

Наследование

- **Наследование** – это способ повторного использования кода, при котором новые классы создаются на баз существующих классов путем заимствования их элементов (данных и функций).
- Новый класс объявляется **наследником** ранее определенного класса (**базового класса**).
- Новый класс называют **производным классом**.
- Производный класс может стать базовым для других классов.
- Таким образом создается **иерархия** классов.

Наследование

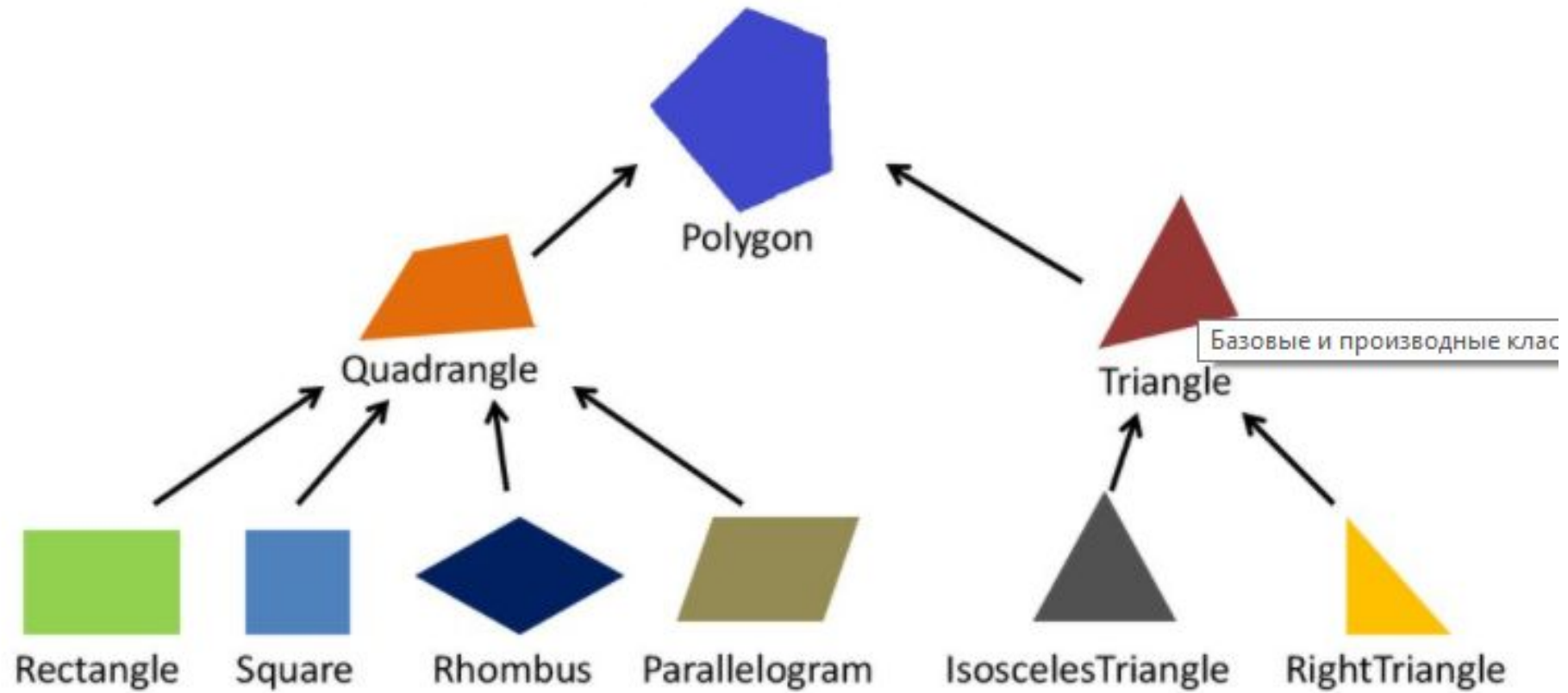
Наследование является мощнейшим инструментом ООП и применяется для следующих взаимосвязанных целей:

- исключения из программы повторяющихся фрагментов кода;
- упрощения модификации программы;
- упрощения создания новых программ на основе существующих.
- Кроме того, наследование является единственной возможностью использовать объекты, исходный код которых недоступен, но в которые требуется внести изменения.

Наследование.

- Объект производного класса обладает всеми методами и атрибутами базового. Помимо них, в него можно добавить новые.
- Производный класс может быть базовым для какого-то другого класса, т.е. иерархия классов может быть сколь угодно глубокой.

Базовые и производные классы



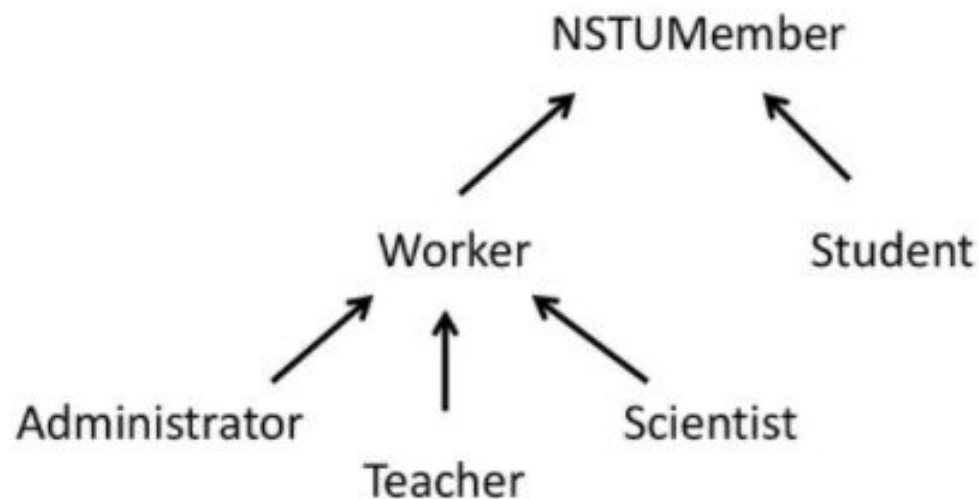
- Наследование порождает иерархические древовидные структуры данных. В них всегда можно добавить новый класс в требуемом месте.

Наследование

- Наследование может быть:
 - **Простое** (производный класс – прямой потомок только одного базового класса).
 - **Множественное** (производный класс – прямой потомок нескольких базовых классов).

Базовые и производные классы

Базовые и производные классы



- Синтаксис определения наследования:

```
class Worker : public NSTUMember { ...
```

- Класс может быть базовым прямо и косвенно:

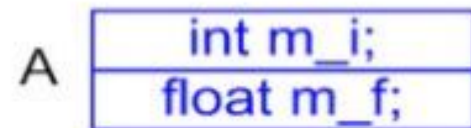
Для `Teacher` прямым базовым классом является `Worker`, а косвенным – `NSTUMember`.

Наследование. Пример.

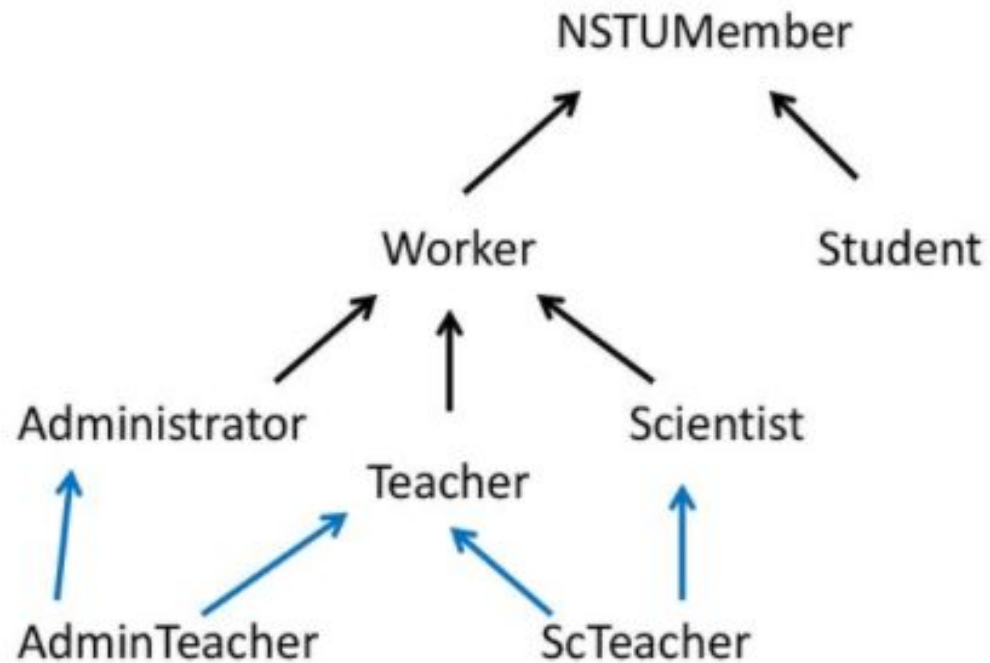
```
class A{
    int m_i;
    float m_f;
};

class B: A{
    double m_d;
};

class C: B{
    long m_l;
};
```



Множественное наследование



```
class ScTeacher : public Teacher, public Scientist { ...
```

```
class AdminTeacher : public Teacher,
```

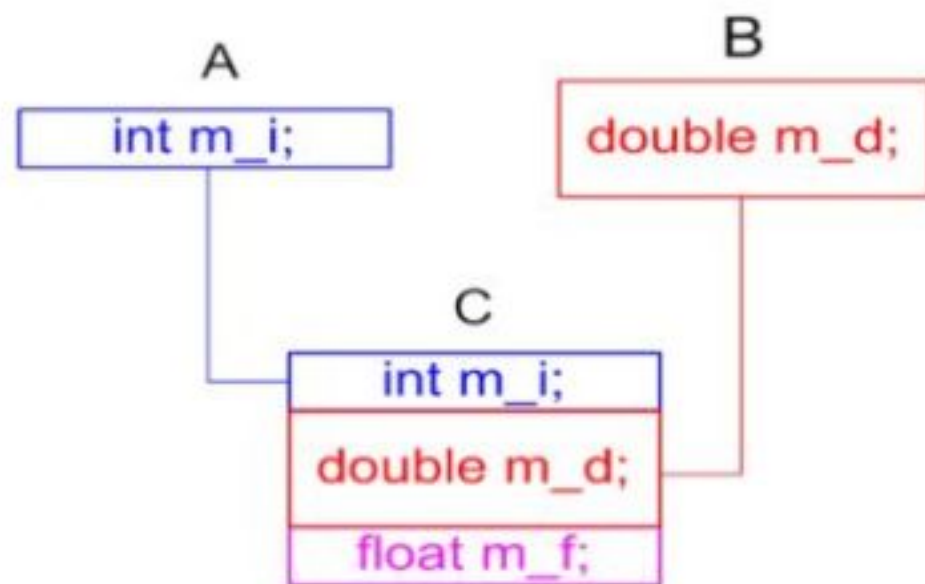
```
public Administrator { ...
```


Множественное наследование.

```
class A{
    int m_i;
};

class B {
    double m_d;
};

class C: A, B{
    float m_f;
};
```



Модификаторы доступа.

- *public*
Поле/метод с таким модификатором доступно отовсюду (из самого класса, из его потомков, из глобальных функций).
- *protected*
Доступно из самого класса и производных от него, но недоступно извне.
- *private*
Доступно только из самого класса.

Модификаторы доступа.

Пример.

```
class A{
    private:
        int m_priv;
    protected:
        int m_prot;
    public:
        int m_pub;
};
class B: public A;
class C: protected A;
class D: private A;
```

- `m_priv` доступно только из класса `A`;
- `m_prot` доступно из классов `A` и производных от него (в `B`, `C` это поле остается `protected`, в `D` становится `private`);
- `m_pub` доступно из классов `A`, в производных от него, а также извне. В классе `B` это поле остается `public`, в классе `C` становится `protected`, в классе `D` – `private`.

Виртуальные функции.

- Виртуальные функции используются для того, чтобы можно было работать с объектами разных типов так, как будто они одного типа.
- Выбор функции, которую необходимо вызвать, производится во время исполнения (не во время компиляции).

Виртуальные функции. Пример.

```
class Shape {  
    int cx, cy;  
    virtual void print();  
};
```

```
class Circle: Shape {  
    int r;  
    virtual void print();  
};
```

```
class Rect: Shape {  
    int w, h;  
    virtual void print();  
};
```

- Пусть у нас есть массив `Shape* shapes[]`;
- Несмотря на то, что тип элементов массива – `Shape*`, т.е. указатель на объект типа `Shape`, реально по указателю может лежать любой объект.
- Если мы напишем такой цикл, то вызываться будет функция `print` того типа, который на самом деле находится по адресу `shapes[i]`:

```
for(int i = 0; i < N; i++) {  
    shapes[i]->print();  
}
```

Общее задание

- Определить иерархию классов (в соответствии с вариантом – выделить базовый и производные).
- Реализовать классы (самостоятельно задать члены-данные и методы класса).
- Написать демонстрационную программу, в которой создаются объекты различных классов.

Классы: человек (имя, дата рождения), абитуриент (количество баллов), студент (курс, группа, факультет), студент-магистрант (тема диссертации)

1) Классы – автомобиль (марка, номер), поезд (номер, количество вагонов, количество пассажиров в вагоне), транспортное средство (средняя скорость, вид топлива, год выпуска)

3) Классы – растение (название, вид), дерево (возраст), цветок (длина стебля), роза (цвет)

2) Классы – млекопитающие (год), парнокопытные (среда обитания), птицы (хищники), животное (вид, род, вес)

4) Классы – печатное издание (издательство, год, название), журнал (номер, месяц), книга (тематика, автор, количество страниц), учебник (назначение)