转换构造函数：这是一个接受单个参数的构造函数（或所有参数都有默认值），允许类的对象从该参数类型隐式转换。例如，如果有一个 Complex 类和一个接受 double 的构造函数，那么可以隐式地将 double 类型的值转换为 Complex 类型。

类型转换运算符：这是一种特殊的类成员函数，用于将一个类的对象转换为另一种类型。例如，你可以为 Complex 类定义一个到 double 的转换运算符，允许将 Complex 类型的对象隐式转换为 double 类型。

conversion constructors

```
class Y{
private:
    int a, b;
    char* name;
public:
    Y(int i){}
    Y(const char* n, int j = 0){}//后面的参数已经给了默认值0，则传入的时候就不需要显式传入后面的参数
    void add(Y){}

}
int main(){
    Y obj1 = 2;//调用函数1
    Y obj2 = "somestring"; //调用函数2，后面的int类型不用传递
    obj1 = 10;
    add(5);
    return 0;
}
```

上述就是把传入的值（正常类型）成功转换成了Y类型

```
class A {
public:
    explicit A() { };
    explicit A(int) { };
};

int main() {
    A z;
    A y = 1;//错误，隐式转换，直接把不同的数据类型赋值过来，不存在任何显式转换的过程
    A x = A(1);
    A w(1);
    A* v = new A(1);
    A u = (A)1;
    A t = static_cast<A>(1);//基本数据类型转换，<>中为需要转换的类型，括号内为转换前的数据
    return 0;
```

```
}
```

这里加了explicit关键字之后代表后面的构造函数为显式构造函数，不能被用于隐式转换，只能用于显式转换

```
class A {
public:
  A() { }       //Default constructor or class A
  A(int) { }    //Conversion constructor with one parameter
};

int main() {
   A a1 = 1.234;   //Standard conversion from 1.234 into an int.
   A as1 = "text string"; //Invalid due to nonstandard conversion
                          //from a string to an int
}
```

由于原本给A的转换构造函数为int类型的输入转换，那么当你输入double类型的其实是可以通过正常的隐式类型转换先将double转换成int后再隐式转换成A类型的，但是string类型就不可以转换成int。

conversion functions

无参数：转换函数不接受任何参数。
隐式返回类型：转换函数的返回类型是转换的目标类型，并且在函数声明中不显式指定。转换类型是由函数名称中的类型标识符确定的。
函数名称：转换函数的名称由 operator 关键字后跟要转换成的类型组成。例如，operator int() 表示转换为 int 类型。

```
#include <iostream>
using namespace std;
class Y {

public:
    int b;
    explicit operator int();
};

Y::operator int() { return b; }

void f(Y obj) {
    int i = int(obj);//显式转换
    cout<<i<<endl;
    int j = (int)obj;//c风格显式转换
    cout<<j<<endl;
    int k = i + obj;//隐式转换,如果上面是explicit，则会报错
    int k = i + int(obj);
    cout<<k<<endl;
}
int main(){
    Y obj;
```

```
    obj.b = 1;
    f(obj);
}
```

首先转换函数声明在类的内部，直接写operator 需要转换成的数据类型();

类的外部定义函数

类名::operator 需要转换成的数据类型(){一般是return属性}

转换函数一定意义上来说是类的成员函数，不需要传入对象本身的参数

同样如果转换函数之前有explicit定义的话，那么也是不可以用隐式转换的，一定要有括号形式的转换

关于连续转换

```
class A {
  int x;
public:
  operator int() {return x;}
};

class B {
  A y;
public:
  operator A() {return y;}
};

int main () {
  B b_obj;
  //int i = b_obj;报错，因为B类型不可以直接转换成int类型，只有B->A->int

  int j = A(b_obj);
    cout<< j <<endl;//正常输出0
  int q = int(A(b_obj));
    cout<< q <<endl;//正常输出0
  return 0;
}
```

关于转换的歧义

```
class Y {
public:
    Y() { x = 1; }
    int x;
    operator int() { return x; }
    operator float() { return (float)x; }
};
```

```
int main() {
    Y y1;
    long a = y1;
    return 0;
}
```

User-defined conversions must be unambiguous;

这里Y类型的数据变量被直接隐式转换成了long类型，可是明明存在两种隐式类型转换，则这个时候就会有两种转换，需要说明清楚，避免歧义