# 内联函数

**出现内联的原因**：为了降低函数调用开销（查函数表，参数进行压栈，赋值）。

内联函数的目的是为了提高程序的执行效率，用关键字**inline进行函数修饰**。在调用时把程序代码进行展开，减少函数调用时的开销。使用内联函数后，编译器可以通过上下文相关的优化技术对结果代码进行深入优化。

构建方法：

1. 在函数声明时，返回类型前面加上inline关键字
2. 在函数定义时，返回类型前面加上inline关键字

一般是建议内联函数的声明和定义放在一起，过于复杂的函数编译器会自动修改内联函数为普通函数。

##### 不宜内联的函数

1. 如果函数体内的代码较长，使用内联函数将导致代码膨胀过大，内联被取消
2. 如果函数体内出现循环，或是其他复杂的控制结构，那么执行函数体内的代码时间将比函数调用的开销大很多，因为内联的意义不大
3. 深度调用函数不建议使用内联，例如递归
4. 对内敛函数不能进行异常接口声明

##### 内联函数VS宏

1. 宏本身也可以做到提高代码执行效率（编译阶段操作不占用执行效率），我们可以借助宏编写精简函数
2. 宏的缺点在于预处理阶段处理器在拷贝宏代码时，会产生边界效应。例如运算符优先级问题，展开无逻辑问题。
3. 宏本身不是代码，无法参与逻辑调试
4. 宏本身无法操作类的私有成员
5. 宏参数不进行类型检查
6. 内联函数增加了类型安全检查，可以操作类的成员

# 友元

**非类的成员函数**，可以访问类的所有成员函数或是成员变量，这种操作方式称为友元操作。

友元分为友元类和友元函数（全局友元函数，成员友元函数），**主要目的是为了共享数据**

##### 前向声明

声明一个类，并不定义它，可以在其他地方去定义类的结构，这种声明方式我们称之为前向声明

##### 注意：

前向声明，在声明之后，定义之前，该类是不完整类型，无法直接操作使用，不完整类型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是返回类型的函数（只能用来声明指针）

### 友元函数

普通友元函数（全局函数）

注意：由于友元函数无法获取类的实例的this指针，所以如果需要操作类数据则需要通过参数传入类的引用或是指针（声明的友元函数不是该类的成员）

参数要点：

1. 要访问非static成员时，需要对象作为参数（引用或是指针）
2. 访问static成员或是全局变量，则不需要对象作为参数
3. 如果做参数的对象是全局对象，则不需要对象做参数

友元函数的位置

友元函数不属于类的成员函数，所以不受访问修饰符的约数

### 某个一类的成员函数是另一个类的友元函数

注意：类的函数必须先要进行声明，才可以成为另一个类的友元，函数的访问域不受限制，放到另一个类中的位置也无限制，调用和使用普通函数一样。

### 友元类

特点：A是B的友元类

1. 友元类的声明位置，不受访问修饰符约束
2. A中可以访问B的所有成员，但是要持有B的有效引用或是指针

# 运算符重载

基本数据类型的重载是无法进行的，我们需要的是进行自定义数据类型的重载。也就是说，如果重载运算符，至少要有一个是用户自定义的数据类型，否则无法完成操作。

### 重载运算符的意义

直观的通过基本认知运算符完成对象数据的计算工作，可以帮助我们更加方便的完成对象数据间的处理转换规则，运算符重载是一种多态的表现。

**成员运算符重载：**本质为类添加了一个特殊的成员函数，函数的名字是运算符。需要使用operator关键字进行修饰

**隐藏内部机理，强调实质，是面向对象编程的目标**

### 注意：

* （成员运算符重载）参数列表中的内容将是重载运算符后与原数据进行操作的数据
* （全局运算符重载）如果运算符需要两个数据进行运算，则需要传入两个参数参与运算。其中第一个参数是运算符左边的参数，第二个是运算符右边的参数
* 无法为指针类型数据进行重载运算符。例如两个指针数据相加，因为指针本身进行成员函数调用需要借用成员引用运算符->，而无法直接调用重载运算符，如下面的例子

class Package& operator+(Package\*, Package\*);//重载了指针之间的加法运算，此操作是违规操作，禁止使用

指针属于地址型数据类型，虽然是复合数据类型，但是指针本身也是数值。并且基本数学运算符均被保留使用，所以无法进行重载指针与指针之间的运算，包括指针和整形之间的运算。

### 友元在运算符重载中的意义

对于自定义的两种数据类型，如果想要实现运算关系，经常遇到私有和受保护数据无法进行互相操作，友元可以方便解决此类问题。