**Министерство образования и науки Российской Федерации**

**Пермское федеральное государственное бюджетное образовательное учреждение высшего образования «Пермский Национальный Исследовательский Политехнический Университет»**

**Электротехнический факультет**

**Кафедра «Информационные технологии и автоматизированные системы»**

**ОТЧЁТ**

По лабораторной работе №18.13 на тему

«Стандартные обобщенные алгоритмы библиотеки STL»

Вариант №11

Выполнил студент группы ИВТ-20-2б

Сабуров Павел Алексеевич

Проверил доцент кафедры ИТАС

Полякова О.А.

Пермь 2021

**Цель работы** –научиться работать с ассоциативными контейнерами в библиотеке STL входящей в поставку с языком программирования C++;

**Постановка задачи**

Задача – реализовать на языке программирования C++ программу, которая демонстрирует возможности работы с ассоциативными контейнерами в библиотеке STL

**Исходные данные для варианта №11:**

Задание 1:

1.  Создать последовательный 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.

Задание 2:

1.  Контейнер - map

2.  Тип элементов - float

Задание 3:

Найти минимальный элемент и добавить его на заданную позицию контейнера

Задание 4:

Найти элементы большие среднего арифметического и удалить их из контейнера

Задание 5:

Каждый элемент домножать на максимальный элемент контейнера

**Анализ задачи**

Для решения задачи были использованы следующие средства:

1. Язык программирования C++ (Microsoft Visual C++)
2. Текстовый редактор Microsoft Visual Studio Code

Ассоциативный массив, массив пар значений:

1. Ключ;
2. Данные;

Ключ как и данные могут быть любых типов, поэтому они называются ассоциативными;

**Задание №1:**

Основная программа, он же полный исходный код:

#include <iostream>

#include <map>

using namespace std;

**//Определятся словарь**

typedef map<int, float> d\_map;

typedef d\_map::iterator it;

**//функция которая создаёт новый словарь**

d\_map make\_map(int count)

{

d\_map new\_map;

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

{

cout << "Enter the number: ";

float number; cin >> number;

new\_map.insert(make\_pair(i, number));

}

return new\_map;

}

**//функция вывода словаря**

void print(d\_map map)

{

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

cout << i << ": " << map[i] << ";\n";

}

float average\_arithmetic(d\_map map)

{

int summa = 0;

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

summa += map[i];

return summa / map.size();

}

int main()

{

cout << "Enter the count of items:\n";

int n; cin >> n;

d\_map map = make\_map(n);

print(map);

**//нахождение среднего арифметического и добавление в начало списка**

float average = average\_arithmetic(map);

cout << "Average arithmetic = " << average << ";\n";

map.insert(make\_pair(map.size(), average));

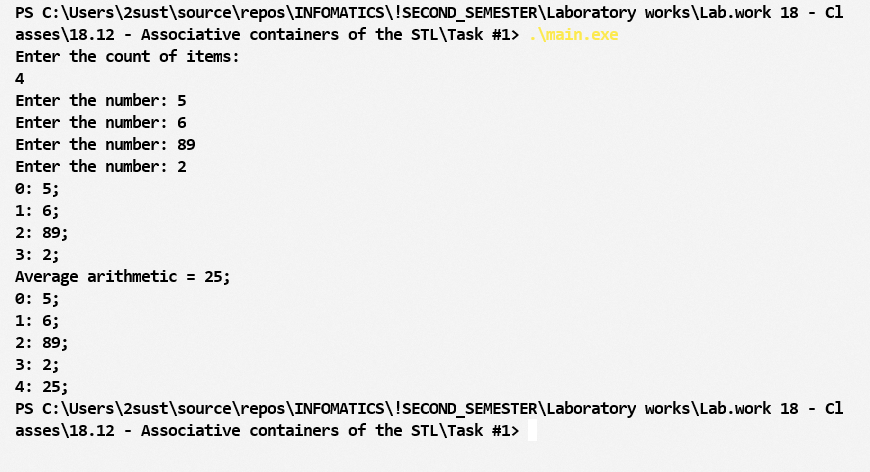
print(map);

map.erase(0);

return 0;

}

**Скриншот выполненной программы:**



Задание №2:

Задание то же самое, но вместо стандартного типа данных используется пользовательский класс «Деньги», который был реализован в предыдущих лабораторных работах.

Вот его заголовочный файл:

#pragma once

#include <iostream>

#include <fstream>

using namespace std;

class Money

{

private:

long \_dollars;

int \_cents;

public:

int Dollars() { return \_dollars; }

void SetDollars(long dollars);

int Cents() { return \_cents; }

void SetCents(int cents);

void PrintMoney();

friend bool operator > (Money& first, Money& second);

friend bool operator < (Money& first, Money& second);

friend bool operator == (Money& first, Money& second);

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

friend ostream& operator << (ostream& out, Money& money);

friend fstream& operator >> (fstream& fin, Money& money);

friend fstream& operator << (fstream& fout, Money& money);

Money& operator++ ();

Money operator++ (int);

Money& operator = (const Money& value);

Money& operator += (Money& adder);

Money& operator -= (Money& subtractor);

Money& operator /= (const Money& divider);

Money();

Money(long dollars, int cents);

Money(const Money& parentMoney);

~Money();

};

В основной же программе изменения минимальны:

1. Вместо float пишется тип данных Money;
2. Меняются сообщения в интерфейсе под денежный лад;

В итоге основная программа принимает следующий вид:

#include <iostream>

#include <map>

#include "Money.h"

using namespace std;

typedef map<int, Money> d\_map;

typedef d\_map::iterator it;

d\_map make\_map(int count)

{

d\_map new\_map;

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

{

cout << "Enter the number: ";

Money number; cin >> number;

new\_map.insert(make\_pair(i, number));

}

return new\_map;

}

void print(d\_map map)

{

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

cout << i << ": " << map[i] << ";\n";

}

Money average\_arithmetic(d\_map map)

{

Money summa(0, 0);

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

summa += map[i];

summa /= Money(map.size(), 0);

return summa;

}

int main()

{

cout << "Enter the count of items:\n";

int n; cin >> n;

d\_map map = make\_map(n);

print(map);

Money average = average\_arithmetic(map);

cout << "Average arithmetic = " << average << ";\n";

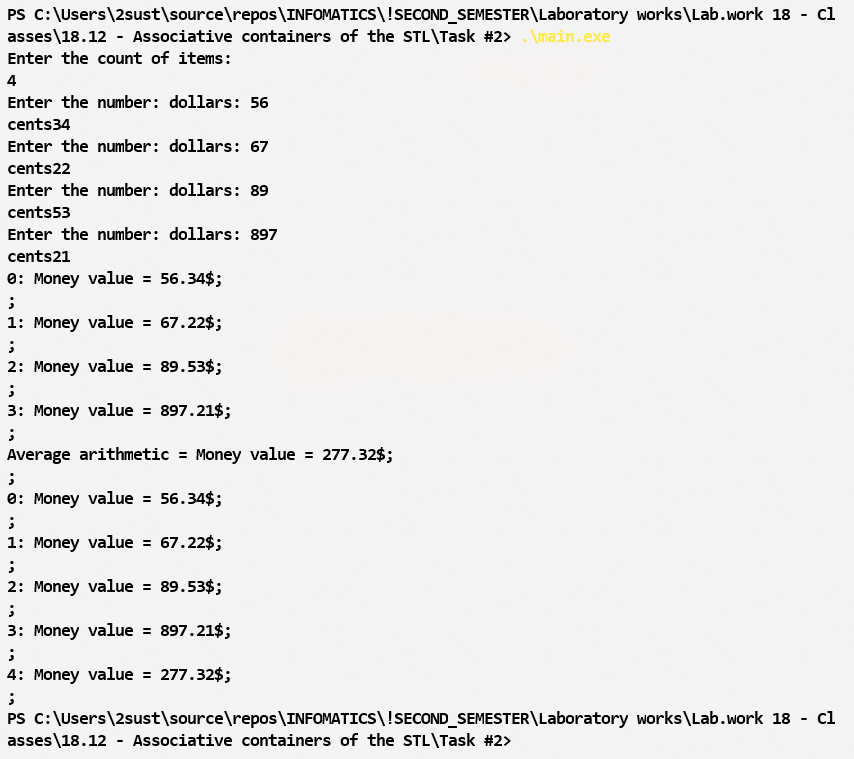
map.insert(make\_pair(map.size(), average));

print(map);

return 0;

}

**Скриншот выполненной программы:**



Задание №3:

Третье задание отличается от первых двух тем, что вместо глобальных функций необходимо реализовать пользовательский параметрический класс, то есть класс с шаблонами.

Вся реализация подобных классов находится внутри одного заголовочного файла:

#pragma once

#include <iostream>

#include <map>

using namespace std;

**//Параметризованный класс**

template <class T>

class UserMap

{

private:

**//Поля класса: словарь и его длина**

map<int, T> m\_map;

int m\_length;

public:

UserMap();

explicit UserMap(int length);

~UserMap();

void PrintMap();

void Add(int index, T item);

T Average();

};

**//конструкторы класса**

template <class T>

UserMap<T>::UserMap() { m\_length = 0; }

template<class T>

UserMap<T>::UserMap(int length)

{

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

{

T item;

cin >> item;

m\_map[i] = item;

}

m\_length = length;

}

**//вывод словаря на экран**

template <class T>

void UserMap<T>::PrintMap()

{

for (int i = 0; i < m\_length; i++)

cout << i << ": " << m\_map[i] << ";\n";

}

**//добавление нового элемента в словарь**

template <class T>

void UserMap<T>::Add(int index, T item)

{

m\_map.insert(make\_pair(index, item));

m\_length++;

}

**//нахождение среднего арифметического в словаре**

template <class T>

T UserMap<T>::Average()

{

T summa = m\_map[0];

summa -= m\_map[0];

for (int i = 0; i < m\_length; i++)

summa += m\_map[i];

summa /= m\_length;

return summa;

}

**//деструктор**

template<class T>

UserMap<T>::~UserMap()

{

}

В связи с изменениями, основная программа принимает следующий вид:

#include "Money.h"

#include "UserMap.h"

using namespace std;

int ReadNaturalNum()

{

int x = -1;

while (x <= 0)

cin >> x;

return x;

}

int main()

{

cout << "----Float----\n";

cout << "Input count of numbers:\n";

int count = ReadNaturalNum();

UserMap<float> floatMap(count);

floatMap.PrintMap();

float average = floatMap.Average();

cout << "average = " << average << ";\n";

floatMap.Add(count, average);

floatMap.PrintMap();

cout << "----Money----\n";

cout << "Input count of numbers:\n";

int moneyCount = ReadNaturalNum();

UserMap<Money> MoneyMap(moneyCount);

MoneyMap.PrintMap();

Money maverage = MoneyMap.Average();

cout << "average = " << maverage << ";\n";

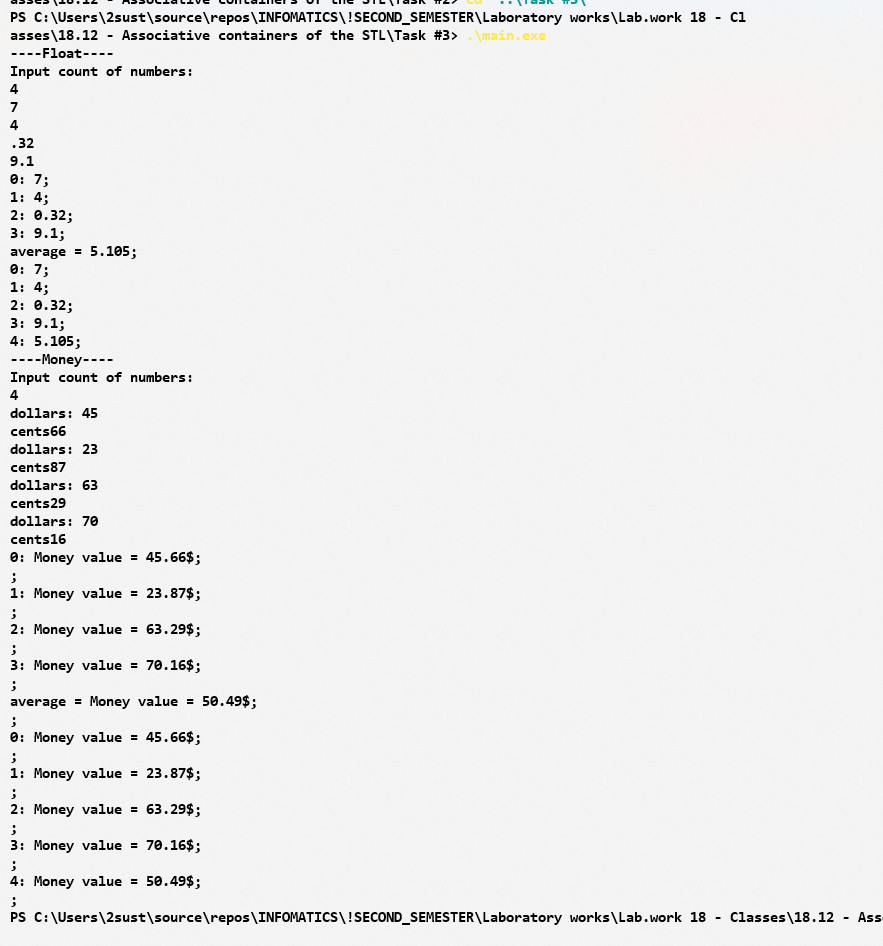
MoneyMap.Add(moneyCount, maverage);

MoneyMap.PrintMap();

return 0;

}

**Скриншот выполненной программы:**



**Полные исходные коды заданий на языке программирования C++:**

Задание №1:

#include <iostream>

#include <map>

using namespace std;

typedef map<int, float> d\_map;

typedef d\_map::iterator it;

d\_map make\_map(int count)

{

d\_map new\_map;

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

{

cout << "Enter the number: ";

float number; cin >> number;

new\_map.insert(make\_pair(i, number));

}

return new\_map;

}

void print(d\_map map)

{

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

cout << i << ": " << map[i] << ";\n";

}

float average\_arithmetic(d\_map map)

{

int summa = 0;

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

summa += map[i];

return summa / map.size();

}

int main()

{

cout << "Enter the count of items:\n";

int n; cin >> n;

d\_map map = make\_map(n);

print(map);

float average = average\_arithmetic(map);

cout << "Average arithmetic = " << average << ";\n";

map.insert(make\_pair(map.size(), average));

print(map);

return 0;

}

Задание №2:

Money.h

#pragma once

#include <iostream>

#include <fstream>

using namespace std;

class Money

{

private:

long \_dollars;

int \_cents;

public:

int Dollars() { return \_dollars; }

void SetDollars(long dollars);

int Cents() { return \_cents; }

void SetCents(int cents);

void PrintMoney();

friend bool operator > (Money& first, Money& second);

friend bool operator < (Money& first, Money& second);

friend bool operator == (Money& first, Money& second);

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

friend ostream& operator << (ostream& out, Money& money);

friend fstream& operator >> (fstream& fin, Money& money);

friend fstream& operator << (fstream& fout, Money& money);

Money& operator++ ();

Money operator++ (int);

Money& operator = (const Money& value);

Money& operator += (Money& adder);

Money& operator -= (Money& subtractor);

Money& operator /= (const Money& divider);

Money();

Money(long dollars, int cents);

Money(const Money& parentMoney);

~Money();

};

Money.cpp

#include "Money.h"

#include <string>

using namespace std;

void Money::SetDollars(long dollars)

{

if (dollars < 0)

{

cout << "Error: dollars less than 0;\n";

\_dollars = 0;

}

else

\_dollars = dollars;

}

void Money::SetCents(int cents)

{

if (cents < 0 || cents >= 100)

{

cout << "Error: cents less than 0 or bigger than 100;\n";

\_cents = 0;

}

else

\_cents = cents;

}

void Money::PrintMoney()

{

string centStr = (\_cents < 10) ? '0' + to\_string(\_cents) : to\_string(\_cents);

cout << "Money value = " << \_dollars << '.' << centStr << "$;\n";

}

Money& Money::operator++()

{

\_cents++;

if (\_cents == 100)

{

\_cents = 0;

\_dollars++;

}

return \*this;

}

Money Money::operator++(int)

{

Money temp(\_dollars, \_cents);

++(\*this);

return temp;

}

Money::Money()

{

\_dollars = 0;

\_cents = 0;

}

Money::Money(long dollars, int cents)

{

SetDollars(dollars);

SetCents(cents);

}

Money::Money(const Money& parentMoney)

{

\_dollars = parentMoney.\_dollars;

\_cents = parentMoney.\_cents;

}

Money::~Money() {}

bool operator>(Money& first, Money& second)

{

if (first.Dollars() == second.Dollars())

return first.Cents() > second.Cents();

else

return first.Dollars() > second.Dollars();

}

bool operator<(Money& first, Money& second)

{

if (first.Dollars() == second.Dollars())

return first.Cents() < second.Cents();

else

return first.Dollars() < second.Dollars();

}

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

{

long dollars;

int cents;

cout << "dollars: ";

cin >> dollars;

cout << "cents";

cin >> cents;

money.SetDollars(dollars);

money.SetCents(cents);

return in;

}

ostream& operator<<(ostream& out, Money& money)

{

money.PrintMoney();

return out;

}

fstream& operator>>(fstream& fin, Money& money)

{

long dollars;

int cents;

fin >> dollars >> cents;

money.SetDollars(dollars);

money.SetCents(cents);

return fin;

}

fstream& operator<<(fstream& fout, Money& money)

{

fout << money.Dollars() << endl << money.Cents() << endl;

return fout;

}

bool operator==(Money& first, Money& second)

{

return (first.Dollars() == second.Dollars()) && (first.Cents() == second.Cents());

}

Money& Money::operator+=(Money& adder)

{

int addedCents = \_cents + adder.\_cents;

if (addedCents >= 100)

{

\_dollars++;

addedCents -= 100;

}

\_dollars += adder.\_dollars;

\_cents = addedCents;

return \*this;

}

Money& Money::operator-=(Money& subtractor)

{

this->SetCents(\_cents - subtractor.Cents());

this->SetDollars(\_dollars - subtractor.Dollars());

return \*this;

}

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

{

this->SetDollars(value.\_dollars);

this->SetCents(value.\_cents);

return \*this;

}

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

{

int dollars = this->\_dollars;

int cents = this->\_cents;

if (dollars / divider.\_dollars != 0)

cents += 100;

this->SetDollars(dollars / divider.\_dollars);

this->SetCents(cents / divider.\_dollars);

return \*this;

}

main.cpp

#include <iostream>

#include <map>

#include "Money.h"

using namespace std;

typedef map<int, Money> d\_map;

typedef d\_map::iterator it;

d\_map make\_map(int count)

{

d\_map new\_map;

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

{

cout << "Enter the number: ";

Money number; cin >> number;

new\_map.insert(make\_pair(i, number));

}

return new\_map;

}

void print(d\_map map)

{

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

cout << i << ": " << map[i] << ";\n";

}

Money average\_arithmetic(d\_map map)

{

Money summa(0, 0);

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

summa += map[i];

summa /= Money(map.size(), 0);

return summa;

}

int main()

{

cout << "Enter the count of items:\n";

int n; cin >> n;

d\_map map = make\_map(n);

print(map);

Money average = average\_arithmetic(map);

cout << "Average arithmetic = " << average << ";\n";

map.insert(make\_pair(map.size(), average));

print(map);

return 0;

}

Задание №3:

UserMap.h

#pragma once

#include <iostream>

#include <map>

using namespace std;

template <class T>

class UserMap

{

private:

map<int, T> m\_map;

int m\_length;

public:

UserMap();

explicit UserMap(int length);

~UserMap();

void PrintMap();

void Add(int index, T item);

T Average();

};

template <class T>

UserMap<T>::UserMap() { m\_length = 0; }

template<class T>

UserMap<T>::UserMap(int length)

{

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

{

T item;

cin >> item;

m\_map[i] = item;

}

m\_length = length;

}

template <class T>

void UserMap<T>::PrintMap()

{

for (int i = 0; i < m\_length; i++)

cout << i << ": " << m\_map[i] << ";\n";

}

template <class T>

void UserMap<T>::Add(int index, T item)

{

m\_map.insert(make\_pair(index, item));

m\_length++;

}

template <class T>

T UserMap<T>::Average()

{

T summa = m\_map[0];

summa -= m\_map[0];

for (int i = 0; i < m\_length; i++)

summa += m\_map[i];

summa /= m\_length;

return summa;

}

template<class T>

UserMap<T>::~UserMap()

{

}

main.cpp

#include "Money.h"

#include "UserMap.h"

using namespace std;

int ReadNaturalNum()

{

int x = -1;

while (x <= 0)

cin >> x;

return x;

}

int main()

{

cout << "----Float----\n";

cout << "Input count of numbers:\n";

int count = ReadNaturalNum();

UserMap<float> floatMap(count);

floatMap.PrintMap();

float average = floatMap.Average();

cout << "average = " << average << ";\n";

floatMap.Add(count, average);

floatMap.PrintMap();

cout << "----Money----\n";

cout << "Input count of numbers:\n";

int moneyCount = ReadNaturalNum();

UserMap<Money> MoneyMap(moneyCount);

MoneyMap.PrintMap();

Money maverage = MoneyMap.Average();

cout << "average = " << maverage << ";\n";

MoneyMap.Add(moneyCount, maverage);

MoneyMap.PrintMap();

return 0;

}

**Ответы на вопросы:**

**Что представляет собой ассоциативный контейнер?**

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

**Перечислить ассоциативные контейнеры библиотеки STL.**

Map, multimap (несколько элементов для одного ключа), set (множество, отсортирован), multiset (несколько элементов для одного ключа)

**Каким образом можно получить доступ к элементам ассоциативного контейнера?**

Ассоциативные контейнеры предоставляют стандартный доступ с помощью итераторов – прямых и обратных.

**Привести примеры методов, используемых в ассоциативных контейнерах.**

|  |  |
| --- | --- |
| **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) |

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

map<string, float> m1,

можно через typedef map<string, float> myMap myMap m2

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

Ассоциативный контейнер map требует, чтобы для типов ключа существовала операция “<” и он хранит свои элементы отсортированными по ключу (по убыванию).

Изменить порядок можно так:

map< string, int, greater<string> > myMap;

(по умолчанию тут lesser)

**Какие операции определены для контейнера map?**

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

**Чем отличаются контейнеры map и multimap?**

В multimap одному индексу может соответсвовать несколько значений.

**Что представляет собой контейнер set?**

В множестве хранятся объекты, упорядоченные по некоторому ключу, являющемуся атрибутом самого объекта.

**Чем отличаются контейнеры map и set?**

Для контейнера set индексы – аттрибут элемента, для map индекс – первый член пары.

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

set<int> set1; // создается пустое

множество int а[5] = { 1. 2. 3. 4, 5};

set<int> set2(a, а + 5);// инициализация копированием set<int>

set3(set2); // инициализация другим множеством

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

По умолчанию – по возрастанию.

Чтобы изменить порядок на обратный- set<int, greater<int>>

**Какие операции определены для контейнера set?**

**Написать функцию для добавления элементов в контейнер set.**

#include <iostream>

#include <set>

using namespace std;

typedef set<int>tset;

tset ::iterator i;

void main()

{

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

tset s(a,a+4);

s.insert(10);

s.insert(6);

for(i=s.begin();i!=s.end();i++) cout<<\*i<<” ”; cout<<endl;

}

**Написать функцию для печати контейнера set.**

#include <iostream>

#include <set>

using namespace std;

typedef set<int>tset;

tset ::iterator i;

void main()

{

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

tset s(a,a+4);

s.insert(10);

s.insert(6);

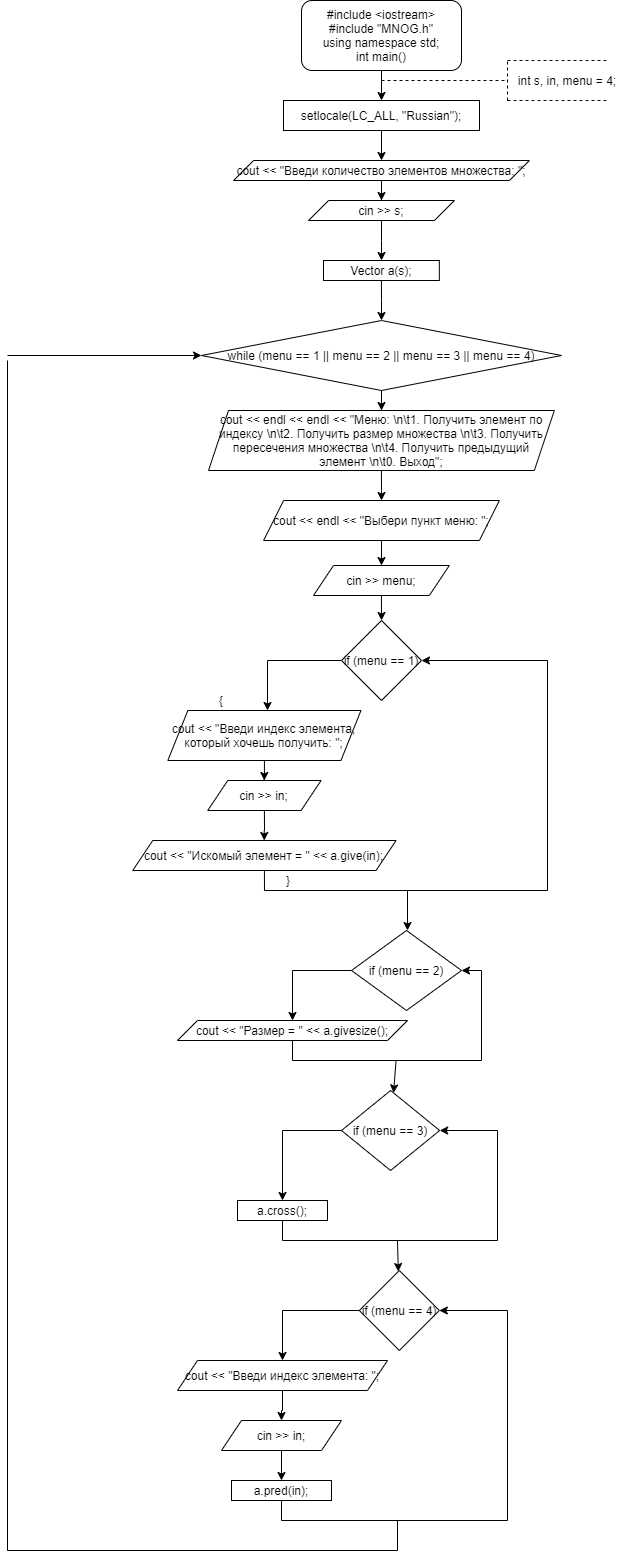
for(i=s.begin();i!=s.end();i++) cout<<\*i<<” ”; cout<<endl;

}

**Чем отличаются контейнеры set и multiset?**

Вотличии от set, в multiset ключи могут повторяться. Элементы с одинаковыми ключами хранятся в множестве в порядке их занесения.

Блок-схема



Vector.cpp:

