Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

**«Пермский национальный исследовательский политехнический университет»**

Кафедра «Информационные технологии и автоматизированные системы»

**ОТЧЕТ**

Тема: «АТД. Контейнеры»

Выполнил работу

Студент группы РИС-22-2Б

Третьяков Н. А.

Проверил

Доцент кафедры ИТАС

Полякова О.А.

Г. Пермь-2023

**Постановка задачи**

1. Определить шаблон класса-контейнера.

2. Реализовать конструкторы, деструктор, операции ввода-вывода, операцию присваивания.

3. Перегрузить операции, указанные в варианте.

4. Инстанцировать шаблон для стандартных типов данных.

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.

Класс- контейнер Множество с ключевыми значениями типа int.

Реализовать операции:

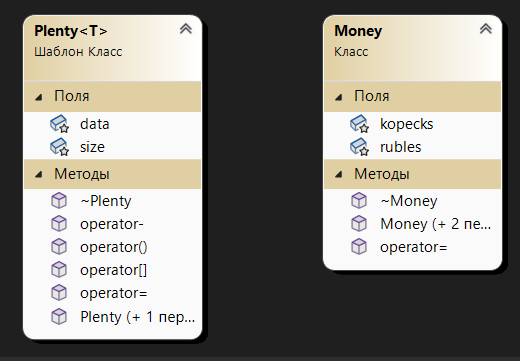
[] - доступа по индексу;

() - определение размера списка;

- - разность множеств а – b=;

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.

**UML - диаграмма**



**Код программы.**

Main.cpp

#include <iostream>

using namespace std;

template <class T>

class Plenty

{

protected:

int size;

T\* data;

public:

Plenty(int s)

{

size = s;

data = new T[size];

for (int k = 0; k < s; k++)

data[k] = 0;

}

Plenty(const Plenty<T>& p)

{

size = p.size;

data = new T[size];

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

data[k] = p.data[k];

}

Plenty& operator = (const Plenty<T>&p)

{

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

if (size > p.size)

{

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

{

data[i] = p.data[i];

}

}

else

{

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

{

data[i] = p.data[i];

}

}

return \*this;

}

Plenty& operator-(const Plenty<T>& p)

{

if (size != p.size)

{

cout << "" << endl;

}

else

{

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

{

data[i] -= p.data[i];

}

return \*this;

}

}

T& operator[](int index)

{

if (index < size) return data[index];

else cout << "" << endl;

}

T operator() ()

{

return size;

}

friend ostream& operator<<(ostream& out, const Plenty<T>& p)

{

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

{

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

}

return out;

}

friend istream& operator>>(istream& in, Plenty<T>& p)

{

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

{

in >> p.data[i];

}

return in;

}

~Plenty()

{

delete[] data;

size = 0;

}

};

class Money

{

protected:

long rubles;

int kopecks;

public:

Money()

{

rubles = 0;

kopecks = 0;

}

Money(long r, int k)

{

rubles = r;

if (k < 100)

{

kopecks = k;

}

else

{

rubles = rubles + k / 100;

kopecks = k % 100;

}

}

Money(const Money& m)

{

rubles = m.rubles;

kopecks = m.kopecks;

}

Money& operator = (const Money& m)

{

Money p;

p.rubles = m.rubles;

p.kopecks = m.kopecks;

return p;

}

friend std::ostream& operator << (std::ostream& out, const Money& p)

{

return(out << p.rubles << ", " << p.kopecks);

}

friend std::istream& operator >> (std::istream& in, Money& p)

{

in >> p.rubles;

in >> p.kopecks;

return in;

}

~Money() {};

};

int main()

{

Plenty<int> P(5);

cin >> P;

cout << P << endl;

Plenty <int> Pl(10);

Pl = P;

cout << Pl << endl;

cout << P[2] << endl;

cout << "size = " << P() << endl;

Money m;

cin >> m;

cout << m;

return 0;

}

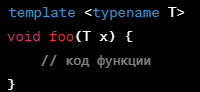
Контрольные вопросы.

1. В чем смысл использования шаблонов?

Использование шаблонов позволяет создавать обобщенный код, который может быть использован с различными типами данных или объектами без необходимости написания отдельного кода для каждого типа или объекта. Это упрощает разработку и поддержку кода.

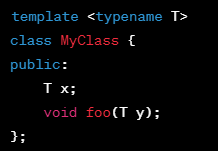
1. Каковы синтаксис/семантика шаблонов функций?

Синтаксис шаблонов функций заключается в использовании ключевого слова "template" перед определением функции и в указании параметров шаблона в угловых скобках.



1. Каковы синтаксис/семантика шаблонов классов?

Синтаксис шаблонов классов похож на синтаксис шаблонов функций, но параметры шаблона указываются после имени класса.



Здесь "T" - это параметр шаблона класса, который может быть заменен на любой тип данных при создании объектов класса.

1. Что такое параметры шаблона функции?

Параметры шаблона функции - это типы данных или константы, которые могут быть переданы в функцию в качестве аргументов. Они используются для создания обобщенного кода, который может работать с различными типами данных.

1. Перечислите основные свойства параметров шаблона функции.

Основные свойства параметров шаблона функции:

* Они могут быть заменены на любой тип данных при вызове функции.
* Они могут быть константами или типами данных.
* Они могут иметь значения по умолчанию.

1. Как записывать параметр шаблона?

Параметр шаблона записывается в угловых скобках после ключевого слова "template". Например: **template <typename T>**

1. Можно ли перегружать параметризованные функции?

Параметризованные функции могут быть перегружены, как и обычные функции. Они должны отличаться друг от друга типами параметров шаблона или аргументами функции.

1. Перечислите основные свойства параметризованных классов.

Основные свойства параметризованных классов:

* Они позволяют создавать обобщенные классы, которые могут работать с различными типами данных.
* Они могут содержать параметры шаблона для определения типов данных, используемых внутри класса.
* Они могут быть унаследованы от других параметризованных классов или обычных классов.

1. Все ли компонентные функции параметризованного класса являются параметризованными?

Не все компонентные функции параметризованного класса должны быть параметризованными. Некоторые функции могут использовать обычные типы данных, не зависящие от параметров шаблона.

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

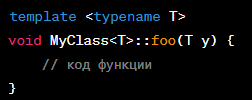
Дружественные функции, описанные в параметризованном классе, могут быть параметризованными или не параметризованными, в зависимости от того, используют ли они параметры шаблона класса или обычные типы данных. Если дружественная функция использует параметры шаблона класса, то она должна быть также параметризованной.

1. Могут ли шаблоны классов содержать виртуальные компонентные функции?

Шаблоны классов могут содержать виртуальные компонентные функции.

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

Компонентные функции параметризованных классов могут быть определены вне определения шаблона класса, так же как и для обычных классов. Однако при определении функции вне шаблона класса необходимо указать параметры шаблона перед именем класса.



1. Что такое инстанцирование шаблона?

Инстанцирование шаблона - это процесс создания экземпляра класса или функции на основе определения шаблона с конкретными параметрами шаблона. Например, если у нас есть шаблон класса **MyClass**, мы можем создать экземпляр этого класса с конкретным типом данных.

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

Генерирование определения класса по шаблону происходит на этапе компиляции. Когда компилятор встречает использование шаблона класса или функции в коде, он генерирует соответствующее определение на основе шаблона с конкретными параметрами шаблона. Это происходит перед созданием объектов или вызовом функций с использованием шаблона.