Министерство науки и высшего образования Российской Федерации  
Федеральное государственное автономное учреждение высшего образования  
«Пермский национальный исследовательский политехнический университет»

Электротехнический факультет  
Кафедра «Информационные технологии и автоматизированные системы»

**О Т Ч Ё Т  
по лабораторной работе**

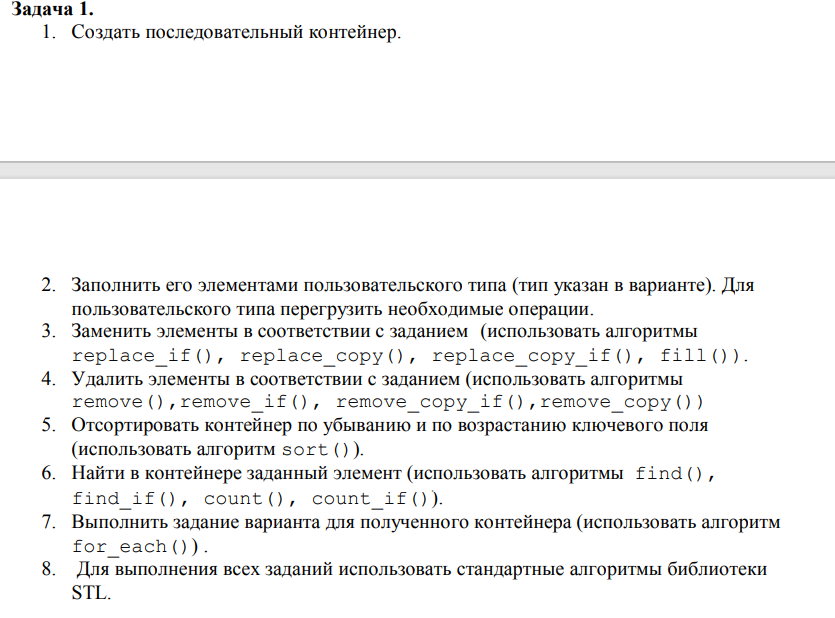
Дисциплина: «Основы алгоритмизации и программирования»

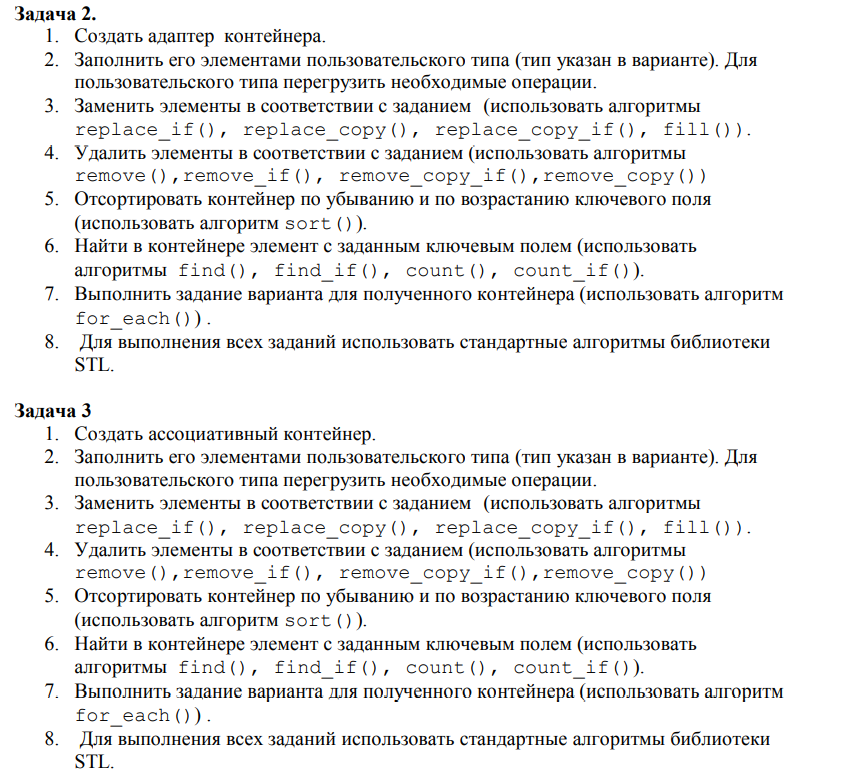
Выполнил  
студент группы ИВТ-21-1б  
Мифтахов И.Г

Проверил  
ст. п. кафедры ИТАС  
Яруллин Д.В.

Пермь 2022

Постановка задачи:





Анализ задачи 1:

1. long rub, int kop для обозначения рублей и копеек соответственно, long buf для более удобной перезаписи переменных, Money s – глобальная переменная для сравнения с заданным значением(в предикате)

2. В файле money.h объявлены все методы, конструкторы, деструктор, перегрузки операторов. В файле money.cpp они реализованы стандартными способами, как в предыдущих лабораторных работах.

3. В ф-ии main() задан предикат с помощью структуры, реализован метод создания вектора, его печати, добавления элементов, нахождение среднего, ф-ия умножения. Для работы с вектором используем итератор it. Операции выполняем с помощью стандартных методов контейнера vector.

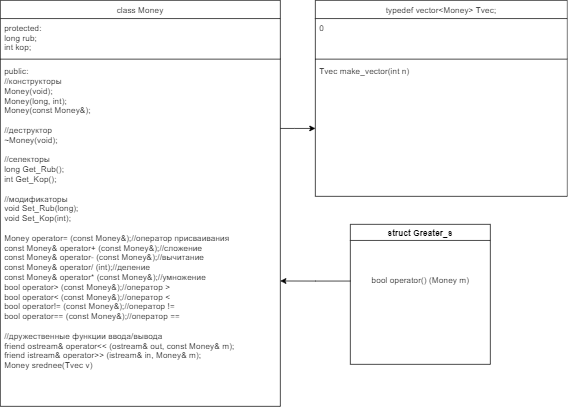
**Анализ задачи 2:**

Все аналогично предыдущей задаче, только вместо вектора использован стек и очередь, и ф-ии, где происходит их копирование в друг друга(адаптеры).

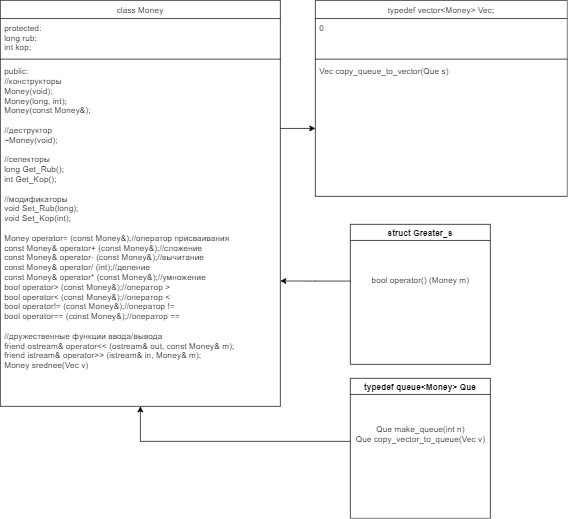
**Анализ задачи 3:**

Опять же, все аналогично, только теперь вместо стека и очереди использован мультисет(бинарное дерево, в котором элементы могут повторяться(дубликаты)). Реализованы соответствующие методы для вывода дерева и его печати. А также добавление.

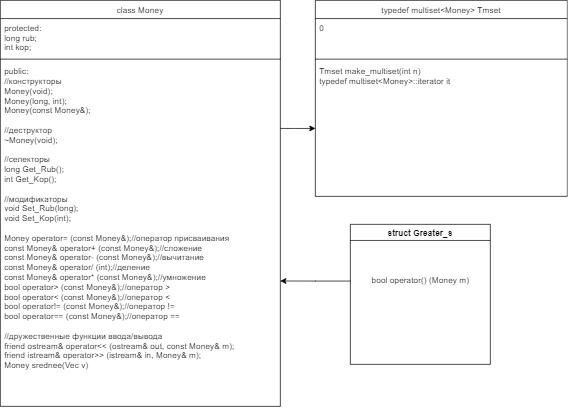
Блок-схема 1:



Блок-схема 2:



Блок-схема 3:



Код программы 1:

#include "Money.h"

#include <vector>

#include <iostream>

#include <algorithm>

using namespace std;

typedef vector<Money> Tvec;

Money s;//добавим глобальную переменную Money для сравнения с заданным значением

//добавим предикат для сравнения заданного значения с объектом Money

struct Greater\_s

{

bool operator() (Money m)

{

if (m > s) return true;

else return false;

}

};

//формирование вектора

Tvec make\_vector(int n)

{

Money a;

Tvec v;

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

{

cin >> a;

v.push\_back(a);

}

return v;

}

//печать вектора

void print\_vector(Tvec v)

{

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

cout << v[i] << " ";

cout << "\n";

}

//поиск среднего

Money srednee(Tvec v)

{

Money sum = v[0];

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

sum = sum + v[i];

return sum / v.size();

}

//функция добавления

void mult(Money& m)

{

m = m \* s;

}

int main()

{

int n;

cout << "Enter n >> ";

cin >> n;

Tvec v = make\_vector(n);

print\_vector(v);

int pos;

cout << "Enter pos for add >> ";

cin >> pos;

Tvec::iterator i;//создали итератор

cout << "Add min: ";

i = min\_element(v.begin(), v.end());

Money min = (\*i);//нашли макс элемент

int count = 0;

for (i = v.begin(); i != v.end(); i++)

{

count++;

if (count == pos) break;

}

v.insert(i, min);//добавили макс элемент в вектор

print\_vector(v);

s = srednee(v);

cout << "Average: " << s << "\n";

cout << "Del more average: ";

v.erase(remove\_if(v.begin(), v.end(), Greater\_s()), v.end());//удалили

print\_vector(v);

cout << "Multiple average: ";

s = srednee(v);

for\_each(v.begin(), v.end(), mult);

print\_vector(v);

return 0;

}

#include "Money.h"

#include <iostream>

using namespace std;

//конструкторы

Money::Money(void)

{

rub = 0;

kop = 0;

}

Money::Money(long R, int K)

{

long buf;

buf = R \* 100 + K;

rub = buf / 100;

kop = buf % 100;

}

Money::Money(const Money& m)

{

rub = m.rub;

kop = m.kop;

}

//деструктор

Money::~Money(void)

{

}

//селекторы

long Money::Get\_Rub()

{

return rub;

}

int Money::Get\_Kop()

{

return kop;

}

//модификаторы

void Money::Set\_Rub(long Rub)

{

rub = Rub;

}

void Money::Set\_Kop(int Kop)

{

long buf;

buf = rub \* 100 + Kop;

rub = buf / 100;

kop = buf % 100;

}

//перегрузка оператора присваивания

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

{

if (&m == this) return \*this;

rub = m.rub;

kop = m.kop;

return \*this;

}

//операция сложения

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

{

long buf;

buf = rub \* 100 + kop + m.rub \* 100 + m.kop;

rub = buf / 100;

kop = buf % 100;

return \*this;

}

//операция вычитания

const Money& Money::operator- (const Money& m)

{

long buf;

buf = rub \* 100 + kop - m.rub \* 100 - m.kop;

rub = buf / 100;

kop = buf % 100;

return \*this;

}

//операция деления

const Money& Money::operator/ (int m)

{

long buf;

buf = (rub \* 100 + kop) / m;

rub = buf / 100;

kop = buf % 100;

return \*this;

}

//операция умножения

const Money& Money::operator\* (const Money& m)

{

long buf;

buf = (rub \* 100 + kop) \* (m.rub \* 100 + m.kop);

rub = buf / 100;

kop = buf % 100;

return \*this;

}

//операторы ввода/вывода

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

{

out << m.rub << "," << m.kop;

return out;

}

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

{

long buf, r;

int k;

cout << "Enter rub >> "; in >> r;

cout << "Enter kop >> "; in >> k;

cout << "\n";

buf = r \* 100 + k;

m.rub = buf / 100;

m.kop = buf % 100;

return in;

}

//оператор >

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

{

if (rub > m.rub) return true;

if (rub == m.rub && kop > m.kop) return true;

return false;

}

//оператор <

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

{

if (rub < m.rub) return true;

if (rub == m.rub && kop < m.kop) return true;

return false;

}

//оператор !=

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

{

if (rub != m.rub && kop != m.kop) return true;

return false;

}

//оператор ==

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

{

if (rub == m.rub && kop == m.kop) return true;

return false;

}

#pragma once

#include <iostream>

using namespace std;

class Money

{

protected:

long rub;

int kop;

public:

//конструкторы

Money(void);

Money(long, int);

Money(const Money&);

//деструктор

~Money(void);

//селекторы

long Get\_Rub();

int Get\_Kop();

//модификаторы

void Set\_Rub(long);

void Set\_Kop(int);

Money operator= (const Money&);//оператор присваивания

const Money& operator+ (const Money&);//сложение

const Money& operator- (const Money&);//вычитание

const Money& operator/ (int);//деление

const Money& operator\* (const Money&);//умножение

bool operator> (const Money&);//оператор >

bool operator< (const Money&);//оператор <

bool operator!= (const Money&);//оператор !=

bool operator== (const Money&);//оператор ==

//дружественные функции ввода/вывода

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

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

};

Код программы 2:

#include "Money.h"

#include <queue>

#include <vector>

#include <iostream>

#include <algorithm>

using namespace std;

typedef queue<Money> Que;//определили стек

typedef vector<Money> Vec;//определили вектор

Money a;

//добавим предикат для сравнения заданного значения с объектом Money

struct Greater\_s

{

bool operator() (Money m)

{

if (m > a) return true;

else return false;

}

};

//ф-ия для формирования стека

Que make\_queue(int n)

{

Que s;

Money m;

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

{

cin >> m;

s.push(m);

}

return s;

}

//копируем стек в вектор

Vec copy\_queue\_to\_vector(Que s)

{

Vec v;

while (!s.empty())

{

v.push\_back(s.front());

s.pop();

}

return v;

}

//копируем вектор в стек

Que copy\_vector\_to\_queue(Vec v)

{

Que s;

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

s.push(v[i]);

return s;

}

//вывод стека

void print\_queue(Que s)

{

Vec temp;

temp = copy\_queue\_to\_vector(s);//копируем стек в вектор

while (!s.empty())

{

cout << s.front() << " ";

s.pop();

}

s = copy\_vector\_to\_queue(temp);//копируем вектор в стек

cout << "\n";

}

//поиск среднего

Money srednee(Vec v)

{

Money sum = v[0];

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

sum = sum + v[i];

return sum / v.size();

}

//добавление

void multi(Money& m)

{

m = m \* a;

}

int main()

{

int n;

cout << "Enter n >> ";

cin >> n;

Que s = make\_queue(n);

print\_queue(s);

int pos;

cout << "Enter pos for add >> ";

cin >> pos;

cout << "Add min: ";

Vec temp = copy\_queue\_to\_vector(s);

Vec::iterator i = min\_element(temp.begin(), temp.end());

Money min = (\*i);

int count = 0;

for (i = temp.begin(); i < temp.end(); i++)

{

count++;

if (count == pos) break;

}

temp.insert(i, min);

s = copy\_vector\_to\_queue(temp);

print\_queue(s);

a = srednee(temp);

cout << "Average: " << a << "\n";

cout << "Del more average: ";

temp = copy\_queue\_to\_vector(s);

temp.erase(remove\_if(temp.begin(), temp.end(), Greater\_s()), temp.end());

s = copy\_vector\_to\_queue(temp);

print\_queue(s);

cout << "Multiple average: ";

temp = copy\_queue\_to\_vector(s);

for\_each(temp.begin(), temp.end(), multi);

s = copy\_vector\_to\_queue(temp);

print\_queue(s);

return 0;

}

#include "Money.h"

#include <iostream>

using namespace std;

//конструкторы

Money::Money(void)

{

rub = 0;

kop = 0;

}

Money::Money(long R, int K)

{

long buf;

buf = R \* 100 + K;

rub = buf / 100;

kop = buf % 100;

}

Money::Money(const Money& m)

{

rub = m.rub;

kop = m.kop;

}

//деструктор

Money::~Money(void)

{

}

//селекторы

long Money::Get\_Rub()

{

return rub;

}

int Money::Get\_Kop()

{

return kop;

}

//модификаторы

void Money::Set\_Rub(long Rub)

{

rub = Rub;

}

void Money::Set\_Kop(int Kop)

{

long buf;

buf = rub \* 100 + Kop;

rub = buf / 100;

kop = buf % 100;

}

//перегрузка оператора присваивания

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

{

if (&m == this) return \*this;

rub = m.rub;

kop = m.kop;

return \*this;

}

//операция сложения

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

{

long buf;

buf = rub \* 100 + kop + m.rub \* 100 + m.kop;

rub = buf / 100;

kop = buf % 100;

return \*this;

}

//операция вычитания

const Money& Money::operator- (const Money& m)

{

long buf;

buf = rub \* 100 + kop - m.rub \* 100 - m.kop;

rub = buf / 100;

kop = buf % 100;

return \*this;

}

//операция деления

const Money& Money::operator/ (int m)

{

long buf;

buf = (rub \* 100 + kop) / m;

rub = buf / 100;

kop = buf % 100;

return \*this;

}

//операция умножения

const Money& Money::operator\* (const Money& m)

{

long buf;

buf = (rub \* 100 + kop) \* (m.rub \* 100 + m.kop);

rub = buf / 100;

kop = buf % 100;

return \*this;

}

//операторы ввода/вывода

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

{

out << m.rub << "," << m.kop;

return out;

}

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

{

long buf, r;

int k;

cout << "Enter rub >> "; in >> r;

cout << "Enter kop >> "; in >> k;

cout << "\n";

buf = r \* 100 + k;

m.rub = buf / 100;

m.kop = buf % 100;

return in;

}

//оператор >

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

{

if (rub > m.rub) return true;

if (rub == m.rub && kop > m.kop) return true;

return false;

}

//оператор <

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

{

if (rub < m.rub) return true;

if (rub == m.rub && kop < m.kop) return true;

return false;

}

//оператор !=

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

{

if (rub != m.rub && kop != m.kop) return true;

return false;

}

//оператор ==

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

{

if (rub == m.rub && kop == m.kop) return true;

return false;

}

#pragma once

#include <iostream>

using namespace std;

class Money

{

protected:

long rub;

int kop;

public:

//конструкторы

Money(void);

Money(long, int);

Money(const Money&);

//деструктор

~Money(void);

//селекторы

long Get\_Rub();

int Get\_Kop();

//модификаторы

void Set\_Rub(long);

void Set\_Kop(int);

Money operator= (const Money&);//оператор присваивания

const Money& operator+ (const Money&);//сложение

const Money& operator- (const Money&);//вычитание

const Money& operator/ (int);//деление

const Money& operator\* (const Money&);//умножение

bool operator> (const Money&);//оператор >

bool operator< (const Money&);//оператор <

bool operator!= (const Money&);//оператор !=

bool operator== (const Money&);//оператор ==

//дружественные функции ввода/вывода

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

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

};

Код программы 3:

#include "Money.h"

#include <set>

#include <iostream>

#include <algorithm>

using namespace std;

typedef multiset<Money> Tmset;//определяем тип работы с мультисет

typedef multiset<Money>::iterator it;//итератор

Money s;

//добавим предикат для сравнения заданного значения с объектом Money

struct Greater\_s

{

bool operator() (Money m)

{

if (m > s) return true;

else return false;

}

};

//создание мн-ва

Tmset make\_multiset(int n)

{

Tmset m;

Money a;

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

{

cin >> a;

m.insert(a);

}

return m;

}

//печать мн-ва

void print\_multiset(Tmset m)

{

it i = m.begin();

while (i != m.end())

{

cout << \*i;

i++;

}

cout << "\n";

}

//среднее

Money srednee(Tmset m)

{

it i = m.begin();

Money sum;//сумма элементов

int n = 1;//количество элементов

while (i != m.end())

{

sum = sum + (\*i);

n++;

i++;

}

return sum / n;

}

//умножение

void multi(Money& m)

{

m = m \* s;

}

bool operator< (const Money& m1, const Money& m2)

{

return (m1 < m2);

}

int main()

{

int n;//размер

cout << "Enter n >> ";

cin >> n;

Tmset m = make\_multiset(n);//создаём мн-во

print\_multiset(m);

cout << "Add min: ";//добавляем мин элемент

//так как это мн-во, то данные упорядочены, в определённое место вставить элемент не получится

//поэтому нам достаточно просто добавить в начало минимальный элемент

it i = m.begin();

Money min = (\*i);

m.insert(min);

print\_multiset(m);

s = srednee(m);//удалить больше среднего

cout << "Average: " << s << "\n";

cout << "Del more average: ";

m.erase(remove\_if(m.begin(), m.end(), Greater\_s()), m.end());

print\_multiset(m);

cout << "Multiple srednee: ";//умножить на среднее

for\_each(m.begin(), m.end(), multi);

print\_multiset(m);

return 0;

}

#include "Money.h"

#include <iostream>

using namespace std;

//конструкторы

Money::Money(void)

{

rub = 0;

kop = 0;

}

Money::Money(long R, int K)

{

long buf;

buf = R \* 100 + K;

rub = buf / 100;

kop = buf % 100;

}

Money::Money(const Money& m)

{

rub = m.rub;

kop = m.kop;

}

//деструктор

Money::~Money(void)

{

}

//селекторы

long Money::Get\_Rub()

{

return rub;

}

int Money::Get\_Kop()

{

return kop;

}

//модификаторы

void Money::Set\_Rub(long Rub)

{

rub = Rub;

}

void Money::Set\_Kop(int Kop)

{

long buf;

buf = rub \* 100 + Kop;

rub = buf / 100;

kop = buf % 100;

}

//перегрузка оператора присваивания

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

{

if (&m == this) return \*this;

rub = m.rub;

kop = m.kop;

return \*this;

}

//операция сложения

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

{

long buf;

buf = rub \* 100 + kop + m.rub \* 100 + m.kop;

rub = buf / 100;

kop = buf % 100;

return \*this;

}

//операция вычитания

const Money& Money::operator- (const Money& m)

{

long buf;

buf = rub \* 100 + kop - m.rub \* 100 - m.kop;

rub = buf / 100;

kop = buf % 100;

return \*this;

}

//операция деления

const Money& Money::operator/ (int m)

{

long buf;

buf = (rub \* 100 + kop) / m;

rub = buf / 100;

kop = buf % 100;

return \*this;

}

//операция умножения

const Money& Money::operator\* (const Money& m)

{

long buf;

buf = (rub \* 100 + kop) \* (m.rub \* 100 + m.kop);

rub = buf / 100;

kop = buf % 100;

return \*this;

}

//операторы ввода/вывода

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

{

out << m.rub << "," << m.kop;

return out;

}

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

{

long buf, r;

int k;

cout << "Enter rub >> "; in >> r;

cout << "Enter kop >> "; in >> k;

cout << "\n";

buf = r \* 100 + k;

m.rub = buf / 100;

m.kop = buf % 100;

return in;

}

//оператор >

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

{

if (rub > m.rub) return true;

if (rub == m.rub && kop > m.kop) return true;

return false;

}

//оператор <

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

{

if (rub < m.rub) return true;

if (rub == m.rub && kop < m.kop) return true;

return false;

}

//оператор !=

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

{

if (rub != m.rub && kop != m.kop) return true;

return false;

}

//оператор ==

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

{

if (rub == m.rub && kop == m.kop) return true;

return false;

}

#pragma once

#include <iostream>

using namespace std;

class Money

{

protected:

long rub;

int kop;

public:

//конструкторы

Money(void);

Money(long, int);

Money(const Money&);

//деструктор

~Money(void);

//селекторы

long Get\_Rub();

int Get\_Kop();

//модификаторы

void Set\_Rub(long);

void Set\_Kop(int);

Money operator= (const Money&);//оператор присваивания

const Money& operator+ (const Money&);//сложение

const Money& operator- (const Money&);//вычитание

const Money& operator/ (int);//деление

const Money& operator\* (const Money&);//умножение

bool operator> (const Money&);//оператор >

bool operator< (const Money&);//оператор <

bool operator!= (const Money&);//оператор !=

bool operator== (const Money&);//оператор ==

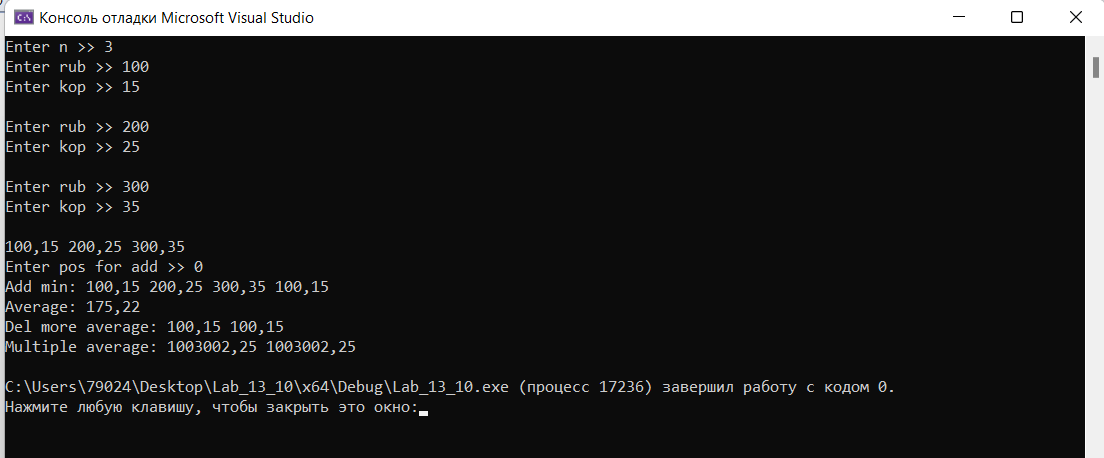
//дружественные функции ввода/вывода

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

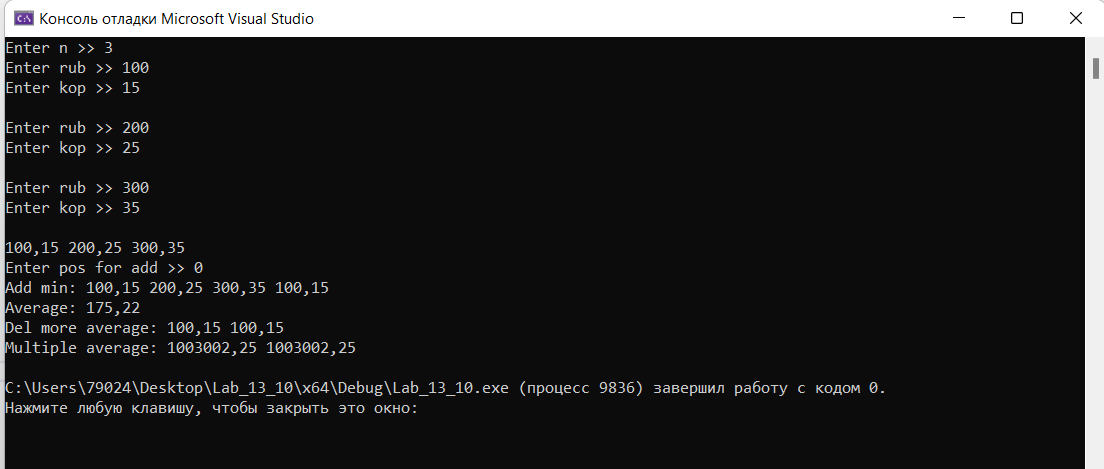
friend istream& operator>> (istream& in, Money& m);

};

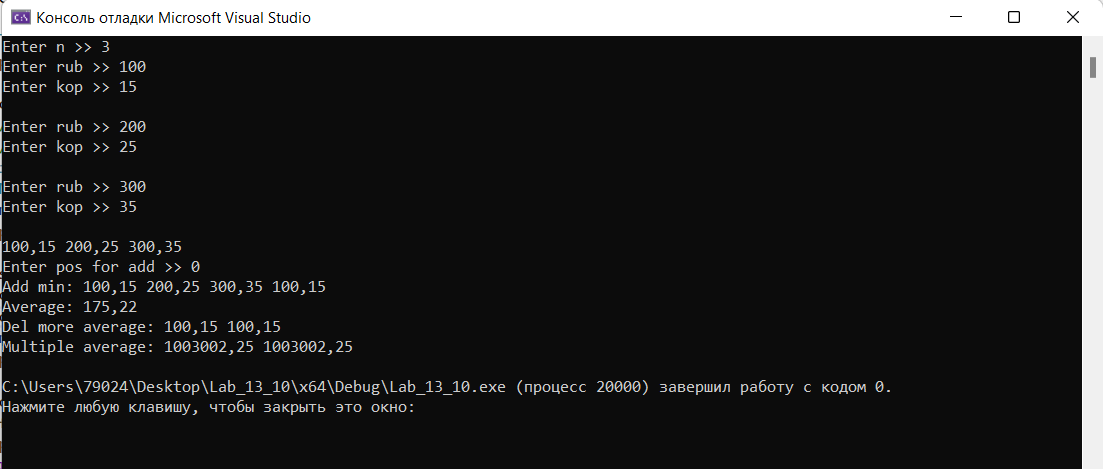
Скриншоты результатов 1:



Скриншоты результатов 2:



Скриншоты результатов 3:



Анализ результатов :

Программа работает корректно. Выводятся результаты без ошибок, все реализовано в соответствие с задачей.