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 №5»**

**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 абстрактный класс.
    2. Определить иерархию классов, в основе которой будет находиться абстрактный класс(см.лабораторнуюработу№4).
    3. Определить класс Вектор, элементами которого будут указатели на объекты иерархии классов.
    4. Перегрузить для класса Вектор операцию вывода объектов с помощью потоков.
    5. В основной функции продемонстрировать перегруженные операции и полиморфизм Вектора.

**Исходные данные**

Вариант 10:

#include <iostream>

using namespace std;

class Object

{

public:

Object();

~Object();

virtual void Print() = 0; //чисто вирутальная функция

};

class Triad : Object

{

protected:

int first;

int second;

int third;

public:

Triad()

{

first = 0;

second = 0;

third = 0;

}

Triad(int f, int s, int t)

{

first = f;

second = s;

third = t;

}

Triad(const Triad& tmp)

{

first = tmp.first;

second = tmp.second;

third = tmp.third;

}

void setFirst(int f)

{

first = f;

}

void setSecond(int s)

{

second = s;

}

void setThird(int t)

{

third = t;

}

int getFirst() { return first; }

int getSecond() { return second; }

int getThird() { return third; }

void IncrementFirst()

{

first = first + 1;

}

void IncrementSecond()

{

second = second + 1;

}

void IncrementThird()

{

third = third + 1;

}

friend istream& operator>>(istream& in, Triad& tmp);

friend ostream& operator<<(ostream& out, const Triad& tmp);

virtual ~Triad()

{

cout << "Delete Triad..." << endl << endl;

}

void Print() override

{

cout << "Вывод тройки чисел: " << first << ", " << second << ", " << third;

}

};

istream& operator>>(istream& in, Triad& tmp)

{

cout << "Введите первое число: "; in >> tmp.first;

cout << "Введите второе число: "; in >> tmp.second;

cout << "Введите третье число(год): "; in >> tmp.second;

return in;

}

ostream& operator<<(ostream& out, const Triad& tmp)

{

return (out << "First NUM: " << tmp.first << ", Second NUM: " << tmp.second << ", Third NUM: " << tmp.third);

}

class Date : public Triad

{

public:

Date()

{

first = 0;

second = 0;

third = 0;

}

Date(int f, int s, int t)

{

first = f;

second = s;

third = t;

}

Date(const Date& tmp)

{

first = tmp.first;

second = tmp.second;

third = tmp.third;

}

void setFirst(int f)

{

first = f;

}

void setSecond(int s)

{

second = s;

}

void setThird(int t)

{

third = t;

}

int getFirst() { return first; }

int getSecond() { return second; }

int getThird() { return third; }

void PlusDays(int n)

{

first = first + n;

while (first > 31)

{

first -= 31;

second += 1;

}

while (second > 12)

{

second -= 12;

third += 1;

}

}

friend istream& operator>>(istream& in, Date& tmp);

friend ostream& operator<<(ostream& out, const Date& tmp);

~Date()

{

cout << endl << "Delete Date..." << endl << endl;

}

};

istream& operator>>(istream& in, Date& tmp)

{

cout << endl << "Сформируем дату: " << endl;

cout << "Введите день: "; in >> tmp.first;

cout << "Введите месяц: "; in >> tmp.second;

cout << "Введите год: "; in >> tmp.third;

return in;

}

ostream& operator<<(ostream& out, const Date& tmp)

{

return (out << endl << "Days: " << tmp.first << ", Month: " << tmp.second << ", Year: " << tmp.third << endl);

}

class Vector

{

private:

Object\*\* beg;//указатель на первый элемент вектора

int size;//размер

int cur;//текущ.позиция

public:

Vector()//конструктор без параметров

{

beg = 0;

size = 0;

cur = 0;

}

Vector(int n)//конструктор с параметрами

{

beg = new Object \* [n];

cur = 0;

size = n;

}

~Vector()//Деструктор

{

if (beg != 0)

{

delete[] beg;

beg = 0;

}

}

void AddObj(Object\* p) //Добавление объекта, на который указывает указатель p в вектор

{

if (cur < size)

{

beg[cur] = p;

cur++;

}

}

friend ostream& operator <<(ostream& out, const Vector& v);

};

ostream& operator<<(ostream& out, const Vector& v)

{

if (v.size == 0)

{

out << "Empty" << endl;

}

Object\*\* p = v.beg; //указатель на указатель типа Object

for (int i = 0; i < v.cur; i++)

{

(\*p)->Print(); //вызов метода Show() - (позднее связывание)

p++;

}

return out;

}

int main()

{

setlocale(LC\_ALL, "RUS");

//cin >> a;

//cout << a << endl;

Vector v(5); //Вектор из пяти элементов

Triad a; //объект класса Triad

cin >> a;

Date b; //объект класса Date

cin >> b;

Object\* p = &a; //ставим указатель на объект класса Car

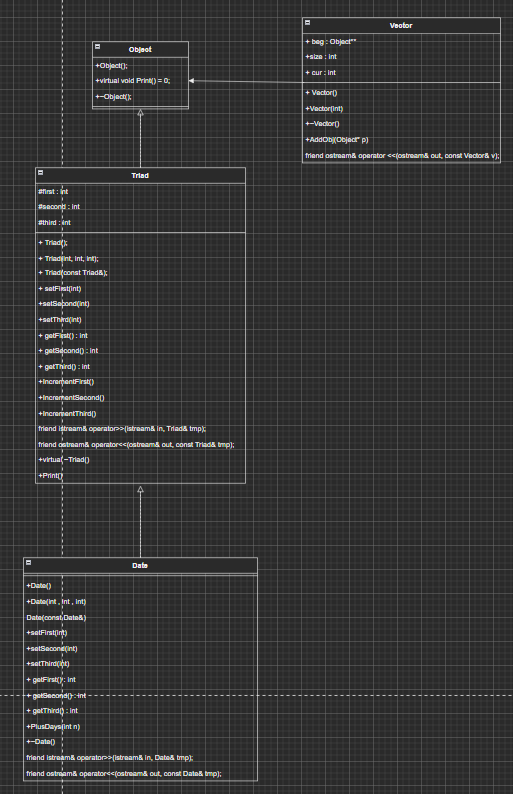
v.AddObj(p);//добавляем объект в вектор

p = &b;//ставим указатель на объект класса Date

v.AddObj(p); //добавляем объект в вектор

cout << v; //вывод вектора

}

**UML-диаграмма**  


**Контрольные вопросы:**

* 1. Какой метод называется чисто виртуальным? Чем он отличается от виртуального метода?

«Виртуальный» означает, что метод может быть переопределен в подклассах, но имеет непосредственную реализацию в базовом классе.

«Чистый виртуальный» означает, что это виртуальный метод без прямой реализации. Такой метод должен быть переопределен хотя бы один раз в иерархии наследования - если у класса есть какие-либо нереализованные виртуальные методы, объекты этого класса не могут быть созданы, и компиляция завершится неудачно.

Чисто виртуальный метод содержит признак=0 вместо тела, например: virtual void f(int)=0; А виртуальный метод в отличие от чисто виртуального еще и описывается.

*Виртуальным*называется метод, ссылка на который вычисляется на этапе выполнения программы.

* 1. Какой класс называется абстрактным?

Класс, который имеет хотя бы один чисто виртуальный метод.  
Базовый класс, объекты которого никогда не будут реализованы называется *абстрактным*классом.

* 1. Для чего предназначены абстрактные классы?

Абстрактные классы предназначены для представления общих понятий, которые предполагается конкретизировать в производных классах

ИЛИ

Такой класс может существовать с единственной целью – быть родительским классом к производным классом, объекты которых будут реализованы.

* 1. Что такое полиморфные функции?

Функции, которые способны работать с объектом любого типа в пределах одной иерархии

* 1. Чем полиморфизм отличается от принципа подстановки?

Принцип подстановки отвечает за спецификацию классов, а полиморфизм за их поведение. (*Полиморфизм*–  возможность замещения методов объекта родителя методами объекта-потомка, имеющих то же имя.)  
  
Чтобы использовать полиморфизм, необходимо чтобы:

1)      все классы-потомки являлись наследниками одного и того же базового класса

2)      функция, реализующая метод,  должна быть объявлена виртуальной в базовом классе

* 1. Привести примеры иерархий с использованием абстрактных классов.

Есть абстрактный класс Food и у него есть чистая виртуальная функция “eat”, которая отвечает за его использование, также существуют классы drinks, fast food, soup и все они являются наследниками абстрактного класса Food, так как в самом классе Food действие для каждого наследника займет очень много кода, мы описываем метод eat в наследнике с ключевым словом override, т.е. задаем новое поведение для метода eat в зависимости от класса и логики взаимодействия с объектом.