6-29运行下面的程序，观察执行结果，指出该程序是如何通过指针造成安全性隐患的，思考如何避免这种事情的发生。

|  |
| --- |
| #include<iostream>  using namespace std;  int main()  { int arr[] = { 1,2,3 };  double \*p = reinterpret\_cast<double\*>(&arr[0]);  \*p = 5;  for (int i = 0; i < 3; i++) { cout << arr[i] << ends; }  system("pause");  return 0;  } |

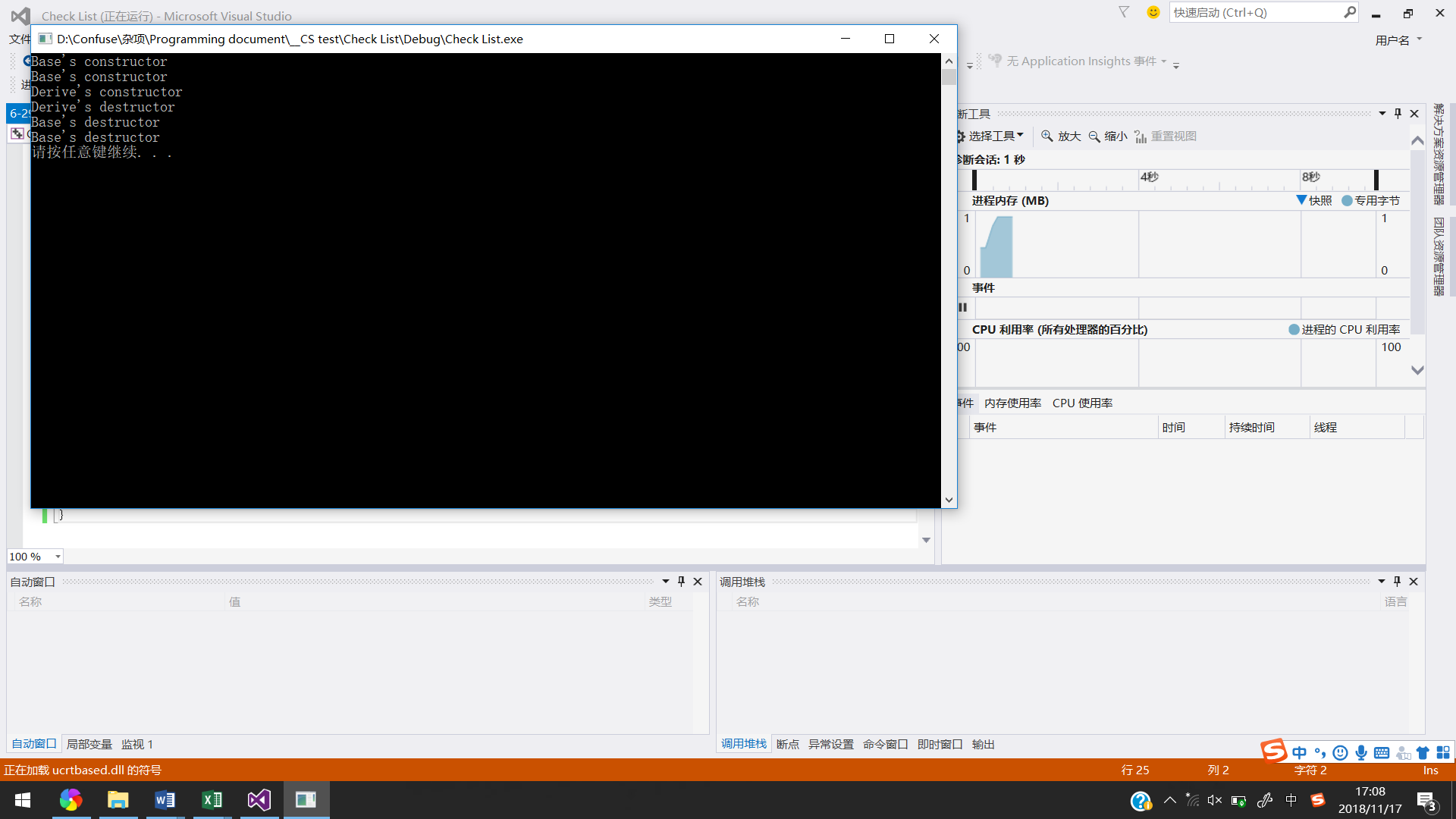
数组不应进行强制类型转换。

7-2派生类构造函数执行的次序是怎样的？

答：先调用基类的构造函数，然后再构造内嵌对象的构造函数。其中，基类构造函数的调用顺序是按照派生类定义时的顺序；而内嵌对象的构造函数调用顺序是按照成员在类中声明的顺序。

7-7定义一个基类及其派生类，在构造函数中输入提示信息，构造派生类的对象，观察构造函数的执行状况。

|  |
| --- |
| #include<iostream>  class Base  {public:  Base(int i) { std::cout << "Base's constructor" <<std::endl; };  ~Base() { std::cout << "Base's destructor" << std::endl; };  };  class Derive :public Base  {public:  Derive(int a, int b) :Base(a), member(b)  {std::cout << "Derive's constructor" << std::endl;};  ~Derive() { std::cout << "Derive's destructor" << std::endl; };  private:  Base member;  };  int main()  {  Derive(2, 5);  system("pause");  return 0;  } |

执行结果为：

7-9定义一个基类Base，有两个共有成员函数fn1()，fn2()，私有派生出Derived类，如何通过Derived类的对象调用基类的函数fn1()?

答：因为Derived类是由Base类私有派生出的，Base类的公有对象和保护对象不能通过Derived类的对象在类外直接访问。因此我们可以在Derived类中添加这样一行代码，使Derived类的私有对象依旧能在类外访问：void fn1() {Base::fn1();}