```
//四：隐式转换和explicit
//编译系统，在私下干了很多我们所不知道和不了解的事
//Time myTime40 = 14; //编译系统肯定有个行为，把14这个数字 转换成了一个Time类型 ，调用了单参数的构造函数
//Time myTime41 = (12, 13, 14, 15, 16); //调用了单参数的构造函数。
//func1(16); //16被转换成了一个临时的Time对象导致func1的调用能够成功。调用了单参数的构造函数

Time myTime100 = { 16 }; //这个写法咱们认为正常，带一个参数16.可以让系统明确的知道调用哪个构造函数
Time myTime101 = 16; //含糊不清的写法，就存在临时对象隐式转换。
func1(16); //也是含糊不清的写法，存在临时对象或者隐式转换的问题
```

```
////是否可以强制系统,明确要求构造函数不能做隐式类型转换呢？ 可以，如果构造函数声明中带有explicit,
//            //则这个构造函数只能用于初始化和显式类型转换：

Time myTime = Time(12, 13, 52); //创建类对象
Time myTime2(12, 13, 52);
Time myTime3 = Time{ 12,13,52 };
Time myTime4{ 12,13,52 };
Time myTime5 = { 12,13,52 }; //隐式类型转换。

复制列表初始化不能使用标记为"显式"的构造函数
```

```
//Time myTime100 = Time { 16 }; //这个写法咱们认为正常，带一个参数16.可以让系统明确的知道调用哪个构造函数
//Time myTime101 = Time(16); //含糊不清的写法，就存在临时对象隐式转换。
//func1(Time(16)); //也是含糊不清的写法，存在临时对象或者隐式转换的问题
//
////对于单参数的构造函数，一般都声明为explicit，除非你有特别原因。

Time time200{};
//Time myTime201 = {};

//func1({});
//func1({1,2,3});
func1(Time{});
func1(Time{ 1,2,3 });
```