Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Пермский национальный исследовательский политехнический университет»
Электротехнический факультет
Кафедра «Информационные технологии и автоматизированные системы»

ЛАБОРАТОРНАЯ РАБОТА №13

«Стандартные обобщенные алгоритмы библиотеки STL»

Выполнила:
студентка группы ИВТ-23-2б
Соловьева Екатерина
Александровна
Проверила:
доцент кафедры ИТАС
О. А. Полякова

2024 г.

Постановка задачи

11 Вариант:

Задача 1 1. Контейнер - вектор 2. Тип элементов Money (см. лабораторную работу №3). Задача 2 Адаптер контейнера - очередь. Задача 3 Ассоциативный контейнер - словарь		
Задание 3	Задание 4	Задание 5
Найти среднее арифметическое и добавить его в начало контейнера	Найти элемент с заданным ключом и удалить их из контейнера	Из каждого элемента вычесть минимальный элемент контейнера

Код программы на C++

Money.h

```
#pragma once
#include <iostream>

using namespace std;

class Money {
	long int rub;
	int kop;
public:
	Money() { rub = 0; kop = 0; };
	Money(long int r, int k) { rub = r; kop = k; };
	Money(const Money& temp) { rub = temp.rub; kop = temp.kop; };
	~Money() {};

	int getRub() { return rub; };
	int getKop() { return kop; };

	void setRub(long int r) { rub = r; }
	void setKop(int k) { kop = k; }

	Money& operator =(const Money&);
	Money& operator ++();
	Money operator / (const Money&);
	Money operator / (const int&);
	Money operator ++(int);
	Money& operator -(const Money&);
	Money operator +(const Money&);
	bool operator > (const Money&);
	bool operator < (const Money&);
	bool operator ==(Money&);

	friend istream& operator >>(istream& in, Money& temp);
	friend ostream& operator <<(ostream& out, const Money& temp);
};
```

Method1.h

```
#pragma once
#include "Money.h"
#include <vector>
#include <iostream>

using namespace std;

typedef vector<Money>vec;
vec makeVector(int size) {
	vec v;
	for (int i = 1; i <= size; i++) {
		Money tmp(rand() % 100, rand() % 100);
		v.push_back(tmp);
	}
	return v;
}

void printVector(vec v) {
	for (int i = 0; i < v.size(); i++) {
		cout << v[i] << " ";
	}
	cout << endl;
}

int average(vec v) {
	int sum = 0;
	for (int i = 0; i < v.size(); i++) {
		sum += (v[i].getRub() + v[i].getKop());
	}
	int n = v.size();
	cout << "Average: " << sum / n << endl;
	return sum / n;
}

void minElem(vec v) {
	Money min = v[0];
	for (int i = 0; i < v.size(); i++) {

		if (min > v[i]) { min = v[i]; }
	}
	cout << "Min element = " << min << endl;

	for (int i = 0; i < v.size(); i++) {
		v[i] = v[i] - min;
	}
	printVector(v);
}
```

Method2.h

```
#pragma once
#include "Money.h"
#include <queue>
#include <vector>
#include <cmath>

using namespace std;

typedef queue<Money> q;
typedef vector<Money> v;

q makeQueue(int size) {
	q queue;
```

```
	Money m;
	for (int i = 0; i < size; i++) {
		cin >> m;
		queue.push(m);
	}
	return queue;
}

v copyQueue(q queue) {
	v vector;
	while (!queue.empty()) {
		vector.push_back(queue.front());
		queue.pop();
	}
	return vector;
}

q copyVector(v vector) {
	q queue;
	for (int i = 0; i < vector.size(); i++) {
		queue.push(vector[i]);
	}
	return queue;
}

void printQueue(q queue) {
	cout << "Queue: ";
	while (!queue.empty()) {
		cout << queue.front() << " ";
		queue.pop();
	}
	cout << endl;
}

Money averageQ(q queue) {
	v vec = copyQueue(queue);
	int n = 1;
	Money sum = queue.front();
	queue.pop();
	while (!queue.empty()) {
		sum = sum + queue.front();
		queue.pop();
		n++;
	}
	queue = copyVector(vec);
	cout << "Average (queue): " << sum / n << endl;
	return sum / n;
}

void addQueue(q& queue, Money m, int index) {
	v vec; Money mm; int i = 1;
	while (!queue.empty()) {
		mm = queue.front();
		if (i == index) { vec.push_back(m); }
		vec.push_back(mm);
		queue.pop();
		i++;
	}
	queue = copyVector(vec);
}

Money minElem(q queue) {
	Money min = queue.front();
	v vec = copyQueue(queue);
	while (!queue.empty()) {
		if (queue.front() < min) { min = queue.front(); }
```

```
                queue.pop();
        }
        queue = copyVector(vec);
        cout << "Min elem: " << min << endl;

        v vec2; Money m;
        while (!queue.empty()) {
                m = queue.front();
                vec2.push_back(m - min);
                queue.pop();
        }
        queue = copyVector(vec2);

        return min;
}
```

Money.cpp

```
#pragma once
#include <iostream>
#include "Money.h"

using namespace std;
bool Money::operator == (Money& m) {
        if (rub == m.rub && kop == m.kop) return 1;
        else return 0;
}

Money& Money::operator - (const Money& a) {
        Money* tmp = new Money;
        tmp->setRub(this->rub); tmp->setKop(this->kop);
        tmp->rub -= a.rub;
        tmp->kop -= a.kop;
        return  *tmp;
}

Money Money::operator / (const Money& m) {
        int tmp1 = rub * 100 + kop;
        int tmp2 = m.rub * 100 + m.kop;
        Money p;
        p.rub = (tmp1 + tmp2) / 100;
        p.kop = (tmp1 + tmp2) % 100;
        return p;
}
Money Money::operator / (const int& i) {
        Money p;
        p.rub = rub / i;
        p.kop = kop / i;
        return p;
}

bool Money::operator <(const Money& m) {
        if (rub < m.rub) return true;
        if (rub == m.rub && kop < m.kop) return true;
        return false;
}

bool Money::operator >(const Money& m) {
        if (rub > m.rub) return true;
        if (rub == m.rub && kop > m.kop) return true;
        return false;
}

Money& Money::operator =(const Money& temp) {
        if (&temp == this) {
```

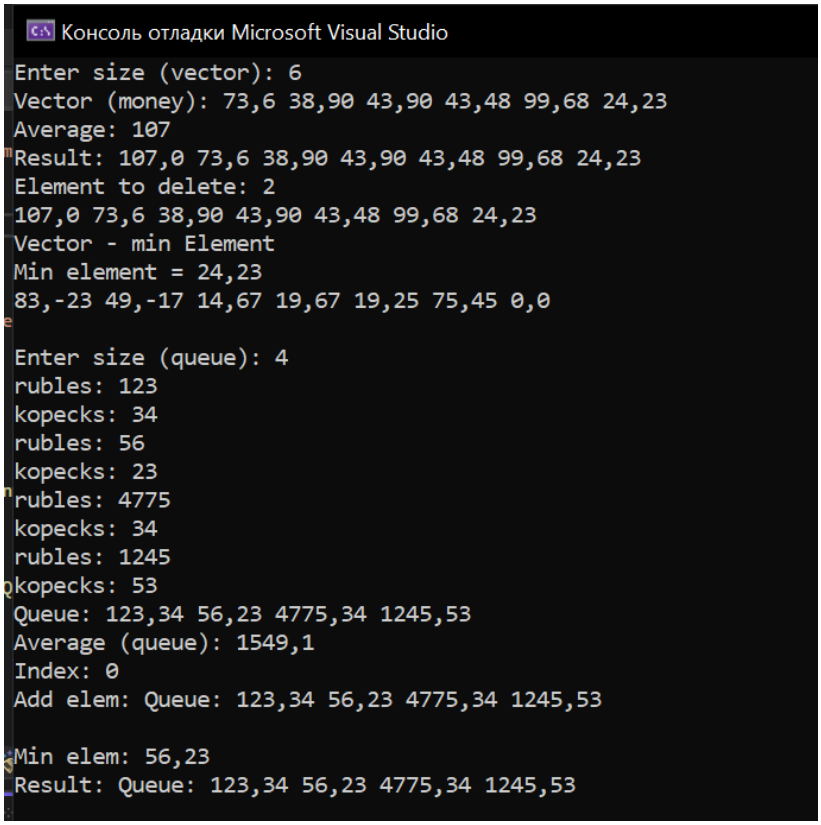
```
                return *this;
        }
        rub = temp.rub;
        kop = temp.kop;
        return *this;
}

Money& Money::operator ++() {
        int temp = rub * 100 + kop;
        temp++;
        kop = temp / 100;
        kop = temp % 100;
        return *this;
}

Money Money::operator ++(int) {
        int temp = rub * 100 + kop;
        temp++;
        Money ex1(rub, kop);
        kop = temp / 100;
        kop = temp % 100;
        return ex1;
}

Money Money::operator +(const Money& temp) {
        int ex1 = rub * 100 + kop;
        int ex2 = temp.rub * 100 + temp.kop;
        Money temp2;
        temp2.rub = (ex1 + ex2) / 100;
        temp2.kop = (ex1 + ex2) % 60;
        return temp2;
}

istream& operator >>(istream& in, Money& temp) {
        cout << "rubles: ";
        in >> temp.rub;
        cout << "kopecks: ";
        in >> temp.kop;
        return in;
}

ostream& operator <<(ostream& out, const Money& temp) {
        return (out << temp.rub << "," << temp.kop);
}
```

Main.cpp

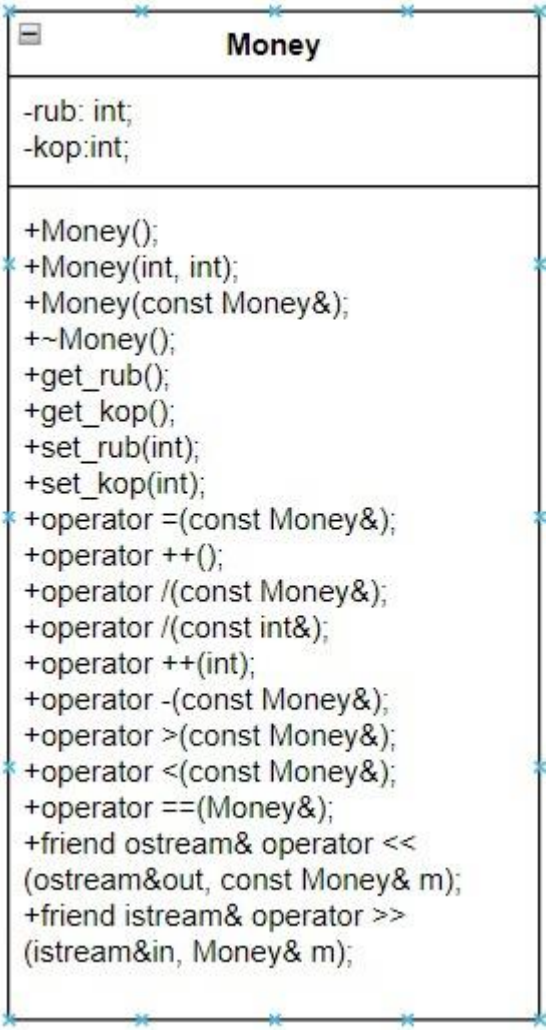
```
#include <iostream>
#include "Method1.h"
#include "Method2.h"

using namespace std;

int main() {
        srand(time(0));
        try {

                vector<Money> v;
                int size;
                cout << "Enter size (vector): "; cin >> size;
                v = makeVector(size);
                cout << "Vector (money): "; printVector(v);

                Money a(average(v), 0);
                v.insert(v.begin(), a);
                cout << "Result: "; printVector(v);
```

```
            int elem;
            cout << "Element to delete: "; cin >> elem;

            printVector(v);

            cout << "Vector - min Element" << endl;
            minElem(v);
            cout << endl;

            Money mm;
            q que;
            int sizeq;
            cout << "Enter size (queue): "; cin >> sizeq;
            que = makeQueue(sizeq);
            printQueue(que);
            averageQ(que);
            int index;
            cout << "Index: "; cin >> index;
            addQueue(que, mm, index);
            cout << "Add elem: "; printQueue(que);
            cout << endl;

            minElem(que);
            cout << "Result: "; printQueue(que);
            cout << endl;
        }
        catch (int) { cout << "Error" << endl; }

        return 0;
}
```

Вывод:



UML-диаграмма



Анализ результатов

Программа сработала корректно и вывела необходимые результаты.