继承

继承的定义和作用

继承是一种允许我们创建一个类（派生类）来继承另一个类（基类）属性和方法的机制。这意味着可以在不修改原有类的情况下，扩展其功能。继承的主要好处包括：

- **代码复用**：通过继承，可以复用基类中的代码，减少重复代码编写。
- **设计灵活性**：继承使得修改和扩展功能变得更加容易，有助于构建灵活的系统架构。
- **基类**：也称为父类，是其他类继承的类。
- **派生类**：也称为子类，继承自基类的类。
- 继承体现了类间的层次关系，如同生物学中的父子关系。
- 子类自动获得父类的所有成员变量和成员函数，但可以添加或覆盖（重写）自己的方法和属性。

继承的语法

在C++中，继承的语法如下：

```
class BaseClass {
    // 基类的成员
};
class DerivedClass :  BaseClass {//默认为Private
    // 派生类的成员
};
class DerivedClass : public BaseClass {
    // 派生类的成员
};
class DerivedClass : Protected BaseClass {
    // 派生类的成员
};
class DerivedClass : Private BaseClass {
    // 派生类的成员
};
```

访问控制和继承类型

继承类型决定了基类成员在派生类中的访问权限。C++支持三种继承类型：

- **公有继承（Public）**：基类的公有成员和保护成员在派生类中保持其访问级别。
- **保护继承（Protected）**：基类的公有成员和保护成员在派生类中都成为保护成员。
- **私有继承（Private）**：基类的公有成员和保护成员在派生类中都成为私有成员。

示例

```
class Base {
public:
    int publicMember;
protected:
    int protectedMember;
private:
    int privateMember;
```

```
};

class DerivedPublic : public Base {
    // publicMember 是公有的
    // protectedMember 是保护的
    // privateMember 不可访问
};

class DerivedProtected : protected Base {
    // publicMember 是保护的
    // protectedMember 是保护的
    // privateMember 不可访问
};

class DerivedPrivate : private Base {
    // publicMember 是私有的
    // protectedMember 是私有的
    // privateMember 不可访问
};
```

各种权限梳理

类的权限

类的定义

```
class ClassName {
public:
    // 公有成员
protected:
    // 受保护成员
private:
    // 私有成员
};
```

访问修饰符

- **public**：公有成员可以在任何地方被访问，包括类外部和派生类。
- **protected**：受保护成员仅能在类的内部及其派生类中访问。
- **private**：私有成员只能在类的内部访问，派生类和类外部均无法直接访问。
- 类的权限是限制类对象的。
- 类内没有权限的概念

实战例程

```
#include <cstdlib>
#include <iostream>
using namespace std;
class Parent
{
private:
    int a = 1000;
protected:
    int b = 2000;
public:
```

```
    void print_a() { cout << "Parent print_a a = " << a << endl; };
    void print_b() { cout << "Parent print_a b = " << b << endl; };
};
int main(int argc, char *argv[])
{
    Parent parent;

    parent.print_a();
    parent.print_b();
//    printf("parent.b = %d\n", parent.b);
//    printf("parent.a = %d\n", parent.a);
    return EXIT_SUCCESS;
}
```

log:

Parent print_a a = 1000
Parent print_a b = 2000

说明:

在print_a()访问了私有成员a，说明在类中是没有权限的概念。所用成员都可以访问

实例化对象parent后可以调用共有成员函数打印a和b的值

实战例程

```
#include <cstdlib>
#include <iostream>
using namespace std;
class Parent
{
private:
    int a = 1000;
protected:
    int b = 2000;
public:
    void print_a() { cout << "Parent print_a a = " << a << endl; };
    void print_b() { cout << "Parent print_a b = " << b << endl; };
};
int main(int argc, char *argv[])
{
    Parent parent;

    parent.print_a();
    parent.print_b();
    printf("parent.b = %d\n", parent.b);
    printf("parent.a = %d\n", parent.a);
    return EXIT_SUCCESS;
}
```

log:

```
error C2248: “Parent::a”: 无法访问 private 成员(在“Parent”类中声明)
[C:\work\c++base\build\MyProject.vcxproj]
error C2248: “Parent::b”: 无法访问 protected 成员(在“Parent”类中声明)
```

说明：

1. 实例化对象parent后
2. 可以调用共有成员函数打印a和b的值
3. 没有权限调用保护成员和私有成员

继承的权限问题

要搞懂继承的权限，首先要明白类的权限

1. 子类只是继承了父类，本质上他还是类，是类就得遵循类权限的规则，这个道理很重要
2. 子类继承了父类的成员，子类的成员等于父类的成员加自己的成员，
3. 子类的成员通过类的定义，可以清晰的知道，成员的访问权限
4. 从父类继承来的成员访问权限不总是保持原有的访问权限，而是根据继承的类型有所改变。
5. **公有继承（Public）**：父类的公有成员和保护成员在子类中保持其访问级别。
6. **保护继承（Protected）**：父类的公有成员和保护成员在子类中都成为保护成员。
7. **私有继承（Private）**：父类的公有成员和保护成员在子类中都成为私有成员。

父类中的成员	**父类成员在子类中的权限**	**父类成员在子类中的权限**	**父类成员在子类中的权限**	**父类成员**	**父类成员**	**父类成员**
--	公有继承	保护继承	私有	在父类内部	在子类内部	在类外（类对象）
公有	公有	保护	私有	✔ 可访问	✔ 可访问	✔ 可访问
私有	私有	私有	私有	✔ 可访问	✘ 不可访问	✘ 不可访问
保护	保护	保护	私有	✔ 可访问	✔ 可访问	✘ 不可访问

实战示例 3.app 公有继承

```
#include <cstdlib>
#include <iostream>
using namespace std;
class Parent
{
private:
    int a = 1000;
protected:
    int b = 2000;
public:
    int c = 3000;
    void print_a() { cout << "Parent print_a a = " << a << endl; };
    void print_b() { cout << "Parent print_a b = " << b << endl; };
};
class Child : public Parent
```

```
{
public:
    Child() { b = 4000; }
    // void print_a() { cout << "Child print_b b = " << a << endl; };
    //error C2248: “Parent::a”: 无法访问 private 成员(在“Parent”类中声明)
    void print_b() { cout << "Child print_b b = " << b << endl; };
    void print_c() { cout << "Child print_c c = " << c << endl; };
};
int main(int argc, char *argv[])
{
    Parent parent;
    Child child;
    parent.print_a();
    parent.print_b();
    child.print_a();
    child.print_b();
    child.c = 5000;
    child.print_c();
    // child.b=100;
    // error C2248: “Parent::b”: 无法访问 protected 成员(在“Parent”类中声明)
    return EXIT_SUCCESS;
}
```

log:

```
Parent print_a a = 1000
Parent print_a b = 2000
Parent print_a a = 1000
Child print_b b = 4000
Child print_c c = 5000
```

这段代码定义了两个类：`Parent`和`Child`。`Child`类是`Parent`类的子类，通过公共继承(`public Parent`)实现。

类有三个成员变量：`a`、`b`和`c`。

其中`a`是私有的，只能在`Parent`类内部访问；

是受保护的，可以在`Parent`类和其所有子类（如`Child`）内部访问；

是公共的，可以在任何地方访问。

`Parent`类还有两个公共方法：`print_a()`和`print_b()`，它们分别打印`a`和`b`的值。

类继承了`Parent`类，因此它可以访问`Parent`类的公共和受保护成员。

在`Child`的构造函数中，它修改了继承自`Parent`的受保护成员`b`的值。

`Child`类还定义了两个公共方法：`print_b()`和`print_c()`，它们分别打印`b`和`c`的值。

在`main`函数中，首先创建了`Parent`类的一个对象`parent`和`Child`类的一个对象`child`。

然后，调用`parent`的`print_a()`和`print_b()`方法，以及`child`的`print_a()`、`print_b()`和`print_c()`方法

注意，`child.print_a()`实际上调用的是`Parent`类的`print_a()`方法，因为`Child`类没有重写这个方法。

最后，修改了`child`的公共成员`c`的值，并打印出来。

这段代码中注释掉的`child.b=100;`会导致编译错误，因为`b`是受保护的成员，不能在类的外部直接访问。

同样，注释掉的`Child`类的`print_a()`方法也会导致编译错误，因为它试图访问`Parent`类的私有成员`a`。

公有继承总结：

1. 父类中可以访问父类的所有成员
2. 子类中可以访问父类的公有成员和 保护成员，
3. 子类中不能访问父类中的私有成员
4. 类外只能访问到公有成员
5. 因为是公有继承父类中的公有成员在子类中还是声明成公有访问，所以类外可以被访问到

实战示例4.cpp 保护继承

```
#include <cstdlib>
#include <iostream>
using namespace std;
class Parent
{
private:
    int a = 1000;
protected:
    int b = 2000;
public:
    int c = 3000;
    void print_a() { cout << "Parent print_a a = " << a << endl; };
    void print_b() { cout << "Parent print_a b = " << b << endl; };
};
class Child : protected Parent
{
public:
    Child() { b = 4000; }
    void print_b() { cout << "Child print_b b = " << b << endl; };
    void print_c() { cout << "Child print_c c = " << c << endl; };
};

int main(int argc, char *argv[])
{
    Parent parent;
    Child child;
    parent.print_a();
    parent.print_b();
    // child.print_a();//error C2248: “Parent::print_a”: 无法访问 protected 成员
(在“Child”类中声明)
    child.print_b();
    // child.c = 5000; //error C2248: “Parent::c”: 无法访问 protected 成员
(在“Child”类中声明)
    child.print_c();
    return EXIT_SUCCESS;
}
```

log:

```
Parent print_a a = 1000
Parent print_a b = 2000
Child print_b b = 4000
Child print_c c = 3000
```

这段代码定义了两个类：`Parent` 和 `Child`。`Child` 类是从 `Parent` 类中以保护方式继承的。

在 `Parent` 类中，我们有三个成员变量：`a`、`b` 和 `c`，

它们的访问级别分别是 `private`、`protected` 和 `public`。

我们还有两个公开的成员函数 `print_a` 和 `print_b`，它们分别打印 `a` 和 `b` 的值。

在 `Child` 类中，我们重写了 `Parent` 类中的 `b` 变量，并在构造函数中将其值设置为 `4000`。

我们还定义了两个公开的成员函数 `print_b` 和 `print_c`，它们分别打印 `b` 和 `c` 的值。

在 `main` 函数中，我们创建了 `Parent` 和 `Child` 的实例，并调用了它们的 `print` 函数。

注意，尽管 `Child` 类从 `Parent` 类继承，但是我们不能直接访问 `Parent` 类中的 `print_a` 函数和 `c` 变量，

因为它们在 `Child` 类中被声明为 `protected`。所以在类外时不能被访问的

这就是为什么 `child.print_a()` 和 `child.c = 5000` 这两行代码被注释掉，并且注释说明了它们会导致错误的原因。

保护继承总结：

1. 父类中可以访问父类的所有成员
2. 子类中可以访问父类的公有成员和 保护成员，
3. 子类中不能访问父类中的私有成员
4. 类外只能访问到公有成员
5. 因为是保护继承父类中的公有成员在子类声明成保护访问，所以在类外不可以被访问到

私有继承说明：

私有继承父类中的成员在子类中被声明成私有，

子类使用的效果和保护继承是相同的。

不同点：

保护继承：保护继承后的子类在被孙子类继承时在孙子类中是可以访问父类中的保护成员和私有成员的

私有继承：因为此时父类中的成员都被声明成子类私有成员了，私有继承后的子类在被孙子类继承时，，在孙子类中不可以访问父类中的保护成员和私有成员，因为在子类中他们已经是私有成员了

父承的概念回顾

- **父子关系**：继承描述了类之间的层次结构，其中子类（派生类）继承了父类（基类）的属性和行为。
- **成员继承**：子类自动拥有父类的公有成员变量和成员函数，以及受保护成员。
- **访问级别**：
- public继承保持原访问级别，
- private继承将父类所有成员变为private，
- protectede继承将父类中公有成员变为protected。
- **多态**：通过虚函数，子类可以覆盖父类方法，实现动态绑定。

访问级别设置原则：

- 对外公开的成员设为public
- 类内专用的成员设为private
- 类和子类间共享的设为protected
- private成员在子类中存在但不可访问