1. 构造函数
2. 析构函数
3. 类的组合

4.3.1 构造函数

- 重点掌握概念、特点、作用、实现
- 概念：是和类名相同的成员函数。

4.3.1 构造函数特征

1. 没有返回值，void也不行
2. 函数名字和类名相同
3. 必须是public
4. 自动调用
5. 构造函数可以是内联函数、重载函数、带默认参数值的函数

默认构造函数

- 无参数的构造函数都是默认构造函数。
- 全部参数都有默认值的也是默认构造函数
- 当不定义构造函数时，编译器自动产生默认构造函数，无参数。
- 可以自定义默认构造函数

默认构造函数

- 下面两个都是默认构造函数，如果在类中同时出现，将产生编译错误：
  - Clock();
  - Clock(int h=0,int m=0,int s=0);

4.3.1 构造函数作用

- 作用是在对象被创建时使用特定的值构造对象，将对象初始化为一个特定的初始状态。
- 有同学会问，能不能在类声明时对成员数据初始化呢？

构造函数的作用

```
class Student{
    char number[10]="20194223";
    char name[50]="陈顺鹏";
    char sex[3]="男";
};
Student zhangyihang,zhengkaili,liushunshun;
```

这样初始化是错误的，因为类是数据抽象，不是实体，不占用内存空间。

构造函数的实现

```
class Clock {
public:
  Clock(int h,int m,int s);
  void setTime(int h, int m, int s);
  void showTime();
private:
  int hour, minute, second;
};
```

构造函数的实现（续）

```
Clock::Clock(int h, int m, int s): hour(h),
  minute(m), second(s) {   }


Clock::Clock(int h, int m, int s)
{
    hour=h;
    minute=m;
    second=s;
}
```

构造函数的自动调用

```
int main() {
   Clock c1(0,0,0),c2(12,12,12);
   //定义对象时自动调用构造函数
   c1.showTime();
   c2.showTime();
   return 0;
}
```

构造函数的调用时机

- 定义对象时，自动调用
- 特别注意：对象的定义和构造函数要一致

```
class Clock{
public:
  Clock(int h,int m,int s)//构造函数
  {  hour=h;
     minute=m;
     second=s;
  }
   void showTime();
private:
  int hour, minute, second;
};
int main() {
  Clock c(1,2,3);
  c.showTime();
  return 0;
}
```

课堂练习

- 定义一个Book、Date类
- 定义构造函数

4.3.2 复制构造函数

- 复制构造函数是一种**特殊的构造函数**，其形参为本类的**对象引用**。作用是用一个已存在的对象去**初始化**同类型的新对象。

```
class 类名 {
public :
    类名（类名 &对象名）；//复制构造函数
        ……
} ;
```

4.3.2 复制构造函数

■复制构造函数被调用的第一种情况

- 定义一个对象时，以本类另一个对象作为初始值

```
class Point {   //Point 类的定义
public:
    Point(int xx, int yy)
  { x = xx; y = yy; } //构造函数，内联
    Point(Point& p); //复制构造函数
private:
    int x, y; //私有数据
};
 Point::Point (Point& p) {
   x = p.x;  y = p.y;
   cout << "Calling the copy constructor " << endl;
 }
 int main() {
   Point a(4, 5);
   Point b = a;    //调用复制构造函数
   Point c(a);   //调用复制构造函数
}
```

4.3.2 复制构造函数的调用

■复制构造函数被调用的其他两种情况

- 如果函数的形参是类的对象，调用函数时，将使用实参对象初始化形参对象，发生复制构造；
- 如果函数的返回值是类的对象，函数执行完成返回主调函数时，将使用return语句中的对象初始化一个临时无名对象，传递给主调函数，此时发生复制构造。

```
//形参为Point类对象的函数
void fun1(Point p) {
  cout << p.getX() << endl;
}
//返回值为Point类对象的函数
Point fun2() {
  Point a(1, 2);
  return a;
}
//主程序
int main() {
  Point a(4, 5);
  fun1(a);          //情况二，对象B作为fun1的实参
  Point b = fun2();//情况三，函数的返回值是类对象，
  return 0;
}
```

隐含的复制构造函数

- 如果程序员没有为类声明复制构造函数，则编译器自己生成一个隐含的复制构造函数。

```
class Point {    //Point 类的定义
public:
  Point(int xx, int yy)
  { x = xx; y = yy; } //构造函数，内联
  Point::Point (Point& p)//隐含的复制构造函数
  {  x = p.x;  y = p.y;
  }

private:
  int x, y; //私有数据
};
int main() {
  Point a(4, 5);
  Point b = a;    //调用复制构造函数
  Point c(a);   //调用复制构造函数
}
```

4.3.3 析构函数

- 完成对象被删除前的一些清理工作。
- 在对象的生存期结束的时刻系统自动调用它，然后再释放此对象所属的空间。
- 如果程序中未声明析构函数，编译器将自动产生一个隐含的析构函数。

构造函数和析构函数举例

```
#include <iostream>
using namespace std;
class Point {
public:
  Point(int xx,int yy);
  ~Point();
private:
  int x, y;
};
```

```
Point::Point(int xx,int yy)
{
  x = xx;
  y = yy;
}
Point::~Point() {
}
```

*编译器默认提供的函数

```
class Empty{
public:
    Empty(){}
    Empty(Empty& rhs){...}
    ~Empty(){}
};
```

4.4.1组合

- 类中的成员数据是另一个类的对象。
- 可以在已有抽象的基础上实现更复杂的抽象。

类组合的构造函数设计

■原则：不仅要负责对本类中的基本类型成员数据赋初值，也要对对象成员初始化。

■声明形式：

```
类名::类名(对象成员所需的形参，本类成员形参)
        :对象1(参数)，对象2(参数)，……
{
//函数体其他语句
}
```

构造组合类对象时的初始化次序

- 首先对构造函数初始化列表中列出的成员（包括基本类型成员和对象成员）进行初始化，初始化次序是成员在类体中定义的次序。
  - •成员对象构造函数调用顺序：按对象成员的声明顺序，先声明者先构造。
  - •初始化列表中未出现的成员对象，调用用默认构造函数（即无形参的）初始化
- 处理完初始化列表之后，再执行构造函数的函数体

例4-4 类的组合，线段（Line）类

```
//4_4.cpp
#include <iostream>
#include <cmath>
using namespace std;
class Point {  //Point类定义
public:
  Point(int xx = 0, int yy = 0) {
    x = xx;
    y = yy;
      cout<<"Calling the constructor of Point" << endl;
  }
  Point(Point &p);
  int getX() { return x; }
  int getY() { return y; }
private:
  int x, y;
};
Point::Point(Point &p) {  //复制构造函数的实现
  x = p.x;
  y = p.y;
  cout << "Calling the copy constructor of Point" << endl;
}
```

例4-4（续）

```
//类的组合
class Line {   //Line类的定义
public:  //外部接口
  Line(Point xp1, Point xp2);
  Line(Line &l);
  double getLen() { return len; }
private: //私有数据成员
  Point p1, p2;     //Point类的对象p1,p2
  double len;
};
//组合类的构造函数
Line::Line(Point xp1, Point xp2) : p1(xp1), p2(xp2) {
  double x = p1.getX() - p2.getX();
  double y = p1.getY() - p2.getY();
  len = sqrt(x * x + y * y);
  cout << "Calling the constructor of Line" << endl;
}
Line::Line (Line &l): p1(l.p1), p2(l.p2) {//组合类的复制构造函数
  len = l.len;
  cout << "Calling the copy constructor of Line" << endl;
}
```

例4-4（续）

```
//主函数
int main() {
  Point myp1(1, 1), myp2(4, 5);//建立Point类的对象
  Line line(myp1, myp2);  //建立Line类的对象
  Line line2(line); //利用复制构造函数建立一个新对象
  cout << "The length of the line is: ";
  cout << line.getLen() << endl;
```

运行结果如下：

```
Calling the copy constructor of Point
Calling the copy constructor of Point
Calling the copy constructor of Point
Calling the copy constructor of Point
Calling constructor of Line
Calling the copy constructor of Point
Calling the copy constructor of Point
Calling the copy constructor of Line
The length of the line is: 5
The length of the line2 is: 5
```

小结

■主要内容

•面向对象的基本概念、类和对象的声明、构造函数、析构函数、内联成员函数、复制构造函数、类的组合

■达到的目标

•掌握类的定义及应用