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 №7
«Шаблоны классов»

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

2024 г.

Постановка задачи

1. Определить шаблон класса-контейнера (см. лабораторную работу №6).
2. Реализовать конструкторы, деструктор, операции ввода-вывода, операцию присваивания.
3. Перегрузить операции, указанные в варианте.
4. Инстанцировать шаблон для стандартных типов данных (int, float, double).
5. Написать тестирующую программу, иллюстрирующую выполнение операций для контейнера, содержащего элементы стандартных типов данных.
6. Реализовать пользовательский класс (см. лабораторную работу №3).
7. Перегрузить для пользовательского класса операции ввода-вывода.
8. Перегрузить операции необходимые для выполнения операций контейнерного класса.
9. Инстанцировать шаблон для пользовательского класса.
10. Написать тестирующую программу, иллюстрирующую выполнение операций для контейнера, содержащего элементы пользовательского класса.

11 Вариант:

Класс- контейнер СПИСОК с ключевыми значениями типа int. Реализовать операции: []– доступа по индексу;
int() – определение размера списка;
+ вектор – сложение элементов списков a[i]+b[i];

Пользовательский класс Money для работы с денежными суммами. Число должно быть представлено двумя полями: типа long для рублей и типа int для копеек. Дробная часть числа при выводе на экран должна быть отделена от целой части запятой.

Код программы на C++

List.h

```
#include <iostream>
#pragma once
using namespace std;

template <class T>
class List {
private:
	T size;
	T* array;
public:
	List(T size);
	virtual void show();
```

```
	~List();
	int& operator ()();
	List operator +(const List& other);
	T& operator [](int index);
};

template<class T>
List<T>::List(T size) {
	this->size = size;
	array = new T[size];
	for (int i = 0; i < size; i++) {
		array[i] = rand() % 100 + 1;
	}
}
template<class T>
void List<T>::show() {
	for (int i = 0; i < size; i++) {
		cout << array[i] << " ";
	}
	cout << endl;
}

template<class T>
List<T>::~List() {}

template<class T>
int& List<T>::operator()() {
	return size;
}

template<class T>
List<T> List<T>::operator+(const List& other) {
	List<T> tmp(size);
	for (int i = 0; i < size; i++) {
		tmp[i] = array[i] + other.array[i];
	}
	delete[]array;
	return tmp;
}

template<class T>
T& List<T>::operator [](int index) {
	if (index >= 0 && index < size) {
		return array[index];
	}
	else {
		cout << "Error" << endl;
	}
}
```

Money.h

```
#include <iostream>
#pragma once
using namespace std;

template <class T, class T1>
class Money {
private:
	T rub;
	T kop;
public:
	Money(long, int);
	~Money();
	Money(const Money&);
	Money operator =(const Money&);
```

```
        Money operator +(const Money&);
        void show();
};

template <class T, class T1>
Money<T, T1>::Money(long R, int K) {
        rub = R;
        kop = K;
}

template <class T, class T1>
Money<T, T1>::~Money() {}

template <class T, class T1>
Money<T, T1> Money<T, T1>::operator =(const Money& money) {
        rub = money.rub;
        kop = money.kop;
}

template <class T, class T1>
void Money<T, T1>::show() {
        cout << "\nResult money: " << rub << "," << kop;
}

template <class T, class T1>
Money<T, T1>Money<T, T1>::operator +(const Money& money) {
        int M = rub * 100 + kop;
        int monM = money.rub * 100 + money.kop;
        M += monM;

        Money temp(M / 100, M % 100);
        return temp;
}
```

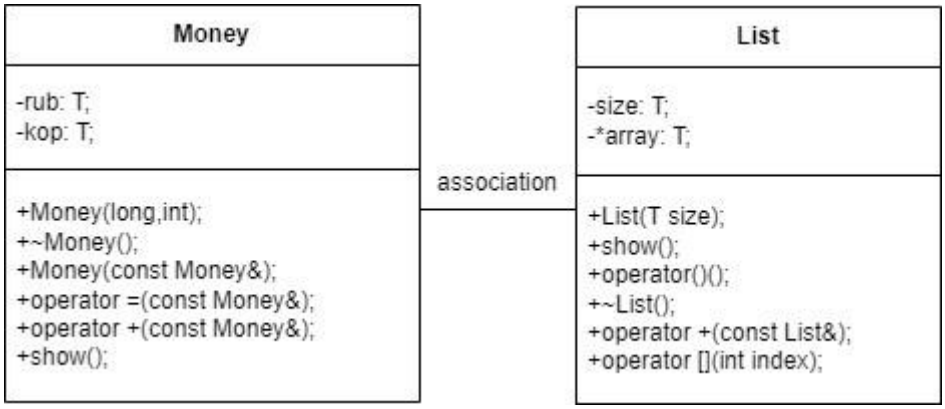
Main.cpp

```
#include <iostream>
#include "Money.h"
#include "List.h"
using namespace std;

int main() {
        int k = 2;
        int size = 6;
        List<int> A(size);
        cout << "Result A: "; A.show();
        cout << "Result A[k]" << A[k] << endl;
        cout << "Size A: " << A() << endl;

        List<int> B(size);
        cout << "Result B: "; B.show();
        cout << "Size B: " << B() << endl;

        List<int> C(size);
        C = A + B;
        cout << "Result C = A + B: "; C.show();


        Money<long, int> a(10203040, 110);
        a.show();
        Money<long, int> b(1000000, 21);
        b.show();

        return 0;
}
```

Вывод:

```
Result A: 42 68 35 1 70 25
Result A[k]35
Size A: 6
Result B: 79 59 63 65 6 46
Size B: 6
Result C = A + B: 121 127 98 66 76 71

Result money: 10203040,110
Result money: 1000000,21
```

UML-диаграмма



Анализ результатов

Программа сработала корректно и вывела необходимые результаты.