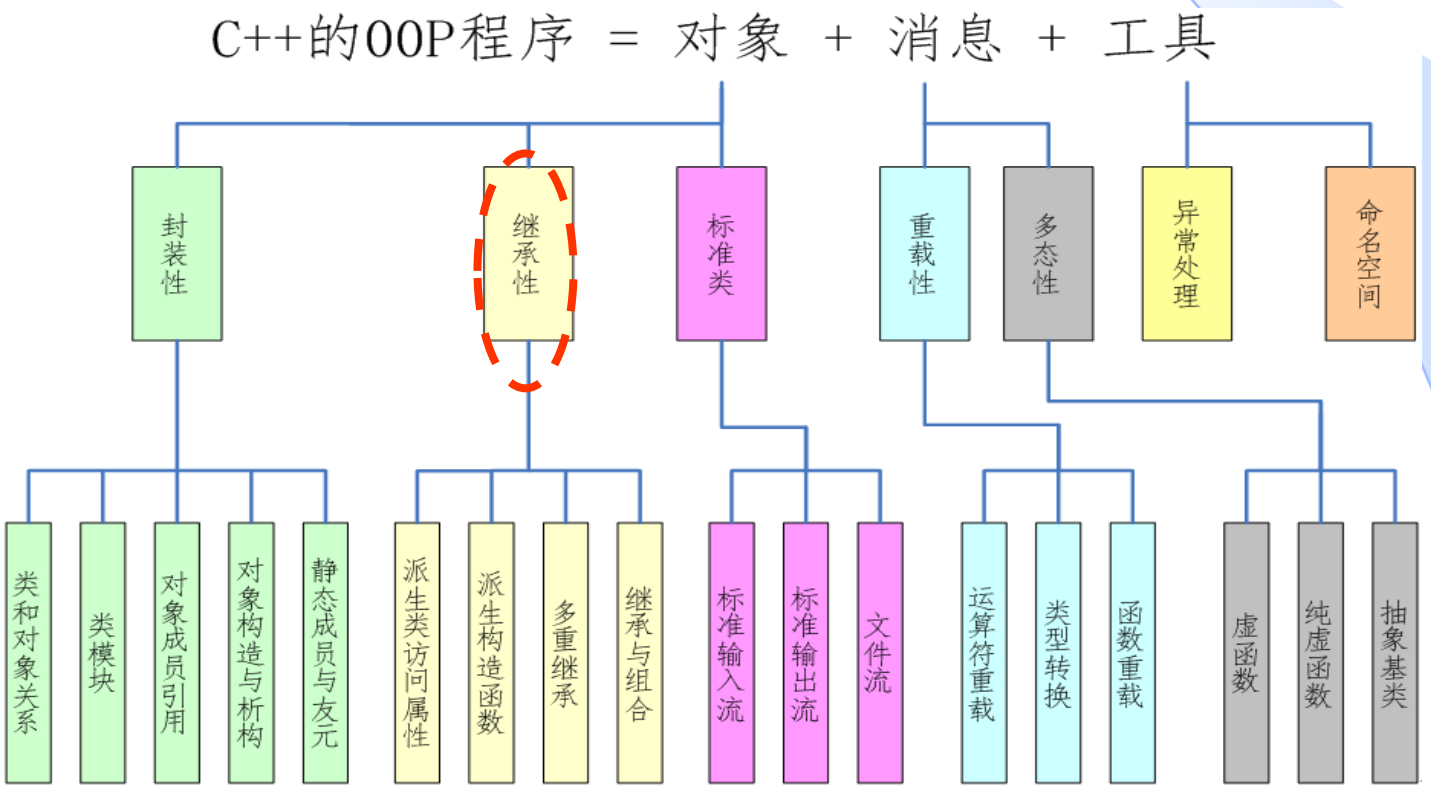
第6 讲 继承与派生（上）

6.1 继承与派生的概念

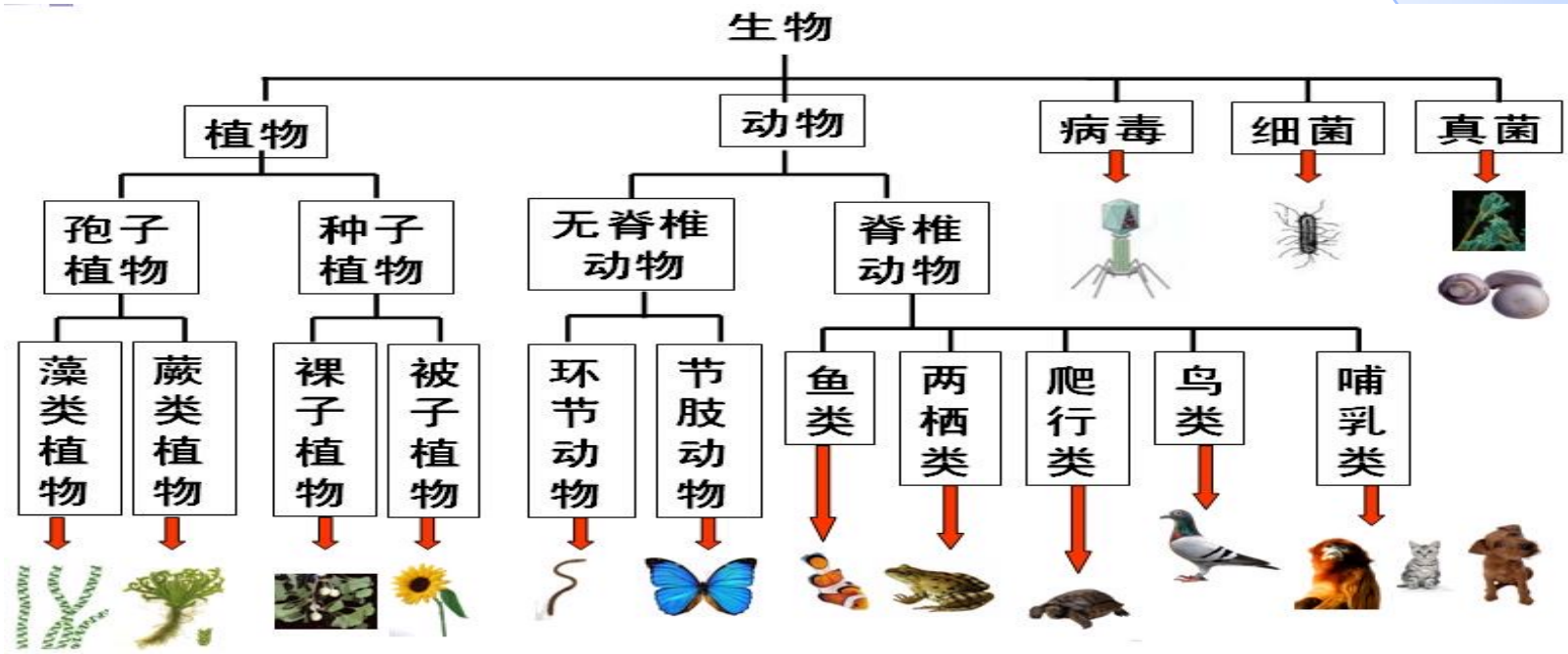
6.2 派生类的声明和构成

6.3 派生类成员的访问属性

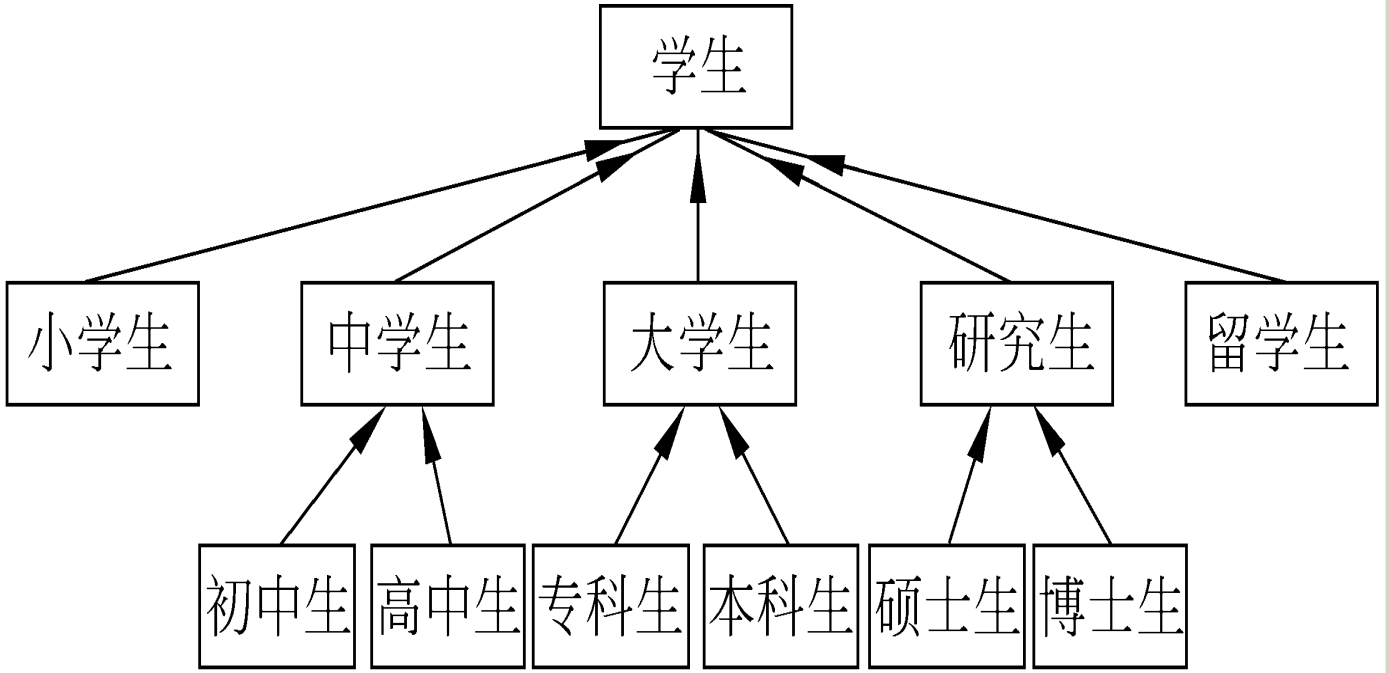


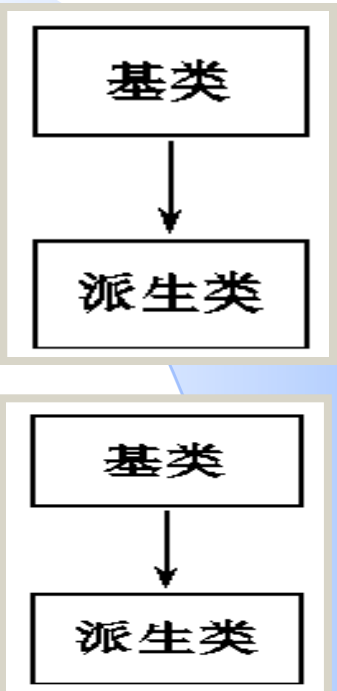
6.1 继承与派生的概念

- 问题提出？
  - ✓如果类A中包含了若干数据成员和成员函数。新类B中包含了A类中部分相同成员。如何高效地申明新类B呢？
  - ✓C++提供了继承机制，继承(inheritance)体现C++可重用性；
  - ✓继承法则也体现了大自然的基本规律。

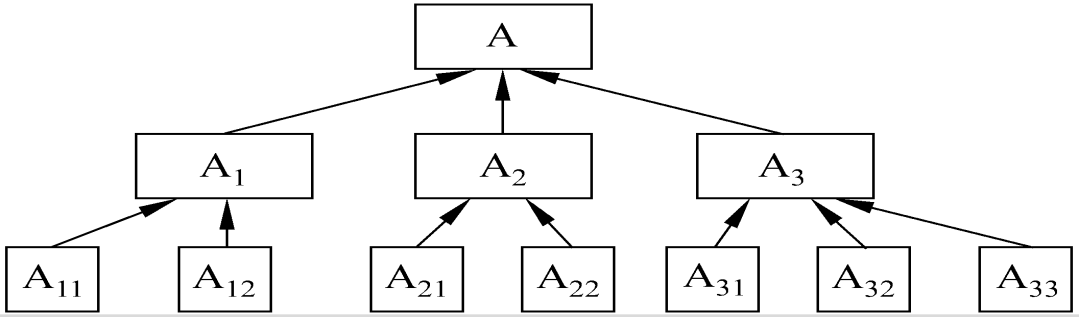


- 类继承：新类（派生类）从已有类（基类）获得其特性；
- 类派生：已有父类（基类）产生新的子类（派生类）；
- 一个基类可派生出多个派生类，每一个派生类又可作为基类再派生出新的派生类，基类和派生类是相对而言。

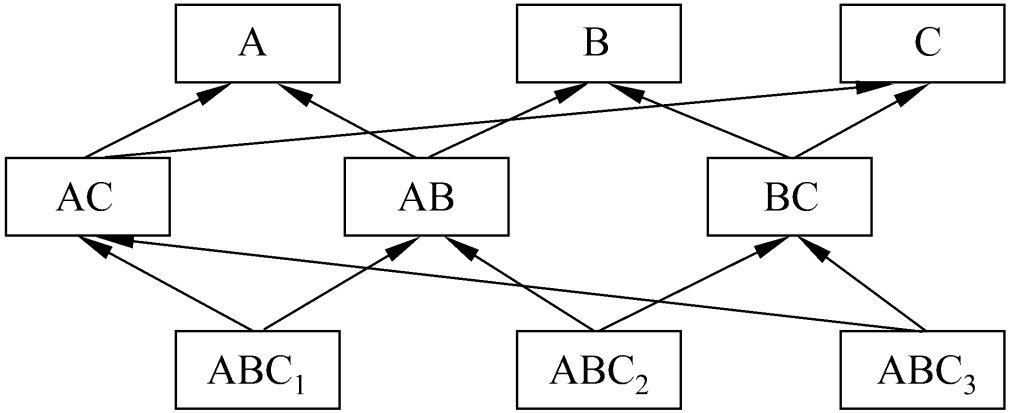


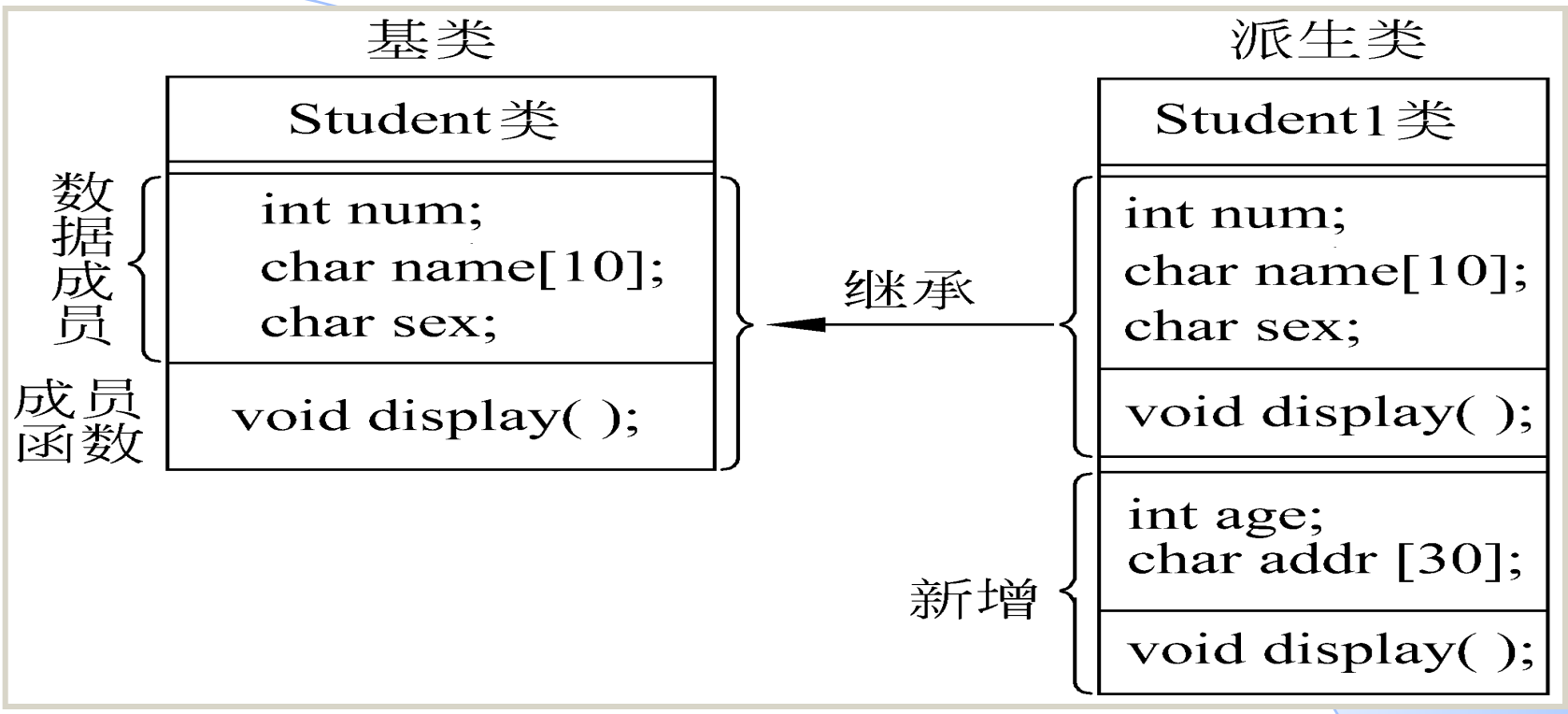


- 单继承(single inheritance):一个派生类只从一个基类派生，这种继承关系所形成的层次是一个树形结构；



- 多重继承(multiple inheritance):一个派生类可以从多个基类派生；






```
1  class Student1: public Student//声明基类是Student
2  {public:
3  void display_1( )   //新增加的成员函数
4  {cout<<"age: "<<age<<endl;
5  cout<<"address: "<<addr<<endl;}
6  private:
7  int age;          //新增加的数据成员
8  string addr; };  //新增加的数据成员
9
```

```
3  class student
4  {public:
5  void display();
6  private:
7    int num;
8    char name[10];
9    char sex;
10 };
```

6.2 派生类的声明和构成

声明派生类的一般形式：

```
class 派生类名：［继承方式］ 基类名
{新增加的成员
  } ;
```

继承方式：public,private和protected，默认为private

继承性质	基类中成员访问权限	基类成员在派生类中的访问权限的演变
public	public protected private	public protected 不可访问
protected	public protected private	protected protected 不可访问
private	public protected private	private private 不可访问

派生类的构造

(1) 遗传：从基类接收成员。派生类把基类全部的成员(不包括构造函数和析构函数)接收过来；

(2) 变异：在声明派生类时改变基类访问属性；增加新成；同名屏蔽；

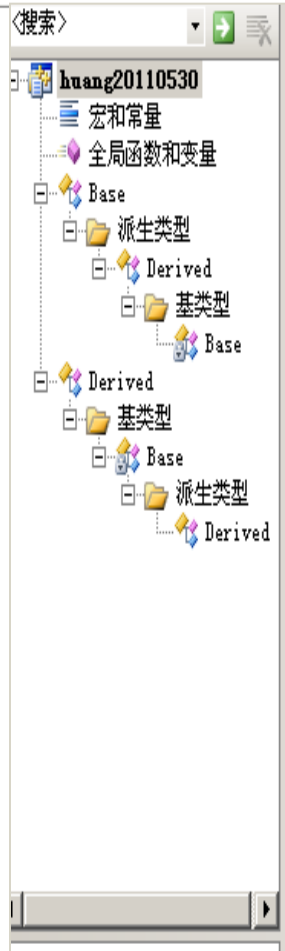
(3) 派生类的构造函数和析构函数。构造函数和析构函数是不能从基类继承，另外定义。

面向对象程序设计核心思想之一

（1）声明一个抽象基类，只提供某些最基本的功能；

（2）在声明派生类时加入某些具体的功能，形成适用于某一特定应用的派生类。

```cpp
#include<iostream>
;using namespace std;
class Base
{public:
   void setx( int n)
   {x=n;}
   void showx()
   {cout<<x<<endl;}
private:
   int x;
};
class Derived: private Base{
   public:
   void setxy(int n, int m)
   {setx(n);
    y=m;}
   void showxy()
   { cout<<x;
   cout<<y<<endl;}
private:
   int y;
};
int main( )
{  Derived obj;
   obj.setx(10);
   obj.showx();
   obj.setxy(20,30);
   obj.showxy();
   return(0);}
```




```
uang20110530\huang20110530\huang.cpp(10) : 参见“Base::x”的声明
uang20110530\huang20110530\huang.cpp(4) : 参见“Base”的声明
0530\huang20110530\huang.cpp(25) : error C2247: “Base::setx”不可访问，
uang20110530\huang20110530\huang.cpp(5) : 参见“Base::setx”的声明
uang20110530\huang20110530\huang.cpp(12) : 参见“Derived”的声明
uang20110530\huang20110530\huang.cpp(4) : 参见“Base”的声明
0530\huang20110530\huang.cpp(26) : error C2247: “Base::showx”不可访问，
```

```cpp
#include<iostream>
;using namespace std;
class Base
{public:
   void setx( int n)
   {x=n;}
   void showx()
   {cout<<x<<endl;}
private:
   int x;
};
class Derived: private Base{
   public:
   void setxy(int n, int m)
   {setx(n);
    y=m;}
   void showxy()
   {showx();
   cout<<y<<endl;}
private:
   int y;
};
int main( )
{  Base obj;
   Derived obj1;
   obj.setx(10);
   obj.showx();
   obj1.setxy(20,30);
   obj1.showxy();
   return(0);}
```

6.3 派生类成员的访问属性

公用继承（public)的派生类访问属性调整

表 11.1　公用基类在派生类中的访问属性

公用基类的成员	在公用派生类中的访问属性
私有成员	不可访问
公用成员	公用
保护成员	保护

可类外、类内访问

只能类内访问

派生类成员的访问方式

(1) 类内:在派生类内可直接访问基类public、protected成员;

(2) 类外:在派生类外（通过对象）能访问基类的public成员;

(3) 派生类新增成员按照访问属性来限定。

```
1.  class Employee {
2.      String name; //name是私有成员，在一切派生类中均不可见
3.      Date birthday;
4.      char sex;
5.      short department;
6.  public:
7.      Employee();
        String getName() const { return name; }
8.      void Print() const
9.      { cout << name << " is an employee" << endl; }
10. };
11. class Manager : public Employee {
12. public:
13.     Manager();
14.     void Print() const //覆盖基类同名同类型成员函数
15.     {
16.         cout << name << "is a manager" << endl;//错误
17.         cout << getName() << " is a manager" << endl;//正确
18.     }
19. };
```

```
1.  class Employee {
2.  protected:
3.      String name; //name是保护成员，向一切派生类开放
4.      Date birthday;
5.      char sex;
6.      short department;
7.  public:
8.      Employee();
9.      void Print() const
10.     { cout << name << " is an employee" << endl; }
11. };

12. class Manager : public Employee {
13. public:
14.     Manager();
15.     void Print() const
16.     {   cout << name << "is a manager" << endl; //正确
17.     }
18. }
```

```
1.  class Employee {
2.      String name;
3.      Date birthday;
4.      char sex;
5.      short department;
6.  public:
7.      Employee();
8.      void Print() const
9.      { cout << name << endl; }
10. };

11. class Manager : public Employee {
12.     short level;
13. public:
14.     Manager();
15.     void Print() const
16.     {   Employee::Print();      //打印Employee部分的信息
17.         cout << level << endl; //打印Manager特有信息
18.     }
19. }
```

```
void main()
{
    Employee e1;
    Manager m1;
    ...
    e1.Print();
    m1.Print();
}
```

必须用Employee::表明调用基类的Print()函数，若直接写Print()将调用派生类的Print()函数自己！

例. 访问公有基类的成员(指出是否错误)

```
1  Class Student//声明基类
2  {public:                                  //基类公用成员
3  void get_value( )
4  {cin>>num>>name>>sex;}
5  void display( )
6  {cout<<" num: "<<num<<endl;
7  cout<<" name: "<<name<<endl;
8  cout<<" sex: "<<sex<<endl;}
9  private :                                 //基类私有成员
10    int num;
11    string name;
12    char sex;};
13
14 class Student1: public Student             //以public方式声明派生类Student1
15 {public:
16 void display_1( )
17    { cout<<" num: "<<num<<endl;            //引用基类的私有成员，？
18       cout<<" name:"<<name<<endl;          //引用基类的私有成员，？
19       cout<<" sex:"<<sex<<endl;            //引用基类的私有成员，？
20       cout<<"age:"<<age<<endl;             //引用派生类的私有成员，？
21       cout<<"address:"<<addr<<endl;}       //引用派生类的私有成员，？
22 private:
23       int age;
24       string addr;};
```

```
1  Class Student//声明基类
2  {public:                                  //基类公用成员
3  void get_value( )
4  {cin>>num>>name>>sex;}
5  void display( )
6  {cout<<" num: "<<num<<endl;
7  cout<<" name: "<<name<<endl;
8  cout<<" sex: "<<sex<<endl;}
9  private :                                 //基类私有成员
10    int num;
11    string name;
12    char sex;};
13
14 class Student1: public Student             //以public方式声明派生类Student1
15 {public:
16 void display_1( )
17    { cout<<" num: "<<num<<endl;            //企图引用基类的私有成员，错误
18       cout<<" name:"<<name<<endl;          //企图引用基类的私有成员，错误
19       cout<<" sex:"<<sex<<endl;            //企图引用基类的私有成员，错误
20       cout<<"age:"<<age<<endl;             //引用派生类的私有成员，正确
21       cout<<"address:"<<addr<<endl;}       //引用派生类的私有成员，正确
22 private:
23       int age;
24       string addr;};
```

```
#include "string"
#include "iostream"
using namespace std;
class Student//声明基类
{public:                                  //基类公用成员
void get_value( )
{cin>>num>>name>>sex;}
void display( )
{cout<<"num:"<<num<<endl;
cout<<"name:"<<name<<endl;
cout<<"sex:"<<sex<<endl;}
private :                                 //基类私有成员
   int num;
  string name;
   char sex;};
class Student1: public Student//以public方式声明派生类Student1
{public:
void display_1( )
{cout<<"age:"<<age<<endl;          //引用派生类的私有成员，正确
cout<<"address:"<<addr<<endl;}     //引用派生类的私有成员，正确
private:
int age;
string addr;};
int main( )
{Student1  stud;    //定义派生类Student1的对象stud
stud.display( );    //调用基类的公用成员函数，输出基类中3个数据成员的值
stud.display_1();  //调用派生类的公用成员函数，输出派生类中两个数据成员的值
return 0;}
```

```
C:\WINDOWS\system32\cmd.exe
num:1458808
name:
sex:
age:1241948
address:
请按任意键继续. . .
```

6.3 派生类成员的访问属性

私有继承(private)派生类访问属性调整

表 11.2　私有基类在派生类中的访问属性

私有基类中的成员	在私有派生类中的访问属性
私有成员	不可访问
公用成员	私有
保护成员	私有

只能类内访问

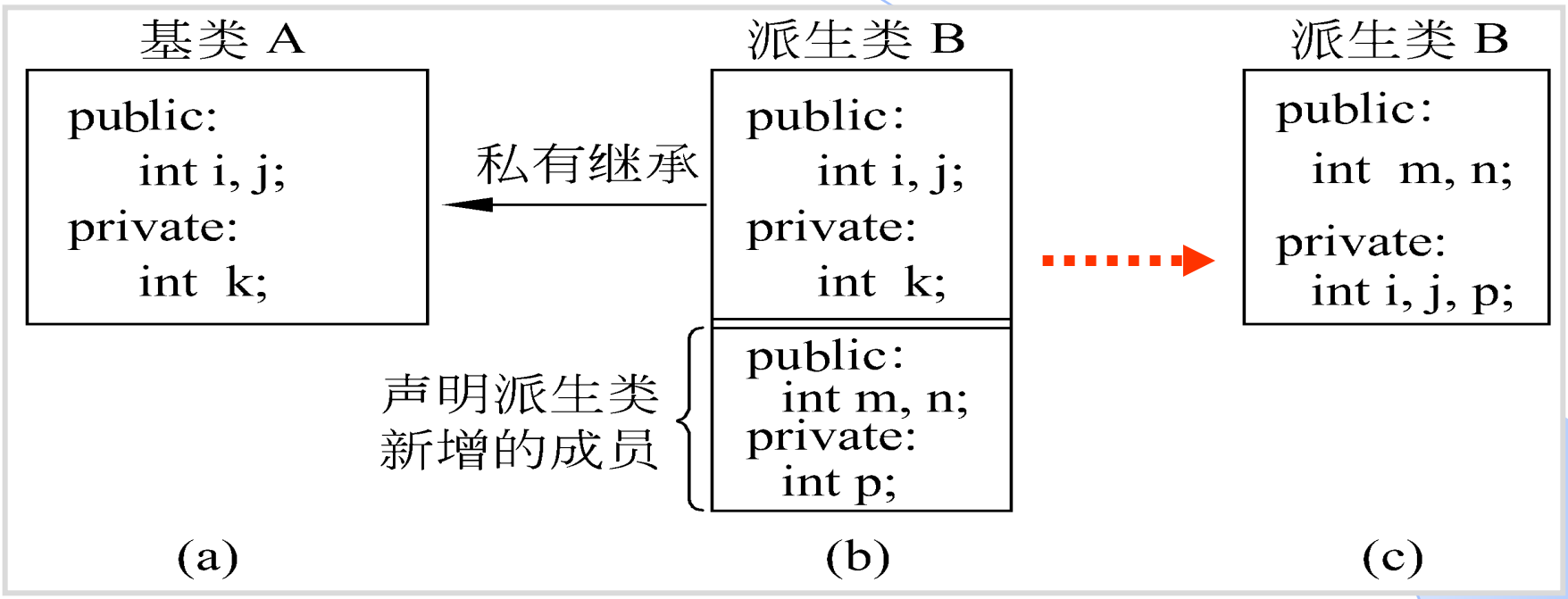
派生类成员的访问方式

（1）类内:在派生类内可访问基类的公用成员和保护成员;

（2）类外:在派生类外（通过派生类对象）不能访问基类的任何成员;

（3）派生类新增成员按public、protected、private访问规则。

私有继承的派生类中访问属性变异



程序设计思想之二

采用私有继承可对于基类中不需要再往下继承的功能把它隐蔽起来，这样下一层派生类无法访问基类的任何成员；

```
#include "string"
#include "iostream"
using namespace std;
class Student//声明基类
{public:
void get_value( )
{cin>>num>>name>>sex;}
void display( )
{cout<<"num:"<<num<<endl;
cout<<"name:"<<name<<endl;
cout<<"sex:"<<sex<<endl;}
private :
   int num;
  string name;
   char sex;};
class Student1: private Student//用私有继承方式声明派生类Student1
{public:
void display_1( )                  //输出两个数据成员的值
{cout<<"age:"<<age<<endl;          //引用派生类的私有成员，正确
cout<<"address:"<<addr<<endl;}     //引用派生类的私有成员，正确
private:
int age;
string addr;
};
int main( )
{Student1  stud1;       //定义一个Student1类的对象stud1
 stud1.display();       //?，私有基类的公用成员函数在派生类中是私有函数
stud1.display_1( );     //?，Display_1函数是Student1类的公用函数
stud1.age=18;           //?，
return 0; }
```

输出

显示以下输出(S): 生成

```
1>------ 已启动全部重新生成: 项目: test2, 配置: Debug Win32 ------
1>正在删除项目“test2”(配置“Debug|Win32”)的中间文件和输出文件
1>正在编译...
1>1.cpp
1>.\1.cpp(27) : error C2247: 'Student::display' not accessible because 'Student1' uses 'private' to inhe
1>        .\1.cpp(8) : see declaration of 'Student::display'
1>        .\1.cpp(16) : see declaration of 'Student1'
1>        .\1.cpp(5) : see declaration of 'Student'
1>.\1.cpp(29) : error C2248: 'Student1::age' : cannot access private member declared in class 'Student1'
1>        .\1.cpp(22) : see declaration of 'Student1::age'
1>        .\1.cpp(16) : see declaration of 'Student1'
1>生成日志保存在“file://e:\vc-execise\test2\test2\Debug\BuildLog.htm”
1>test2 - 2 个错误，0 个警告
========== 全部重新生成: 0 已成功, 1 已失败, 0 已跳过 ==========
```

```
#include "string"
#include "iostream"
using namespace std;
class Student//声明基类
{public:
void get_value( )
{cin>>num>>name>>sex;}
void display( )
{cout<<"num:"<<num<<endl;
cout<<"name:"<<name<<endl;
cout<<"sex:"<<sex<<endl;}
private :                              //基类私有成员
   int num;
  string name;
   char sex;};
class Student1: private Student//用私有继承方式声明派生类Student1
{public:
  void display_1( ) //输出5个数据成员的值
{display();                    //调用基类的公用成员函数，输出3个数据成员的值
  cout<<"age:"<<age<<endl;        //引用派生类的私有成员，正确
  cout<<"address:"<<addr<<endl;}   //引用派生类的私有成员，正确
private:
  int age;
  string addr;};

int main( )
{Student1 stud1;
 stud1.display_1( );//display_1函数是派生类Student1类的公用函数
 return 0;}
```

```
C:\WINDOWS\system32\cmd.exe
num:1458808
name:
sex:
age:1241948
address:
请按任意键继续. . .
```

程序设计思想之三

在派生类中可用using 基类::成员恢复基类防控级别；

(1)基类private成员不能在派生类中任何地方用using声明;

(2)基类protected成员可在派生类中任何地方用using声明;

当在public下声明时,可派生类外（通过对象）访问该成员，但不能用基类对象访问该成员；当在protected下声明时，该成员可以被继续派生下去；当在private下声明时，不能载派生类外（通过对象）访问;

(3)基类public成员，可在派生类中任何地方用using声明。当在public下声明时,可派生类外（通过对象）访问该成员，但不能用基类对象访问该成员；当在protected下声明时，该成员可以被继续派生下去；当在private下声明时，不能载派生类外（通过对象）访问。

```
Class CAnimal
{public:
 void GetWeight();
Private:
 floagt m_fweight;};
Class CPig:private CAnimal
{public:
using CAnimal::GetWeight;
void GetPorkWeight();
Private:
Float m_fPorkWeight;};
```

```
#include <iostream>
Using namespace std;

void main(void)
{
Cpig apig;
apig.GetWeight();
apig.GetPorkWeight()
;
}
```

```
1  class Base
2  {
3  protected:
4      void test1() { cout << "test1" << endl; }
5      void test1(int a) {cout << "test2" << endl; }
6
7      int value = 55;
8  };
9
10 class Derived : Base     //使用默认继承
11 {
12 public:
13     //using Base::test1;     //using只是声明，不参与形参的指定
14     //using Base::value;
15     void test2() { cout << "value is " << value << endl; }
16 };
```

问题：分析上述编译会出错吗？为什么？怎么纠错？

```cpp
#include "pch.h"
#include <iostream>

#include "string"
using namespace std;
class tester
{
public:
    tester() : i(5), ch('x') {};
private:
    int i;
    char ch;
};
class  tester1 :public tester
{
private:
    int y = 0;
};

int main(void)
{
    tester1 myTester1;
    char* p = NULL;
    p = (char*)&myTester1 + sizeof(int);
    cout << "Address of ch = " << (void*)p << endl;
    cout << "ch = " << *(p) << endl;
    //getchar();
    *p = 'y';
    cout << "Now ch = " << *(p) << endl;
    return 0;
}
```

```
C:\Users\hp\source\repos\Con
Address of ch = 0136FDF0
ch = x
```

```
Address of ch = 010FFC60
ch = x
Now ch = y

C:\Users\hp\source\repos\ConsoleApplicati
若要在调试停止时自动关闭控制台，请启用“工
按任意键关闭此窗口...
```

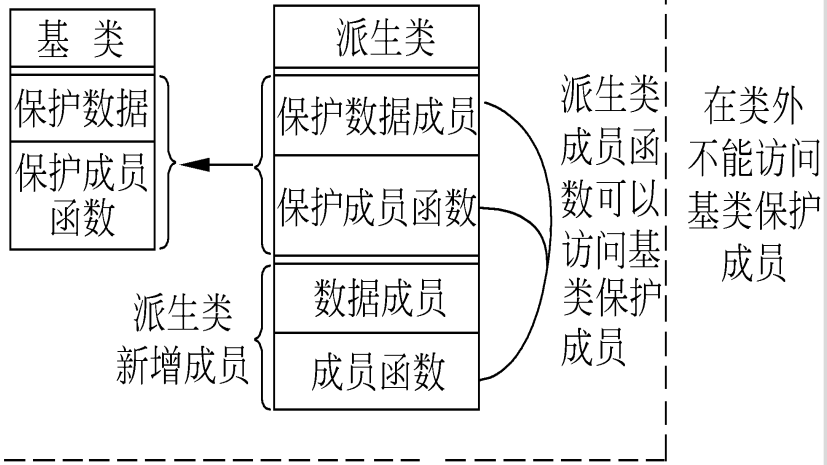
6.3 派生类成员的访问属性

■ **保护成员**(protected):在访问属性等价于私有(Private)成员；但在继承方式区别于Private成员.

□ 保护继承(protected)在派生类的访问属性调整

表 11.3 基类成员在派生类中的的访问属性

基类中的成员	在公用派生类中的访问属性	在私有派生类中的访问属性	在保护派生类中的访问属性
私有成员	不可访问	不可访问	不可访问
公用成员	公用	私有	保护
保护成员	保护	私有	保护



□protected派生类访问方式

（1）类内:在派生类内可访问基类公用和保护成员；

（2）类外：在派生类外（用对象）不能访问基类任何成员；

（3）新增成员按访问属性规定。

```
1.  class Employee {
2.  protected:
3.      String name;
4.      Date birthday;
5.      char sex;
6.      short department;
7.  public:
8.      Employee();
9.      void Print() const
10.     { cout << name << "is an employee" << endl; }
11. };

12. class Manager : protected Employee {
13.     short level;
14. public:
15.     Manager();
16.     void Print() const
17.     {   //派生类可以访问基类的保护成员
18.         cout << name << "is a manager" << endl;
19.     }
20. }
```

```
void main()
{
    Employee e1;
    Manager m1;
    ...
    m1.Print();
    cout << m1.name;//错误，
        //外界不能访问保护成员
}
```

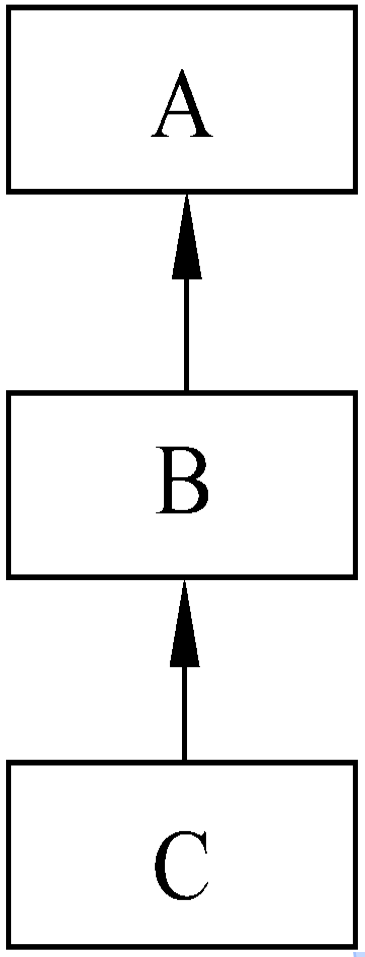
6.3 派生类成员的访问属性

- 多级派生时的访问属性

（1）如果有下图所示的派生关系：类A为基类，类B是类A的派生类；类C是类B的派生类，则类C也是类A的派生类；

（2）类B称为类A的直接派生类，类C称为类A的间接派生类。类A是类B的直接基类，是类C的间接基类；

（3）在多级派生下，各成员的访问属性仍按前面的原则确定。



```
class A          //基类
{public:
 int i;
protected:
 void f2( );
 int j;
private:
 int k;};
```

```
class B: public A   //public方式
{public:
void f3( );
 protected:
void f4( );
 private:
  int m;};
```

```
class C: protected B //protected方式
{public:
void f5( );
 private:
  int n;};
```

	i	f2	j	k	f3	f4	m	f5	n
基类A	公用	保护	保护	私有					
公用派生类B	公用	保护	保护	不可访问	公用	保护	私有		
保护派生类C	保护	保护	保护	不可访问	保护	保护	不可访问	公用	私有

- 无论何种继承，派生类中都不能访问基类的私有成员；
- 如多级派生时都采用公用继承，那么直到最后一级派生类都能访问基类的公用成员和保护成员。

```
#include <iostream>
#include <string>
using namespace std;
class Student//声明基类
{public:                              //基类公用成员
  void display( );
 protected :                          //基类保护成员
int num;
string name;
char sex;};

void Student::display( )              //定义基类成员函数
   {cout<<"num:"<<num<<endl;
    cout<<"name:"<<name<<endl;
    cout<<"sex:"<<sex<<endl;}

class Student1: protected Student     //用protected方式声明派生类Student1
{public:
void display1( );                     //派生类公用成员函数
private:
int age;                              //派生类私有数据成员
string addr; };                       //派生类私有数据成员

void Student1::display1( )            //定义派生类公用成员函数
    {cout<<"num: "<<num<<endl;        //引用基类的保护成员，？
     cout<<"name:"<<name<<endl;       //引用基类的保护成员，？
    cout<<"sex:"<<sex<<endl;          //引用基类的保护成员，？
     cout<<"age:"<<age<<endl;         //引用派生类的私有成员，？
     cout<<"address:"<<addr<<endl; }  //引用派生类的私有成员，？

int main( )
 {Student1 stud1;                     //stud1是派生类Student1类的对象
  stud1.display1( );                  //?, display1是派生类中的公用成员函数
//  stud1.num=10023;                  //?, 外界不能访问保护成员
```

```
C:\WINDOWS\system32\cmd.exe
num: 1458808
name:
sex:
age:1241948
address:
请按任意键继续. . .
```

本讲重点分析

- 派生类声明形式:class 派生类名：［继承方式］ 基类名
- 继承方式：public, protected和private

- 类成员访问属性
  - ✓ 基类成员:public,protected,private
  - ✓ 派生类成员：public,protected, private 和不可访问

- 派生类的成员访问方式
  - ✓ 类新成员访问：public, protected, private
  - ✓ 类内对基类继承成员访问： public, protected
  - ✓ 类外(通过对象）对基类继承成员：public

第6次作业（必做题）

第1题:教材本章后面习题中的第8题。阅读程序，分析程序运行结果，并回答问题。

第2题：在上次（第5次）作业的第1题的基础上，从People(人员）类派生出student(学生）类，添加属性：班号char classNo[7]；从People(人员)类派生出teacher(教师)类，添加属性：职务char principalship[11]、部门char department[21]；从student类中派生出graduate(研究生)类、添加属性：专业char subject[21]、导师teacher advisor；从teacher类和graduate类派生出TA(助教)类，添加属性：RA。要求编制能管理上述四类人员的程序，并实现数据项录入、显示操作。

选做题

1. 参考教材本章综合程序的代码，在前面作业基础上进行修改：要求采用继承方式，新增员工设岗，设立3类岗位：经理（manager)1名，技术（technician)岗10名和其余为销售岗位（salesman)。岗位不同，待遇不同。经理拿固定月薪12000元，技术人员按每小时260拿月薪，销售岗按当月的销售额提成5%。新增加类属性可以自由添加；其他细节部分自由设计。鼓励创新！！！

2. 上机调试本PPT 中的第23页中的程序。如果将类tester中的数据成员i改为：long int. 那如何实现现有程序的功能？

笑话一则：小明为啥在起跑线上？

- 妈妈：“小明，要是你再不努力学习，小学学习基础没打好，你就会输在人生起跑点上！”
- 小明：“我早就输在起人生跑点上啦。”
- 妈妈：“你输了什么？”
- 小明：“继承与遗传呀”

因此，同学们应该感恩“好的基类”；同时注意设计“好的基类”。