ООП. Тема 3: Виртуальные методы

Задание 1. «Домашние питомцы»

1. Создать абстрактный класс Pet, имеющий два чисто виртуальных метода voice и name
2. Создать класс Cat, производный от Pet, в котором определены виртуальные методы voice и name
3. Создать класс Dog, производный от Pet, в котором определены виртуальные методы voice и name
4. Создать:

   объект класса Cat (cat) и указатель на этот объект (pCat);
   объект класса Dog (dog) и указатель на этот объект (pDog);
   указатель на класс Pet (pPet).

5. Вызвать виртуальные методы voice и name:

   для объектов cat и dog;
   через указатели pCat и pDog;
   через указатель pPet, загружая в него адрес объекта cat;
   через указатель pPet, загружая в него адрес объекта dog;

6. Создать класс FamilyPets, содержащий
   - **закрытые данные** – максимальное и реальное количество питомцев, массив ссылок на питомцев (массив указателей на объекты классов Dog и Cat).
   - **открытые**:
     конструктор, создающий «пустую» семью
     метод voice, «прослушивание» голосов всех членов FamilyPets
     метод name, вывод имен всех членов FamilyPets
     перегруженную операцию + для добавления нового члена FamilyPets
   - **дружественную функцию**:
     проверяющую есть ли член FamilyPets с указанным именем и определяющую, *к какому классу он относится (может быть несколько одинаковых имен)*
     перегружающую операцию >> для вывода информации о всех членах FamilyPets

7. Создать функцию main для тестирования разработанных классов

Задание 2. Что будет выведено?

1) Пусть объявлены классы:

```
class A
{
    private:
    A() { std::cout << "A() "; }
};
class B
{
public:
    A a1;
```

```
    B() {std:: cout << "B() "; }
};
```

Выполнение программы начинается с создания объектов:

```
A am;
B bm;
```

ошибка коомпиляции(закрытый конструктор)

Что будет выведено?

2) Пусть объявлены классы:

```
class A
{
   public:
    A() { std::cout << "A() "; }
};
class B
{
public:
    A a1;
    B() {std:: cout << "B() "; }
};
```

A()
A()
B()

Выполнение программы начинается с создания объектов:

```
A am;
B bm;
```

Что будет выведено?

Задание 3. Укажите порядок вызова конструкторов и деструкторов?

1) Пусть объявлен класс:

```
class A
{
public:
    A(int p) :     p_(p) {  std::cout << "A(p)_";   }
    A(const A& copy) : p_(copy.p_) { std::cout << "A(const A&)_"; }
    ~A() {  std::cout << "~A()_";   }
    int getP() const { return p_; } ;
    void setP(int p) { p_ = p; }

private:
     int p_;
};

Определена функция:
bool isEqual(A x, A y)
{
    return x.getP() == y.getP();
}
```

A(p)_
A(p)_
A(const A&)_
A(const A&)_
A(const A&)_
^A()_
^A()_
^A()_
^A()_
^A()_

Созданы объекты и выполнена функция:

```
{
  A ob1(3);
  A ob2(5);
  A ob3(ob1);
  bool result = isEqual(ob1, ob2);
}
```

Укажите порядок вызова конструкторов и деструкторов.

2) Пусть объявлен класс:

```
class A
{
public:
    A(int p) :     p_(p) {  std::cout << "A(p)_";   }
    A(const A& copy) : p_(copy.p_) { std::cout << "A(const A&)_"; }
    ~A() {  std::cout << "~A()_";   }
    int getP() const { return p_; } ;
     void setP(int p) { p_ = p; }

private:
     int p_;
};
```

Определена функция:

```
bool isEqual(const A& x, const A & y)
{
    return x.getP() == y.getP();
}
```

A()_
A()_
A(const &A)_
^A()_
^A()_

Созданы объекты и выполнена функция:

```
{
  A ob1(3);
  A ob2(5);
  A ob3(ob1);
  bool result = isEqual(ob1, ob2);
}
```

Укажите порядок вызова конструкторов и деструкторов.

Задание 4. Ответьте на вопросы:

1) Сколько аргументов требуется для определения перегруженной унарной операции?
2) Можно ли при перегрузке изменить приоритет операции?
3) Истинно ли следующее утверждение: перегруженная операция требует на один аргумент меньше, чем количество операндов?
4) При каких способах передачи параметра в функцию объект-параметр может изменить свое состояние?

1) 1
2) нет
3) зависит от реализации
4) по ссылке, по значению