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

两种程序设计思想

- 面向过程的程序设计(Structured Programming)
  - 以功能为中心，通过分解功能，采用函数来描述操作
  - 数据与函数分离，数据(类型或结构)一旦发生改变，函数也要相应改变
  - 例如排序函数：void sort(int a[],int n);只能排序整数数组
- 面向对象程序设计(Object Oriented Programming)
  - 以数据为中心，采用对象来描述内部属性和操作方法
  - 将数据和函数当作一个统一体，采用软件对象模拟客观世界中实际对象
  - 例如：钟表类、日期类

OOP方法的特征

- 抽象：实体特征+实体功能
  - 如钟表、日期、分数等实体的抽象描述
- 封装：数据和函数封装组成统一体
  - 通过公有成员函数修改数据、显示数据、取得数据
- 继承：将对象(实体)之间的差异通过继承性加以抽象描述
  - 动物→
  - {人、大象、鸟...}→
  - {学生、工、农、兵...} →
  - {大学生、中学生...} →...
- 多态性：指相同语法结构(数据或操作)代表多种功能
  - 如+、—、*、/

OOP方法的优点

- 开发时间短，效率高，可靠性高
- 重用，共享，可维护性，精简
- 适合于大型程序长时间的团队开发工作

思考为什么要引入

- 为什么要引入数组？
  - 太多的变量（全班同学的C++成绩）
- 为什么要引入函数？
  - 结构化程序设计：功能分解，代码重用
- 为什么要引入指针？
  - 加快速度、参数传递、动态分配内存
- 为什么要引入结构体？
  - 复合数据结构，不同数据类型的聚合在一起

为什么要引入类

- 采用人们认识客观世界的自然方式去模拟客观世界中对象
  - 将客观世界中的实体完整性的描述(即数字化)
- 提高开发程序的效率和可靠性
  - 数据与代码的聚合(以便更高级的代码复用)

日期类的抽象描述

- 客观世界的日期信息抽象描述如下：
  - 数据成员：年、月、日
    - int year,month,day;
  - 函数成员：初始化数据、输出日期、取日期值
    - void init(int y,int m,int d);
    - void print_ymd();
    - void print_mdn();
    - int get_year();
    - int get_month();
    - int get_day();

日期类封装定义

```
class Date
{
        int year,month,day;
public:
        void init(int,int,int );
        void print_ymd();
        void print_mdy();
        int get_year() { return year; }
        int get_month() { return month; }
        int get_day() { return day; }
};
void Date::init(int  yy,int  mm,int  dd)
{       month=(mm>=1&&mm<=12) ? mm:1;
        year=(yy>=1900&&yy<=9999) ? yy:1900;
        day=(dd>=1&&dd<=31) ? dd:1;
};
void Date::print_ymd()
{       cout<<year<<"-"<<month<<"-"<<day<<endl;}
void Date::print_mdy()
{       cout<<month<< "-" <<day<< "-" <<year<<endl;}
```

- 日期数据修改靠init()函数
- 日期数据显示靠一系列print函数
- 日期数据获得靠一系列get函数
- 对象中的数据输入输出有接口函数
- 确保对象中的数据安全性
- 日期对象整体性与离散性操作自如

日期时间类继承性设计思考

- 日期时间类的特征成员：
  - 年、月、日、时、分、秒
- 日期时间类的功能成员
  - 设置日期时间函数
  - 显示日期时间函数
  - 一系列得到特征数据值的函数
- 假设已存在(定义)日期类、时间类
- 如何定义日期时间类呢？

已定义完成的日期类和时间类如下：

```
class Date  //日期类定义
{
    int year,month,day;
public:
    void init_date(int,int,int );
    void print_ymd();
    void print_mdy();
    int get_year() { return year; }
    int get_month() { return month; }
    int get_day() { return day; }
};
```

```
class Time  //时间类定义
{
    int hour,minute,second;
public:
    void init_time(int,int,int );
    void print_hms();
    int get_hour() { return hour; }
    int get_minute() { return minute; }
    int get_second() { return second; }
};
```

定义日期时间类的两种可能方式

1. 重新完整定义日期时间类，代码重复
2. 利用已存在的类定义日期时间类，代码复用
   - ☞ 这就是继承性抽象描述实体的基本思路，第8章详细介绍

```
class DateTime  //重新完整定义
{
        int year,month,day,hour,minute,second;
public:
        void init_datetime( int,int,int,int,int,int );
        void show();
        int get_year() { return year; }
        int get_month() { return month; }
        int get_day() { return day; }
        int get_hour() { return hour; }
        int get_minute() { return minute; }
        int get_second() { return second; }
};
```

```
class DateTime:public date,time          //继承定义
{
public:
    void init_datetime(int y,int m,int d,int h,int mi,int s){init_date(y,m,d);init_time(h,mi,s);};
    void show(){print_ymd();print_hms();};
};
```

日期时间类多态性设计思考

- 假设有两个日期时间的对象：mydatetime，yourdatetime
- 计算这两个对象之差：mydatetime-yourdatetime
- 结果是什么呢？相隔多少年？相隔多少日？相隔多少分？
- 毫无疑问这些结果都是有意义的
- 问题是能不能直接计算表达式： mydatetime-yourdatetime
- 减号两边是整数、浮点数、双精度数都可以计算
- 当然希望减号两边也可以是日期时间对象，这就是表达式的**多态性**
- 当然希望减号两边也可以是分数对象
- 第9章详细介绍

感谢收看！