|  |  |
| --- | --- |
| 1. 成员访问运算符 2. 对箭头运算符返回值的限定 | String& operator\*()const  {  Return (\*p)[curr];  }  String\* operator->()const  {  Return &this->operator\*();  }  StrBlobPtr p (a1);  \*p = “okay”;  Cout<<p->size()<<endl;  Cout<<(\*p).size()<<endl;//等价p->size();  Cout<<p.operator->size()<<endl; |
| 1. 成员访问运算符 2. \*与-> 3. 箭头运算符必须是类的成员，解引用通常也是不过可以不用 4. ->返回string的指针 5. 对箭头运算符返回值的限定,这里看不懂。 6. 箭头运算符永远不能丢掉成员访问这个最基本的含义?不知道什么意思，说可以改变的是箭头从哪个对象当中获取成员，而箭头获取成员这一事实永远不会改变 7. 形如point->mem的表达式来说，point必须是指向类对象的指针或者是一个重载了operator->类的对象。 8. 调用：   1．\*point.mem;// point是一个指针  2. point.operator()->mem;//point是类一个对象   1. 怎么说，就是若point是指针，则执行第一个操作,先解引用指针，然后从所得的对象中获取指定的成员。   若point是一个对象，则使用point.operator->()的结果来获取mem。若获取的结果是一个指针，则执行第一步，解引用，若结果本身含有重载的operator->(),重复调用当前步骤。最终得到所需内容？？？？  Note:重载的箭头运算符必须返回类的指针或者自定义了箭头运算符的某个类的对象 | |