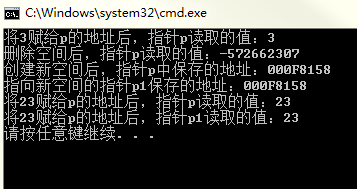
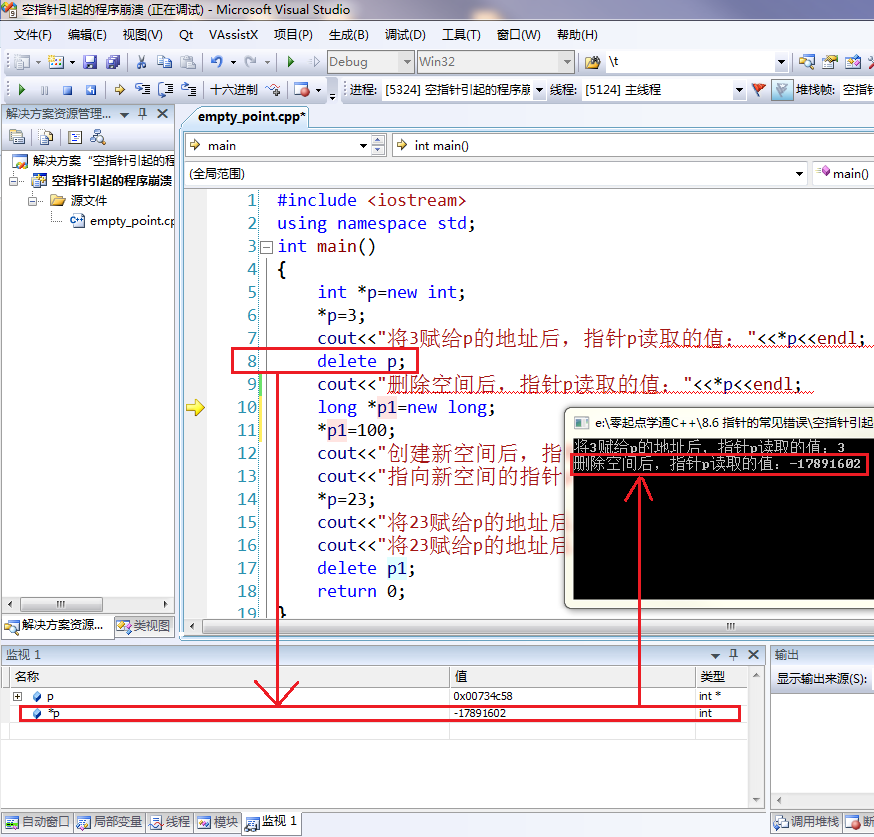
  众所周知，最开始我们用new来创建一个指针，那么等我们用完它之后，一定要用delete将该指针删掉。但是，值得注意的是，难道就仅仅是删除这个指针这么简单的么？下面，我们用一个程序来说明这个问题：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19 | #include <iostream>  using namespace std;  int main()  {      int \*p=new int;      \*p=3;      cout<<"将3赋给p的地址后，指针p读取的值："<<\*p<<endl;      delete p;      cout<<"删除空间后，指针p读取的值："<<\*p<<endl;      long \*p1=new long;      \*p1=100;      cout<<"创建新空间后，指针p中保存的地址："<<p<<endl;      cout<<"指向新空间的指针p1保存的地址："<<p1<<endl;      \*p=23;      cout<<"将23赋给p的地址后，指针p读取的值："<<\*p<<endl;      cout<<"将23赋给p的地址后，指针p1读取的值："<<\*p1<<endl;      delete p1;      return 0;  } |

       在上面这个程序中，我们在第8行就将指针p利用delete删掉了。但是，我们来看看程序的输出结果：



      对照着上面的程序，我们来分析一下这个输出。首先，我们在程序的第5行初始化了一个指针p。之后输出指针p读取的值。由于第6行的原因，程序肯定会输出3了。之后，我们在程序的第8行删除了这个指针p。但是我们惊奇的发现，在程序的第9行竟然可以输出指针p读取的值。我们不是已经把它删了么？其实不然，debug，上图：



      从监视窗口中，我们可以看见虽然程序的第8行已经将指针p删除了，但是在监视窗口中p仍然存在，只是\*p所指向的值不再是原来的3了，而是一个随机数。这里就说明了一个非常重要的概念：**我们在删除一个指针之后，编译器只会释放该指针所指向的内存空间，而不会删除这个指针本身。**

      然后我们接着往下分析。在程序的第10行我们又创建了一个long型的指针p1。在12行与13行的输出中我们惊奇地发现，指针p保存的地址居然和指针p1保存的地址一模一样！这个就说明了指针p和指针p1都指向内存的同一个地方！！！出现这种状况的原因其实是由于编译器。**编译器默认将释放掉的内存空间回收然后分配给新开辟的空间。**所以在第11行由于我们新开辟了一个可以保存long型变量的空间并且由p1来指向它，那么这里的p1指向的其实就是在程序第8行释放掉的内存空间，即p指向的内存空间！所以，这就导致了两个指针同时指向同一个内存空间。这是多不安全的一件事情啊！要知道，我们是把指针p删了的啊！如果再重新对\*p进行赋值操作，那么不是会连着\*p1一起改动么？

      果然，让我们担心的事情出现了。我们明明在程序的第11行中定义了\*p1的值为100，但是在输出上面，指针p1读取的值竟然也是23。这个原因就是因为野指针p造成的。我们可以看到，在程序的第14行我们将23赋给了\*p。又由于p和p1指向的是同一块内存单元，所以在这里相当于也将p1所指向的内存单元中的值（原来是100），改成了23！这样必然会导致程序的出错！

       那么我们就不禁要问了，对于这种由于野指针造成的问题，有没有解决的方法呢？答案当然是有的了。我们只需要牢记下面这句话：

**在删除一个指针之后，一定将该指针设置成空指针（即在delete \*p之后一定要加上： p=NULL）**

      我们来看一下在stdio.h中关于关键字NULL的定义：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | /\* Define NULL pointer value \*/    #ifndef NULL  #ifdef \_\_cplusplus  #define NULL    0  #else  #define NULL    ((void \*)0)  #endif  #endif |

      注意上面定义的第5行。这里其实就说明了NULL就是0。也就是说，我们在删除完指针p之后，一定要把它变成空指针！只有这样，才会杜绝上面程序中出现的野指针的错误。

p.s. 对于NULL的应用，我们不应该仅限于上面的方法，还可以应用NULL来判断指针是否初始化成功了，如下例if中的判断方法：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21 | #include <iostream>  using namespace std;    int main()  {      int \*p=new int;      if (p==NULL)      {          //判断指针p是不是空指针，如果是空指针，那么程序在这里就应该报错          //报错的方法有很多，比如说返回一个ERROR值：          //return ERROR;      }        //判断了操作成功之后我们才能进行一系列的操作      //...        //用完指针p之后，一定要将其删掉。这样可以杜绝野指针的存在      delete p;      //删除指针p之后，一定要加上下面这句话，免得成为野指针      p=NULL;  } |

         好了，下次一定要记住，在分配空间给指针之后，一定要用**NULL**来判断一下是否成功了。然后在删除这个指针的时候，也要用**NULL**来赋给指针，杜绝成为野指针！

O(∩\_∩)O哈哈~写完，收工~~~~