第五章 程序实现

周 瑞

QQ群：235458708

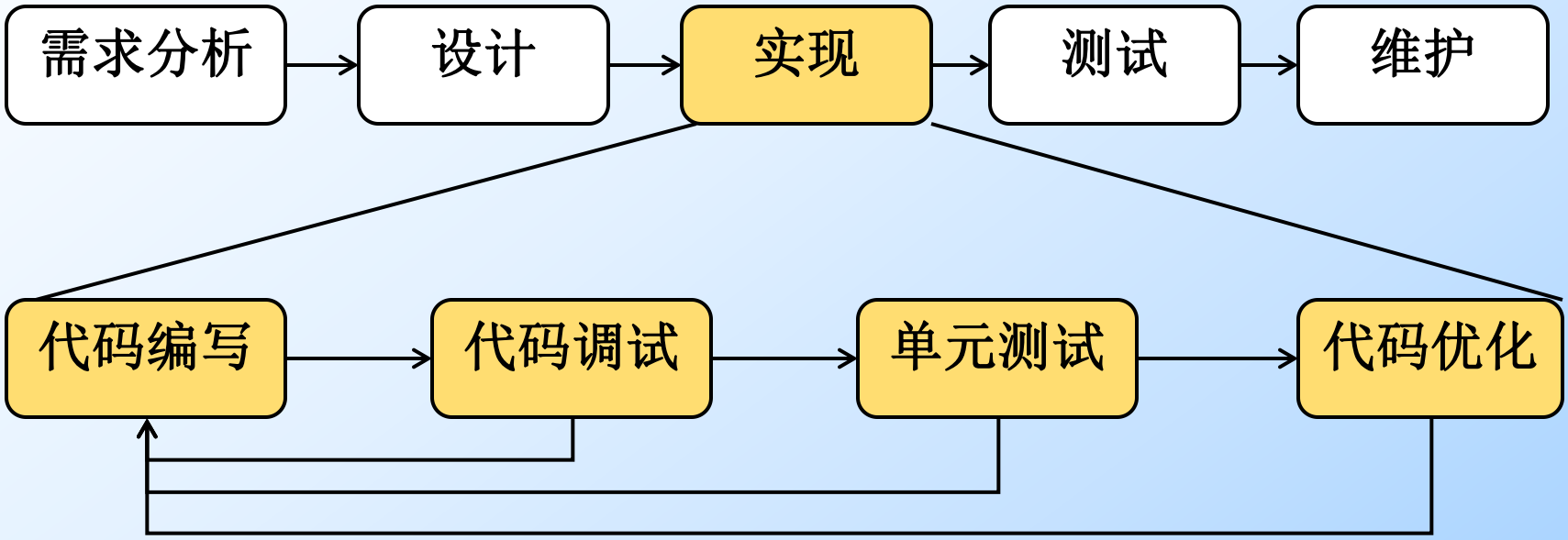
信息与软件工程学院

本章学习目标

1. 了解选择程序设计语言的一般原则
2. 理解编程规范
3. 培养良好的编程习惯

程序实现

- 将详细设计的结果翻译成用某种程序设计语言编写的并且能够执行的程序代码。



程序设计语言

- 程序设计语言是：人与计算机通信的最基本工具
- 程序设计语言的特性不可避免地会影响开发人员的思路和解决问题的方式，会影响代码的可理解性和可维护性
- 编码之前的一项重要工作就是选择一种适当的编程语言

程序设计语言的发展

- 第一代语言：机器语言
  - 机器指令代码、不同机器的机器语言不同
  - 二进制形式
  - 使用绝对地址
  - 运行效率高、但是出错率高、可读性差

- 第二代语言：汇编语言
  - 符号指令对应于机器指令
  - 存储空间由机器安排
  - 不同类型处理器具有不同汇编语言

程序设计语言的发展

- 第三代语言：高级程序设计语言
  - 从20世纪50年代开始
  - Fortran、Basic、 Pascal、 C、C++、C#、Java等

- 第四代语言：4GL
  - 查询语言和报表生成器：SQL
  - 图形语言：SQL Windows
  - 应用生成器：Power Builder
  - 形式化规格说明语言：Z、NPL、SPECINT

不同的语言适用于不同的应用

- COBOL语言：数据处理程序
- PHP语言：专门用来编写网页处理程序
- Perl语言：更适合文本处理
- C语言：被广泛用于系统软件开发
- JAVA语言：用于跨平台的应用软件开发

程序设计语言的技术特点

- 一旦确定了软件需求之后，待选用的程序设计语言的技术特性就显得非常重要
  - 如果需要复杂的数据结构，就要仔细衡量有哪些语言能提供这些复杂的数据结构描述
  - 如果软件要求高性能及实时处理能力，就该选用适合于实时处理的语言，如C或汇编语言
  - 如果应用有许多输出报告或繁杂的文件处理，则选用PowerBuilder、Delphi或SQL比较合适

程序设计语言的基本成分

- 所有程序设计语言的基本成分都可归纳为4种：
  - 数据成分：指明该语言能接受的数据，如各种类型的变量、数组、指针、记录等
  - 运算成分：指明该语言可执行的运算，如+，-，*、/
  - 控制成分：顺序结构、条件选择结构和循环结构
  - 传输成分：数据的传输方法，例如输入、输出函数

程序设计语言的分类

- 按语言级别分类，有低级语言和高级语言之分
- 按照用户要求，有过程式语言和非过程式语言之分
- 按照应用范围，有通用语言和专用语言之分
- 按照使用方式，有交互式语言和非交互式语言之分
- 按照成分性质，有顺序语言、并发语言和分布式语言之分
- 第四代语言，有数据库查询语言DEV2000、程序代码生成器以及其他一些原型语言、形式化规格说明语言等等。

选择一种语言要考虑的因素

- 在选择与评价程序设计语言时，首先要从应用要求入手，对比各项要求的相对重要性，然后再根据这些要求和相对重要性来选择合适的编程语言。
- 具体选择的考虑：
  - 编程人员的水平和编程经历
  - 待开发软件的类型
  - 算法和计算复杂性
  - 数据结构的复杂性
  - 软件的开发成本和时间要求
  - 软件的可移植性要求
  - 可用的软件工具

选择一种语言要考虑的因素

- 编程人员的水平和编程经历
  - 虽然程序员学习一门新的语言并不困难，但是要熟练地掌握和精通一门语言是需要长期的实践开发积累的。
  - 因此，在选择语言时一定要考虑到时间限制和程序员掌握语言的程度，尽可能选择一种程序员熟悉的语言。
- 待开发软件的类型
  - 一般分为数据库应用软件、实时控制软件、系统级软件、人工智能类软件、军用软件等等。
  - 根据软件的类型选择合适的开发语言，例如，FORTRAN/C语言适合科学计算，PowerBuilder、Delphi、C#等语言适合于信息系统的开发，LISP、PROLOG语言适合于人工智能领域。

选择一种语言要考虑的因素

- 算法和计算复杂性
  - 待开发软件算法的复杂性不同，应该选择合适的语言
  - 例如，科学计算领域以前大都选择FORTRAN，因为它的运行性能比较好，但是当今计算机硬件的发展使得运算速度已不再成为瓶颈，因此许多计算型软件普遍采用C/C＋＋语言。
  - 计算复杂度很高的软件采用汇编语言、人工智能类的语言肯定不合适，前者编写代码的工作复杂度太高，后者的运行效率太低，并且这两类语言的科学计算库都很少，可复用的软件元素较少。

选择一种语言要考虑的因素

- 数据结构的复杂性
  - 有些语言，例如FORTRAN、BASIC语言，定义数据类型的能力非常差，一旦设计中有比较复杂的数据结构，程序员实现时会感到很棘手。而PASCAL、C++、JAVA之类的语言其数据结构描述能力非常强大，为程序员创造了一个很广阔的编程空间。
- 考虑软件的开发成本和时间要求
  - 不仅要考虑当前的开发成本，还要考虑今后的维护成本，如果选择的语言很生僻，即使现在以很快的速度开发出来，将来的维护工作量不得不考虑。

选择一种语言要考虑的因素

- 软件的可移植性要求
  - 如果目标系统的运行环境不能确定，例如，可能运行在小型机的UNIX操作系统上，也可能运行在大型机的OS/400操作系统上，甚至还要运行在PC机的Windows操作系统环境中，这时选择的开发语言最好是JAVA。这样可以保证软件的跨平台运行。
- 可用的软件工具
  - 选择语言时，特别是为大型软件选择语言时，一定要考虑可用的软件工具。如果某种语言有支持开发的工具，则开发和调试都会容易。

良好的编程习惯

- 程序员编写的代码除了交给计算机运行外，还必须让其他程序员或设计人员能够看懂。
- 如果程序代码的可读性好，则调试和维护的成本就可以大幅度降低，同时可以减小程序运行期间软件失效的可能性，提高程序的可靠性。
- 对于代码编写而言，要求程序具有良好的结构和风格。

如果我不是程序员，我能读懂吗？

程序设计规范

1）基本要求
2）可读性要求
3）正确性与容错性要求
4）可移植性要求
5）输入和输出要求
6）重用性要求

程序设计规范：基本要求

- 程序结构清晰且简单易懂，单个函数的行数一般不要超过 100行
- 算法设计应该简单，代码要精简，避免出现垃圾程序
- 尽量使用标准库函数（类方法）和公共函数（类方法）
- 最好使用括号以避免二义性

程序设计规范：可读性要求

1. 源程序文档化
   - 标识符的命名：
     - 文件、模块、变量、常量、函数等的命名
     - 反映它所代表的东西，见名知意，英文单词或其组合（每个单词首字母大写或用下划线连接）
     - 精炼和意义明确，最小长度下的最大信息
     - 例子：
       - name, address
       - number_of_reservations, postal_address
       - getName, setName
     - 不好的命名：a, iii, postaladdress

程序设计规范：可读性要求

- 标识符的命名：
  - 给名字加注释
  - 一个变量一种用途
  - 避免局部变量和全局变量同名
  - 变量声明放在代码块的开头
  - 尽量在变量声明时初始化

程序设计规范：可读性要求

- 程序注释
  - 序言性注释：标题、版本、功能、算法、接口、编程者、日期、版权信息等
  - 功能性注释：程序体内
  - 一般用自然语言
  - 注释不宜过多，也不宜过少
  - 不要注释含义已经很清楚的代码

    ```
    Total = amount + total; //add amount to total
    ```

    这个注释好吗？
  - 保持注释和代码一致，修改代码时同时修改注释
  - 注释应当准确无二义性

程序设计规范：可读性要求

- 代码版式
  - 空格
    - 关键字和括号之间
    - 参数列表中逗号之后
    - 二元运算符之间
    - 语句中的表达式之间
  - 例子

    (a< -17) ANDNOT (B < =49) ORC

    => (a < -17) AND NOT (B <= 49) OR C

程序设计规范：可读性要求

- 代码版式
  - 空行
    - 源代码的各个节之间
    - 函数、类、接口的定义之间
    - 变量和语句之间
    - 语句的功能块之间
  - 对齐和缩进
    - 采用阶梯状缩进，以显示嵌套结构的层次
    - 代码行不宜过长，可分行

程序设计规范举例：

规范这段代码：

```
#include<stdio.h>
int main()
{
    int a[20],s,i,j;
    printf("Please input 10 numbers:\n");
    for(i=1;i<=10;i++) scanf("%d",&a[i]);
    for(i=1;i<=9;i++)
        for(j=i+1;j<=10;j++)
            if(a[i]<a[j]) {s=a[i];a[i]=a[j];a[j]=s;}
    printf("After sort:\n");
    for(i=1;i<=10;i++)
        printf("%d ",a[i]);
        return 0;
}
```

程序设计规范举例：

```
/*
The program …
*/

#include <stdio.h>

/*
The function…
*/
int main()
{
    int array[20], temp, i, j;

    printf( “Input 10 numbers:\n”);
    for(i=1;i<=10;i++)
        scanf("%d",&array[i]);
    for(i=1;i<=9;i++)
        for(j=i+1;j<=10;j++)
            if(array[i]<array[j])
            {
                temp=array[i];
                array[i]=array[j];
                array[j]=temp;
            }

    printf("After sort:\n");
    for(i=1;i<=10;i++)
        printf("%d ",array[i]);

    return 0;
}
```

程序设计规范：可读性要求

2. 数据说明标准化

- 数据说明的次序规范化，使数据容易查找，如：
  - ① 常量说明
  - ② 简单变量说明
  - ③ 数组说明
  - ④ 公用数据块说明
  - ⑤ 文件说明
- 当多个变量在一条语句中声明时，可按字母排序
- 对于复杂数据结构，使用注释

程序设计规范：可读性要求

3. 语句结构简单化

- 程序编写简单、清楚（换位思考）
- 清晰第一、效率第二（除非有特殊要求）
- 尽量采用顺序、选择和循环三种基本结构
- 避免大量的循环嵌套和条件嵌套
- 使用库函数
- 避免使用复杂的条件测试

程序设计规范：可读性要求

3. 语句结构简单化

- 一行内一条语句

```
if(a[i]<a[j]) {s=a[i];a[i]=a[j];a[j]=s;}
                =>
 if(a[i]<a[j])
 {
     s=a[i];
     a[i]=a[j];
     a[j]=s;
 }
```

- 使用括号清晰表达逻辑表达式和算术表达式

```
year%4==0&&year%100!=0||year%400==0
=>  (year%4==0)&&(year%100!=0)||(year%400==0)
```

程序设计规范：正确性与容错性要求

- 程序首先是正确的，其次考虑优美和效率。
- 对所有的用户输入，必须进行合法性和有效性检查。
- 不要单独进行浮点数的比较。在计算机中用二进制表示十进制数时，有时二进制数不能准确地表达十进制数，这时浮点数的表示具有不准确性。用它们做比较，其结果常常发生异常情况。解决办法是在严格的容差级范围内检验两个值的差异，其形式为：$|x0 - x1| < \varepsilon$ 其中 ε 是容差级，其大小取决于具体应用中的总体精度要求及所用数值的精度。

程序设计规范：正确性与容错性要求

- 所有变量在调用前必须被初始化。
- 改一个错误时可能产生新的错误，因此在修改前首先考虑对其它程序的影响。
- 单元测试也是编程的一部分，提交联调测试的程序必须通过单元测试。
- 单元测试时，必须针对类里的每一个方法进行测试：测试其正确的输入，是否得到正确的输出；错误的输入是否得到相应的容错处理（如异常捕捉处理，返回错误提示等）。

程序设计规范：可移植性要求

- 应当尽量使用语言的标准部分，避免使用第三方提供的接口，以确保程序不受具体的运行环境影响，和平台无关。
- 对数据库的操作，使用符合语言规范的标准接口类例如JDBC，除非程序是运行于特定的环境下，并且有很高的性能优化方面的要求。
- 程序中涉及到的数据库定义和操纵语句，尽量使用标准SQL数据类型和SQL语句

程序设计规范：输入和输出要求

- 对所有输入数据进行检验，保证输入数据的有效性
- 检查输入项的各种重要组合的合理性
- 输入格式和步骤尽量简单
- 尽量允许自由格式输入
- 允许默认值
- 以交互方式进行输入时，要在屏幕上使用提示符以及输入提示，并指明输入项的种类和取值范围，同时在输入过程中和输入结束时，在屏幕上给出状态信息
- 输出数据尽量表格化和图形化
- …

程序设计规范：重用性要求

- 可重复使用的、功能相对独立的算法或接口，应该考虑封装成公共的控件或类，如时间、日期处理，字符串格式处理，数据库连接，文件读写等，以提高系统中程序的可复用性。
- 相对固定和独立的程序实现方式和过程，应考虑做成程序模板，增强对程序实现方式的复用，如对符合一定规范的XML数据的解析等过程。

本章复习要点

- 选择程序设计语言的一般原则
- 程序设计规范