作用：当创建一个对象时，往往需要做一些初始化工作，例如对数据成员赋值等

特点：是一种特殊的成员函数，它的名字和类名相同，没有返回值，不需要用户调用（用户也不能调用），而是在创建对象时自动执行

例子：

```
#include<iostream>
usingnamespace std;
class Student{
private:
char*name;
int age;
float score;
public:
//声明构造函数
Student(char*,int,float);
//声明普通成员函数
voidsay();
};
//定义构造函数
Student::Student(char*name1,int age1,float score1){
    name = name1;
    age = age1;
    score = score1;
}
//定义普通成员函数
void Student::say(){
    cout<<name<<"的年龄是 "<<age<<"，成绩是 "<<score<<endl;
}
intmain(){
//根据构造函数创建对象
    Student stu("小明",15,90.5f);//传参形式类似于函数调用
    stu.say();
return0;
}
```

运行结果：
小明的年龄是 15，成绩是 90.5

在类中我们定义了一个构造函数 Student()，它的作用是给3个 private 属性的成员变量赋值。在 main 函数中，我们根据构造函数创建了一个对象 stu；因为构造函数有参数，所以创建对象时要相应地传入实参，形式类似于函数调用。

读者要注意：一旦在类中定义了构造函数，那么创建对象时一定会被执行；如果构造函数有参数，创建对象时就要传参。

另外，构造函数主要用来进行初始化，没有返回值（有返回值没有任何意义），这就意味着：

- 不管是声明还是定义，函数名前面都不能出现返回值类型，即使是 void 也不允许；
- 函数体中不能有 return 语句。

默认构造函数 和java 是一样的

如果用户自己没有定义构造函数，那么编译器会自动生成一个默认的构造函数，只是这个构造函数的函数体是空的，也没有参数，不执行任何操作。比如上面的 Student 类，默认生成的构造函数如下：

```
Student(){}
```

一个类，必须有构造函数，要么用户自己定义，要么编译器自动生成。一旦用户自己定义了构造函数，不管它是 public 属性的，还是 private、protected 属性的，编译器都不再自动生成。上面的 Student 类，只有一个构造函数，就是我们自己定义的。实际上，编译器只有在必要的时候才会生成默认构造函数，而且它的函数体一般不为空。默认构造函数的目的是帮助编译器做初始化工作，而不是帮助程序员。这是C++的内部实现机制，这里不再深究，初学者可以按照上面说的“一定有一个空函数体的默认构造函数”来理解。

构造函数的重载

和普通成员函数一样，构造函数是允许重载的。一个类可以提供多个构造函数，让用户在创建对象时进行选择，编译器会根据创建对象时传递的参数来确定调用哪一个构造函数。也就是说：

- 只有一个构造函数会被执行；

- 创建对象时提供的参数必须和其中的一个构造函数匹配，否则编译错误。