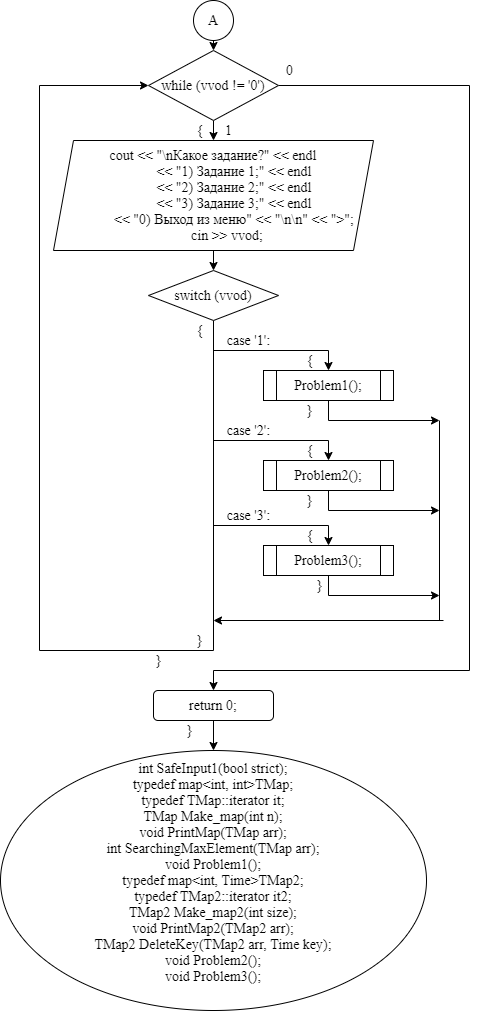


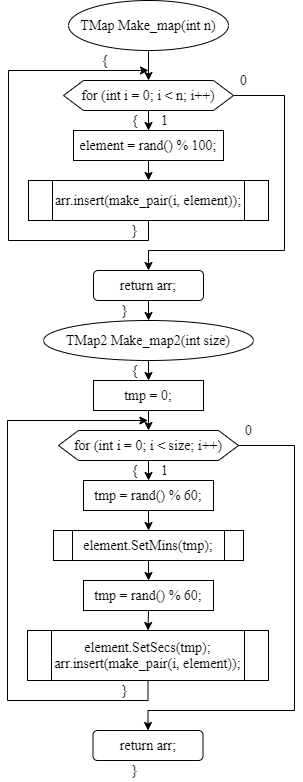
**5.6.** Для добавления среднего арифметического ко всем элементам контейнера используется перегрузка оператора +.

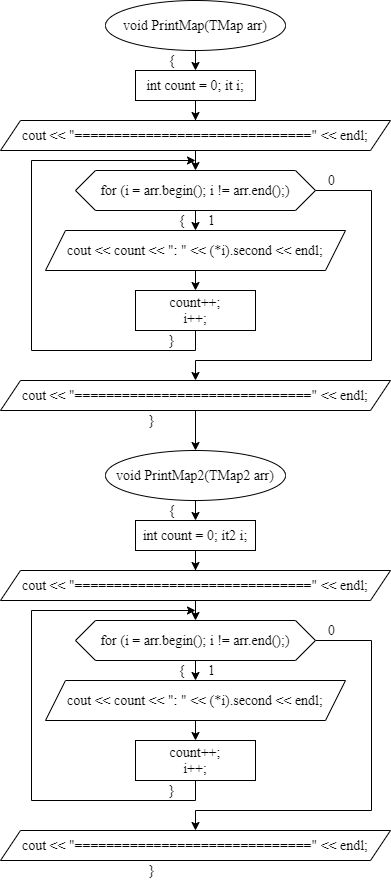


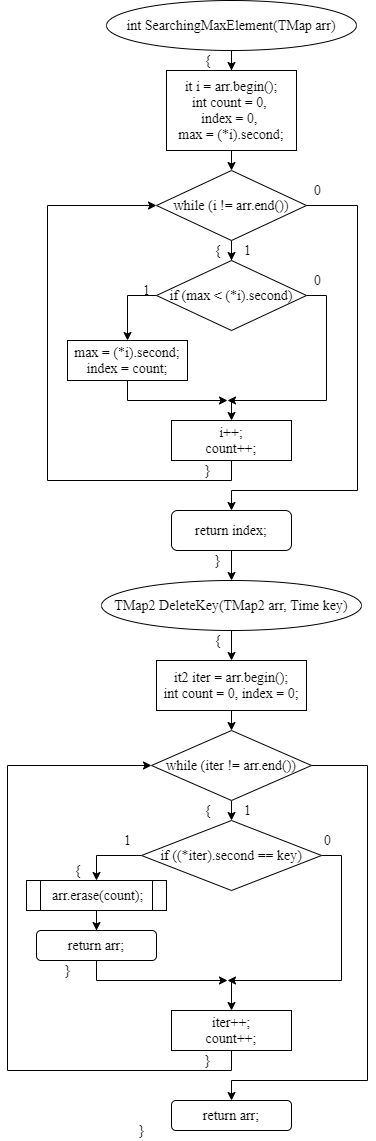
**Блок-схема**

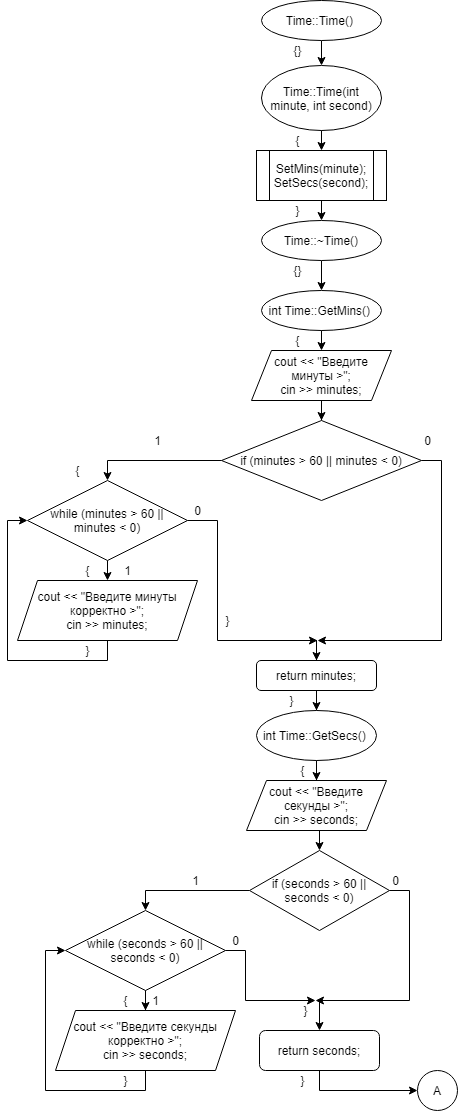
****

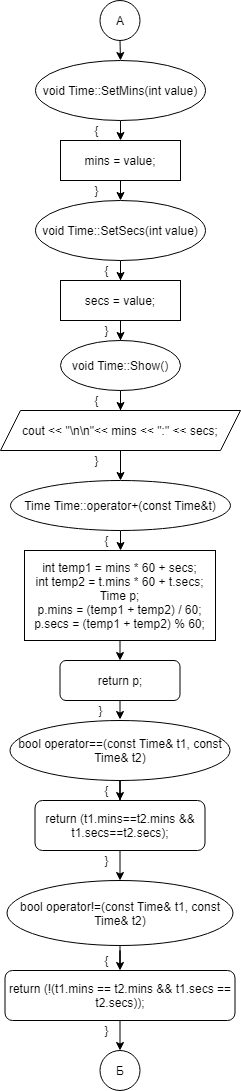
****

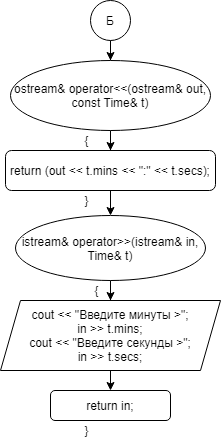
****

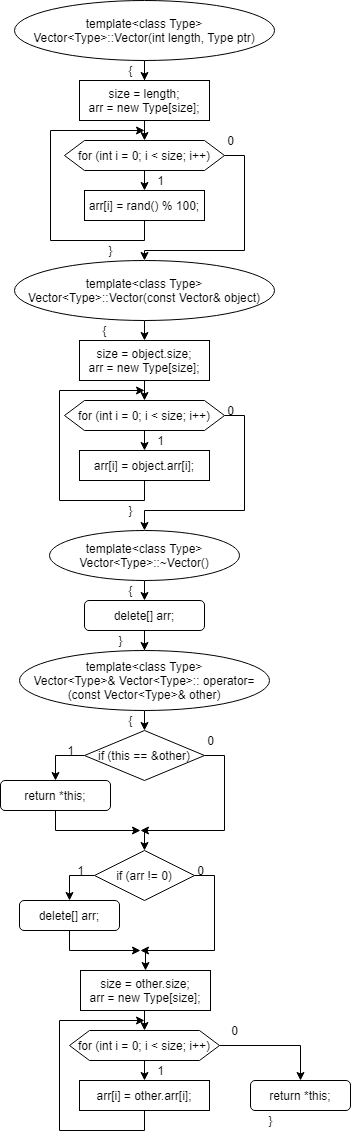
****

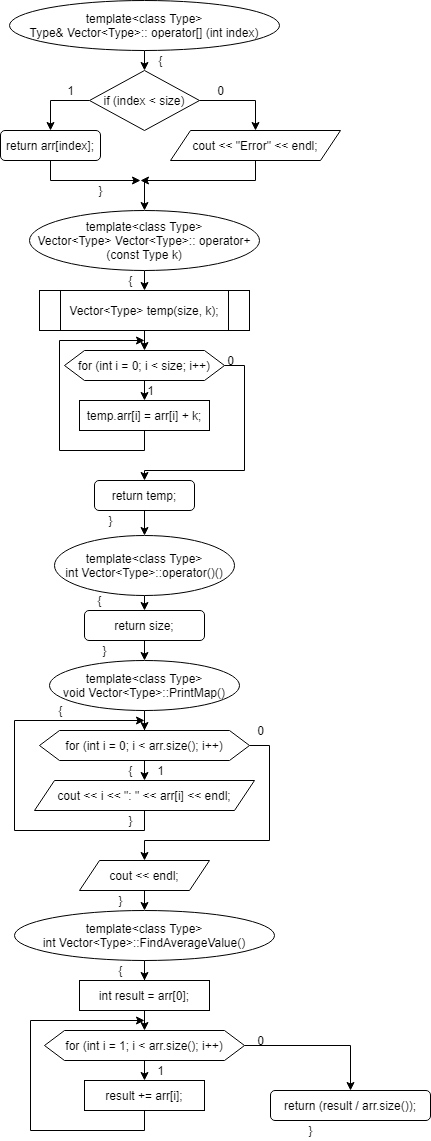
****

****

****

****



****

**Код**

#include"Time.h"

#include"Vector.h"

#include <ctime>

#include <iterator>

int SafeInput1(bool strict);

typedef map<int, int>TMap;

typedef TMap::iterator it;

TMap Make\_map(int n);

void PrintMap(TMap arr);

int SearchingMaxElement(TMap arr)

{

it i = arr.begin();

int count = 0, index = 0, max = (\*i).second;

while (i != arr.end()) {

if (max < (\*i).second) {

max = (\*i).second;

index = count;

}

i++;

count++;

}

return index;

}

void Problem1()

{

TMap array;

cout << "Введите размерность ассоциативного контейнера map: ";

int size = SafeInput1(true);

array = Make\_map(size);

PrintMap(array);

int indexOfMaxEl = SearchingMaxElement(array);

cout << "MAX=" << array[indexOfMaxEl] << endl;

int tmp = array[indexOfMaxEl];

int i = array.size();

array.insert(make\_pair(i, tmp));

cout << "NEW MAP" << endl;

PrintMap(array);

}

typedef map<int, Time>TMap2;

typedef TMap2::iterator it2;

TMap2 Make\_map2(int size)

{

Time element;

TMap2 arr;

int tmp = 0;

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

tmp = rand() % 60;//= element.GetMins();

element.SetMins(tmp);

tmp = rand() % 60;//element.GetSecs();

element.SetSecs(tmp);

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

}

return arr;

}

void PrintMap2(TMap2 arr);

TMap2 DeleteKey(TMap2 arr, Time key)

{

it2 iter = arr.begin();

int count = 0, index = 0;

while (iter != arr.end()) {

if ((\*iter).second == key) {

arr.erase(count);

return arr;

}

iter++;

count++;

}

return arr;

}

void Problem2()

{

TMap2 array;

it2 iter;

cout << "Введите размерность ассоциативного контейнера map: ";

int size = SafeInput1(true);

array = Make\_map2(size);

PrintMap2(array);

Time key;

cout << "Введите ключ, который необходимо удалить " << endl;

int tmp = key.GetMins();//= element.GetMins();

key.SetMins(tmp);

tmp = key.GetSecs();//element.GetSecs();

key.SetSecs(tmp);

array = DeleteKey(array, key);

cout << "NEW MAP" << endl;

PrintMap2(array);

}

void Problem3()

{

cout << "Введите размерность ассоциативного контейнера map: ";

int size = SafeInput1(true);

Vector<int> array(size);

array.PrintMap();

int avg = array.FindAverageValue();

cout << "Среднее арифметическое = " << avg << endl << "NEW MAP" << endl;

array + avg;

array.PrintMap();

}

int main()

{

setlocale(LC\_ALL, "ru");

srand(time(0));

char vvod = '6';

while (vvod != '0') {

cout << "\nКакое задание?" << endl

<< "1) Задание 1;" << endl

<< "2) Задание 2;" << endl

<< "3) Задание 3;" << endl

<< "0) Выход из меню" << "\n\n" << ">";

cin >> vvod;

switch (vvod)

{

case '1':

{

Problem1();

}

break;

case '2':

{

Problem2();

}

break;

case '3':

{

Problem3();

}

break;

default:

if (vvod != '0')

cout << "ERROR!";

break;

}

}

return 0;

}

int SafeInput1(bool strict)

{

int result;

while (!(cin >> result) || (cin.peek() != '\n') || (strict && result <= 0)) {

cin.clear();

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

cout << "Введите корректное число!" << endl;

}

cin.get();

return result;

}

TMap Make\_map(int n)

{

TMap arr;

int element;

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

element = rand() % 100;

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

}

return arr;

}

void PrintMap(TMap arr)

{

it i;

int count = 0;

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

for (i = arr.begin(); i != arr.end();) {

cout << count << ": " << (\*i).second << endl;

count++;

i++;

}

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

}

void PrintMap2(TMap2 arr)

{

it2 i;

int count = 0;

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

for (i = arr.begin(); i != arr.end();) {

cout << count << ": " << (\*i).second << endl;

count++;

i++;

}

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

}

#pragma once

#include<iostream>

using namespace std;

class Time {

protected:

int mins, secs;

public:

Time();

Time(int minute, int second);

~Time();

int GetMins();

int GetSecs();

int getM() { return mins; };

int getS() { return secs; };

void SetMins(int value);

void SetSecs(int value);

void Show();

Time operator+(const Time& t);

friend bool operator==(const Time& t1, const Time& t2);

friend bool operator!=(const Time& t1, const Time& t2);

friend ostream& operator<<(ostream& out, const Time& t);

friend istream& operator>>(istream& in, Time& t);

};

#include "Time.h"

using namespace std;

int SafeInput(bool strict)

{

int result;

while (!(cin >> result) || (cin.peek() != '\n') || (strict && result <= 0) || (result > 60)) {

cin.clear();

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

cout << "Введите корректное число!" << endl;

}

cin.get();

return result;

}

Time::Time() {

mins = 0; secs = 0;

}

Time::Time(int minute, int second)

{

SetMins(minute);

SetSecs(second);

}

Time::~Time() {

}

int Time::GetMins()

{

int minutes;

cout << "Введите минуты >";

minutes = SafeInput(true);

return minutes;

}

int Time::GetSecs()

{

int seconds;

cout << "Введите секунды >";

seconds = SafeInput(true);

return seconds;

}

void Time::SetMins(int value)

{

mins = value;

}

void Time::SetSecs(int value)

{

secs = value;

}

void Time::Show()

{

cout << "\n\n" << mins << ":" << secs;

}

Time Time::operator+(const Time& t)

{

int temp1 = mins \* 60 + secs;

int temp2 = t.mins \* 60 + t.secs;

Time p;

p.mins = (temp1 + temp2) / 60;

p.secs = (temp1 + temp2) % 60;

return p;

}

bool operator==(const Time& t1, const Time& t2)

{

return (t1.mins == t2.mins && t1.secs == t2.secs);

}

bool operator!=(const Time& t1, const Time& t2)

{

return (!(t1.mins == t2.mins && t1.secs == t2.secs));

}

ostream& operator<<(ostream& out, const Time& t)

{

return (out << t.mins << ":" << t.secs);

}

istream& operator>>(istream& in, Time& t)

{

cout << "Введите минуты >";

in >> t.mins;

cout << "Введите секунды >";

in >> t.secs;

return in;

}

#pragma once

#include<iostream>

#include<map>

using namespace std;

template<class Type>

class Vector

{

protected:

map<int, Type> arr;//Type\* arr;

int size;

public:

Vector();

Vector(int length); //конструктор с параметрами

Vector(const Vector& object); //конструктор копирования

~Vector(); //очистка памяти

Vector& operator=(const Vector<Type>& other);

Type& operator[] (int index); //получение значения по индексу

void operator+(const Type k); //сложение элемента массива с константой

int operator()(); //получение размера

void PrintMap();

int FindAverageValue();

void insert(int pos, Type element);

friend ostream& operator<< (ostream& out, const Vector<Type>& p)

{

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

out << p.arr[i] << " ";

}

cout << endl;

return out;

}

friend istream& operator>> (istream& in, Vector<Type>& p)

{

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

in >> p.arr[i];

}

return in;

}

};

#include "Vector.cpp"

#ifndef \_VECTOR\_CPP\_

#define \_VECTOR\_CPP\_

#include "Vector.h"

#include <assert.h>

using namespace std;

template<class Type>

Vector<Type>::Vector()

{

size = 0;

}

template<class Type>

Vector<Type>::Vector(int length)

{

size = length;

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

arr[i] = rand() % 100;

}

}

template<class Type>

Vector<Type>::Vector(const Vector& object)

{

size = object.size;

arr = new Type[size];

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

arr[i] = object.arr[i];

}

}

template<class Type>

Vector<Type>::~Vector()

{

//delete[] arr;

}

template<class Type>

Vector<Type>& Vector<Type>:: operator=(const Vector<Type>& other)

{

if (this == &other)

return \*this;

if (arr != 0)

delete[] arr;

size = other.size;

arr = new Type[size];

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

arr[i] = other.arr[i];

}

return \*this;

}

template<class Type>

Type& Vector<Type>:: operator[] (int index)

{

if (index < size)

return arr[index];

else

cout << "Error" << endl;

}

template<class Type>

void Vector<Type>:: operator+(const Type k)

{

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

arr[i] = arr[i] + k;

}

}

template<class Type>

int Vector<Type>::operator()()

{

return size;

}

template<class Type>

void Vector<Type>::PrintMap()

{

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

cout << i << ": " << arr[i] << endl;

}

cout << endl;

}

template<class Type>

int Vector<Type>::FindAverageValue()

{

int result = arr[0];

for (int i = 1; i < arr.size(); i++) {

result += arr[i];

}

return (result / arr.size());

}

template<class Type>

void Vector<Type>::insert(int pos, Type element)

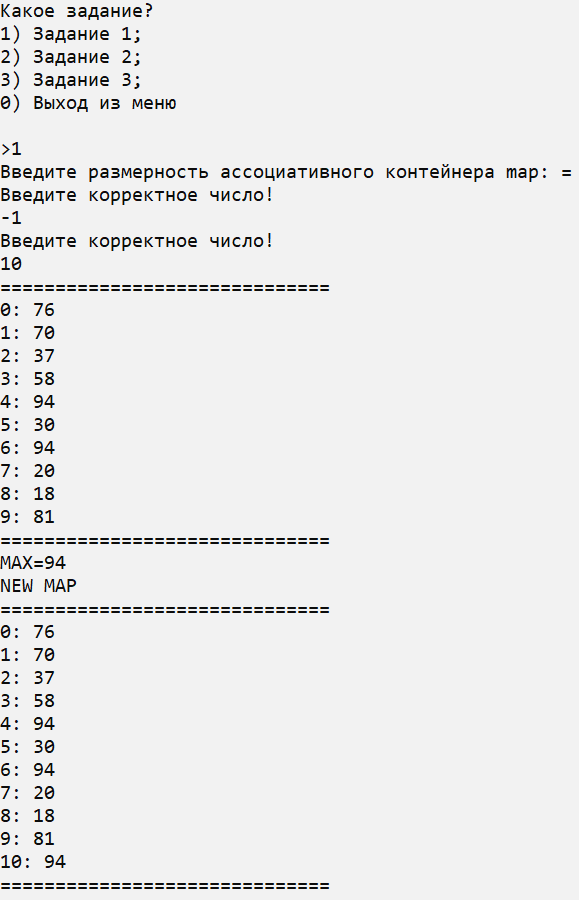
{

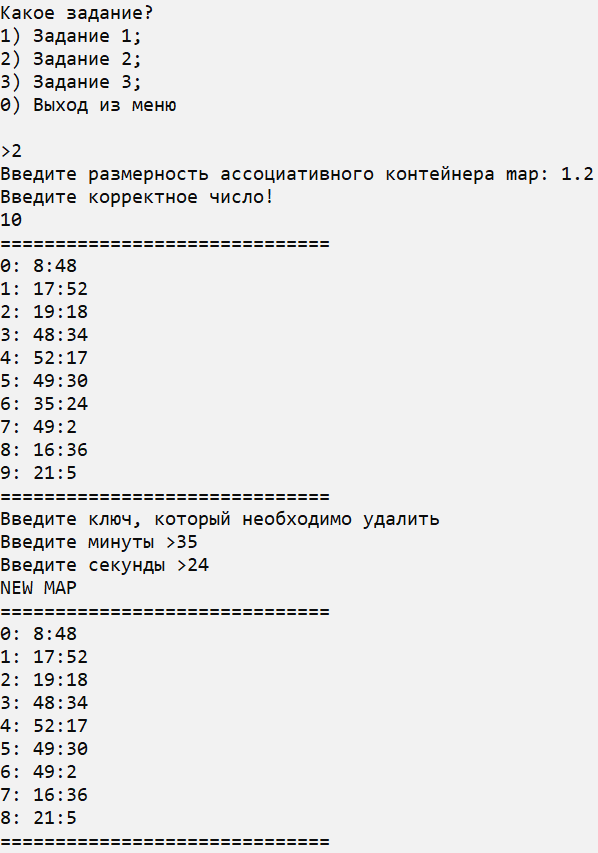
arr.insert(make\_pair(pos, element));

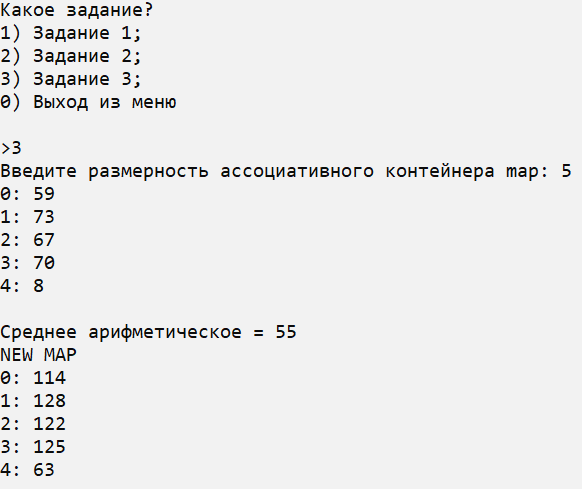
}

#endif

**Скриншоты**







**Контрольные вопросы**



1. 

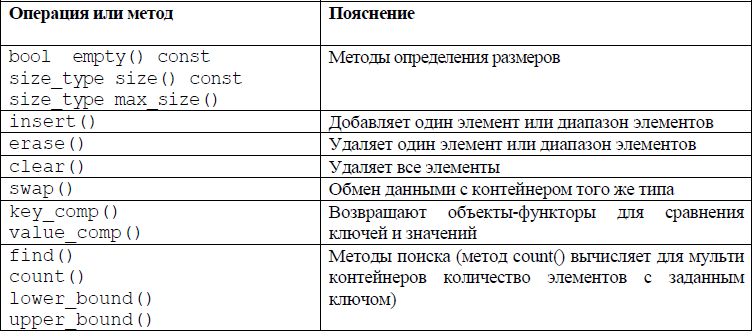


2. map, multimap, multiset, set.

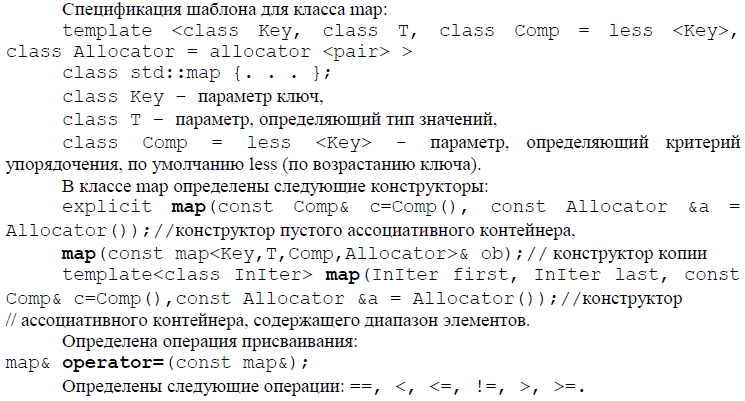


3. Использовать итератор для получения доступа к элементам ассоциативного контейнера.

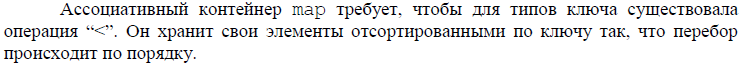


4. 



5. 



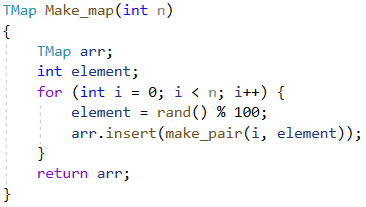
6. 



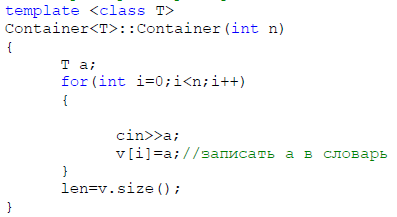
7. 



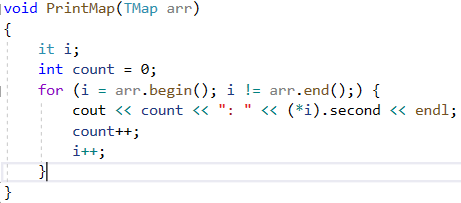


8. 

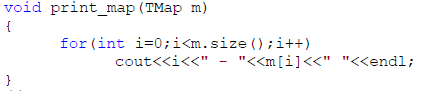


9. 

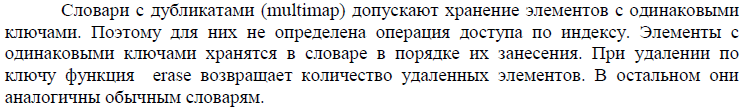


10. 



11. 



12. 

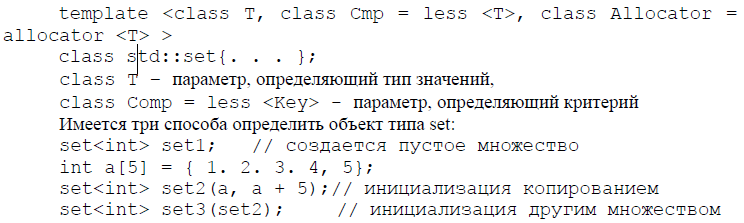


13. 

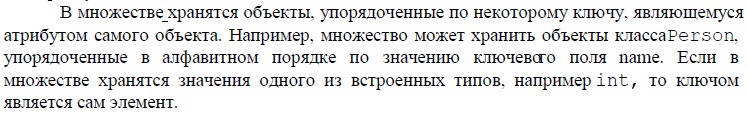


14. В множестве set значения не играют роли, так что рассматриваются только ключи.



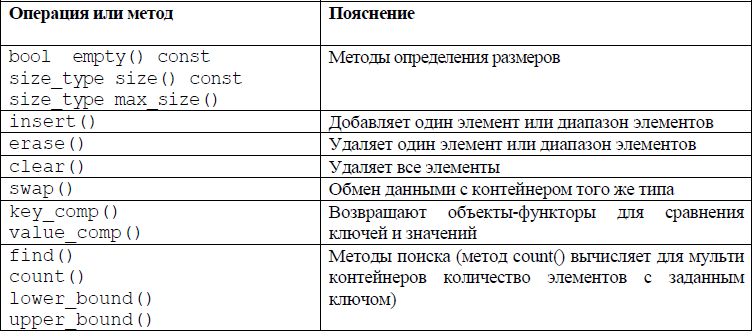
15. 



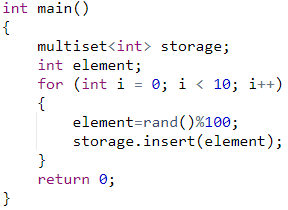
16. 



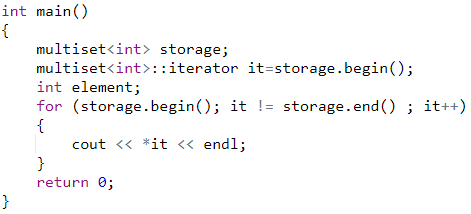
17. К множествам применимы общие для всех контейнеров методы.





18. 



19. 



20. 