1. Для создания дочерних классов, которые имеют общие свойства с базовым классом
2. Член класса может использоваться любой функцией, которая является членом данного или производного класса. Также к членам возможен доступ извне через имя объекта
3. Член класса может использоваться только функциями – членами данного класса и функциями друзьями своего класса. В производном классе он недоступен
4. Тоже самое, что и Private, но также член класса с данным атрибутом доступа может использоваться функциями-членами и функциями друзьями классов, производных от данного
5. Например, так:

class Man : public People

{список компонентов класса}

1. Не наследуются
2. В
3. Сначала конструируется базовый класс, потом компоненты-объекты (если присутствуют), потом производный класс
4. Сначала уничтожается производный класс, потом его компоненты-объекты, после этого базовый класс
5. Виртуальные функции используются в тех случаях, когда в каждом производном классе требуется свой вариант компонентной функции. Любая нестатическая функция базового класса может быть сделана виртуальной, для чего используют ключевое слово virtual.

Связывание — это процесс, который используется для конвертации идентификаторов (таких как имена переменных или функций) в адреса. В некоторых программах невозможно знать наперёд, какая функция будет вызываться первой. В таком случае используется позднее связывание (или «динамическая привязка»). В C++ для выполнения позднего связывания используются указатели на функции

1. Конструкторы не могут. Деструкторы могут, при этом почти каждый класс, имеющий виртуальную функцию, должен иметь виртуальный деструктор
2. Да
3. Производный класс является подтипом базового, он замещает реализации всех функций-членов, специфичных для типа базового класса и наследует общие для типа и подтипа функции
4. Производный класс напрямую не поддерживает открытый интерфейс базового, но используется его реализацией, предоставляя свой собственный открытый интерфейс
5. Наследующий класс должен дополнять, а не замещать поведение базового класса
6. int stage; string post; string name
7. Student::Student()

{

}

Employee::Employee()

{

}

Teacher::Teacher()

{

}

1. Student::Student(string N, int A)

{

Name = N;

Age = A;

}

Employee::Employee(string N, int A)

{

Name = N;

Age = A;

}

Teacher::Teacher(string N, int A)

{

Name = N;

Age = A;

}

1. Student::Student(const Student& student)

{

Name = student.name;

Age = student.age;

}

Employee::Employee(const Employee& employee)

{

Name = employee.name;

Age = employee.age;

}

Teacher::Teacher(const Teacher& teacher)

{

Name = teacher.name;

Age = teacher.age;

}

1. Student& Student::operator = (const Student& s)

{

if (&s == this) return \*this;

name = s.name;

age = s.age;

return \*this;

}

Employee& Employee::operator = (const Employee & e)

{

if (&e == this) return \*this;

name = e.name;

age = e.age;

return \*this;

}

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

{

if (&t == this) return \*this;

name = t.name;

age = t.age;

return \*this;

}