软件开发技术

第11章 实现工作流

浙江大学 机械工程学院 航空制造工程研究所

王 青 李江雄

本章内容

- 1 、实现工作流任务
- 2、面向对象设计语言
- 3、程序设计风格
- 4、面向对象程序设计风格

01 实现工作流任务

01 实现工作流任务

浙江大学 ZHEJIANG UNIVERSITY

- 实现工作流主要包括两项工作:

  把面向对象的设计结果翻译成用某种程序语言书写的面向对象程序

  测试并调试面向对象的程序

面向对象程序的质量基本上由面向对象设计的质量决定，但是，所采用的程序语言的特点和程序设计风格也将对程序的可靠性、可重用性及可维护性产生深远的影响。

软件测试是保证软件可靠性的主要措施，面向对象测试的目标是用尽可能低的测试成本发现尽可能多的软件错误。面向对象程序中特有的封装、继承和多态等机制，给面向对象测试带来一些新的特点，增加了测试和调试的难度。

面向对象的设计结果，可以用面向对象语言来实现，也可以用非面向对象语言来实现。

由于面向对象语言本身充分支持面向对象概念的实现，所以，编译程序可以自动把面向对象概念映射到目标程序中，而使用非面向对象语言编写面向对象程序，则必须有程序员自己将面向对象概念映射到目标程序中。

显然，对于面向对象的设计，使用面向对象语言远比使用非面向对象语言方便。

02 面向对象设计语言

2.1面向对象语言的技术特点

（1）支持类与对象概念的机制

所有面向对象语言都允许动态创建对象，并且可以用指针引用动态创建的对象。

（2）实现整体-部分（聚集）结构的机制

实现聚集一般有两种方法：使用指针；独立的关联对象。

（3）实现继承（泛化）结构的机制

（4）实现属性和服务的机制

- 对于属性：包括：支持实例连接的机制、属性的可见性控制、对属性值的约束
- 对于服务：支持消息连接的支持，控制服务的可见性、动态绑定

（5）类库

大多数面向对象语言都提供一个可以使用的类库。许多软件基础代码就不必由程序员从头编写，为实现软件重用提供了很多便利。

类库中往往包含通用数据结构的类，如:动态数据、表、队列、堆栈、树。

更完整的类库通常还提供独立与具体设备接口的接口类，此外用于窗口系统的用户界面类也往往被开发集成系统做成类库。

2.2 选择面向对象语言

开发人员在有条件选择面向对象语言时，应该着重考虑如下一些因素：

(1) 将来能否占主导地位

(2) 可重用性

应该优先选用能够最完整、最准确表达问题域的面向对象语言。这类语言编写的代码可重用性最强。

(3) 类库和开发环境

- 开发环境和类库对程序的可重用性也有重大影响。
- 考虑类库时，应该考虑类库中提供了哪些有价值的类。
- 开发环境中是否集成了类库编辑和浏览工具。

(4) 其他因素

- 能够提供开发人员使用的开发工具、开发平台、发行平台
- 对机器性能和内存要求
- 集成已有软件的难易程度
- 类型检查 ……等等

03

程序设计风格

03 程序设计风格

- 程序不只是给机器执行的，也是供人阅读的。
  - 编写程序的人要阅读程序
  - 参与测试的人要阅读程序
  - 维护的人要阅读程序
- 程序的可读性，对测试、维护尤其重要。

  为了提高程序的可读性，提高程序质量，在编写程序时，需要讲究编程风格。
- 程序设计风格至少包括3个方面：
  - 源程序文档化
  - 数据说明的方法
  - 语句结构

浙江大学 ZHEJIANG UNIVERSITY

3.1 源程序文档化

源程序文档化包括标识符的命名、安排注释、程序文本排版等。

●标识符的命名

➢使用有意义的命名

标识符包括模块名、变量名、标号名、子程序、数据区、缓冲区、类名、属性名、方法名、消息名等等，这些名字应能反映它所代表的实际内容，使其能够见名知意，有助与对程序功能的理解。如times表示次数、total表示总量、average表示平均值，用sum表示和。

➢用 i 做前缀表示该变量为整型，f 做前缀表示浮点型，d做前缀表示双精度型，c做前缀表示字符型，ch或char做前缀表示字符串，ptr做前缀表示指针或地址。

➢ 使用一致的表示方法

如有四个变量名：averageFreq，frequencyMaximum，minFr，frqnTotl。

（1）Freq、frequency、Fr、frqn作为同一意义的话，则应该用相同的表示，如Freq，或frequency，或frqn。

（2）两个词组合的顺序也应该一致，理想可能是freqAverage，freqMaximum，freqMinmum，freqTotal。

➢ 名字不是越长越好，过长的名字会增加工作量，应该选择精炼的、意义明确的名字，简化程序语句，必要时可使用缩写：

如：frequency缩写为freq, temperature缩写为tmptr

03 程序设计风格

浙江大学 ZHEJIANG UNIVERSITY

●程序的注释

正确的注释能够帮助读者理解程序，为测试和维护提供明确的指导，注释绝不是可有可无的。

大多数程序设计语言允许用自然语言写注释。正规的程序文本中，注释行的数量占到整个源程序的1/3到1/2

注释分序言性注释和功能性注释

➢序言性注释

序言性注释通常置于每个程序模块的开头部分，它对程序的整体进行说明，对于理解程序本身起引导作用

程序名（可能为类名、函数名、方法名等等）

功能和目的

主要算法

接口说明，包括调用形式、参数描述、子程序清单

重要变量描述（名称、用途、约束条件）

调用的其他程序

开发简历（设计者时间、审查者时间、修改信息（修改原因、内容、时间）

```
/*
Title:      Function average
Purpose: To calculate the average value of
an array
Sample  Call: aver=average(score, n)
Inputs:   score – an array of float data
            n  – the number of the float data
Return: the average value of score
Functions Referenced: no
Author: M.WRIGHT 10/30/95
Auditor: D.Carrie 11/2/95
*/
float average(score, n)
int n;
float score[];
{
 int i;
float aver, sum = score[0];
for (i=1; i<n; i++)
    sum = sum + score[i];
aver = sum/n;
return(aver);
}
```

➢功能性注释

功能性注释嵌入在源程序中，用以描述其后的语句或程序段，也就是解释下面要做什么，或是执行了下面会怎么样。一般不要描述下面怎么做。（因为怎么做往往由代码描述了）

如：

/* 把月销售额计入年度总额 */（该注释描述下面的目的，对阅读程序有帮助）

```
total = amount + total;
```

/* Add amount to total*/（该注释描述怎么做，与程序语句基本重复）

```
total = amount + total;
```

书写功能性注释，要注意以下几点：

- ✓用于描述一段程序，而不是每一个语句
- ✓用缩进和空行，使程序和注释容易区别
- ✓注释要正确

程序文本排版

一个程序如果写的密密麻麻分不出层次，常常是很难看懂的。应该利用空格、空行、移行组织程序的版面。

- 空格

  恰当地利用空格，可以突出运算的优先性，避免发生运算错误。例如

  (a<-17)&&!(b<=49)||c 与 (a < -17) && !(b <= 49) || c 显然后面的容易理解

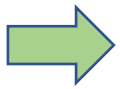
- 空行

  自然的程序段之间用空行隔开，有利于阅读

- 移行（缩格）

  用缩格来分清层次

```
if（…）
if(…)
{
……
}
else
{
……
}
else
{
……
```



```
if（…）
    if(…)
    {
        ……
    }
  else
  {
      ……
    }
else
{
  ……
}
```

3.2 数据说明标准化

为了使程序中的数据说明更易于理解和维护，在编写程序时，需要注意数据说明的风格。具体需要注意以下几点

（1）数据说明的次序应该规范化，使数据属性容易查找，也有利于测试、排错和维护。一般次序为：常量、简单变量、数组，类型排序为：整型、浮点型、双精度型、字符型、指针、逻辑变量。

（2）当一行说明多个变量时，可按出现的先后次序或按字母排序

（3）对于复杂的数据结构，应当使用注释对其进行说明

浙江大学 ZHEJIANG UNIVERSITY

3.3 语句结构简单化

- 语句结构要力求简单、直接，不能为了片面追求效率而使语句复杂难懂
- 在一行内只写一条语句，并且采用适当的移行格式，使程序的逻辑和功能变得更加明确。
- 程序编写首先应当考虑清晰性，不要刻意追求技巧性。
- 程序编写要简单清楚，直截了当地说明程序员的用意。
- 尽可能不用中间变量
- 尽可能使用库函数
- 尽可能不使用goto语句

```
a[i] = a[i] + a[t]
a[t] = a[i] - a[t]
a[i] = a[i] - a[t]
```

使用技巧，减少一个存储单元，但不清晰。

⟺

```
temp = a[i]
a[i] = a[t]
a[t] = temp
```

使用临时变量，程序清晰、易懂。

```
for(i=1;i<=n;i++)
    for(j=1;j<=n;j++)
        V[i][j]=(i%j)*(j%i)
```

程序难懂，不直截了当

⟺

```
for(i=1;i<=n;i++)
    for(j=1;j<=n;j++)
        if(i==j)
            V[i][j] = 1;
        else
            V[i][j]=0
```

直截了当说明目的

04

面向对象程序设计风格

04 面向对象程序设计风格

良好的程序设计风格对面向对象实现来说尤其重要，不仅能明显减少维护或扩充的开销，而且有助于在新项目中重用已有的程序代码。

良好的面向对象程序设计风格，既包括前面的程序设计风格准则，也包括为使用面向对象方法所特有的概念（如继承）而必须遵循的一些新的准则。

主要从两个方面考虑面向对象编程风格:

- 提高代码的可重用性
- 提高程序的可扩充性

4.1 提高代码的可重用性的准则

面向对象方法的一个主要目标就是提高软件的可重用性，软件重用有多个层次，在实现阶段主要涉及代码的重用问题。代码重用有两种:

- 本项目内的代码重用：内部重用主要是找出设计中相同或相似部分，然后利用继承机制共享它们。
- 新项目重用旧项目代码：为做到外部重用（即一个项目重用另一个项目的代码）则必须有长远眼光，需要反复精心设计。

有助于实现这两类重用的程序设计准则是相同的。

- 准则（1）提高方法的内聚

一个方法应该只完成单个功能，如果某个方法涉及两个或多个不相关的功能，则应该把它分解成几个更小的方法。

- 准则（2）减小方法的规模

如果某个方法的程序太长，则应该把它分解为几个更小的方法。

- 准则（3）保持方法的一致性

功能相似的方法应该有一致的名字、参数特征（包括参数个数、类型和次序）、返回值类型、使用条件及出错条件。

准则（4）把策略与实现分开

从方法所完成的功能看，方法可以分成两类：策略方法和实现方法。

策略方法：检查系统运行状态，根据系统运行状态做出决策，并处理出错情况，它们并不直接进行计算或实现复杂的算法。这类方法通常紧密依赖于具体应用，这类方法比较容易编写，也比较容易理解。

实现方法：负责完成具体的操作，却不做出是否执行这个操作的决定，也不知道为什么执行这个操作。仅仅针对具体数据完成特定处理，通常用于实现复杂的算法。它不做决策，执行完成后，只是返回执行状态或执行结果。相对独立于具体应用，比较容易实现重用。

为了提供代码的重用性，在编程时不要把策略和实现放在同一个方法中，应该把算法的核心部分放在一个单独的具体实现方法中。为此需要从策略方法中提取出具体参数，作为调用实现方法的参数。

准则（5）全面覆盖

如果输入条件的各种组合都可能出现，则针对所有组合写出方法，而不仅仅针对当前用到的组合情况写方法。

例如在当前项目中需要写一个方法，以获取表中的第一个元素，则至少还应该为获取表中最后一个元素再写一个方法，尽管这个方法对于本项目没有用处。

一个方法不应该只能处理正常值，对空值、边界值、界外值等异常情况也应该能过做出有意义的相应。

准则（6）尽量不使用全局信息

应该尽量降低方法与外界的耦合程度，不使用全局信息是降低耦合度的一项主要措施。

准则（7）利用继承机制

在面向对象程序中，使用继承机制是实现共享和提高重用程度的主要途径。

（1）调用子过程：最简单的继承做法为把公共的代码分离出来，构成一个被其他方法调用的公用方法。可以在基类中定义这个公用方法，供派生类中的方法调用。

```
CMyDialog：CButon
CMyDialog：：OnOK()
{
        /*添加退出前的处理*/
    CButton::OnOK();
      /* 添加退出后的处理*/
}
```

（2）分解因子 这是提高相似类代码可重用性的一个有效途径，从不同类的相似方法中分解出不同的代码（因子），把余下的代码作为公用方法中的代码，声明一个基类实现公用方法，分解出的因子作为名字相同的而算法不同的方法放在不同的子类中。让子类继承基类　　中定义的公用方法，可以明显降低为添加新子类而需付出的工作量，只需在新子类中编写其特有的代码即可。

4.2 提高代码的可扩充性的准则

准则（1）封装实现策略

应该把类的实现策略（包括描述属性的数据结构、修改属性的算法等）封装起来，对外提供公有的接口，否则将降低今后修改数据结构或算法的自由度。

准则（2）不要用一个方法遍历多条关联链

一个方法应该只包含对象模型中的有限内容，违反这条准则将导致方法过分复杂，既不易理解，也不易修改。

准则（3）避免使用多分支语句

一般可以利用DO-CASE语句测试对象对象的内部状态，而不要用来根据对象类型选择应有的行为，否则在增添新类时将不得不修改原有的代码。应该合理地利用多态型机制，根据对象当前类型，自动决定应有的行为。

准则（4）精心确定公有方法

公有方法是向外公布的接口。对这类方法的修改往往会涉及许多其他类，因此，修改公有方法的代价通常比较高。为提高可修改型，降低维护成本，必须精心选择和定义公有方法。私有方法是仅在类内使用的方法，删除、增加修改又有方法所涉及的面要窄得多。

4.3 提高代码健壮性准则

所谓健壮性就是在硬件故障、输入的数据无效或操作错误等意外环境下，系统能做出适当响应的程度。对于任何一个使用软件，健壮性是不可忽视的质量指标。

准则（1）预防用户的操作错误

软件系统必须具有处理用户操作错误的能力。当用户在输入数据是发生错误，不应引起程序运行中断，更不应该造成“死机”。任何一个接收用户输入数据的方法，对其接收到的数据都必须进行检查，即使发现了非常严重的错误，也应该给出恰当的提示信息，并准备再次接收用户的输入。

准则（2）检查参数的合法性

每个方法，都应该一开始就检查输入参数的合法性，对有不合法的输入的参数，及时做出响应。

准则（3）不要预先确定限制条件

在设计阶段，往往很难准确地预测出应用系统中使用的数据结构的最大容量需求。因此不应该预先设定限制条件。如果有必要和可能，则应该使用动态内存分配机制，创建为预设限制条件的数据结构。

THANK YOU

浙江大学机械工程学院航空制造工程研究所
王 青 李江雄