西北工业大学
NORTHWESTERN POLYTECHNICAL UNIVERSITY

C++程序设计
Programming in C++

1011018

主讲：魏英，计算机学院

类的定义

- 3、类的数据成员
- 4、类的成员函数
- 5、类的声明

- 1. 在类中声明数据成员
- 正如我们所见，类的数据成员的声明类似于普通变量的声明。如果一个类具有多个同一类型的数据成员，则这些成员可以在一个成员声明中指定。

```
class Cube { //Cube类表示立方体
    … //其他成员
    long color; //数据成员
    double x,y,z,side; //数据成员
};
```

- 类的数据成员可以是基本类型、数组、指针、引用、共用体、枚举类型、void指针、const限定等数据类型。例如：

```
class ADT { //类成员数据类型
    … //成员函数
    long color;
    double x,y,z,side; //基本类型
    int a[10]; //数组
    char *s; //指针
    char &r; //引用
    void *p; //void指针
};
```

- 类的数据成员还可以是成员对象（member object），即类类型或结构体类型的对象。若类A中嵌入了类B的对象，称这个对象为子对象（subobject）。例如：类Line嵌入了类Point的子对象start、end。

```
class Point { //Point类表示点
public:
    void set(int a,int b);
    int x,y;
};
class Line { //Line类表示线
public:
    void set(Point a,Point b);
    Point start, end; //成员对象
};
```

- 2．在类中定义或声明数据类型
- 除了定义数据成员和成员函数之外，类还可以定义自己的局部类型，并且使用类型别名来简化。

- 在类中定义或声明的数据类型的作用域是类内部，因此，它们不能在类外部使用。

- 在类定义中，可以定义结构体和共用体类型、嵌套的类定义，声明枚举类型。

```
class ADT { //类定义
      struct Point { int x,y; }; //定义结构体
      union UData {Point p; long color; }; //定义共用体
      enum COLORS {RED,GREEN,BLUE,BLACK,WHITE }; //定义枚举类型
      class Nested { //嵌套类定义
            … //成员函数
            Point start; //数据成员
            UData end; //数据成员
            COLORS color; //数据成员
      };
      typedef Point* LPPOINT; //声明类型别名
      … //成员函数
      … //数据成员
}; //类定义结束
```

- 1. 在类的外部定义成员函数

- 如果成员函数仅有声明在类定义中，则在类外部必须有它的实现，其一般形式为：

```
返回类型 类名::函数名(形式参数列表)
{
        函数体
}
```

```
class Data { //Data类定义
public:
  void set(int d); //成员函数原型声明
  int get() { //成员函数定义
              return data;
     } //get函数定义结束
private:
     int data; //数据成员
};  //Data类定义结束
void Data::set(int d) //成员函数的外部定义，使用 Data:: 限定
{
        data=d; //访问类的数据成员
}
void set(int d) //全局普通函数
{
        … //函数体
}
```

- 说明：
- （1）（::）是作用域限定符（field qualifed）。如果在作用域限定符的前面没有类名，或者函数前面既无类名又无作用域限定符，例如：

```
::set(10) 或 set(10)
```

- 则表示set函数不属于任何类，这个函数不是成员函数，而是全局的普通函数。此时的（::）不是类作用域限定符的含义，而是命名空间域限定符的含义。

- （2）在成员函数中可以访问这个类的任何成员，无论它是公有的或是私有的，是类内部声明的还是类外部定义的。

- （3）虽然成员函数在类的外部定义，但在调用成员函数时会根据在类中声明的函数原型找到函数的定义（即函数代码），从而执行该函数。因此类的成员函数原型声明必须出现在成员函数定义之前，否则编译时会出错。

- （4）在类的内部声明成员函数，而在类的外部定义成员函数，这是一个良好的编程习惯。因为不仅可以减少类体的长度，使类体结构清晰，便于阅读，而且有助于类的接口和实现分离。

- （5）如果一个成员函数，其函数体不太复杂，只有4～5行时，一般可在类体中定义。

- 2．内联成员函数

- 类的成员函数可以指定为inline，即内联函数。
-
- 默认情况下，在类体中定义的成员函数若不包括循环等控制结构，符合内联函数要求时，C++会自动将它们作为内联函数处理（隐式inline）。

- 也可以显式地将成员函数声明为inline。例如：

```
class Data { //Data类定义
      int getx() { return x;} //内联成员函数
      inline int gety() { return y;} //显式指定内联成员函数
      inline void setxy(int _x, int _y); //显式指定内联成员函数
      void display();
      int x, y;
};
inline void Data::setxy(int _x, int _y) //内联成员函数
{
      x=_x, y=_y;
}
void Data::display() //非内联成员函数
{
      … //函数体
}
```

- 判断成员函数是否是内联的，有以下几条：

- ① 符合内联函数要求；
- ② 符合①的条件，并且在类体中定义，自动成为内联的；
- ③ 符合①的条件，在类体显式指明inline，或在外部定义时显式指明inline，或者同时显式指明，则函数是内联的；
- ④ 在类外部定义，并且既没有在类体，也没有在外部定义时显式指明inline，则函数不是内联的。

- 3. 成员函数重载及默认参数
- 可以对成员函数重载或使用默认参数。例如：

```
class MAX {
      int Max(int x,int y) { return x>y?x:y; }
      int Max()
      {
           return Max(Max(a,b),Max(c,d));
      } //重载Max
      int Set(int i=1,int j=2,int k=3,int l=4)
      {
           a=i,b=j,c=k,d=l;
      } //默认参数
      int a,b,c,d;
};
```

- 需要注意，声明成员函数的多个重载版本或指定成员函数的默认参数，只能在类内部中进行。

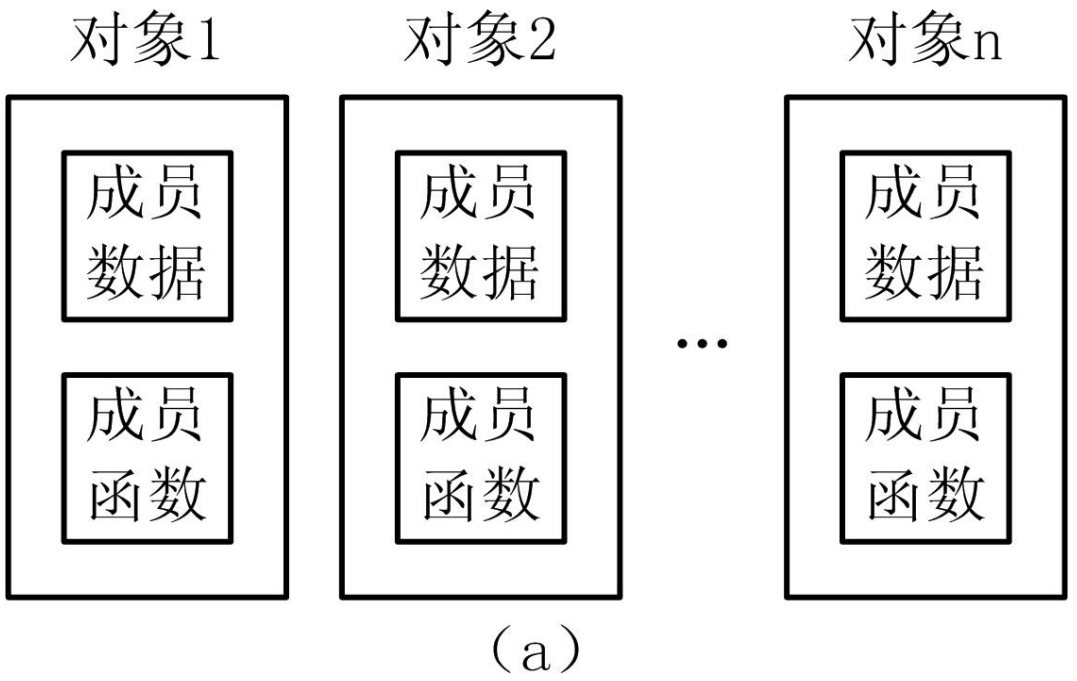
- 因为类定义中的声明先于成员函数的外部实现，根据重载或默认参数函数的要求，必须在第1次出现函数声明或定义时就明确函数是否重载或有默认参数。

- 4．成员函数的存储方式

- 用类实例化一个对象时，系统会为每一个对象分配存储空间。如果一个类包括了数据成员和成员函数，则要分别为数据和函数的代码分配存储空间。
- 通常，C++会为每个对象的数据成员分配各自独立的存储空间，像结构体成员那样。

- 那么在类中的成员函数是否会如图所示那样也分配各自独立的存储空间呢？

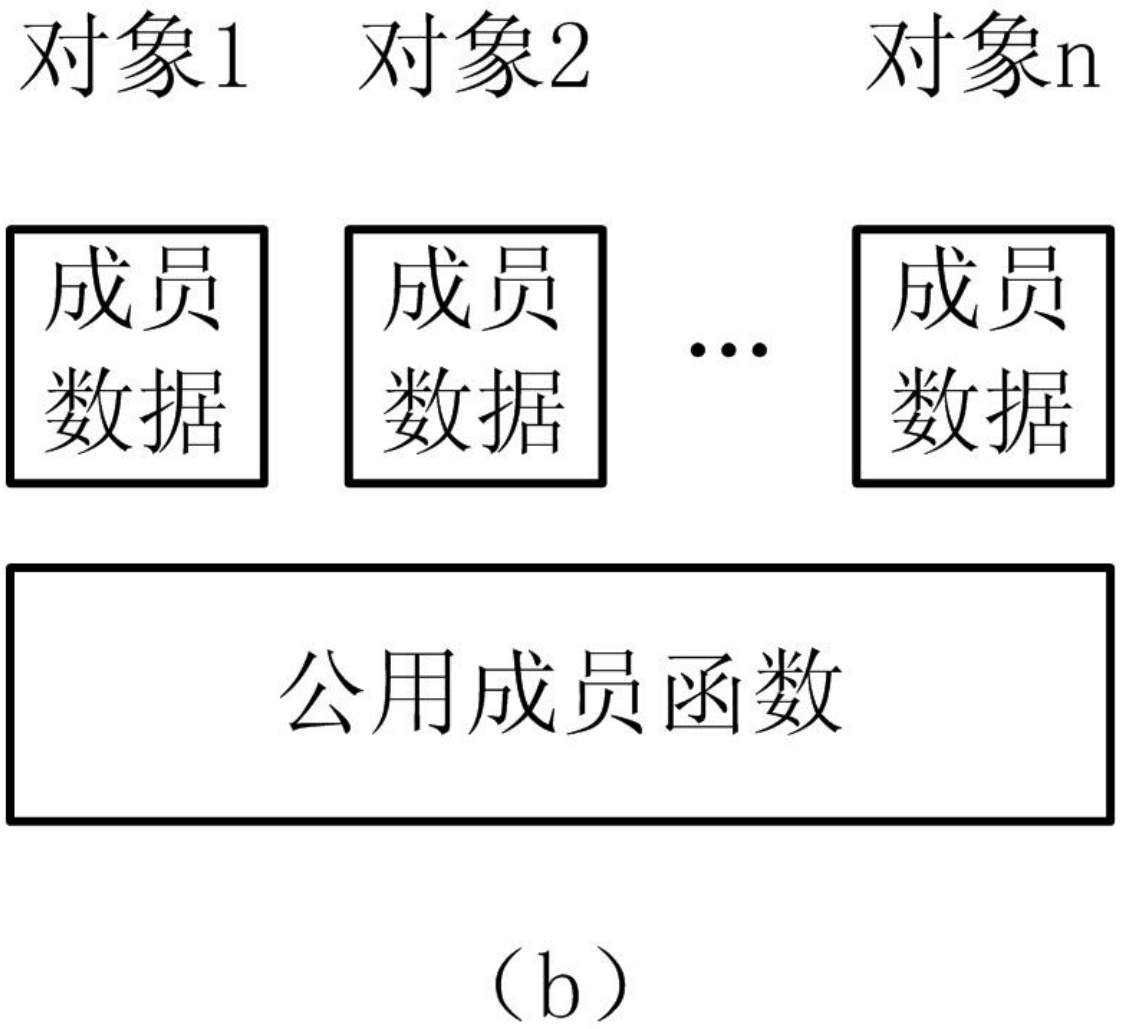
图25.1 成员函数的存储方式



- 由于不论调用哪一个对象的函数代码，实际调用的都是同样内容的代码。因此，若像上图那样存放相同代码的多份副本，既浪费空间又无必要。

- 实际上，成员函数代码只有公用的一段存储空间，调用不同对象的成员函数时都是执行同一段函数代码。

图9.1 成员函数的存储方式



- 例如定义了一个类

```
class Time { //Time类
      int h,m,s; //数据成员
      void settime(int a,int b,int c)
      { h=a,m=b,s=c;} //成员函数
};
```

- sizeof(Time)的值是12。显然，Time类的存储空间长度只取决于数据成员h、m、s所占的空间，而与成员函数settime无关。C++把成员函数的代码存储在对象空间之外的地方。

- 一旦遇到类体后面的右大括号，类的定义就结束了。
- 在一个给定的源文件中，一个类只能被定义一次。
- 通常将类的定义放在头文件中，这样可以保证在每个使用该类的文件都以同样的方式定义类。

- 可以只声明一个类而不定义它：

```
class Point; //Point类声明，非Point类定义，因为没有类体
```

- 这个声明，称为前向声明（forward declaration），表示在程序中引入了Point类类型。
- 在声明之后、定义之前，类Point是一个不完全类型，即已知Point是一个类，但不知道它包含哪些成员。因此不能定义该类型的对象，只能用于定义指向该类型的指针及引用，或者用于声明（而不是定义）使用该类型作为形参类型或返回类型的函数。

- 在创建类的对象之前，必须完整地定义该类。这样，编译器就会给类的对象准备相应的存储空间。
- 同样地，在使用引用或指针访问类的成员之前，必须已经定义类。
- 类不能具有自身类型的数据成员。然而，只要类名一经出现就可以认为该类已声明。因此，类的数据成员可以是指向自身类型的指针或引用。

- 例如：

```
class Point; //Point类声明，非Point类定义，因为没有类体
class Line {
      Point a; //错误，不能使用仅有类声明而没有类定义的类定义数据对象
      Point *pp, &rp; //正确，只有类声明，即可用它定义该类的指针或引用
      Line b; //错误，类不能具有自身类型的数据成员
      Line *pl, &rl; //正确，类可以有指向自身类型的指针或引用的数据成员
};
```

C程序设计