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 №18.13**

Дисциплина: «Информатика»

Тема: “ Объектно-ориентированное программирование.

Стандартные обобщённые алгоритмы библиотеки STL.”

Вариант 10

Выполнил:

Студент группы ИВТ-20-2Б Галинов О. Ю.

Проверила:

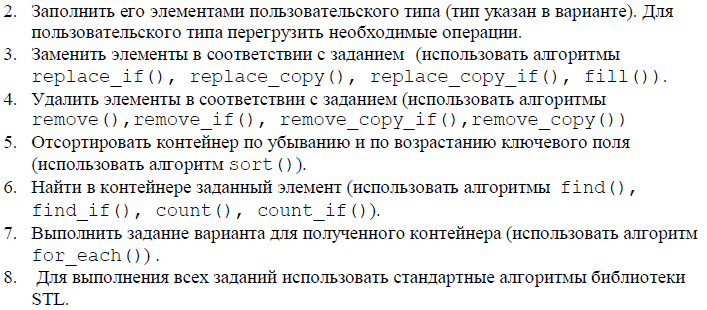
Доцент кафедры ИТАС

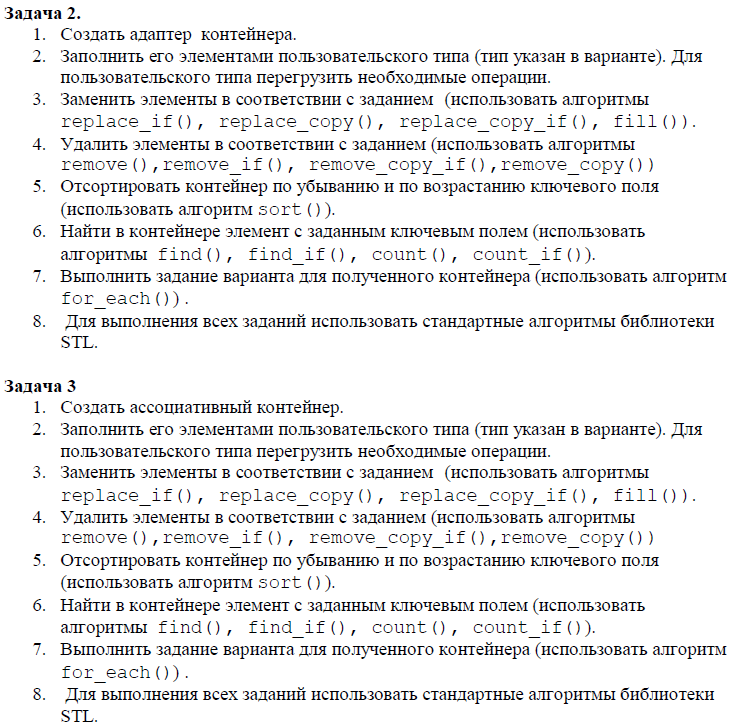
Полякова О.А.

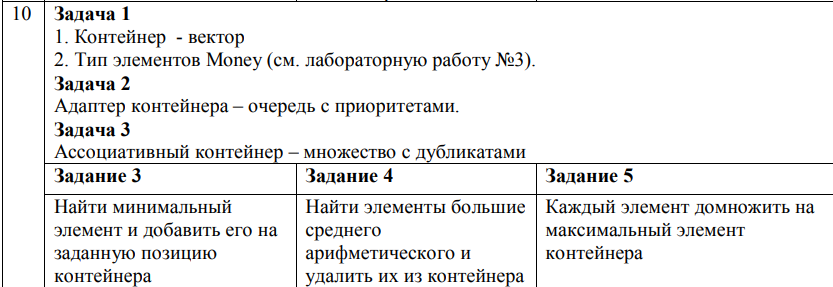
Пермь, 2021

**Постановка задачи**









**Анализ задачи**

**1.** Для решения задач необходимо:

**1.1.** Организовать функцию Task1(), в которой будет использоваться библиотека vector для создания вектора типа Money, для решения задачи №1.

**1.2.** Организовать функцию Task2(), в которой будет использоваться библиотека priority\_queue для создания очереди типа int, для решения задачи №2.

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

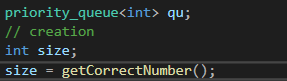
**1.4.** Организовать функцию Task3(), в которой будет использоваться библиотека multimap для создания словаря с дубликатами типа Money, для решения задачи №3.

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

**2.1.** Функция Task1(), в которой будет использоваться библиотека vector для создания вектора типа Money, для решения задачи №1. Переменная size устанавливает размер двунаправленной очереди.

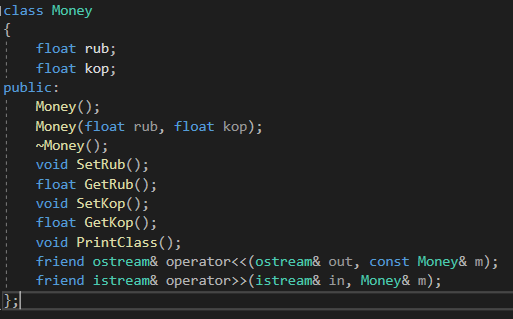


**2.2.** Функция Task2(), в которой будет использоваться библиотека priority\_queue для создания очереди типа int, для решения задачи №2. Переменная size устанавливает размер очереди.



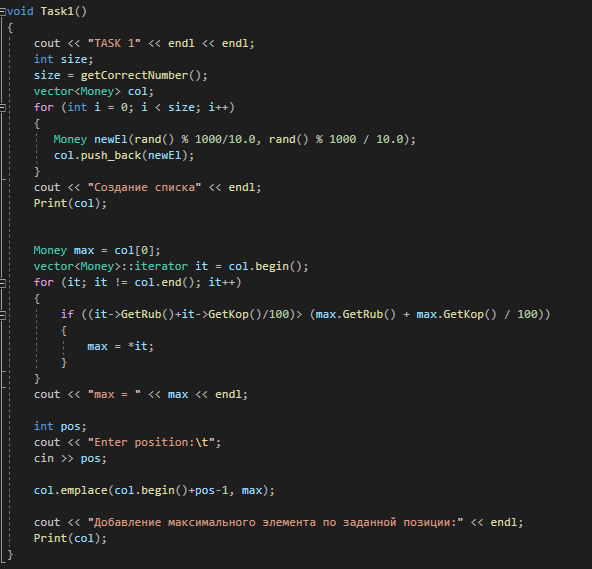
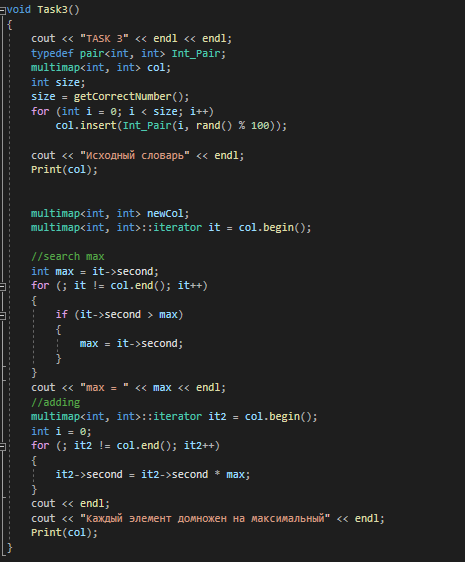
**2.3.** Функция Task3(), в которой будет использоваться, в которой будет использоваться библиотека map для создания словаря с дубликатами (multimap) - col, типа int для ключа и типа int для хранящихся данных, для решения задачи №3. Переменная size устанавливает размер контейнера.





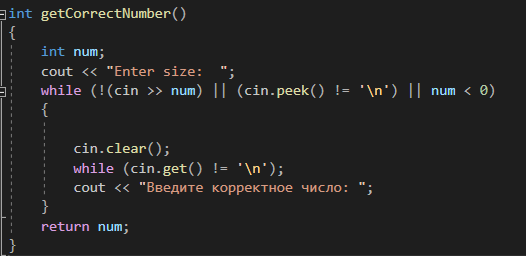
**3.** Для решения задачи данные были представлены в следующем виде:

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

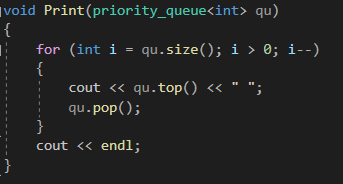
  

**4.** Для операций ввода и вывода использовались следующие операторы и функции:

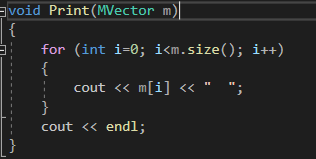
**4.1.** Для ввода целочисленных значений используется функция getCorrectNumber (), которая проверяет введённое данное. Таким образом будут игнорироваться данные типа char, double, float.



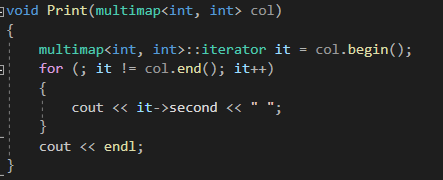
**4.2.** Для печати очереди типа int используется функция Print().



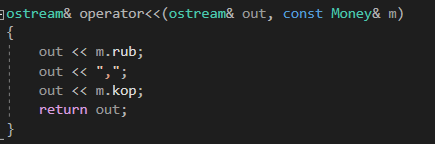
**4.3.** Для печати вектора типа Money используется функция Print().



**4.4.** Для печати словаря с дубликатами типа int используется функция Print().

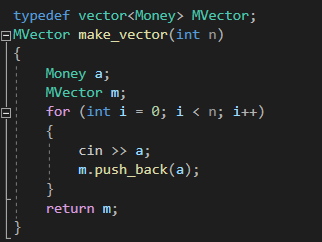


**4.5.** Для печати объектов класса Money используется перегрузка оператора сдвига.

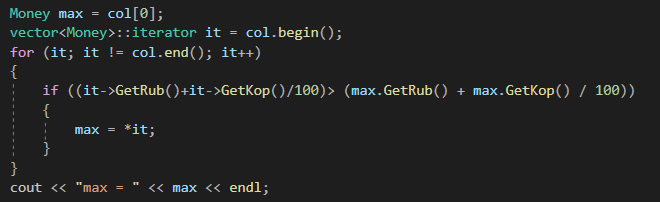


**5.** Поставленные задачи будут решены следующими действиями:

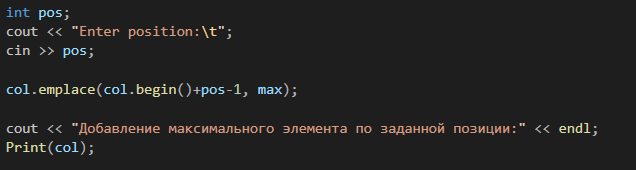
**5.1.** Для формирования вектора типа Money в функции Task1() используется функция Make\_vector(). Используется цикл for, который итерируется по всему контейнеру и элементы добавляются через метод push\_back, прописанный в библиотеке vector.



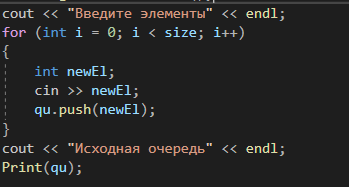
**5.2.** Для поиска максимального элемента используется цикл, в котором с помощью условия происходит поиск максимального элемента.



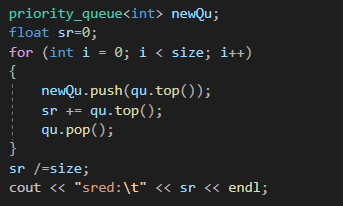
Далее данный элемент вставляется на заданную позицию вектора с помощью метода emplace, описанного в библиотеке vector.



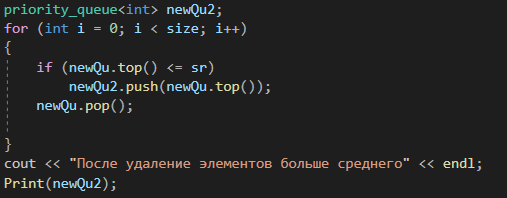
**5.3.** Для формирования очереди типа int в функции Task2() используется цикл for, который итерируется по всему контейнеру и элементы добавляются через метод push, прописанный в библиотеке queue.



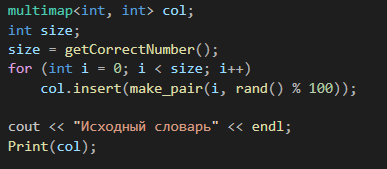
Далее создается еще одна очередь, в которую записываются все элементы из прошлой, а также считается среднее арифметическое очереди.



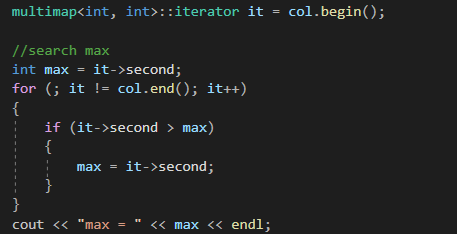
**5.4.** Для удаления элементов больше или равных среднему арифметическому создается еще одна очередь, в которую записываются все элементы, которые подходят под условие.



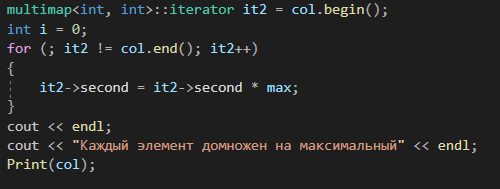
**5.5.** Для формирования контейнера multimap типа Money в функции Task3() используется функция цикл, где каждый элемент генерируется случайно (0-100).

****

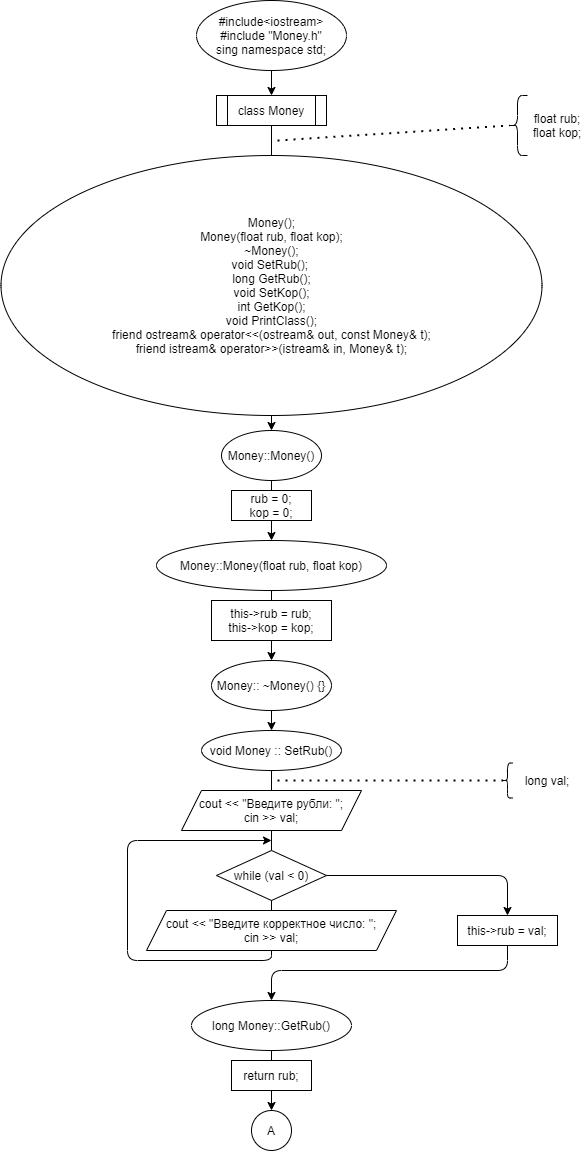
**5.6.** Для поиска максимального элемента используется цикл, в котором при помощи итератора происходит проход по всему контейнеру и каждый элемент сравнивается с максимумом (который изначально равен первому элементу).

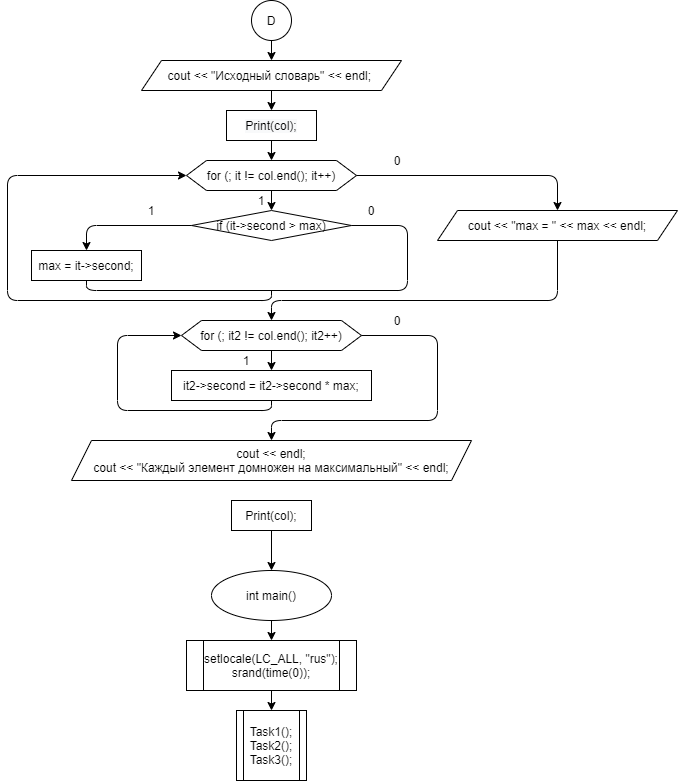
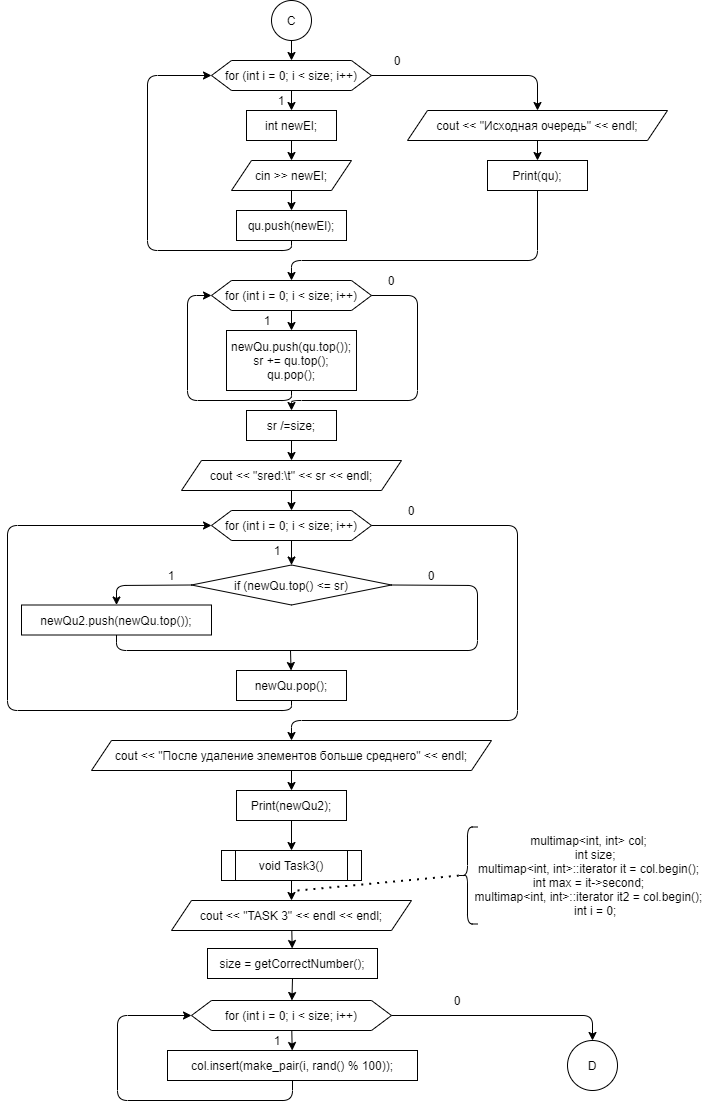
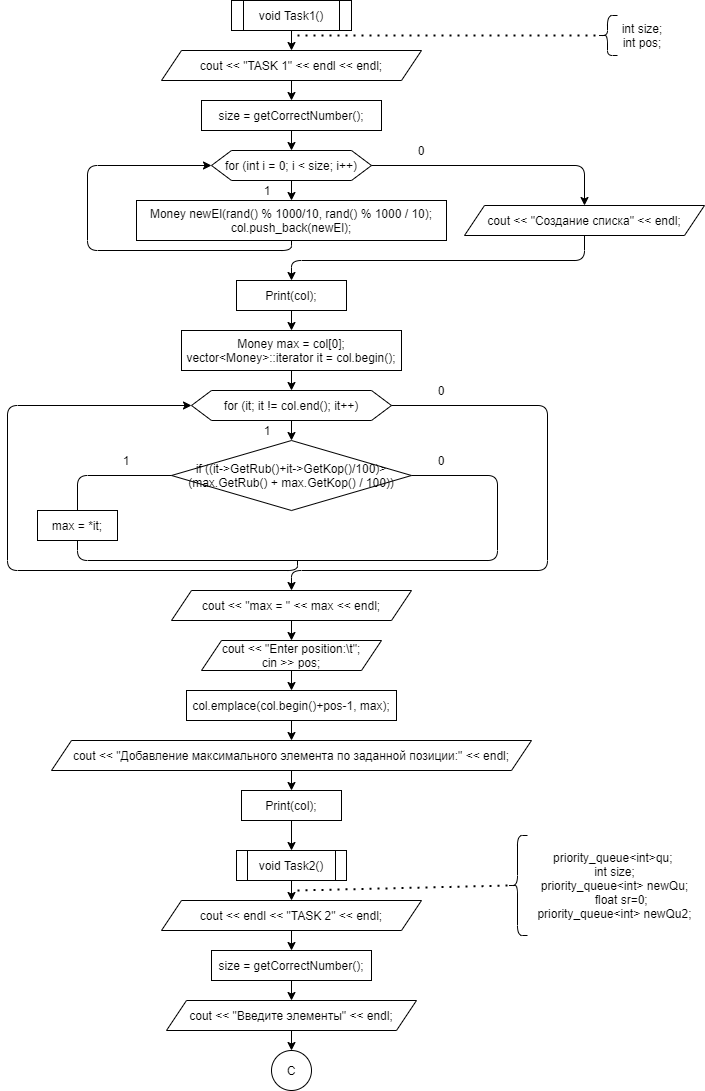
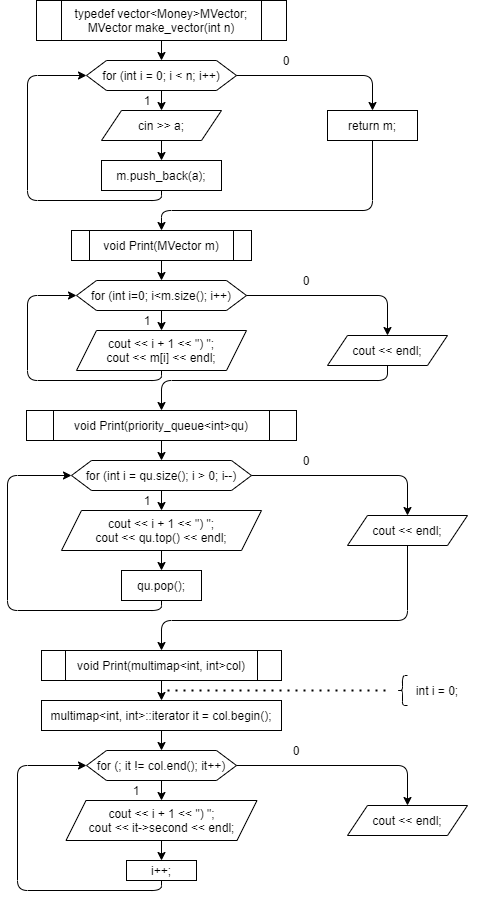
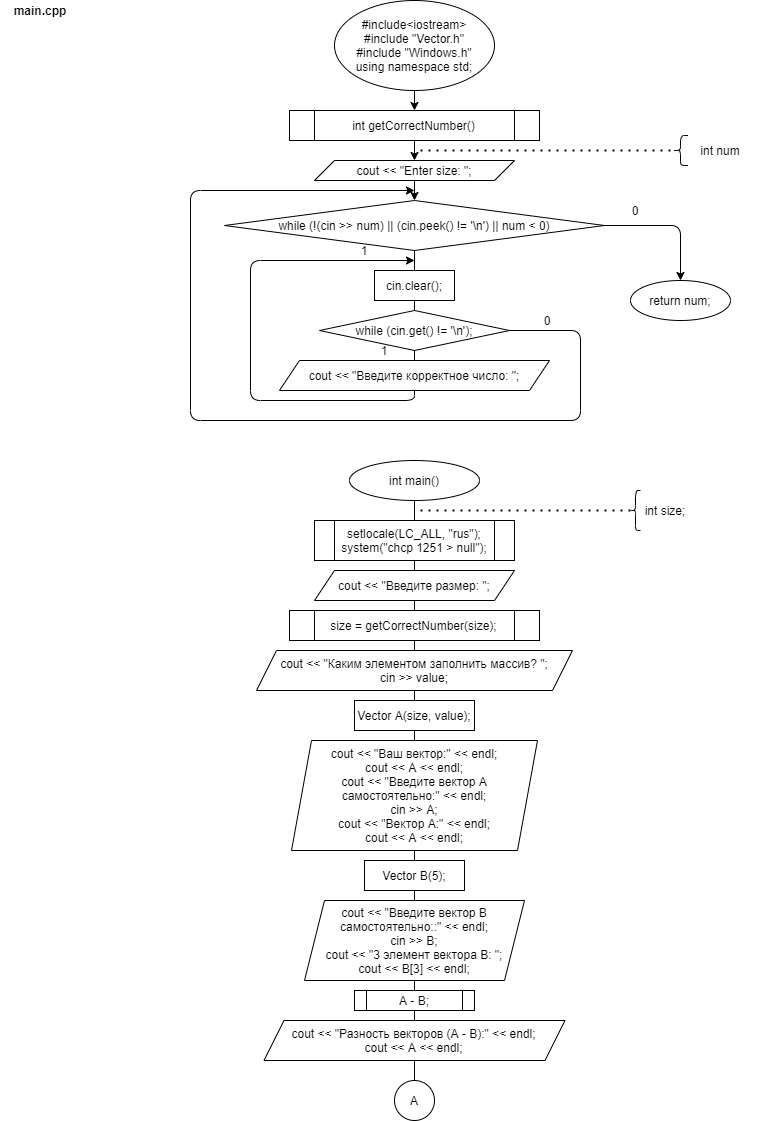
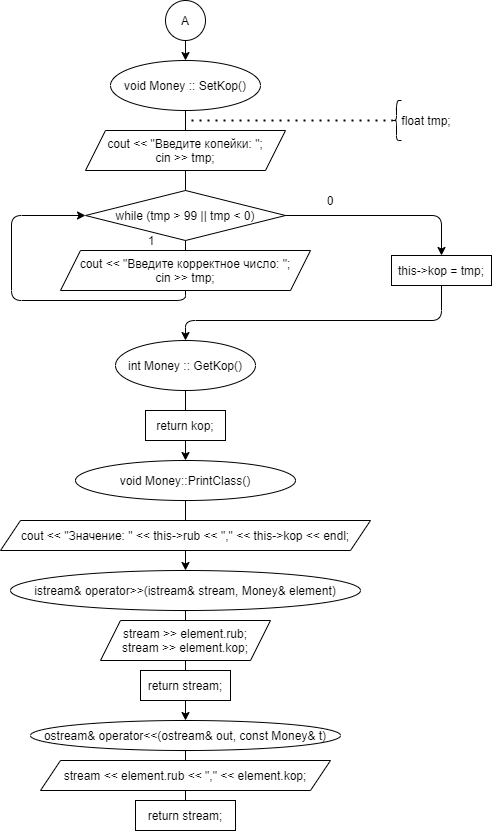


Далее все элементы контейнера умножаются на максимальный.



**Блок-схема**





**Код**

#include <iostream>

#include <iterator>

#include <vector>

#include "Money.h"

#include <map>

#include <queue>

#include<algorithm>

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;

}

typedef vector<Money> MVector;

MVector make\_vector(int n)

{

Money a;

MVector m;

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

{

cin >> a;

m.push\_back(a);

}

return m;

}

void Print(MVector m)

{

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

{

cout << i + 1 << ") ";

cout << m[i] << endl;

}

cout << endl;

}

void Print(priority\_queue<int> qu)

{

for (int i = qu.size(); i > 0; i--)

{

cout << i + 1 << ") ";

cout << qu.top() << endl;

qu.pop();

}

cout << endl;

}

void Print(multimap<int, int> col)

{

int i = 0;

multimap<int, int>::iterator it = col.begin();

for (; it != col.end(); it++)

{

cout << i + 1 << ") ";

cout << it->second << endl;

i++;

}

cout << endl;

}

void Task1()

{

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

int size;

size = getCorrectNumber();

vector<Money> col;

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

{

Money newEl(rand() % 1000/10, rand() % 1000 / 10);

col.push\_back(newEl);

}

cout << "Создание списка" << endl;

Print(col);

Money max = col[0];

vector<Money>::iterator it = col.begin();

for (it; it != col.end(); it++)

{

if ((it->GetRub()+it->GetKop()/100)> (max.GetRub() + max.GetKop() / 100))

{

max = \*it;

}

}

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

int pos;

cout << "Enter position:\t";

cin >> pos;

col.emplace(col.begin()+pos-1, max);

cout << "Добавление максимального элемента по заданной позиции:" << endl;

Print(col);

}

void Task2()

{

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

priority\_queue<int> qu;

// creation

int size;

size = getCorrectNumber();

cout << "Введите элементы" << endl;

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

{

int newEl;

cin >> newEl;

qu.push(newEl);

}

cout << "Исходная очередь" << endl;

Print(qu);

priority\_queue<int> newQu;

float sr=0;

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

{

newQu.push(qu.top());

sr += qu.top();

qu.pop();

}

sr /=size;

cout << "sred:\t" << sr << endl;

priority\_queue<int> newQu2;

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

{

if (newQu.top() <= sr)

newQu2.push(newQu.top());

newQu.pop();

}

cout << "После удаление элементов больше среднего" << endl;

Print(newQu2);

}

void Task3()

{

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

multimap<int, int> col;

int size;

size = getCorrectNumber();

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

col.insert(make\_pair(i, rand() % 100));

cout << "Исходный словарь" << endl;

Print(col);

multimap<int, int>::iterator it = col.begin();

//search max

int max = it->second;

for (; it != col.end(); it++)

{

if (it->second > max)

{

max = it->second;

}

}

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

//adding

multimap<int, int>::iterator it2 = col.begin();

int i = 0;

for (; it2 != col.end(); it2++)

{

it2->second = it2->second \* max;

}

cout << endl;

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

Print(col);

}

int main()

{

setlocale(LC\_ALL, "rus");

srand(time(0));

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();

float GetRub();

void SetKop();

float GetKop();

void PrintClass();

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

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

};

#include "Money.h"

#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;

}

float 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;

}

float 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;

}

**Скриншоты**