# 第7章　数据的抽象与封装

| | | | |
|---|---|---|---|
| 1、实体、对象与类的概念 | 2、类的定义 | 3、对象声明与引用 | 4、私有、公有与保护 |
| 5、日期类的设计 | 6、两种程序设计思想 | 7、汽车类的设计 | 8、几何图形圆类的设计 |
| 9、构造函数的定义 | 10、重载构造函数 | 11、析构函数的定义 | 12、整数翻译函数 |
| 13、实际意义的析构函数 | 14、Person类的设计 | **15、对象与指针** | 16、this指针 |

# 对象与指针

- 指向对象的指针

  类名 *指针变量名表;

  例：

```
Person person1(“Zhang3”,19,’f’);
Person *ptr=&Person1;
ptr->ShowMe();
```

- 动态存储

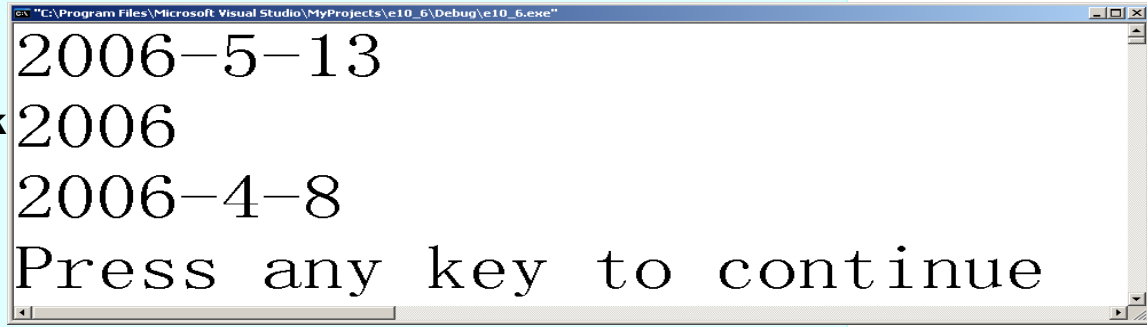
  对象指针＝new 类名(名字初始化值);

  delete 名字指针;

  例:

```
Person *p=new Person;
delete p;
```

```
#include<iostream>//日期类定义
using namespace std;
class Date
{
public:
        int year,month,day;
        void init(int y,int m,int k
        void print_ymd();
};
void Date::init(int  yy,int  mm,int
{       month = mm;       year
void Date::print_ymd()
{       cout<<year<<"-"<<month<<"-"<<day<<endl; }
int main()
{       Date date1;
        Date *p1 = &date1;
        p1->init(2006,5,13);
        p1->print_ymd();
        int *p2 = &date1.year;
        cout << *p2 <<endl;
        void (Date::*p3)(int ,int ,int);  //普通函数指针定义不行
        void (Date::*p4)();
        p3 = Date::init;
        p4 = Date::print_ymd;
        (date1.*p3)(2006,4,8);
        (date1.*p4)();
        return 0;
}
```



```
#include<iostream>
#include<string.h>
using namespace std;
class Person
{private:
        char  Name[9];
        char  Sex;
        int     Age;
public:
        Person( )
        {       strcpy(Name,"XXX");
                Age = 0;
                Sex = ' ';
        }
        ~Person( )
        {       cout<<"Now destroying Person"<<endl; }
        void Register( char *name, int age, char sex);
        void ShowMe( );
};
void  Person::Register(char *name,int age,char sex)
{       strcpy(Name,name);
        Age = age;
        Sex = sex;
}
void Person::ShowMe()
{       cout<<Name<<'\t'<<Age<<'\t'<<Sex<<endl;}
```

```
int main()
{
        Person  *p1,*p2;                    //声明两个指向对象的指针
        p1=new Person;                      //动态生成一个Person对象
        cout << "person1: \t";
        p1->ShowMe();
        p1->Register("Zhang3", 19, 'm');
        cout << "person1: \t";
        p1->ShowMe();
        p2=new Person;                      //动态生成一个Person对象
        cout << "person2: \t";
        p2->ShowMe();
        *p2 = *p1;                          //对象之间的赋值
        cout << "person2: \t";
        p2->ShowMe();
        delete p1;                          //释放p1指针指向对象所占的空间
        delete p2;                          //释放p2指针指向对象所占的空间
        return 0;
}
```

# 感谢收看！