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 ПО ЛАБОРАТОРНОЙ РАБОТЕ**

Дисциплина: «Основы алгоритмизации и программирования»

Тема : "Шаблоны классов"

Выполнила работу

Студентка группы РИС-22-1Б

Верхоланцева Е. С.

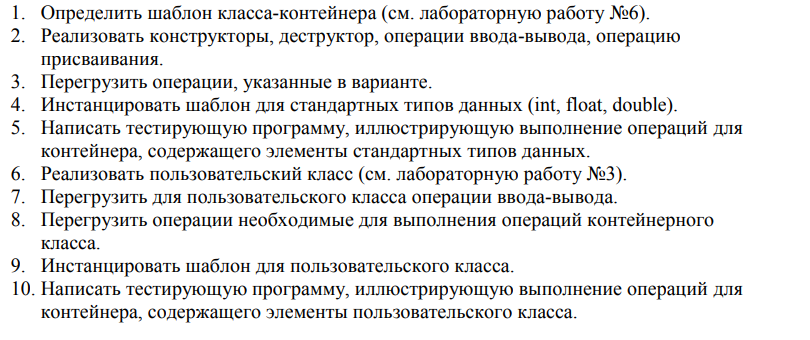
Проверила

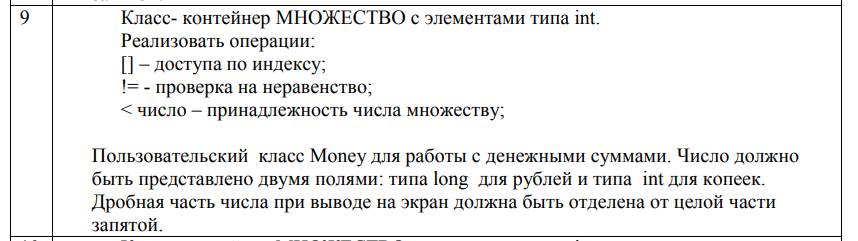
Доцент кафедры ИТАС

Полякова О.А.

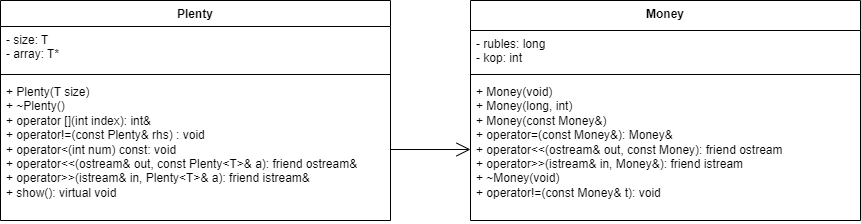
Г. Пермь-2023

**Постановка задачи**





**Диаграмма классов**



**Описание параметритизированного класса**

template<class T>

class Plenty

{

private:

T size;

T\* array;

public:

Plenty(T size);

~Plenty() { delete[] array; size = 0; }

int& operator [](int index);

void operator!=(const Plenty& rhs);

void operator<(int num) const;

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

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

virtual void show()

{

cout << "Множество: ";

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

{

cout << " " << array[i];

}

cout << endl;

}

};

**Определение компонентных функций**

template<class T>

Plenty<T>::Plenty(T size)

{

this->size = size;

array = new T[size];

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

{

array[i] = rand() % 100 + 1;

}

}

// операция проверки на неравенство

template<class T>

void Plenty<T>::operator!=(const Plenty& rhs) {

bool f = false;

int i = 0;

while (!f) {

if (array[i] != rhs.array[i]) f = true;

i++;

}

if (f == 1) cout << "Множества не равны." << endl;

else cout << "Множества равны." << endl;

}

// операция доступа по индексу

template<class T>

int& Plenty<T>::operator [](int index)

{

if (index >= 0 && index < size)

{

return array[index];

}

else

{

cout << endl << "Error" << endl;

}

}

// проверка числа на принадлежность множеству

template<class T>

void Plenty<T>::operator<(int num) const

{

bool f = 0;

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

if (array[i] == num) {

f = 1;

cout << "Число " << num << " принадлежит множеству." << endl << endl;

break;

}

}

if (f == 0) { cout << "Число " << num << " не принадлежит множеству. " << endl << endl; }

}

template<class T>

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

{

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

{

out << a.array[i] << " ";

}

return out;

}

template<class T>

istream& operator>>(istream& in, Plenty<T>&a)

{

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

{

in >> a.array[i];

}

return in;

}

**Описание пользовательского класса Money**

class Money

{

public:

Money(void);

Money(long, int);

Money(const Money&);

Money& operator=(const Money&);

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

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

virtual ~Money(void) { rubles = 0; kop = 0; };

void operator!=(const Money& t);

private:

long rubles;

int kop;

};

**Определение компонентных функций**

Money::Money(void)

{

rubles = 0;

kop = 0;

}

Money::Money(long r, int k)

{

rubles = r;

kop = k;

}

Money::Money(const Money& t)

{

rubles = t.rubles;

kop = t.kop;

}

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

{

rubles = t.rubles;

kop = t.kop;

return \*this;

}

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

{

out << t.rubles << " , " << t.kop;

return out;

}

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

{

cout << "Введите рубли: "; in >> t.rubles;

cout << "Введите копейки: "; in >> t.kop;

return in;

}

void Money::operator!=(const Money& t)

{

if (rubles == t.rubles && kop == t.kop) cout << "Равны." << endl;

else cout << "Не равны." << endl;

}

**Функция main**

int main()

{

setlocale(LC\_ALL, "rus");

srand(time(0));

int k, size = 5, n = 0;

cout << "Введите k - индекс множества: "; cin >> k;

Plenty<int> a(size);

// вывод множества а

a.show();

cout << endl << "Число с выбранным индексом: " << a[k] << endl;

cout << "Введите число, которое хотите проверить на принадлежность множетству: "; cin >> n;

a < n;

cout << "Проверка операции равенства множеств." << endl;

Plenty<int> b(size);

b.show();

Plenty<int> c(size);

c.show();

b != c;

cout << "Тестирование класса Money:" << endl;

Money t1;

cin >> t1;

cout << "Первый экземпляр класса Money:" << endl;

cout << t1; cout << endl;

Money t2;

cin >> t2;

cout << "Второй экзепляр класса Money: " << endl;

cout << t2; cout << endl;

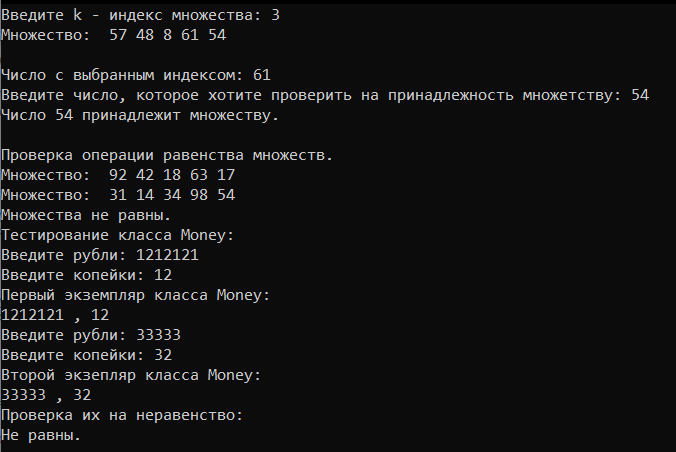
cout << "Проверка их на неравенство: " << endl;

t1 != t2;

return 0;

}

**Результаты работы программы**



**Ответы на контрольные вопросы**

1. В чем смысл использования шаблонов?

Шаблоны вводятся для того, чтобы автоматизировать создание функций, обрабатывающих разнотипные данные.

2. Каковы синтаксис/семантика шаблонов функций?

template<параметры\_шаблона>

заголовок функции

{тело функции}

пример:

template<class type>

type abs(type x)

{

if (x<0) return -x;

else return x;

}

3. Каковы синтаксис/семантика шаблонов классов?

template<параметры\_шаблона>

class имя\_класса

{ ... };

пример:

template<class T>

class Point

{

T x, y;

public:

Point(T x = 0, T y = 0):x(x), y(y){}

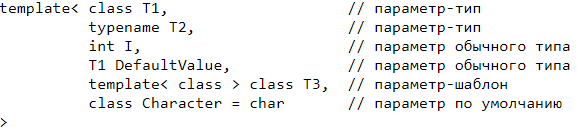
void Show();

{;

4. Что такое параметры шаблона функции?

Параметр шаблона - то, что указывается в <>.

Параметрами шаблонов могут быть:

* параметры-типы;
* параметры обычных типов;
* параметры-шаблоны.
* 

5. Перечислите основные свойства параметров шаблона функции.

- шаблоны не могут быть виртуальными

- шаблоны могут содержать статические элементы, дружественные функции и классы

- шаблоны могут быть производными как от шаблонов, так и от обычных классов, а также являться базовыми и для шаблонов, и для обычных классов.

6. Как записывать параметр шаблона?

в угловых скобочках < >

7. Можно ли перегружать параметризованные функции?

Да, меняя тип данных параметра, либо меняя параметры местами, в том случае, если они разного типа, а также можно выполнять перегрузку функции добавляя или исключая параметры.

8. Перечислите основные свойства параметризованных классов.

- Параметры шаблона могут быть любыми типами данных, включая примитивные типы, пользовательские типы и другие шаблоны классов.

- Параметры шаблона могут быть как значениями, так и типами. Например, можно создать шаблон класса с параметром, который задает размер массива.

-Параметры шаблона могут использоваться внутри класса для объявления переменных, функций и типов.

-Каждый конкретный экземпляр параметризованного класса является отдельным типом, который определяется параметрами шаблона.

-Шаблонные классы могут быть унаследованы от других шаблонных классов и могут иметь шаблонные функции-члены.

9. Bce ли компонентные функции параметритизированного класса являются параметризованными?

Да, все компонентные функции (методы) параметризированного класса являются параметризованными, то есть они используют те же параметры шаблона, что и сам класс.

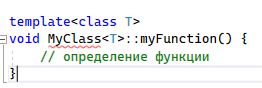
10. Являются ли дружественные функции, описанные в  
классе, параметризованными?  
Да, дружественные функции, описанные внутри класса, являются параметризованными, если класс является параметризованным. Дружественные функции имеют доступ к закрытым членам класса, но они также должны быть параметризованы, чтобы иметь доступ к параметрам шаблона класса.

11. Могут ли шаблоны классов содержать виртуальные компонентные функции?

Да, шаблоны классов могут содержать виртуальные компонентные функции. Виртуальные функции позволяют реализовать полиморфизм и позволяют вызывать функцию на объекте, указатель на который имеет тип базового класса, но указывает на объект, созданный из класса-наследника.

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

Компонентные функции параметризованных классов могут быть определены вне определения шаблона класса с помощью использования template<class T> перед каждым определением функции, где T - это параметр шаблона.



13. Что такое инстанцирование шаблона?

Инстанцирование шаблона - это процесс создания конкретной реализации шаблона класса или функции на основе заданных параметров шаблона. При инстанцировании компилятор создает отдельный экземпляр шаблона для конкретных аргументов шаблона. То есть компилятор подставляет значения параметров шаблона в шаблонный код и создает реализацию класса или функции, которая может быть использована в программе.

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

На этапе компиляции.