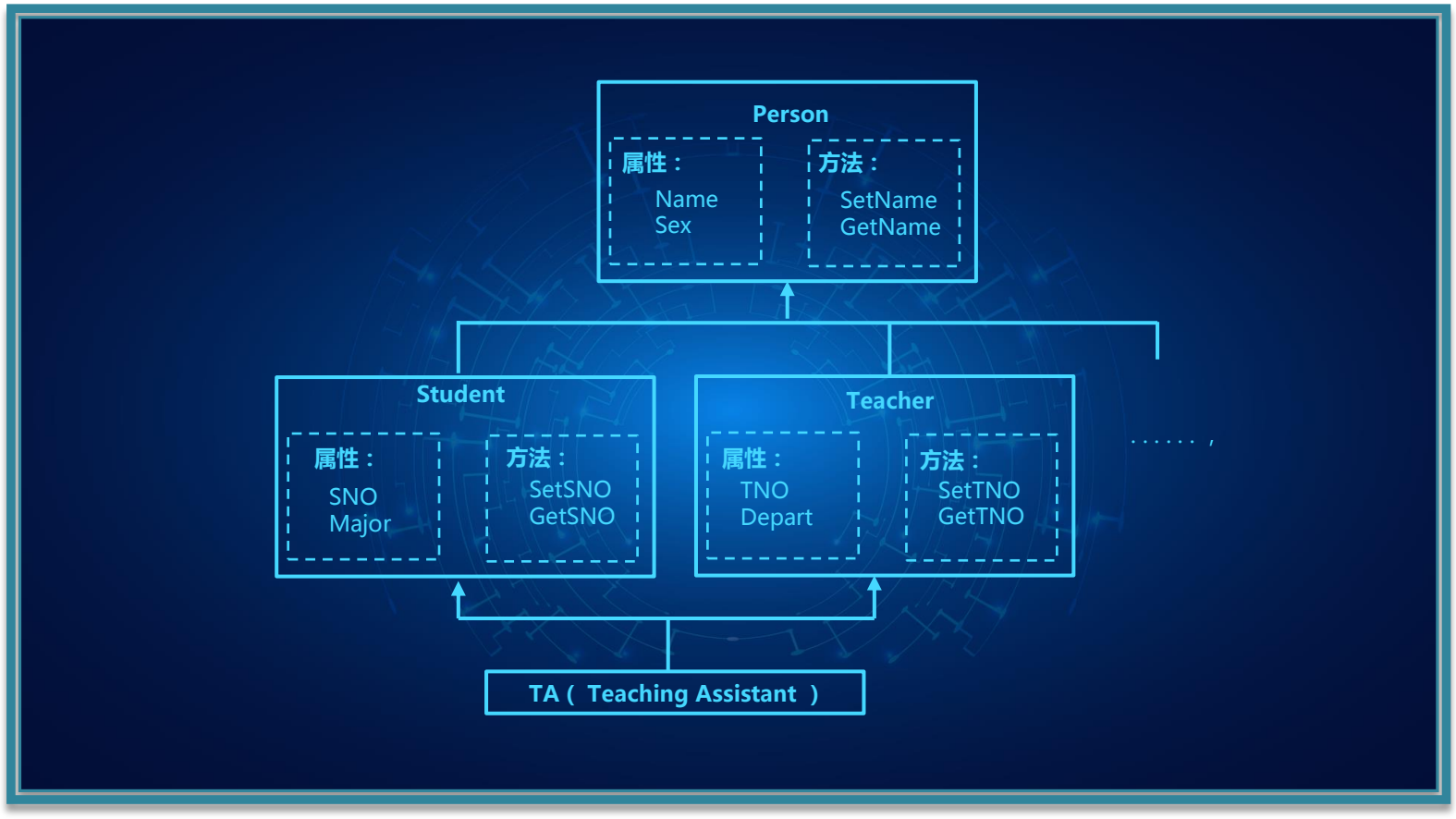
多重继承下的派生类定义


Person
属性：
Name
Sex
方法：
SetName
GetName
Student
属性：
SNO
Major
方法：
SetSNO
GetSNO
Teacher
属性：
TNO
Depart
方法：
SetTNO
GetTNO
.....,
TA (Teaching Assistant)

通过对Student类和Teacher类进行多重继承得到TA类。在一个多重继承关系中，定义派生类的语法为：

```
class 派生类名：继承方式 基类名1, 继承方式 基类名2, …, 继承方式 基类名n
{
    派生类成员声明;
};
```

其中，不同的基类可以使用不同的继承方式。

【例3-3】实现图中的程序代码。

```
// Person.h
#include <iostream>
using namespace std;
#ifndef _PERSON_H
#define _PERSON_H
class Person
{
public:
    Person(char *name, bool sex)
    {
        strcpy(m_name, name);
        m_sex = sex;
        cout<<"Person类构造函数被调用！"<<endl;
    }
    ~Person()
    {
        cout<<"Person类析造函数被调用！"<<endl;
    }
    void SetName(char *name)
    {
        strcpy(m_name, name);
    }
    char* GetName()
    {
        return m_name;
    }
private:
    char m_name[20];        // 姓名
    bool m_sex;  // 性别（true：男，false：女）
};
#endif
```

```
// Student.h
#include "Person.h"
class Student : public Person
{
public:
    Student(char *sno, char *name, bool sex, char *major) : Person(name, sex)
    {
        strcpy(m_sno, sno);
        strcpy(m_major, major);
        cout<<"Student类构造函数被调用！"<<endl;
    }
    ~Student() { cout<<"Student类析构函数被调用！"<<endl; }
    void SetSNO(char *sno) { strcpy(m_sno, sno); }
    char* GetSNO() { return m_sno; }
    void SetMajor(char *major) { strcpy(m_major, major); }
    char* GetMajor() { return m_major; }
private:
    char m_sno[8];              // 学号
    char m_major[20];           // 专业
};
```

```
// Teacher.h
#include "Person.h"
class Teacher : public Person
{
public:
    Teacher(char *tno, char *name, bool sex, char *depart) : Person(name, sex)
    {
        strcpy(m_tno, tno);
        strcpy(m_depart, depart);
        cout<<"Teacher类构造函数被调用！"<<endl;
    }
    ~Teacher() { cout<<"Teacher类析构函数被调用！"<<endl; }
    void SetTNO(char *tno) { strcpy(m_tno, tno); }
    char* GetTNO() { return m_tno; }
    void SetDepart(char *depart) { strcpy(m_depart, depart); }
    char* GetDepart() { return m_depart; }
private:
    char m_tno[6];              // 教师号
    char m_depart[20];          // 系
};
```

```
// TA.h
#include "Student.h"
#include "Teacher.h"
class TA : public Student, public Teacher
{
public:
    TA(char *sno, char *name, bool sex, char *major, char *tno, char *depart)
        : Teacher(tno, name, sex, depart), Student(sno, name, sex, major)
    {
        cout<<"TA类构造函数被调用！"<<endl;
    }
    ~TA() { cout<<"TA类析构函数被调用！"<<endl; }
};
```

```
// testTA.cpp
#include "TA.h"
int main()
{
    TA ta("1210102", "王五", true, "计算机应用", "09110", "计算机科学与技术系");
    return 0;
}
```

在创建通过多重继承定义的派生类对象时，也会先调用基类的构造函数，再调用派生类的构造函数。各基类构造函数的调用顺序与多重继承时的继承顺序一致。在创建TA类时，先继承Student类、再继承Teacher类，因此，会先调用Student类的构造函数、再调用Teacher类的构造函数。Student类和Teacher类又是Person类的派生类，因此，在执行Student类构造函数前会先调用Person类构造函数、在执行Teacher类构造函数前也会先调用Person类构造函数。在创建TA类对象时，程序会在屏幕上输出：

Person类构造函数被调用！

Student类构造函数被调用！

Person类构造函数被调用！

Teacher类构造函数被调用！

TA类构造函数被调用！

提示：析构函数调用顺序总是与构造函数调用顺序相反。