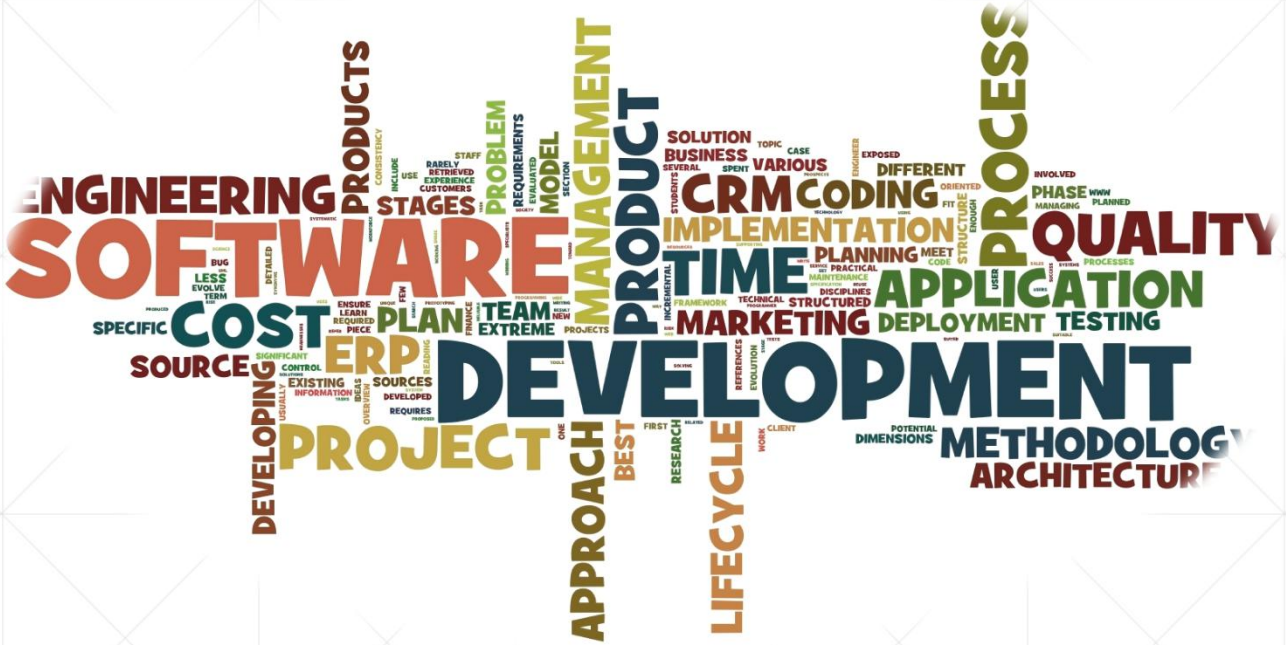
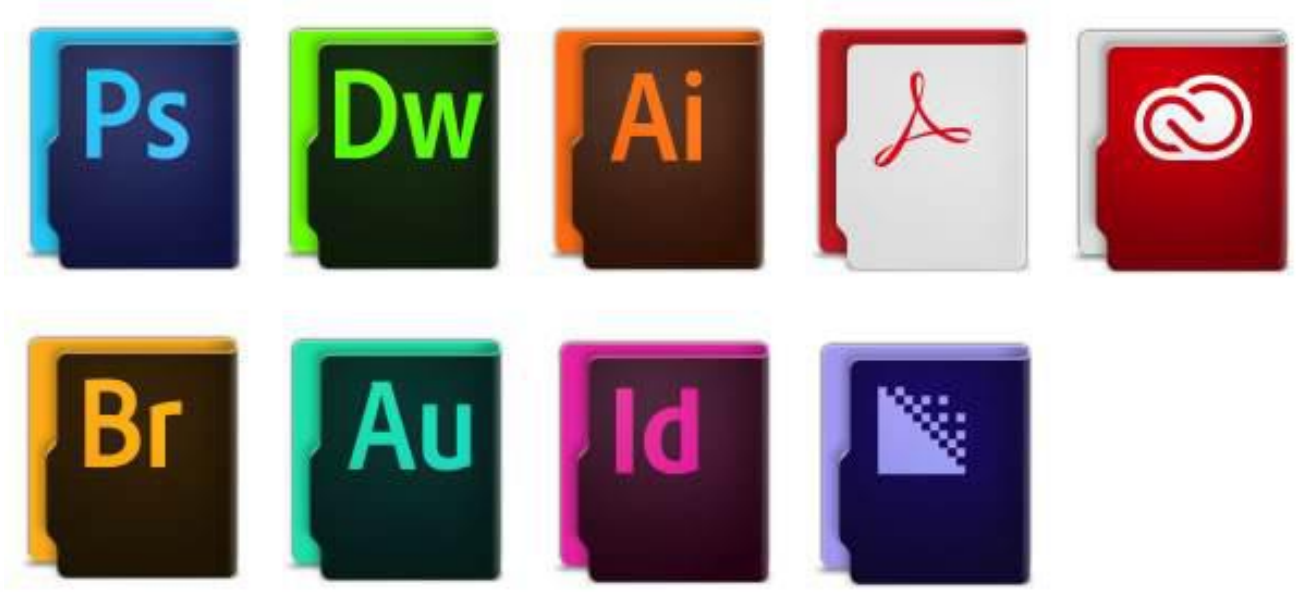


编程习惯

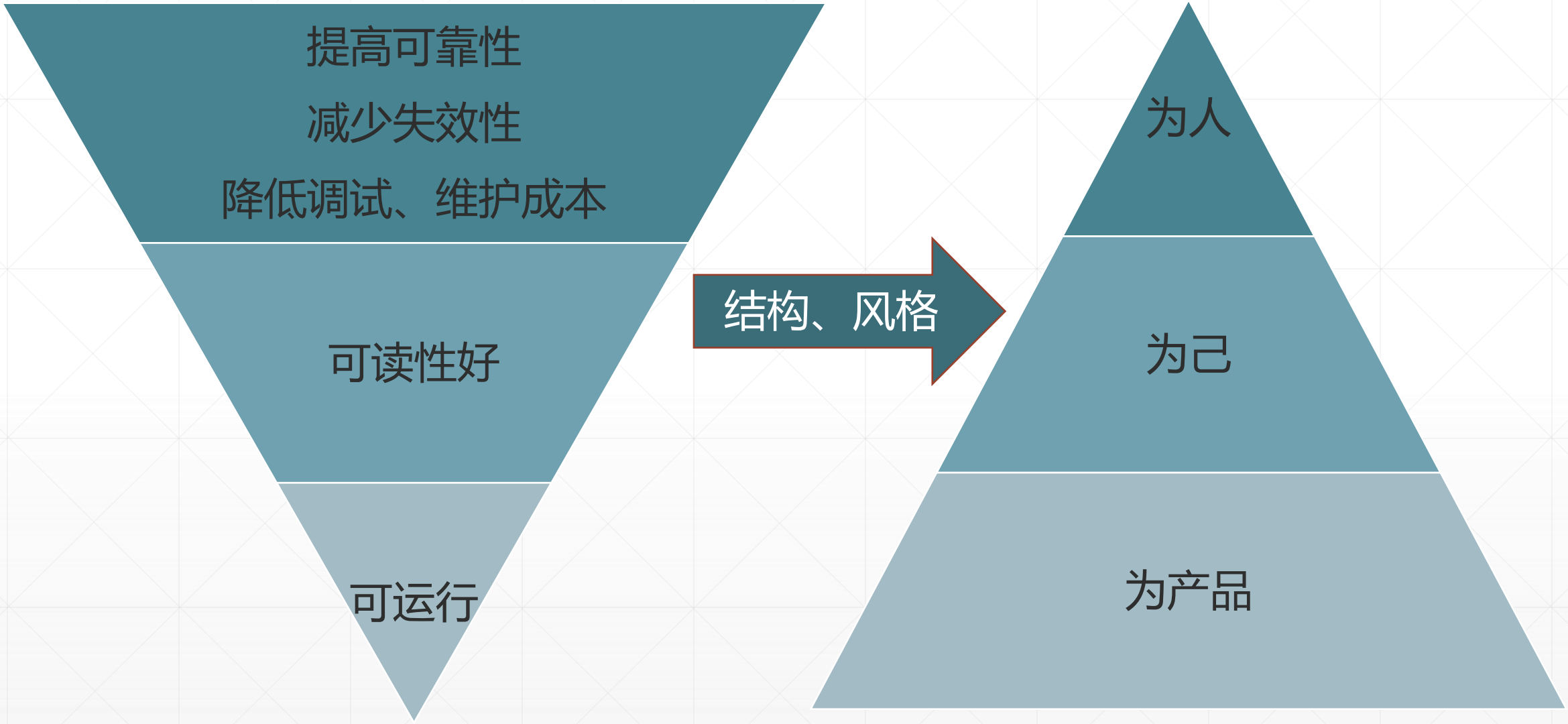
授课教师：吴祖峰　　电子邮箱：wuzufeng@uestc.edu.cn



1. 编程习惯

- 重要性
- 结构化程序设计原则
- 程序设计风格

重要性



结构化程序设计原则

简单结构

- 尽量使用语言提供的基本控制结构，即顺序结构、选择结构和重复（循环）结构。

组合嵌套

- 复杂结构应该用基本控制结构组合或嵌套实现。

一致性

- 语言中没有的控制结构，可用一段等价的程序段模拟，但要求该程序段在整个系统中应前后一致。

块机制

- 将程序组织成容易识别的块，每块只有一个入口和一个出口。

GOTO语句

- 严格控制GOTO语句。

结构化程序设计原则

自顶向下	先考虑总体，后考虑细节；先考虑全局目标，后考虑局部目标
逐步细化	对于复杂问题，进行分解为子目标。
模块化	再将子目标分解为小目标，予以结构化实现。

自顶向下、逐步细化

[例] 用筛选法求100以内的素数。

[细化] ：具体做法就是从2到100中去掉2，3，…，9，10的倍数，剩下的就是100以内的素数。

```
main( )
    {
      建立2到100的数组A[ ]，其中A[i]= i；                    ------1
      建立2到10的素数表B[ ]，存放2到10以内素数；      ------2
     若A[i] = i是B[ ]中任意一个数的倍数，则剔除A[i]；  ------3
     输出A[ ]中所有没有被剔除的数；                          ------4
    }
```

程序设计风格

程序设计风格

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

程序设计风格—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。

程序设计风格—可读性要求

注　释

- 程序头,函数头说明；接口说明；子程序清单,有关数据的说明；模块位置；开发历史等
- 主要变量(结构、联合、类或对象):含义的注释。
- 应保持注释与代码完全一致。
- 处理过程的每个阶段和典型算法前都有相关注释说明，但是不要对每条语句注释。

程序设计风格—可读性要求

格 式

- **程序格式清晰：**
- **一行只写一条语句，不要密密麻麻，分不出层次**
- **显示程序的逻辑结构，**利用空格、空行和缩进进行，**缩进量一般为4个字节。**

```
if (A<-17)AND NOT(B<=49)OR Cthen A=A-1
if A>100 thenA=A*2 endif elseA=A+1
endif
```

写成

```
if  (A < -17)  AND  NOT (B <= 49) OR C
     then A=A-1
          if A>100
             then A=A*2
          endif
     else A=A+1
end if
```

程序设计风格—可读性要求

程序本身

语句力求简单、清晰，不要片面追求效率,程序编写得过于紧凑,使语句复杂化。

例如：

V是一个N×N单位矩阵,

当I≠J时,V(I,J)=0;

当I=J时,V(I,J)=1。

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

写成

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

程序设计风格—可读性要求

程序本身-续

- 简单变量的运算速度比下标（数组）变量的运算要快,程序员可能把语句：

  X=A[I]+1/A[I]

  写成

  AI= A[I]; X=AI+1/AI

编程时尽可能使用已有的库函数。

尽量用公共过程或子程序代替具有独立功能的重复代码段 。

使用括号清晰地表达算术表达式和逻辑表达式的运算顺序。

尽量使用三种基本控制结构编写程序,使用IF THEN ELSE结构实现分支;使用DO UNTIL或DO WHILE来实现循环。

避免采用过于复杂的条件测试，少用含有“否定”运算符的条件语句，例如:

IF NOT ((CHAR< =’0’) OR (CHAR> =’9’)) THEN ……

改成

IF (CHAR>’0’) AND (CHAR<’9’) THEN ……

程序设计风格—可读性要求

程序本身-续

- 避免使用空的ELSE语句和IF THEN IF语句

```
IF(CHAR>=’A’) THEN
IF(CHAR<=’Z’) THEN
PRINT “This is a letter。”
ELSE//这个语句的配套IF逻辑上不明确
PRINT "This is not a letter。”
```

程序设计风格—可读性要求

程序本身-续

- 避免使用ELSE GOTO和ELSE RETURN结构。
- 避免过多的循环嵌套和条件嵌套。
- 数据结构要有利于程序的简化。
- 模块功能尽可能单一化，模块间的耦合能够清晰可见。利用信息隐蔽确保每一个模块的独立性。
- 对递归定义的数据结构尽量使用递归过程。
- 尽量不要修补结构差的程序,而应重新设计和编码。
- 对太大的程序，要分块编写、测试，后再集成

程序设计风格—可读性要求

数据说明

数据说明的先后次序规范化

简单变量类型说明、数组说明、公用数据块说明、文件说明

每个类型说明中可按如下顺序排列

整型量说明、实型量说明、字符量说明、逻辑量说明

同一条说明语句中可按字母顺序排列

例如：INTEGER cost, length, price, width

程序设计风格—正确性与容错性要求

程序首先是正确，其次是考虑优美和效率。

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。

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，因此修改前首先考虑其影响。

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public方法进行测试，测试其正确的输入，是否得到正确的输出；错误的输入是否有容错处理。

程序设计风格—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

程序设计风格—输入和输出要求

√ 任何程序都会有输入输出，输入输出的方式应当尽量方便用户的使用。在需求分析和设计阶段就应确定基本的输入输出风格，要避免因设计不当带来操作和理解的麻烦。

√ 对所有的输入数据进行检验，从而识别错误的输入，以保证每个数据的有效性。

√ 检查输入项各种重要组合的合理性，必要时报告输入状态信息。

√ 输入的步骤和操作尽可能简单，并且要保持简单的输入格式。

√ 有些输入信息应提供缺省值。

√ 输入一批数据时，最好使用输入结束标志，而不要由用户指定输入数据数目。

√ 在以交互式方式进行输入时，要显示提示信息、选择项和取值范围，便于操作。同时，在输入数据的过程和输入数据结束时，也要在屏幕上给出状态信息。

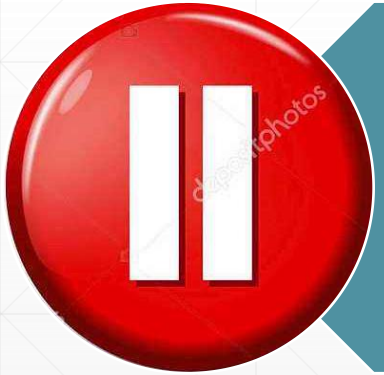
√当程序设计语言对输入格式有严格要求时，应保持输入格式与输入语句的要求一致。

√ 给所有的输出加上注解信息。

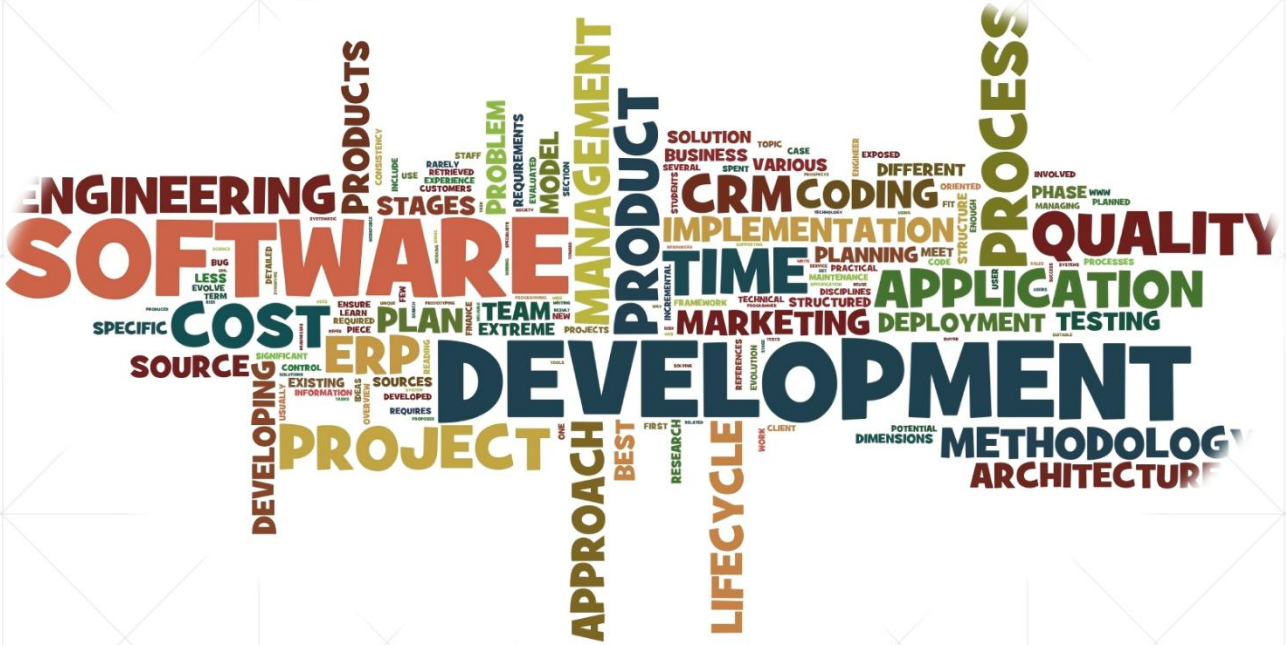
√ 按照用户的要求设计输出报表格式。

程序设计风格—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。



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



感谢观看！

授课教师：吴祖峰　　电子邮箱：wuzufeng@uestc.edu.cn