Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение

высшего образования

«Пермский национальный исследовательский политехнический университет» (ПНИПУ)

Электротехнический факультет

Кафедра информационных технологий и автоматизированных систем

Направление подготовки 09.03.01 «Информатика и вычислительная техника»

**Лабораторная работа «Класс №7»**

**по дисциплине**

**«Основы алгоритмизации и программирования»**

**(Семестр** 2)

Выполнил студент гр. ИВТ-21-1б

Гребнев Алексей Дмитриевич

Проверил:

Яруллин Денис Владимирович

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

(оценка) (подпись)

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

(дата)

г.Пермь -2022

**Постановка задачи**

1.Определить шаблон класса-контейнера(см.лаб.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.лаб.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.

**Исходные данные**

Вариант 10:

Класс-контейнер МНОЖЕСТВО с элементами типа int.

Реализовать операции:

[] – доступа по индексу;

() – определение размера множества;

- -разность множеств;

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.

#include <iostream>

using namespace std;

template <class T>

class Variety

{

private:

int size;

T\* data;

public:

Variety(T size)

{

this->size = size;

data = new T[size];

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

{

data[i] = rand() % 96;

}

}

void ShowVariety()

{

cout << "Variety: ";

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

{

cout << " " << data[i] << " ";

}

}

int& operator[](int num)

{

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

{

return data[num];

}

else

{

cout << endl << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl << "Ошибка" << endl << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

}

}

Variety operator-(const Variety& a) //-a

{

Variety<T> temp(size);

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

{

temp[i] = data[i] - a.data[i];

}

delete[] data;

return temp;

}

int& operator() ()

{

return size;

}

~Variety()

{

}

};

class Money

{

private:

long rub;

int cop;

public:

Money();

Money(long, int);

~Money();

Money operator-(const Money& a);

Money& operator=(const Money& a);

void ShowMoney()

{

cout << "Money: " << rub << ", " << cop << ";" << endl;

}

};

Money::Money()

{

rub = 0;

cop = 0;

}

Money::Money(long rub, int cop)

{

this->rub = rub;

this->cop = cop;

}

Money Money::operator-(const Money& a)

{

Money tmp(rub, cop);

tmp.rub = this->rub - a.rub;

tmp.cop = this->cop - a.cop;

while (tmp.cop >= 100)

{

tmp.cop = tmp.cop - 100;

++tmp.rub;

}

return tmp;

}

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

{

this->rub = a.rub;

this->cop = a.cop;

return \*this;

}

Money::~Money()

{

}

int main()

{

setlocale(LC\_ALL, "RUS");

int size;

int index;

cout << "Задайте размер(size): ";

cin >> size;

Variety<int> a(size);

cout << "class Variety a: ";

a.ShowVariety();

cout << endl << "Определение размера множества a: " << a() << endl;

cout << endl << "Задайте индекс(index) для поиска: ";

cin >> index;

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

{

cout << "Доступ по заданному индексу: " << a[index] << endl;

}

else

{

cout << endl << "Доступ по заданному индексу невозможен!" << endl;

}

Variety<int> b(size);

cout << "class Variety b: ";

b.ShowVariety();

Variety<int>c(size);

cout << "class Variety c: ";

c = a - b;

c.ShowVariety();

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl << endl;

Money A(543, 23);

cout << "Первый 'Money': ";

A.ShowMoney();

Money B(542, 22);

cout << "Второй 'Money': ";

B.ShowMoney();

Money С(52, 66);

cout << "Третий 'Money': ";

С.ShowMoney();

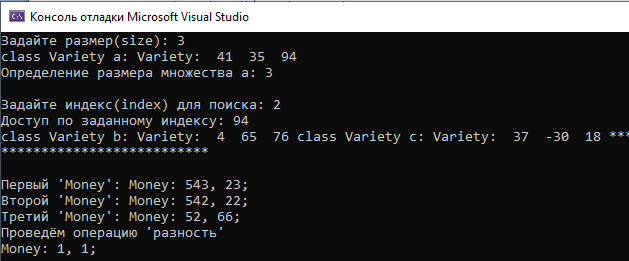
cout << "Проведём операцию 'разность'" << endl;

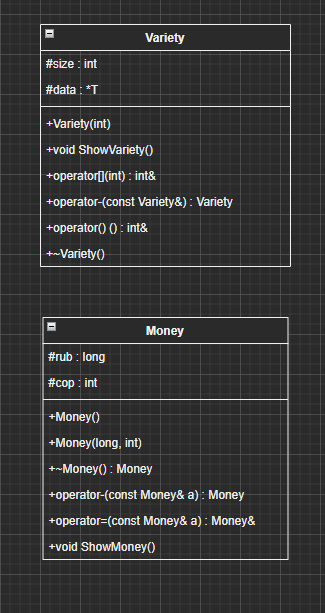
С = A - B;

С.ShowMoney();

}

**Результат**



**UML – диаграмма   
 **

**Контрольные вопросы:**

* 1. **В чем смысл использования шаблонов?**

Шаблоны позволяют обобщать функции и классы для работы с различными типами данных, например если нам потребуется изменить вводимый тип данных для переменной, то во всем коде нам не придется переписывать тип данных там, где используется эта переменная.

* 1. **Каковы синтаксис/семантика шаблонов функций?**

Для создания шаблона нужно написать ключевое слово template, поставить треугольные скобки, в них написать typename (тип данных) и указать название типа данных, например T.

Пример с функцией:

template <typename T>

T abs(T x)

{

if (x<0) return -x;

else return x;

}

* 1. **Каковы синтаксис/семантика шаблонов классов?**

template<параметры\_шаблона>

classимя\_класса

{…};

Пример с классом:

template <class T>

class Point

{

private:

T x,y;

public:

Point(T x, T y)

{

this->x=x;

this->y=y;

}

void Show();

};

template <class T>

Void Point::Show()

{

cout<<”(“<<x<<”,”<<y<<”)”;

}

* 1. **Что такое параметры шаблона функции?**

**Параметры** **шаблона** **являются** **его** **формальными** **аргументами**, а типы тех аргументов, которые используются в конкретных обращениях к функции, служат фактическими аргументами шаблона.

**Пример:**

template<classT> ;ГдеT – параметр шаблона.

* 1. **Перечислите основные свойства параметров шаблона функции.**
* Имена параметров шаблона должны быть уникальными во всем определении шаблона;
* Список параметров шаблона не может быть пустым;
* В списке параметров шаблона может быть несколько параметров, и каждому из них должно предшествовать ключевое слово class;
* Имя параметра шаблона имеет все права имени типа в определенной шаблоном функции;
* Определенная с помощью шаблона функция может иметь любое количество не параметризованных формальных параметров. Может быть непараметризованно и возвращаемое функцией значение;
* В списке параметров прототипа шаблона имена параметров не обязаны совпадать с именами тех же параметров в определении шаблона;
* При конкретизации параметризованной функции необходимо, чтобы при вызове функции типы фактических параметров, соответствующие одинаково параметризованным формальным параметрам, были одинаковы;
  1. **Как записывать параметр шаблона?**

В треугольных скобках после ключевого слова class или typename (название параметра может быть любое, но последующие не могут повторяться)

* 1. **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.

* 1. **Перечислите основные свойства параметризованных классов.**
* Компонентные функции параметризованного класса автоматически являются параметризованными. Их не обязательно объявлять как параметризованные с помощью template;
* Дружественные функции, которые описываются в параметризованном классе, не являются автоматически параметризованными функциями, т.е. по умолчанию такие функции являются дружественными для всех классов, которые организуются по данному шаблону;
* Если friend-функция содержит в своем описании параметр типа параметризованного класса, то для каждого созданного по данному шаблону класса имеется собственная friend-функция;
* В рамках параметризованного класса нельзя определить friend-шаблоны (дружественные параметризованные классы);
* С одной стороны, шаблоны могут быть производными (наследоваться) как от шаблонов, так и от обычных классов, с другой стороны, они могут использоваться в качестве базовых для других шаблонов или классов;
* Шаблоны функций, которые являются членами классов, нельзя описывать как virtual;
* Локальные классы не могут содержать шаблоны в качестве своих элементов;
* Статические переменные шаблонов классов необходимо инициализировать для каждого используемого типа данных.
  1. **Все ли компонентные функции параметризованного класса являются параметризованными?**

Компонентные функции параметризованного класса автоматически являются параметризованными.

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

Дружественные функции, которые описываются в параметризованном классе, не являются автоматически параметризованными функциями, т.е. по умолчанию такие функции являются дружественными для всех классов, которые организуются по данному шаблону.

* 1. **Могут ли шаблоны классов содержать виртуальные компонентные функции?**

Шаблоны функций, которые являются членами классов, нельзя описывать какvirtual.

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

Реализация компонентной функции шаблона класса, которая находится вне определения шаблона класса, должна включать дополнительно следующие два элемента: Определение должно начинаться с ключевого слова template, за которым следует такой же список\_параметров\_типов в угловых скобках, какой указан в определении шаблона класса.

* 1. **Что такое инстанцирование шаблона?**

На каком этапе происходит генерирование определения класса по шаблону?

Процесс порождения функции или класса из шаблона называется инстнацированием