|  |  |
| --- | --- |
| 1. 使用function生成一个可调用对象 2. 使用mem\_fn生成一个可调用对象 3. 使用bind生成一个可调用对象 | Function<bool(const string&) fcn = &string::empty  Mem\_fn(&string::empty)  Bind(&string::empty,\_1) |
| 1. 因为将成员函数指针当做算法的运算对象是不可的   Fp(\*it)错误,必须有.\*与->\*和对象，但是下面3个方法省去了这两个东西   1. 使用function生成一个可调用对象 2. ((\*it).\*p)(),p是fcn内部的一个指向成员函数的指针，it是迭代器\*it获得对象   2）function中的参数必须根据实际对象是指针还是引用进行调整为\*或者&   1. 使用mem\_fn生成一个可调用对象 2. 这个可以通过对象调用，也可以通过指针调用，就无需像1一样要调整 3. 实际上有一对重载的函数，可以推断可调用对象的类型 4. 使用bind生成一个可调用对象 5. 和funciton类似，必须将函数中用于执行对象的隐式形参转换成显示的 6. 和mem\_fn类似，bind生成的可调用对象的第一个实参既可以是string的指针，也可以是string的调用 | |