Министерство образования и науки Российской Федерации  
Федеральное государственное автономное образовательное учреждение высшего образования

«ПЕРМСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

(ПНИПУ)

Электротехнический факультет

Кафедра Информационные технологии и автоматизированные системы

Лабораторные работы по

«Основам алгоритмизации и программирования»

вариант № 15

за 2 семестр

Выполнил:

студент группы РИС-21-1бз

Тимолянов Григорий Константинович

Проверила:

Доцент кафедры ИТАС,

к.т.н. Полякова О.А.

2024

###### **Лабораторная работа №12**

**Ассоциативные контейнеры библиотеки STL**

**Цель:** 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.

Контейнер - multiset

Тип элементов – float

Найти минимальный элемент и добавить его в конец контейнера;

Найти элемент с заданным ключом и удалить его из контейнера;

К каждому элементу добавить сумму минимального и максимального элементов контейнера.

###### **Функции для решения задачи 1**

#include<iostream>

#include<set>

typedef std::multiset<float> TMSet;

typedef TMSet::iterator it;

TMSet make\_multiset(size\_t n)

{

TMSet m;

for (size\_t i = 0; i < n; i++) //создать элемент и добавить его в множество

{

m.insert(rand() % 10);

}

return m;

}

void print\_multiset(const TMSet& m)

{

for (auto& item : m)

std::cout << item << ": ";

}

float arithmetic\_mean(const TMSet& m)

{

it i = m.begin();

float s{};

for (size\_t j = 0; j < m.size(); j++, i++)

s += \*i;

return s / m.size();

}

it Max(const TMSet& m)

{

auto iter = m.end();

return (--iter);

}

it Min(const TMSet& m)

{

auto iter = m.begin();

return iter;

}

void division(TMSet& m)

{

float min = \*Min(m);

if (!min)

{

std::cout << "Error! Minimum value = 0.\nDivision failed." << std::endl;

return;

}

it iter = m.begin();

float\* arr = new float[m.size()];

size\_t size = m.size();

for (size\_t i = 0; i < m.size(); i++, iter++)

{

arr[i] = \*iter / min;

}

m.clear();

for (size\_t i = 0; i < size; i++)

{

m.insert(arr[i]);

}

delete[] arr;

}

void add\_min\_el(TMSet& m)

{

m.insert(\*Min(m));

}

void remove\_element\_by\_index(TMSet& m, size\_t pos)

{

it i = m.begin();

for (size\_t j = 0; j < pos; i++, j++);

m.erase(i);

}

void add\_sum\_max\_min(TMSet& m)

{

float sum = \*Min(m) + \*Max(m);

it iter = m.begin();

float\* arr = new float[m.size()];

size\_t size = m.size();

for (size\_t i = 0; i < m.size(); i++, iter++)

{

arr[i] = \*iter + sum;

}

m.clear();

for (size\_t i = 0; i < size; i++)

{

m.insert(arr[i]);

}

delete[] arr;

}

###### **Основная программа для решения задачи 1**

int main()

{

size\_t n{};

std::cout << "N? ";

std::cin >> n;

TMSet m = make\_multiset(n); //создать словарь

print\_multiset(m);

float el = arithmetic\_mean(m); //вычислить среднее значение

std::cout << "\nArithmetic mean: " << el << std::endl;

m.insert(el); //добавить в конец

print\_multiset(m);

float max = \*Max(m);

std::cout << "\nMax: " << max << std::endl;

m.erase(max);

print\_multiset(m);

float min = \*Min(m);

std::cout << "\nMin: " << min << std::endl;

division(m);

print\_multiset(m);

add\_min\_el(m);

std::cout << "\nAdd minimal element: " << std::endl;

print\_multiset(m);

std::cout << "\nWhich element should i delete? " << std::endl;

float value;

std::cin >> value;

m.erase(value);

print\_multiset(m);

std::cout << "\nWhich position should i delete? " << std::endl;

size\_t pos;

std::cin >> pos;

remove\_element\_by\_index(m, pos);

print\_multiset(m);

std::cout << "\nAdd the sum of the minimum and maximum elements to each element: " << std::endl;

add\_sum\_max\_min(m);

print\_multiset(m);

return 0;

}

###### Объяснение результатов работы программы.

###### 

Создаем контейнер multiset на 5 элементов и заполняем его случайными числами;

Находим средне-арифметическое значение и вставляем его в контейнер;

Находим максимальное число и удаляем его;

Находим минимльное значение и делим на него все остальные элементы (в данном случае минимальное значение – нуль, деление не происходит);

Вставка минимального элемента в контейнер;

Удаление элемента по значению, затем по позиции;

Добавление к всем элементам суммы минимального и максимального значений контейнера.

###### **Задача 2.**

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.

Тип элементов Money.

###### **Классы для решения задачи 2**

Money.h

 #pragma once

#include<iostream>

class Money

{

int m\_rubles;

int m\_kopeck;

public:

//---------constructor-----------

Money() :m\_rubles(0), m\_kopeck(0) {};

Money(long, short);

Money(const Money&);

~Money();

//-------Getters and Setters----

long Get\_rubles();

short Get\_kopeck();

void Set\_rubles(long);

void Set\_kopeck(short);

//-------overloaded-functions----

Money& operator=(const Money&);

Money operator+(const Money&);

Money operator+(int);

Money& operator+=(const Money&);

Money operator/(const Money&);

Money& operator/=(const Money&);

Money operator/(int);

bool operator>(const Money&);

bool operator<(const Money&);

bool operator!=(const Money&);

bool operator!=(int);

bool operator==(int);

bool operator==(const Money&);

void clear();

//-------friend-functions---------

friend bool operator<(const Money&, const Money&);

friend std::ostream& operator<<(std::ostream&, const Money&);

friend std::istream& operator>>(std::istream&, Money&);

};

Money.cpp

 #include "Money.h"

Money::Money(const long rubles, const short kopeck)

{

if (rubles < 0 || kopeck < 0)

{

m\_rubles = -abs(m\_rubles);

m\_kopeck = -abs(m\_kopeck);

}

long long money = (long long)m\_rubles \* 100 + m\_kopeck;

m\_rubles = money / 100;

m\_kopeck = money % 100;

}

Money::Money(const Money& money)

{

m\_rubles = money.m\_rubles;

m\_kopeck = money.m\_kopeck;

}

Money::~Money()

{

}

long Money::Get\_rubles()

{

return m\_rubles;

}

short Money::Get\_kopeck()

{

return m\_kopeck;

}

void Money::Set\_rubles(const long rubles)

{

if (m\_kopeck >= 0)

m\_rubles = rubles;

else

m\_rubles = -abs(rubles);

}

void Money::Set\_kopeck(const short kopeck)

{

m\_kopeck = kopeck;

if (m\_rubles < 0)

m\_kopeck = -abs(m\_kopeck);

if (m\_kopeck < 0)

m\_rubles = -abs(m\_rubles);

}

Money& Money::operator=(const Money& money)

{

if (&money == this)

return \*this;

m\_rubles = money.m\_rubles;

m\_kopeck = money.m\_kopeck;

return \*this;

}

Money Money::operator+(const Money& money)

{

long long money\_first = (long long)m\_rubles \* 100 + m\_kopeck;

long long money\_second = (long long)money.m\_rubles \* 100 + money.m\_kopeck;

Money p;

p.m\_rubles = (money\_first + money\_second) / 100;

p.m\_kopeck = (money\_first + money\_second) % 100;

return p;

}

Money Money::operator+(int value)

{

long long money = (long long)m\_rubles \* 100 + m\_kopeck;

money += value;

Money temp;

temp.m\_rubles = (money) / 100;

temp.m\_kopeck = (money) % 100;

return temp;

}

Money& Money::operator+=(const Money& money)

{

long long temp1 = (long long)m\_rubles \* 100 + m\_kopeck;

long long temp2 = (long long)money.m\_rubles \* 100 + money.m\_kopeck;

m\_rubles = (temp1 + temp2) / 100;

m\_kopeck = (temp1 + temp2) % 100;

return \*this;

}

Money Money::operator/(const Money& money)

{

Money division;

if (&money == this)

{

division.m\_rubles = 0;

division.m\_kopeck = 1;

return division;

}

long long money\_first = m\_rubles;

money\_first \*= 100;

money\_first += m\_kopeck;

long long money\_second = money.m\_rubles;

money\_second \*= 100;

money\_second += money.m\_kopeck;

division.m\_rubles = (money\_first / money\_second) / 100;

division.m\_kopeck = (money\_first / money\_second) % 100;

return division;

}

Money& Money::operator/=(const Money& money)

{

if (&money == this)

{

m\_rubles = 0;

m\_kopeck = 1;

return \*this;

}

long long money\_first = (long long)m\_rubles \* 100 + m\_kopeck;

long long money\_second = (long long)money.m\_rubles \* 100 + money.m\_kopeck;

m\_rubles = (money\_first / money\_second) / 100;

m\_kopeck = (money\_first / money\_second) % 100;

return \*this;

}

Money Money::operator/(const int value)

{

int temp1 = m\_rubles \* 100 + m\_kopeck;

Money temp\_money;

temp\_money.m\_rubles = (temp1 / value) / 100;

temp\_money.m\_kopeck = (temp1 / value) % 100;

return temp\_money;

}

bool Money::operator>(const Money& money)

{

return (m\_rubles > money.m\_rubles || m\_rubles == money.m\_rubles && m\_kopeck > money.m\_kopeck);

}

bool Money::operator<(const Money& money)

{

return (m\_rubles < money.m\_rubles || m\_rubles == money.m\_rubles && m\_kopeck < money.m\_kopeck);

}

bool Money::operator!=(const Money& money)

{

return !(m\_rubles == money.m\_rubles && m\_kopeck == money.m\_kopeck);

}

bool Money::operator!=(int value)

{

return !(m\_rubles == value && m\_kopeck == value);

}

bool Money::operator==(int value)

{

return (m\_rubles == value && m\_kopeck == value);

}

bool Money::operator==(const Money& money)

{

return (m\_rubles == money.m\_rubles && m\_kopeck == money.m\_kopeck);

}

void Money::clear()

{

m\_kopeck = 0;

m\_rubles = 0;

}

bool operator<(const Money& \_Left, const Money& \_Right)

{

return (\_Left.m\_rubles < \_Right.m\_rubles || \_Left.m\_rubles == \_Right.m\_rubles && \_Left.m\_kopeck < \_Right.m\_kopeck);

}

std::ostream& operator<<(std::ostream& out, const Money& time)

{

return (out << time.m\_rubles << ":" << time.m\_kopeck);

}

std::istream& operator>>(std::istream& in, Money& money)

{

int kopeck\_sup;

std::cout << "Rubles? "; in >> money.m\_rubles;

do

{

std::cout << "Kopeck? "; in >> kopeck\_sup;

if (abs(kopeck\_sup) > 99)

std::cout << "Error!. Please enter a number from -99 to 99" << std::endl;

} while (abs(kopeck\_sup) > 99);

money.m\_kopeck = kopeck\_sup;

if (money.m\_rubles < 0 || money.m\_kopeck < 0)

{

money.m\_rubles = -abs(money.m\_rubles);

money.m\_kopeck = -abs(money.m\_kopeck);

}

return in;

}

###### Функции для решения задачи 2

#include<iostream>

#include<set>

#include"Money.h"

typedef std::multiset<Money> TMSet;

typedef TMSet::iterator it;

TMSet make\_multiset(size\_t n)

{

TMSet m;

Money money;

for (size\_t i = 0; i < n; i++) //создать элемент и добавить его в множество

{

std::cin >> money;

m.insert(money);

}

return m;

}

void print\_multiset(const TMSet& m)

{

for (auto& item : m)

std::cout << item << ": ";

}

Money arithmetic\_mean(const TMSet& m)

{

it i = m.begin();

Money s{};

for (size\_t j = 0; j < m.size(); j++, i++)

s += \*i;

return s / m.size();

}

it Max(const TMSet& m)

{

auto iter = m.end();

return (--iter);

}

it Min(const TMSet& m)

{

auto iter = m.begin();

return iter;

}

void division(TMSet& m)

{

Money min = \*Min(m);

if (min == 0)

{

std::cout << "Error! Minimum value = 0.\nDivision failed." << std::endl;

return;

}

it iter = m.begin();

Money\* arr = new Money[m.size()];

size\_t size = m.size();

for (size\_t i = 0; i < m.size(); i++, iter++)

{

arr[i] = \*iter;

arr[i] /= min;

}

m.clear();

for (size\_t i = 0; i < size; i++)

{

m.insert(arr[i]);

}

delete[] arr;

}

void add\_min\_el(TMSet& m)

{

m.insert(\*Min(m));

}

void remove\_element\_by\_index(TMSet& m, size\_t pos)

{

if (pos > m.size())

{

std::cout << "Error. Position > size" << std::endl;

return;

}

it i = m.begin();

for (size\_t j = 0; j < pos; i++, j++);

m.erase(i);

}

void add\_sum\_max\_min(TMSet& m)

{

Money sum = \*Min(m);

sum += \*Max(m);

it iter = m.begin();

Money\* arr = new Money[m.size()];

size\_t size = m.size();

for (size\_t i = 0; i < m.size(); i++, iter++)

{

arr[i] = \*iter;

arr[i] += sum;

}

m.clear();

for (size\_t i = 0; i < size; i++)

{

m.insert(arr[i]);

}

delete[] arr;

}

###### Основная программа для решения задачи 2

int main()

{

size\_t n{};

std::cout << "N? ";

std::cin >> n;

TMSet m = make\_multiset(n); //создать словарь

print\_multiset(m);

Money el = arithmetic\_mean(m); //вычислить среднее значение

std::cout << "\nArithmetic mean: " << el << std::endl;

m.insert(el); //добавить в конец

print\_multiset(m);

Money max = \*Max(m);

std::cout << "\nMax: " << max << std::endl;

m.erase(max);

print\_multiset(m);

Money min = \*Min(m);

std::cout << "\nMin: " << min << std::endl;

add\_min\_el(m);

std::cout << "\nAdd minimal element: " << std::endl;

print\_multiset(m);

std::cout << "\nWhich element should i delete? " << std::endl;

Money value;

std::cin >> value;

m.erase(value);

print\_multiset(m);

std::cout << "\nWhich position should i delete? " << std::endl;

size\_t pos;

std::cin >> pos;

remove\_element\_by\_index(m, pos);

print\_multiset(m);

std::cout << "\nAdd the sum of the minimum and maximum elements to each element: " << std::endl;

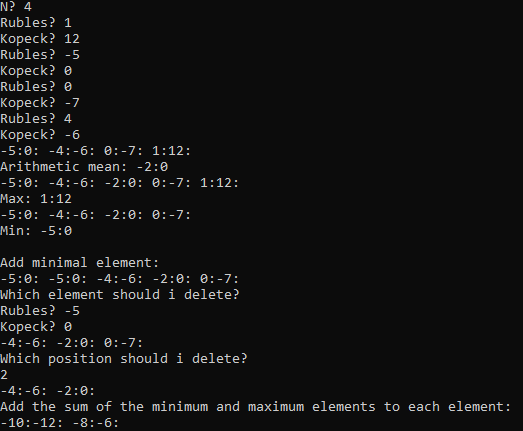
add\_sum\_max\_min(m);

print\_multiset(m);

return 0;

}

###### Объяснение результатов работы программы.



Создаем контейнер multiset на 5 элементов и заполняем его случайными числами;

Находим средне-арифметическое значение и вставляем его в контейнер;

Находим максимальное число и удаляем его;

Вставка минимального элемента в контейнер;

Удаление элемента по значению, затем по позиции;

Добавление к всем элементам суммы минимального и максимального значений контейнера.

###### Задача 3

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.

Параметризированный класс – Вектор

Тип элементов Money.

###### Классы для решения задачи 3

Container.h

#pragma once

#include<iostream>

#include<set>

template <class T>

class Container

{

std::multiset<T> m;

public:

//---------constructor-----------

Container() {};

Container(size\_t);

~Container();

//-------other-functions----

void Print();

T arithmetic\_mean();

void Add(const T&);

typename std::multiset<T>::iterator Max();

typename std::multiset<T>::iterator Min();

void delete\_el(const T&);

void division();

void remove\_element\_by\_index(int);

void add\_min\_el();

void add\_sum\_max\_min();

};

template<class T>

inline Container<T>::Container(size\_t n)

{

T money;

for (size\_t i = 0; i < n; i++) //создать элемент и добавить его в множество

{

std::cin >> money;

m.insert(money);

}

}

template<class T>

inline Container<T>::~Container()

{

}

template<class T>

inline void Container<T>::Print()

{

for (auto& item : m)

std::cout << item << ": ";

}

template<class T>

T Container<T>::arithmetic\_mean()

{

if (!m.size())

throw 0;

class std::multiset<T>::iterator i = m.begin();

T s{};

for (size\_t j = 0; j < m.size(); j++, i++)

s += \*i;

return s / m.size();

}

template<class T>

void Container<T>::Add(const T& el)

{

m.insert(el);

}

template<class T>

typename std::multiset<T>::iterator Container<T>::Max()

{

if (!m.size())

throw 1;

auto iter = m.end();

return (--iter);

}

template<class T>

typename std::multiset<T>::iterator Container<T>::Min()

{

if (!m.size())

throw 2;

auto iter = m.begin();

return iter;

}

template<class T>

void Container<T>::delete\_el(const T& temp)

{

if (!m.size())

throw 3;

m.erase(temp);

}

template<class T>

void Container<T>::division()

{

T min = \*Min();

if (min == 0)

{

std::cout << "Error! Minimum value = 0.\nDivision failed." << std::endl;

return;

}

class std::multiset<T>::iterator iter = m.begin();

T\* arr = new T[m.size()];

size\_t size = m.size();

for (size\_t i = 0; i < m.size(); i++, iter++)

{

arr[i] = \*iter;

arr[i] /= min;

}

m.clear();

for (size\_t i = 0; i < size; i++)

{

m.insert(arr[i]);

}

delete[] arr;

}

template<class T>

inline void Container<T>::remove\_element\_by\_index(int pos)

{

if (pos >= m.size())

{

std::cout << "Error. Position > size" << std::endl;

return;

}

class std::multiset<T>::iterator i = m.begin();

for (size\_t j = 0; j < pos; i++, j++);

m.erase(i);

}

template<class T>

inline void Container<T>::add\_min\_el()

{

m.insert(\*Min());

}

template<class T>

inline void Container<T>::add\_sum\_max\_min()

{

T sum = \*Min();

sum += \*Max();

class std::multiset<T>::iterator iter = m.begin();

T\* arr = new T[m.size()];

size\_t size = m.size();

for (size\_t i = 0; i < m.size(); i++, iter++)

{

arr[i] = \*iter;

arr[i] += sum;

}

m.clear();

for (size\_t i = 0; i < size; i++)

{

m.insert(arr[i]);

}

delete[] arr;

}

###### Основная программа для решения задачи 3

int main()

{

try

{

size\_t n{};

std::cout << "N? ";

std::cin >> n;

Container<Money> m(n); //создать словарь

m.Print();

Money el = m.arithmetic\_mean(); //вычислить среднее значение

std::cout << "\nArithmetic mean: " << el << std::endl;

m.Add(el); //добавить в конец

m.Print();

Money max = \*m.Max();

std::cout << "\nMax: " << max << std::endl;

m.delete\_el(max);

m.Print();

Money min = \*m.Min();

std::cout << "\nMin: " << min << std::endl;

m.add\_min\_el();

std::cout << "\nAdd minimal element: " << std::endl;

m.Print();

std::cout << "\nWhich element should i delete? " << std::endl;

Money value;

std::cin >> value;

m.delete\_el(value);

m.Print();

std::cout << "\nWhich position should i delete? " << std::endl;

size\_t pos;

std::cin >> pos;

m.remove\_element\_by\_index(pos);

m.Print();

std::cout << "\nAdd the sum of the minimum and maximum elements to each element: " << std::endl;

m.add\_sum\_max\_min();

m.Print();

}

catch (const int num\_error)

{

switch (num\_error)

{

case 0:

std::cout << "Error in calculating the arithmetic mean. The container is empty." << std::endl;

break;

case 1:

std::cout << "Error getting the maximum element. The container is empty." << std::endl;

break;

case 2:

std::cout << "Error getting the minimum element. The container is empty." << std::endl;

break;

case 3:

std::cout << "Error deleting an element. The container is empty." << std::endl;

break;

default:

std::cout << "Error!" << std::endl;

break;

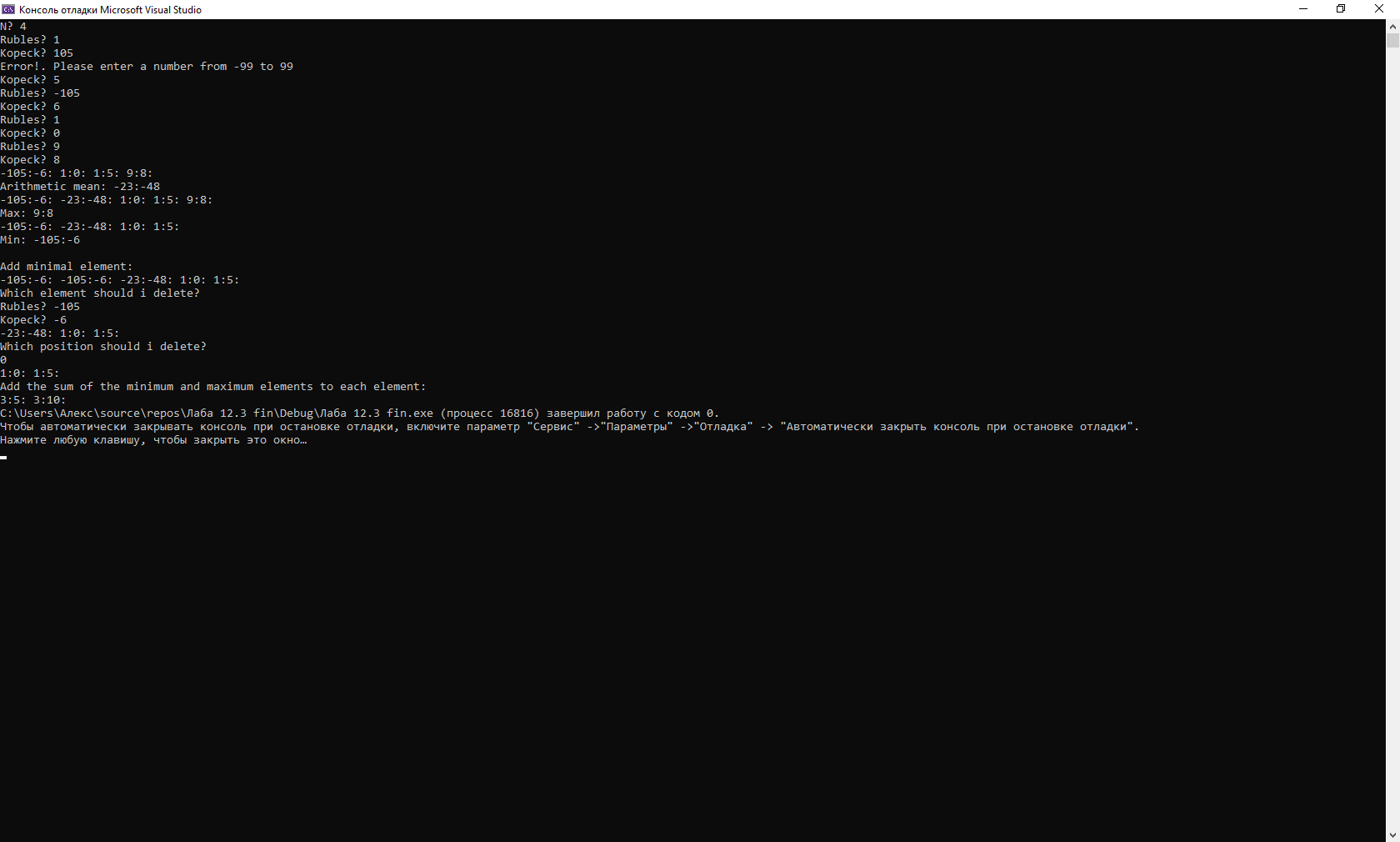
}

}

return 0;

}

###### Объяснение результатов работы программы.



Создаем контейнер multiset на 5 элементов и заполняем его случайными числами;

Находим средне-арифметическое значение и вставляем его в контейнер;

Находим максимальное число и удаляем его;

Вставка минимального элемента в контейнер;

Удаление элемента по значению, затем по позиции;

Добавление к всем элементам суммы минимального и максимального значений контейнера.

###### **Ответы на контрольные вопросы.**

1. Что представляет собой ассоциативный контейнер?

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

2. Перечислить ассоциативные контейнеры библиотеки STL.

Map, multimap, set, multiset.

3. Каким образом можно получить доступ к элементам ассоциативного контейнера?

Доступ к элементам контейнера массива осуществляется с использованием итератора.

4. Привести примеры методов, используемых в ассоциативных контейнерах.

Insert, clear, erase, fine, count, empty.

5. Каким образом можно создать контейнер map? Привести примеры.

map<int, int> m;

m.insert(make\_pair(1, 2));

m.insert(make\_pair(22, 442));

m[98] = 999;

map<int, pair<double, long long>> newMap;

newMap[65] = make\_pair(3.4, 3);

map<int, pair<double, long long>>::iterator it;

it = newMap.begin();

std::cout << it->first << " " << it->second.first << ":" << it->second.second;

6. Каким образом упорядочены элементы в контейнере map по умолчанию? Как изменить порядок на обратный?

Упорядочены по возрастанию. Указать порядок сортировки:

map<int, int, greater<>> x;

7. Какие операции определены для контейнера map?

Операции присваивания, сравнения.

1. Написать функцию для добавления элементов в контейнер map с помощью функции make\_pair().

std::map<int, int>&fillMap(const int count, std::map<int, int>&myMap)

{

int myKey{};

int myValue{};

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

{

std::cout << "Key> ";

std::cin >> myKey;

std::cout << "Value> ";

std::cin >> myValue;

myMap.insert(std::make\_pair(myKey, myValue));

}

return myMap;

}

Функция первым аргументом принимает кол-во элементов в словаре и просит их заполнить, вторым аргументом принимает ссылку на ранее созданный словарь и возвращает её же.

1. Написать функцию для добавления элементов в контейнер map с помощью функции операции прямого доступа [].

std::map<int, int>&fillMap(const int count, std::map<int, int>&myMap)

{

int myKey{};

int myValue{};

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

{

std::cout << "Key> ";

std::cin >> myKey;

std::cout << "Value> ";

std::cin >> myValue;

myMap[myKey] = myValue;

}

return myMap;

}

1. Написать функцию для печати контейнера map с помощью итератора.

void printMap(std::map<int, int>&myMap)

{

std::map<int, int>::iterator it;

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

{

std::cout << it->first << ":" << it->second << std::endl;

}

}

1. Написать функцию для печати контейнера map с помощью функции операции прямого доступа [].

void printMap(std::map<int, int>&myMap)

{

std::map<int, int>::iterator it;

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

{

std::cout << myMap[it->first] << ":" << it->second << std::endl;

}

}

12. Чем отличаются контейнеры map и multimap?

Map хранит элементы с уникальным ключом, multimap позволяет хранить элементы с одинаковым ключом.

13. Что представляет собой контейнер set?

Это контейнер, в котором хранятся только уникальные элементы, и повторения запрещены. Элементы сортируются в соответствии с их значениями.

14. Чем отличаются контейнеры map и set?

Map это set, в котором каждый элемент является парой ключ-значение. Ключ используется для сортировки и индексации данных и должен быть уникальным, а значение — это фактические данные.

15. Каким образом можно создать контейнер set? Привести примеры.

int a[4] = { 1, 3, 5, 7 };

std::set<int> mySet(a, a + 4);

mySet.insert(10);

mySet.insert(6);

16. Каким образом упорядочены элементы в контейнере set по умолчанию? Как изменить порядок на обратный?

По умолчанию упорядочены по возрастанию. Сменить можно так:

set<int, std::greater<>>

17. Какие операции определены для контейнера set?

Count, find, size, insert, emplace, erase/

18. Написать функцию для добавления элементов в контейнер set.

std::set<int>&fillSet(const int count, std::set<int>&mSet)

{

int elem{};

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

{

std::cout << "Elem>";

std::cin >> elem;

mSet.insert(elem);

}

return mSet;

}

19. Написать функцию для печати контейнера set.

void printSet(std::set<int>&mSet)

{

for (const auto& item : mSet)

{

std::cout << item << " ";

}

}

20. Чем отличаются контейнеры set и multiset?

Set хранит только уникальные элементы, multiset допускает повторяющиеся элементы.