Министерство науки и высшего образования Российской Федерации Федеральное государственное автономное образовательное учреждение высшего образования

**«Пермский национальный исследовательский политехнический университет»**

Электротехнический факультет  
Кафедра «Информационные технологии и автоматизированные системы» направление подготовки: 09.03.04 – «Программная инженерия»

# Лабораторная работа "Шаблоны классов."

Выполнил студент гр. РИС-24-3б

Жиряков Леонид Антонович

Проверил:

Доцент кафедры ИТАС   
Ольга Андреевна Полякова

\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (оценка) (подпись)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(дата)

г. Пермь, 2024

Вариант 8.

Постановка задачи:

Общая:

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).

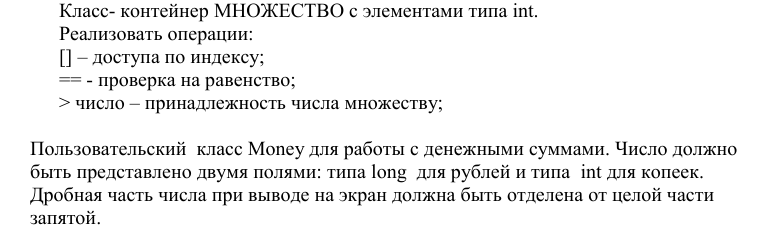
7. Перегрузить для пользовательского класса операции ввода-вывода.

8. Перегрузить операции необходимые для выполнения операций контейнерного класса.

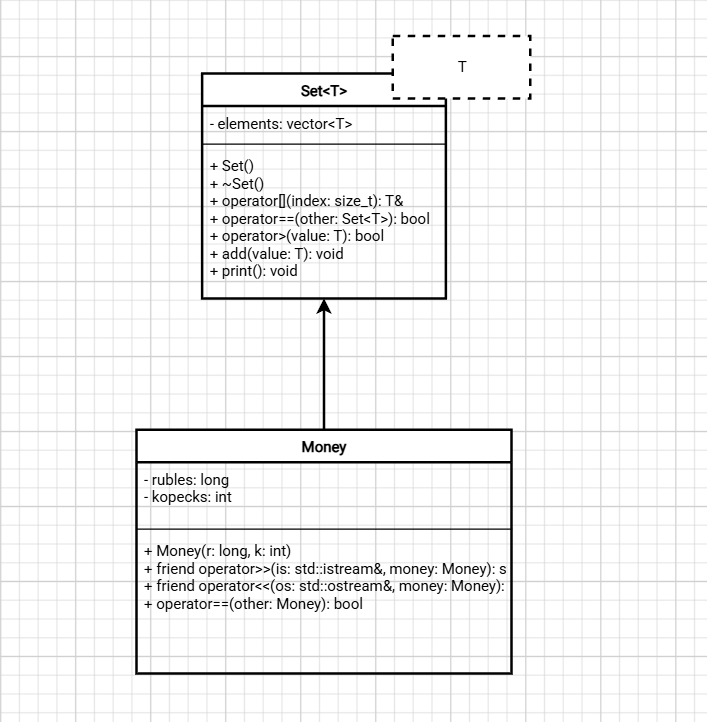
9. Инстанцировать шаблон для пользовательского класса.

10. Написать тестирующую программу, иллюстрирующую выполнение операций для контейнера, содержащего элементы пользовательского класса.

Персональная:



UML – диаграмма:



Программная реализация

**Заголовочный файлы**

**Set.h:**

#pragma once

#include <iostream>

#include <vector>

template<typename T>

class Set

{

private:

std::vector<T> elements;

public:

Set() = default;

~Set() = default;

// Оператор доступа по индексу

T& operator[](size\_t index)

{

return elements[index];

}

// Оператор проверки на равенство

bool operator==(const Set<T>& other) const

{

return elements == other.elements;

}

// Оператор принадлежности

bool operator>(const T& value) const

{

return std::find(elements.begin(), elements.end(), value) != elements.end();

}

void add(const T& value)

{

if (!(\*this > value))

{

elements.push\_back(value);

}

}

void print() const

{

for (const auto& element : elements)

{

std::cout << element << " ";

}

std::cout << std::endl;

}

};

Money.h:

#pragma once

#include <iostream>

#include <iomanip>

class Money

{

private:

long rubles;

int kopecks;

public:

Money(long r = 0, int k = 0) : rubles(r), kopecks(k) {}

// Оператор ввода

friend std::istream& operator>>(std::istream& is, Money& money)

{

is >> money.rubles >> money.kopecks;

return is;

}

// Оператор вывода

friend std::ostream& operator<<(std::ostream& os, const Money& money)

{

os << money.rubles << "," << std::setfill('0') << std::setw(2) << money.kopecks;

return os;

}

bool operator==(const Money& other) const

{

return (rubles == other.rubles) && (kopecks == other.kopecks);

}

};

**Функция main**

#include "Set.h"

#include "Money.h"

int main()

{

// Тест для стандартных типов данных

Set<int> intSet;

intSet.add(1);

intSet.add(2);

intSet.add(4);

intSet.print(); // Ожидается вывод: 1 2 4

// Тест для пользовательского класса Money

Set<Money> moneySet;

Money m1(110, 70);

Money m2(200, 55);

moneySet.add(m1);

moneySet.add(m2);

moneySet.print();

return 0;

}

Результат работы:



Контрольные вопросы:

1. В чем смысл использования шаблонов?

2. Каковы синтаксис/семантика шаблонов функций?

3. Каковы синтаксис/семантика шаблонов классов?

4. Что такое параметры шаблона функции?

5. Перечислите основные свойства параметров шаблона функции.

6. Как записывать параметр шаблона?

7. Можно ли перегружать параметризованные функции?

8. Перечислите основные свойства параметризованных классов.

9. Все ли компонентные функции параметризованного класса являются параметризованными?

10. Являются ли дружественные функции, описанные в параметризованном классе, параметризованными?

11. Могут ли шаблоны классов содержать виртуальные компонентные функции?

12. Как определяются компонентные функции параметризованных классов вне определения шаблона класса?

13. Что такое инстанцирование шаблона?

14. На каком этапе происходит генерирование определения класса по шаблону?

Ответы:

1. **В чем смысл использования шаблонов?**  
   Шаблоны позволяют отделить алгоритм от конкретных типов данных, что делает код более универсальным и повторно используемым. Они позволяют создавать функции и классы, которые могут работать с любыми типами данных, передаваемыми в качестве параметров.
2. **Каковы синтаксис/семантика шаблонов функций?**  
   Шаблон функции определяется с помощью ключевого слова template, за которым следует список параметров шаблона. Синтаксис выглядит так:

template <параметры шаблона>

заголовок функции

{

тело функции

}

1. **Каковы синтаксис/семантика шаблонов классов?**  
   Шаблон класса также определяется с помощью template, за которым следует список параметров шаблона и имя класса. Синтаксис:

template <параметры шаблона>

class имя класса

{

// тело класса

};

1. **Что такое параметры шаблона функции?**  
   Параметры шаблона функции — это типы данных, которые передаются в шаблон и используются в определении функции. Они позволяют функции работать с различными типами данных.
2. **Перечислите основные свойства параметров шаблона функции.**
   * Параметры могут быть любого типа (включая пользовательские типы).
   * Параметры могут быть использованы для определения типов возвращаемых значений и параметров функции.
   * Параметры шаблона могут быть перегружены.
3. **Как записывать параметр шаблона?**  
   Параметр шаблона записывается в заголовке шаблона, например, template<class T> или template<typename T>.
4. **Можно ли перегружать параметризованные функции?**  
   Да, параметризованные функции могут быть перегружены, как и обычные функции.
5. **Перечислите основные свойства параметризованных классов.**
   * Параметризованные классы могут содержать статические элементы.
   * Они могут иметь дружественные функции и классы.
   * Могут быть производными от других шаблонов и обычных классов.
6. **Все ли компонентные функции параметризованного класса являются параметризованными?**  
   Нет, не все компонентные функции обязательно являются параметризованными. Они могут быть определены как обычные функции.
7. **Являются ли дружественные функции, описанные в параметризованном классе, параметризованными?**  
   Да, дружественные функции, описанные в параметризованном классе, могут быть параметризованными.
8. **Могут ли шаблоны классов содержать виртуальные компонентные функции?**  
   Нет, шаблоны методов (функций) не могут быть виртуальными, но виртуальные функции могут быть определены в обычных классах, от которых наследуются шаблоны.
9. **Как определяются компонентные функции параметризованных классов вне определения шаблона класса?**  
   Определение компонентных функций происходит с использованием синтаксиса, который включает имя класса и параметры шаблона, например:

template<class T>

void Point<T>::Show()

{

// тело функции

}

1. **Что такое инстанцирование шаблона?**  
   Инстанцирование шаблона — это процесс генерации компилятором конкретного класса или функции на основе шаблона и аргументов шаблона.
2. **На каком этапе происходит генерирование определения класса по шаблону?**  
   Генерация определения класса по шаблону происходит на этапе компиляции, когда компилятор встречает конкретный экземпляр шаблона, использованный в программе.

Ссылка:

**https://github.com/LeonidZhir/-**