|  |  |
| --- | --- |
| 1. dynam\_cast运算符 2. 指针类型的dynam\_cast 3. 引用类型的dynamic\_ast | Dynamic\_cast<type\*)(e)  Dynamic\_cast<type&(e)  Dynamic\_cast<type&&)(e)  Dynamic\_cast<type\*|&|&&)(e) |
| 1. 运行时类型识别 2. 使用某种类型的指针或引用，并且该类型含有虚函数时，运算符将使用指针或引用所绑定对象的动态类型。在dynamic\_cast的e是动态类型吗，是要求继承体系具有动态类型(即：需要虚函数)，才能发生dynamic 3. dynam\_cast运算符   type是一个类类型，通常有虚函数  e的类型是目标type的公有派生类、  e的类型是目标type的公有基类，向下转型是失败的指针为000，引用抛出bad\_cast  e的类型就是目标type的类型   1. 指针类型的dynam\_cast   e必须是有效的指针  转换失败为0  可以对一个空指针执行，结果是所需类型的空指针   1. 引用类型的dynamic\_cast   e必须是左值  右值引用的话：e不能是左值(试了下，c+14好像可以是左值)  转换失败抛出bad\_cast异常  不存在空引用，所以抛出异常,指针是为0，不会抛出异常  代码：  1.注意需要将类具有虚函数，才能发生动态类型绑定  // 向下转型，错误  ZooAnimal z0;  Bear ba0 = dynamic\_cast<Bear&>(z0);  try  {  ZooAnimal \*z0 = new ZooAnimal;  // 引用抛出异常，不支持向下转型  //C& cp = dynamic\_cast<C&>(\*pbb);  //Bear ba0 = dynamic\_cast<Bear&>(z0);    // 指针不抛出异常但是指针值为0  //C \*pc = dynamic\_cast<C\*>(pbb);  //Bear \*ba00 = dynamic\_cast<Bear\*>(z0);  }  catch (const std::exception& e)  {  cout << e.what() << endl;  } | |