虚析构函数

- 如果使用基类指针释放动态创建的派生类对象，则需要将析构函数声明为虚函数。

【例3-7】虚析构函数示例。

```
// Person.h
#include <iostream>
using namespace std;
class Person
{
public:
    Person(char *name, bool sex)
    {
        m_name = new char[strlen(name)+1];
        strcpy(m_name, name);
        m_sex = sex;
        cout<<"Person类构造函数被调用！"<<endl;
    }
    virtual ~Person()// 将析构函数声明为虚函数
    {
        delete []m_name;
        cout<<"Person类析构函数被调用！"
            <<endl;
    }
protected:
    char *m_name;           // 姓名
    bool m_sex; // 性别（true：男，false：女）
};
```

```
// Student.h
#include "Person.h"
class Student : public Person
{
public:
    Student(char *sno, char *name, bool sex, char *major) : Person(name, sex)
    {
        m_sno = new char[strlen(sno)+1];
        m_major = new char[strlen(major)+1];
        strcpy(m_sno, sno);
        strcpy(m_major, major);
        cout<<"Student类构造函数被调用！"<<endl;
     }
    ~Student()                  // 等价于virtual ~Student()
    {
        delete []m_sno;
        delete []m_major;
        cout<<"Student类析构函数被调用！"<<endl;
    }
private:
    char *m_sno;                // 学号
    char *m_major;              // 专业
};
```

```
// main.cpp
#include "Student.h"
int main()
{
    Person *pPerson = new Student("1210101", "张三", true, "计算机应用");
    delete pPerson;
    return 0;
}
```

程序运行结果为：

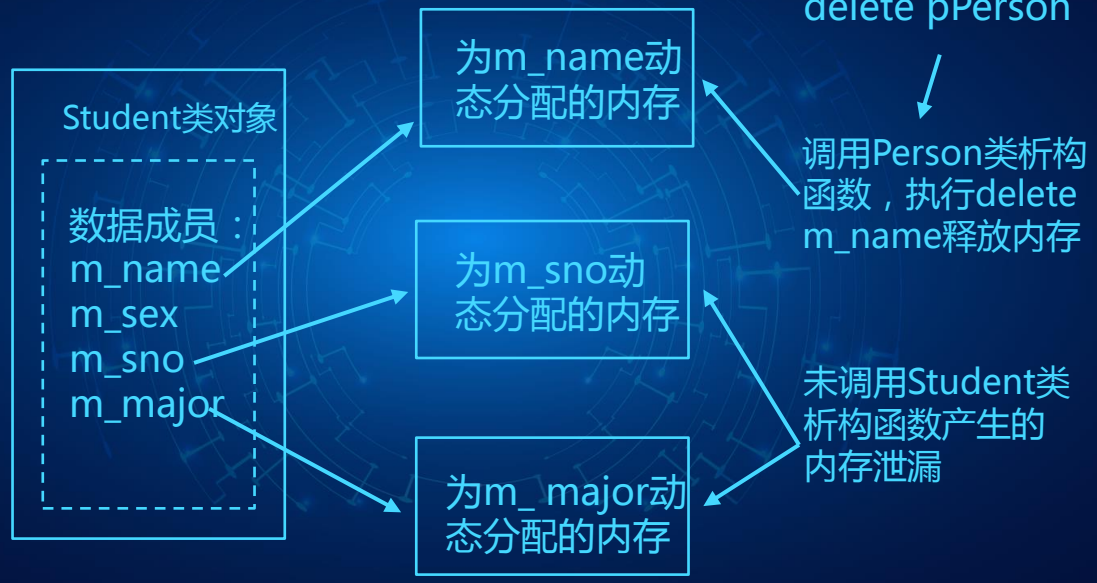
Person类构造函数被调用！

Student类构造函数被调用！

Student类析构函数被调用！

Person类析构函数被调用！

Person*pPerson=new Student;



注意：

当一个类不准备作为基类使用时，一般不要使析构函数成为虚函数，因为它会为类增加一个虚指针和一个虚函数表，使得对象的尺寸翻倍，并会降低其可移植性。实际上，当且仅当类作为基类，才把析构函数声明为虚函数。