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 № 18.7**

Дисциплина: «Информатика»

Тема: Объектно-ориентированное программирование. Шаблоны классов

Вариант 11

Выполнил:

Студент группы РИС-20-1б

Ознобихин Елисей Андреевич

Проверила:

Доцент кафедры ИТАС

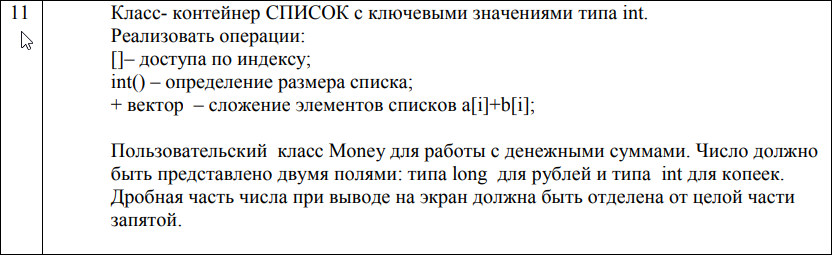
Полякова О. А.

**Пермь, 2021**

**Цель работы**

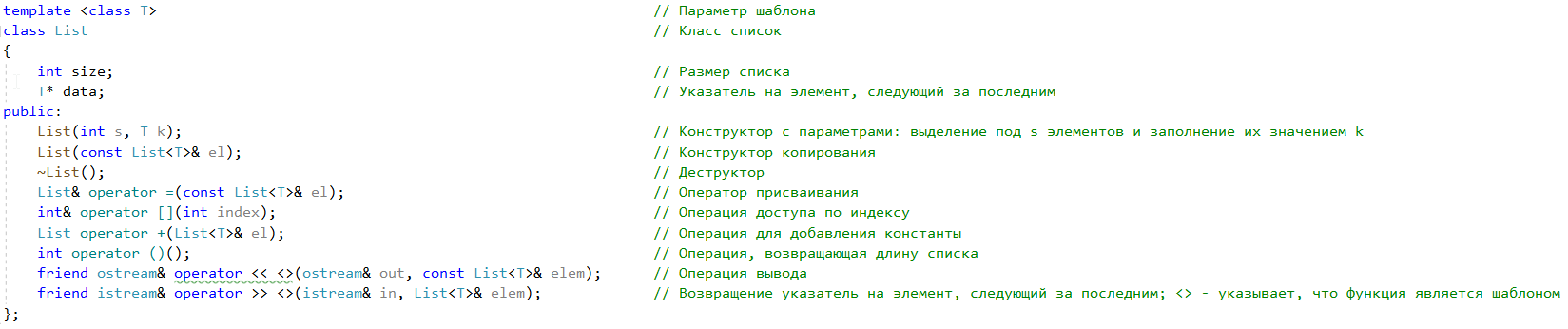
1. Создание консольного приложения, состоящего из нескольких файлов в системе программирования Visual Studio
2. Реализация шаблона класса-контейнера.

**Постановка задачи**

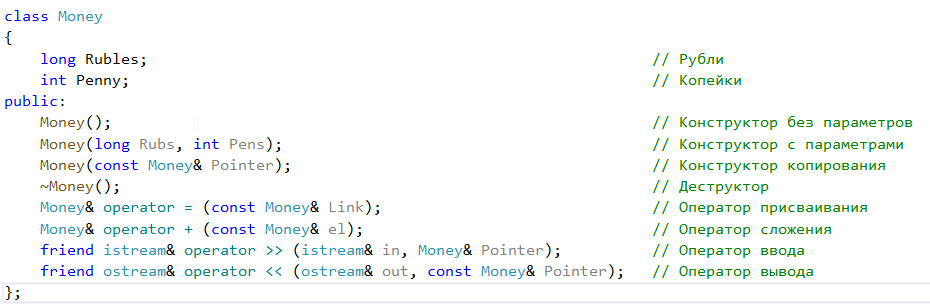
1. Определить шаблон класса-контейнера (см. лабораторную работу №6).
2. Реализовать конструкторы, деструктор, операции ввода-вывода, операцию присваивания.
3. Перегрузить операции, указанные в варианте.
4. Инстанцировать шаблон для стандартных типов данных (int, float, double).
5. Написать тестирующую 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.

**Анализ задачи**

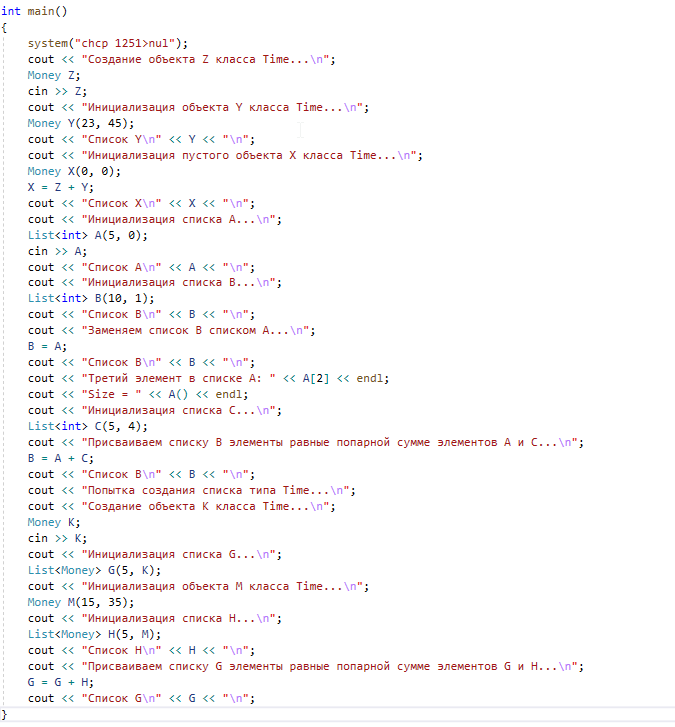
1. Для решения задачи необходимо:
   1. Реализовать определение шаблонного класса List и определение необходимых методов данного класса.



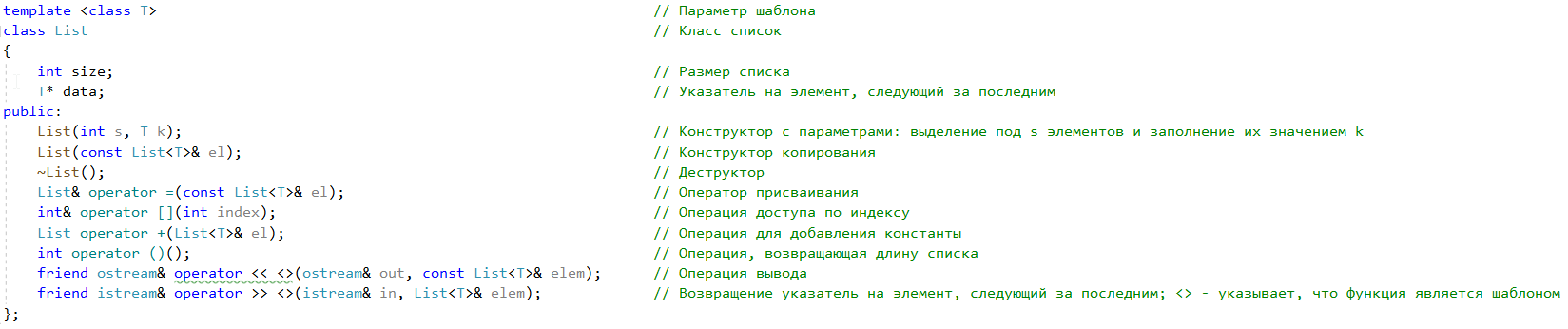
* 1. Реализовать определение класса Money и определение необходимых методов и операторов данного класса.

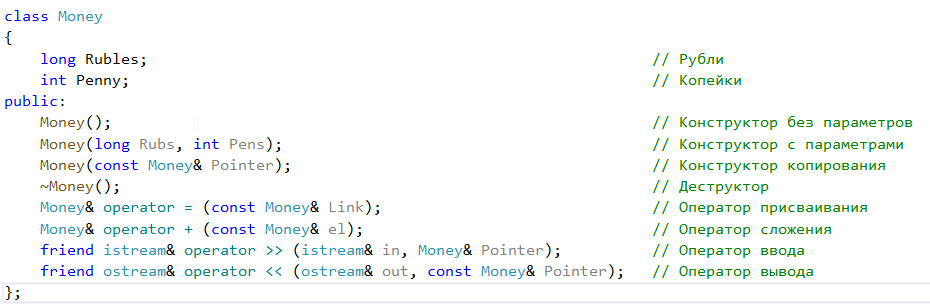


* 1. Реализовать применение этих функций в главной функции.



1. Для решения задачи данные были представлены в следующем виде:
   1. Для работы с данными используются атрибуты класса.





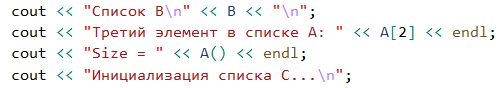
1. Для операций ввода и вывода использовались следующие операторы и функции:
   1. Ввод данных реализован с помощью оператора cin, используемых при реализации в главной функции и функций перегрузки оператора.

C:\Users\Асус\Documents\ShareX\Screenshots\2021-05\devenv_lLfANLTB7k.png

C:\Users\Асус\Documents\ShareX\Screenshots\2021-05\devenv_KNWw3u6Mka.png

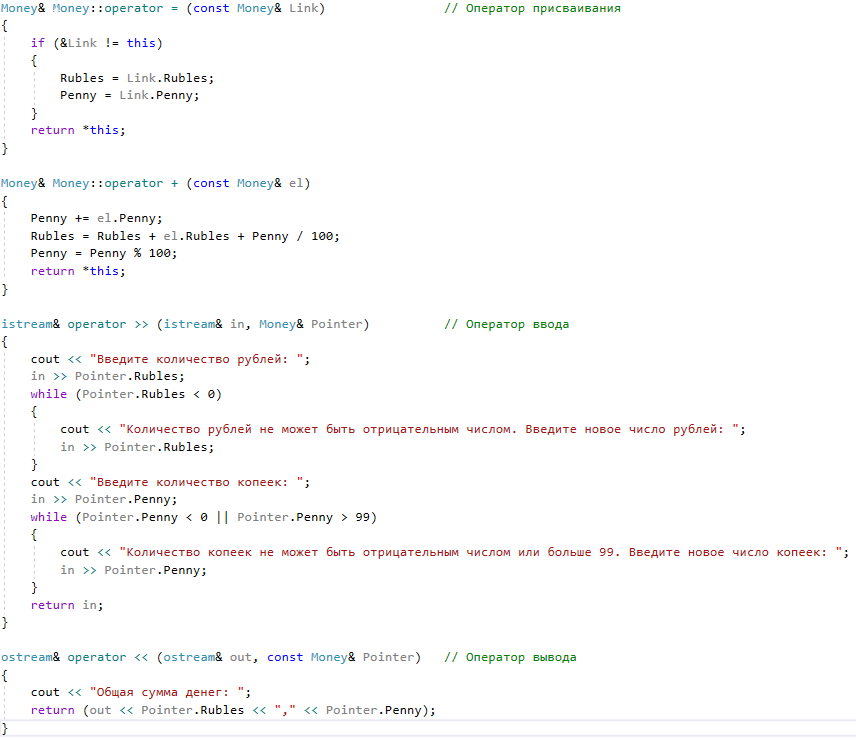
C:\Users\Асус\Documents\ShareX\Screenshots\2021-05\devenv_JyQ2Zf5hYf.png

* 1. Вывод данных реализован с помощью оператора cout, используемого при реализации в главной функции и функций перегрузки операторов ввода-вывода.

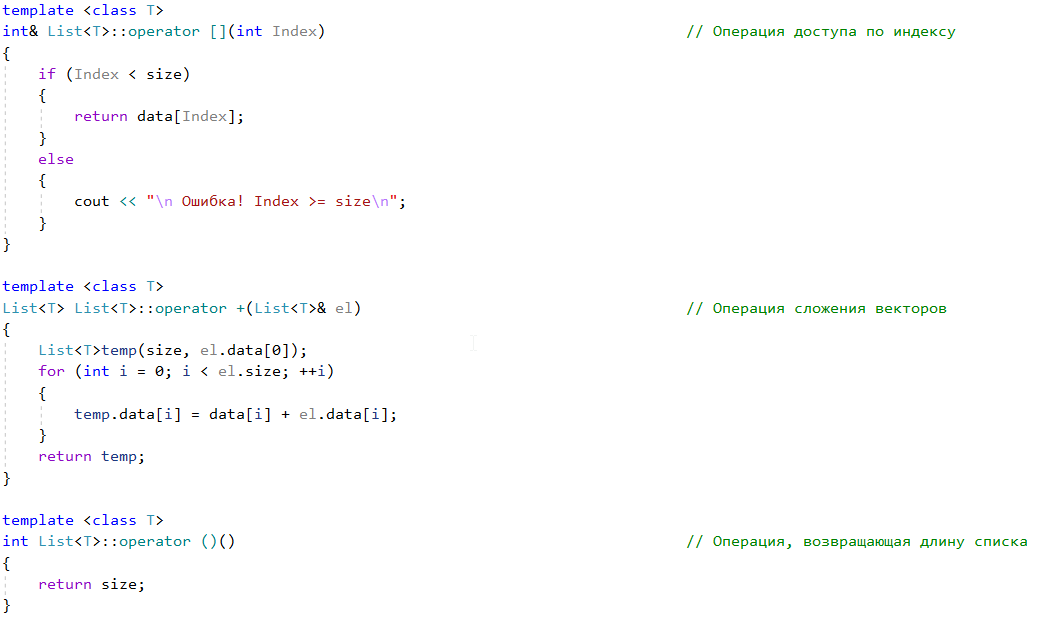


C:\Users\Асус\Documents\ShareX\Screenshots\2021-05\devenv_yAibXQkV9w.png

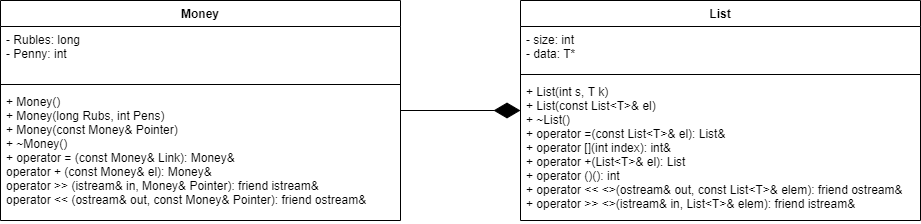
1. Поставленные задачи решены следующими действиями:
   1. В классах Money находится реализация ввода-вывода и сложение рублей и копеек.



* 1. В шаблонном классе List находится реализация списка для различных типов данных, определение размера списка, доступ элемента списка по индексу, сложение векторов.



**UML-Диаграмма**



**Реализация задачи на языке С++**

**Заголовочный файл List.h**

#pragma once

#include <iostream>

using namespace std;

template <class T>

class List

{

int size;

T\* data;

public:

List(int s, T k);

List(const List<T>& el);

~List();

List& operator =(const List<T>& el);

int& operator [](int index); List operator +(List<T>& el);

int operator ()();

friend ostream& operator << <>(ostream& out, const List<T>& elem);

friend istream& operator >> <>(istream& in, List<T>& elem);

};

template <class T>

List<T>::List(int s, T k)

{

size = s;

data = new T[size];

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

{

data[i] = k;

}

}

template <class T>

List<T>::List(const List<T>& el)

{

if (size < el.size)

{

size = el.size;

}

data = new T[size];

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

{

data[i] = el.data[i];

}

}

template <class T>

List<T>::~List(){

delete[] data;

data = 0;

}

template <class T>

List<T>& List<T>::operator =(const List<T>& el)

{

if (this != &el)

{

size = el.size;

if (data != 0)

{

delete[] data;

}

data = new T[size];

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

{

data[i] = el.data[i];

}

}

return \*this;

}

template <class T>

int& List<T>::operator [](int Index)

{

if (Index < size)

{

return data[Index];

}

else

{

cout << "\n Ошибка! Index >= size\n";

}

}

template <class T>

List<T> List<T>::operator +(List<T>& el)

{

List<T>temp(size, el.data[0]);

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

{

temp.data[i] = data[i] + el.data[i];

}

return temp;

}

template <class T>

int List<T>::operator ()()

{

return size;

}

template <class T>

ostream& operator << (ostream& out, const List<T>& el)

{

cout << "Вывод данных из списка\n";

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

{

out << el.data[i] << " ";

}

return out;

}

template <class T>

istream& operator >>(istream& in, List<T>& el) // Операция ввода

{

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

{

cout << "Введите данные " << i + 1 << " объекта в список\n";

in >> el.data[i];

}

return in;

}

**Заголовочный файл Money.h**

#pragma once

#include <iostream>

using namespace std;

class Money

{

long Rubles;

int Penny;

public:

Money();

Money(long Rubs, int Pens);

Money(const Money& Pointer);

~Money();

Money& operator = (const Money& Link);

Money& operator + (const Money& el);

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

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

};

**Файл с описанием методов класса Money.cpp**

#include "Money.h"

#include <iostream>

using namespace std;

Money::Money()

{

Rubles = 0;

Penny = 0;

}

Money::Money(long Rubs, int Pens)

{

Rubles = Rubs;

Penny = Pens;

}

Money::Money(const Money& Pointer)

{

Rubles = Pointer.Rubles;

Penny = Pointer.Penny;

}

Money::~Money()

{

}

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

{

if (&Link != this)

{

Rubles = Link.Rubles;

Penny = Link.Penny;

}

return \*this;

}

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

{

Penny += el.Penny;

Rubles = Rubles + el.Rubles + Penny / 100;

Penny = Penny % 100;

return \*this;

}

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

{

cout << "Введите количество рублей: ";

in >> Pointer.Rubles;

while (Pointer.Rubles < 0)

{

cout << "Количество рублей не может быть отрицательным числом. Введите новое число рублей: ";

in >> Pointer.Rubles;

}

cout << "Введите количество копеек: ";

in >> Pointer.Penny;

while (Pointer.Penny < 0 || Pointer.Penny > 99)

{

cout << "Количество копеек не может быть отрицательным числом или больше 99. Введите новое число копеек: ";

in >> Pointer.Penny;

}

return in;

}

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

{

cout << "Общая сумма денег: ";

return (out << Pointer.Rubles << "," << Pointer.Penny);

}

**Файл с главной программой Лабораторная работа № 18.7.cpp**

#include "List.h"

#include "Money.h"

#include <iostream>

using namespace std;

int main()

{

system("chcp 1251>nul");

cout << "Создание объекта Z класса Time...\n";

Money Z;

cin >> Z;

cout << "Инициализация объекта Y класса Time...\n";

Money Y(23, 45);

cout << "Список Y\n" << Y << "\n";

cout << "Инициализация пустого объекта X класса Time...\n";

Money X(0, 0);

X = Z + Y;

cout << "Список X\n" << X << "\n";

cout << "Инициализация списка A...\n";

List<int> A(5, 0);

cin >> A;

cout << "Список А\n" << A << "\n";

cout << "Инициализация списка B...\n";

List<int> B(10, 1);

cout << "Список B\n" << B << "\n";

cout << "Заменяем список B списком A...\n";

B = A;

cout << "Список B\n" << B << "\n";

cout << "Третий элемент в списке A: " << A[2] << endl;

cout << "Size = " << A() << endl;

cout << "Инициализация списка C...\n";

List<int> C(5, 4);

cout << "Присваиваем списку B элементы равные попарной сумме элементов A и C...\n";

B = A + C;

cout << "Список B\n" << B << "\n";

cout << "Попытка создания списка типа Time...\n";

cout << "Создание объекта K класса Time...\n";

Money K;

cin >> K;

cout << "Инициализация списка G...\n";

List<Money> G(5, K);

cout << "Инициализация объекта M класса Time...\n";

Money M(15, 35);

cout << "Инициализация списка H...\n";

List<Money> H(5, M);

cout << "Список H\n" << H << "\n";

cout << "Присваиваем списку G элементы равные попарной сумме элементов G и H...\n";

G = G + H;

cout << "Список G\n" << G << "\n";

}

**Ответы на вопросы**

1. Адаптировать класс под использование любых типов данных
2. Ключевое слово template
3. Ключевое слово template
4. Поля, принимающие значения любого типа
5. Не могут быть виртуальными, могут быть дружественными и наследуемыми
6. Template <typename T>
7. Да
8. Могут содержать дружественные, но не виртуальные функции. Код прописывается во время компиляции
9. Нет
10. Да
11. Нет
12. Ключевое слово template
13. Генерация кода по определенному типу данных
14. При компиляции