中国科学技术大学软件学院
SCHOOL OF SOFTWARE ENGINEERING OF USTC

Writing the Programs

中国科学技术大学软件学院
孟宁

猜猜它是干嘛的^-^

C/C++ code

```
#include "stdio.h"
#include "conio.h"
long a=10000, b, c=28000, d, e, f[28010], g;
void main()
{
 for( ;b-c; ) f[b++] =a/5;
 for( ; d=0, g=c*2; c-=14,printf("%.4d",e+d/a),e=d%a)
 for(b=c; d+=f[b]*a,f[b] =d%--g,d/=g--,--b; d*=b ) ;
}
```

什么样的代码是好代码?

- 规范整洁
  - 遵守常规语言规范
  - 合理使用空格、空行、缩进、注释
- 逻辑清晰
  - 没有代码冗余、重复
  - 让人清晰明了的命名规则
  - 不只是程序员的编程技术, 更重要的是设计技术的提高, 会让你的代码简洁清晰。
- 优雅端庄
  - 设计的艺术

Contents

Chapter 7 Objectives

- Standards for programming
- Guidelines for reuse
- Using design to frame the code
- Internal and external documentation

7.1 Programming Standards and Procedures

- Standards for you
  - methods of code documentation
- Standards for others
  - Integrators集成商, maintainers, testers
  - Prologue序幕\文件头部 documentation
  - Automated tools to identify dependencies
- Matching design with implementation
  - Low coupling, high cohesion, well-defined interfaces

文件头部的注释

```
/*
 *  linux/init/main.c
 *
 *  Copyright (C) 1991, 1992  Linus Torvalds
 *
 *  GK 2/5/95  -  Changed to support mounting root fs via NFS
 *  Added initrd & change_root: Werner Almesberger & Hans Lermen, Feb '96
 *  Moan early if gcc is old, avoiding bogus kernels - Paul Gortmaker, May '96
 *  Simplified starting of init:  Michael A. Griffith <grif@acm.org>
 */
```

```
/*
 * kernel/ksysfs.c - sysfs attributes in /sys/kernel, which
 *           are not related to any other subsystem
 *
 * Copyright (C) 2004 Kay Sievers <kay.sievers@vrfy.org>
 *
 * This file is release under the GPLv2
 *
 */
```

```
/*****************************************************************
 * Copyright (C) Future Sotware, 2002.
 *
 * $Id: la.h,v 1.1.1.2 2005/03/14 02:48:48 cvsroot Exp $
 *
 * Description: Contains definitions, macros and functions to be used
 *              by  external modules.
 *
 *****************************************************************/
```

```
/*
 * Copyright (C) Future Software Ltd, 1997-2002.
 *
 * $Id: osix.h,v 1.2 2005/08/10 04:27:05 cvsroot Exp $
 *
 * Description: This is the exported file for fsap 3000.
 *              It contains exported defines and FSAP APIs.
 */
```

```
/*****************************************************************/
/* Copyright (C) SSE-USTC, 2009                                  */
/*                                                               */
/*  FILE NAME              :  socketwraper.c                     */
/*  PRINCIPAL AUTHOR       :  Mengning                           */
/*  SUBSYSTEM NAME         :  ChatSys                            */
/*  MODULE NAME            :  ChatSys                            */
/*  LANGUAGE               :  C                                  */
/*  TARGET ENVIRONMENT     :  ANY                                */
/*  DATE OF FIRST RELEASE  :  2009/9/29                          */
/*  DESCRIPTION            :  Socket wraper functions for ChatSys.*/
/*****************************************************************/

/*
 * Revision log:
 * UDP socket API replaced by TCP socket API,modified by Mengning,2009/12/2
 * Created by Mengning,2009/9/29
 * NOTE:only this file call directly socket API.
 *
 */
```

```
1
2  /*****************************************************************************/
3  /* Copyright (C) SSE-USTC(Suzhou), 2010                                      */
4  /*                                                                           */
5  /*  FILE NAME             :  mmdbdatabase.h                                  */
6  /*  PRINCIPAL AUTHOR      :  M-Mencius Group (mengning@ustc.edu.cn)          */
7  /*  SUBSYSTEM NAME        :  MMDB                                            */
8  /*  MODULE NAME           :  FastDB Abstuctation Layer                       */
9  /*  LANGUAGE              :  C++                                             */
10 /*  TARGET ENVIRONMENT    :  ANY                                             */
11 /*  DATE OF FIRST RELEASE :  2010/05/14                                      */
12 /*  DESCRIPTION           :  The exported file,MMDB.                         */
13 /*****************************************************************************/
14
15 /*
16  * Revision log:
17  *
18  * Created by Mengning,2010/05/14
19  *
20  */
```

程序块头部的注释

- 无注释
- 一句话注释
- 函数功能、各参数的含义和输入/输出用途等一一列举

```
/* Function Protypes */
/*********************************************
 * Function Name :  ParseChatSysPdu
 * Description   :  parse pInputChatSysPdu to be  pRequestMsg
 * Input(s)      :  pInputChatSysPdu
 * Output(s)     :  pRequestMsg
 * Return        :  SUCCESS/FAILURE
 *********************************************/
INT4 ParseChatSysPdu(tChatSysPdu * pInputChatSysPdu, tChatSysMsg * pRequestMsg);
```

7.2 Programming Guidelines

Control Structures

- Make the code easy to read
- Build the program from modular blocks
- Make the code not too specific, and not too general
- Use parameter names and comments to exhibit coupling among components
- Make the dependency among components visible

7.2 Programming Guidelines

Example of Control Structures

- Control skips around among the program's statements

```
                        benefit = minimum;
                        if (age < 75) goto A;
                        benefit = maximum;
                        goto C;
                        if (AGE < 65) goto B;
                        if (AGE < 55) goto C;
A:          if (AGE < 65) goto B;
                        benefit = benefit * 1.5 + bonus;
                        goto C;
B:                      if (age < 55) goto C;
                        benefit = benefit * 1.5;
C:          next statement
```

- Rearrange the code

```
if (age < 55) benefit = minimum;
elseif (AGE < 65) benefit = minimum + bonus;
elseif (AGE < 75) benefit = minimum * 1.5 + bonus;
else benefit = maximum;
```

7.2 Programming Guidelines

Algorithms

- Common objective and concern: performance (speed)
- Efficiency may have hidden costs
  - cost to write the code faster
  - cost to test the code
  - cost to understand the code
  - cost to modify the code

7.2 Programming Guidelines

Data Structures

- Several techniques that used the structure of data to organize the program
  - keeping the program simple
  - using a data structure to determine a program structure

7.2 Programming Guidelines

Keep the Program Simple

Example: Determining Federal Income Tax

1. For the first $10,000 of income, the tax is 10%
2. For the next $10,000 of income above $10,000, the tax is 12 percent
3. For the next $10,000 of income above $20,000, the tax is 15 percent
4. For the next $10,000 of income above $30,000, the tax is 18 percent
5. For any income above $40,000, the tax is 20 percent

```
tax = 0.
if (taxable_income == 0) goto EXIT;
if (taxable_income > 10000) tax = tax + 1000;
else{
                tax = tax + .10*taxable_income;
                goto EXIT;
}
if (taxable_income > 20000) tax = tax + 1200;
else{
                tax = tax + .12*(taxable_income-10000):
                goto EXIT;
}
if (taxable_income > 30000) tax = tax + 1500;
else{
                tax = tax + .15*(taxable_income-20000);
                goto EXIT;
}
if (taxable_income < 40000){
                tax = tax + .18*(taxable_income-30000);
                goto EXIT;
}
else
                tax = tax + 1800. + .20*(taxable_income-40000);
EXIT;
```

7.2 Programming Guidelines

Keep the Program Simple Example (continued)

- Define a tax table for each “bracket” of tax liability

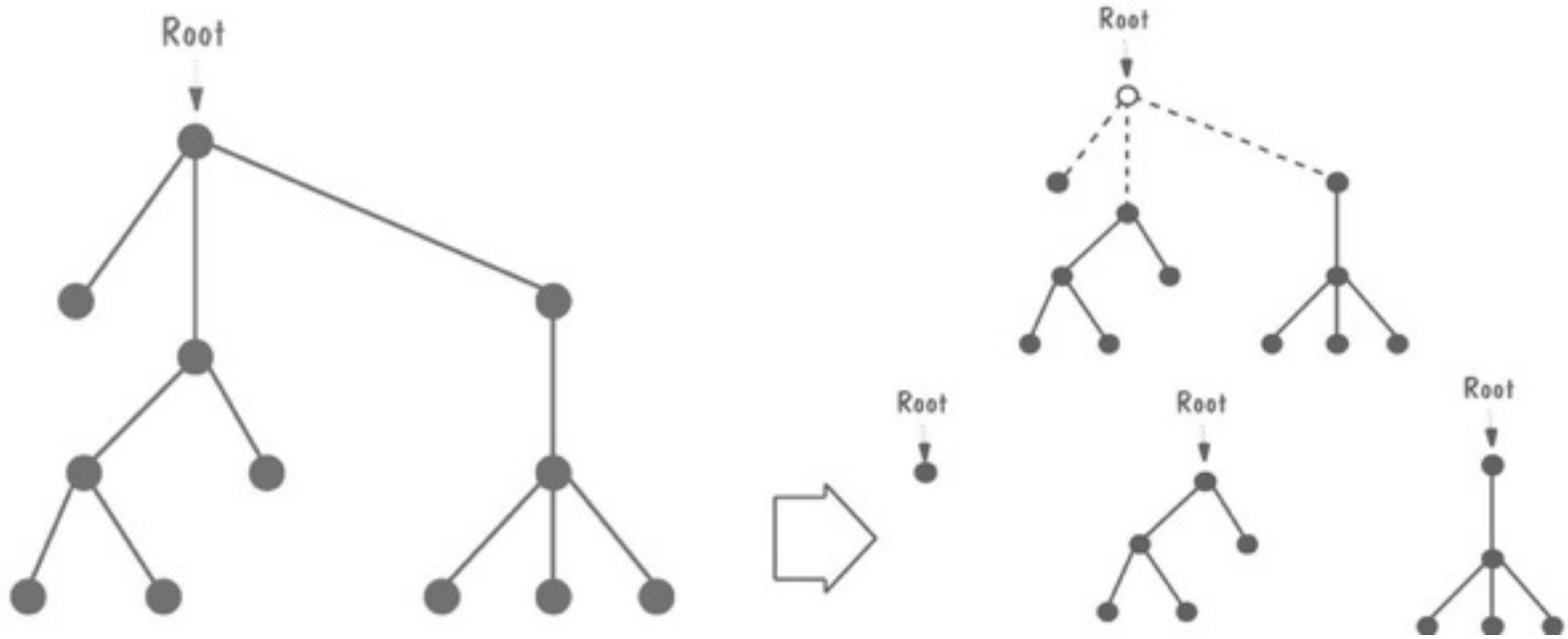
Bracket	Base	Percent
0	0	10
10,000	1000	12
20,000	2200	15
30,000	3700	18
40,000	55000	20

- Simplified algorithm

```
for (int i=2,level=1; i <= 5; i++)
                if (taxable_icome > bracket[i])
                        level = level + 1;
tax= base[level]+percent[level] * (taxable_income - bracket[level]);
```

7.2 Programming Guidelines
Data Structures Example: Rooted Tree

- Recursive data structure
- Graph composed of nodes and lines
  - Exactly one node as root
  - If the lines emanating from the root are erased, the resulting graph is a rooted tree

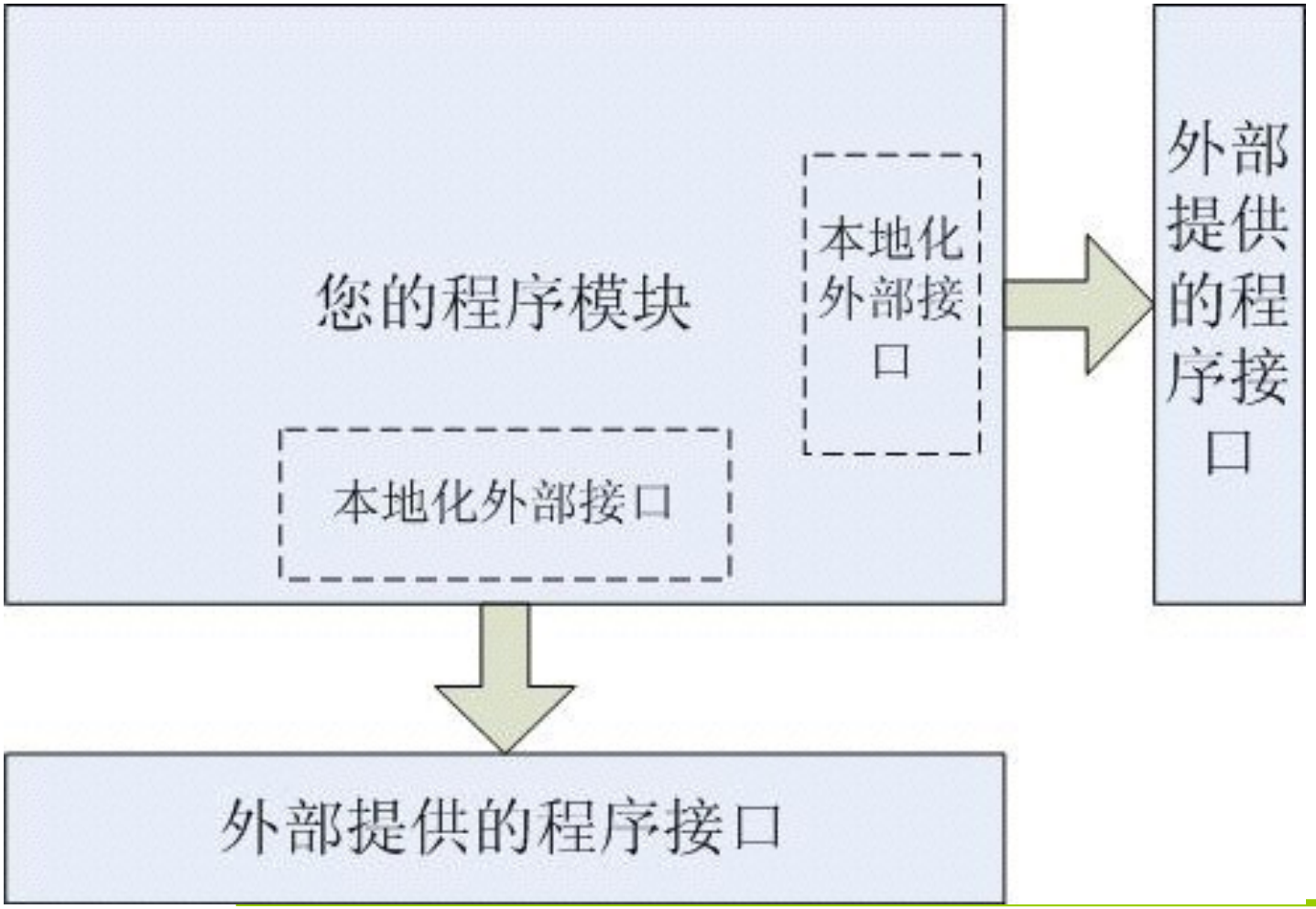


7.2 Programming Guidelines

General Guidelines to Preserve Quality

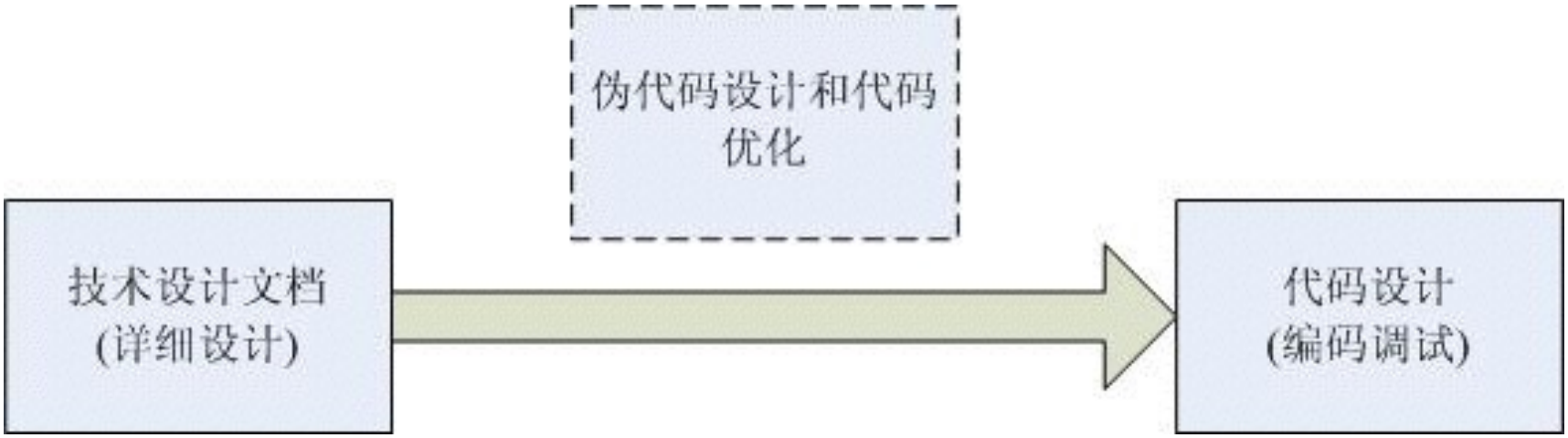
- Localize input and output
- Employ pseudocode
- Revise and rewrite, rather than patch
- Reuse
  - Producer reuse: create components designed to be reused in future applications
  - Consumer reuse: reuse components initially developed for other projects

Localize input and output



在组件中本地化这部分以和其余的代码相分离是适合的。

Include pseudocode



设计通常为每一个程序组件提供一个框架。然后，你用自己的专业知识和创造性来编写代码以实现设计。

在从设计到编码中加入伪代码要好于直接将设计翻译成代码。

Revise and rewrite, rather than patch

- 如果您觉得控制流程盘根错节、判定过程难以理解、或者无条件的分支难以消除，那么就该重新返回到设计了。
- 重新检查设计，搞清楚您遇到的问题是设计中的固有问题，还是设计转化为代码的问题。

7.2 Programming Guidelines

Consumer Reuse

- Four key characteristics to check about components to reuse
  - does the component perform the function or provide the data needed?
  - is it less modification than building the component from scratch?
  - is the component well-documented?
  - is there a complete record of the component's test and revision history?

7.2 Programming Guidelines

Producer Reuse

- Several issues to keep in mind
  - make the components general
  - separate dependencies (to isolate sections likely to change)
  - keep the component interface general and well-defined
  - include information about any faults found and fixed
  - use clear naming conventions
  - document data structures and algorithms
  - keep the communication and error-handling sections separate and easy to modify

7.3 Documentation

- Internal documentation
  - header comment block
  - meaningful variable names and statement labels
  - other program comments
  - format to enhance understanding
  - document data (data dictionary)
- External documentation
  - describe the problem
  - describe the algorithm
  - describe the data

7.3 Documentation

Information Included in Header Comment Block

- What is the component called
- Who wrote the component
- Where the component fits in the general system design
- When the component was written and revised
- Why the component exists
- How the component uses its data structures, algorithms, and control

7.4 The Programming Process

Programming as Problem-Solving

- Polya's (1957) four distinct stages of finding a good solution
  - understanding the problem
  - devising plan
  - carrying out the plan
  - looking back

7.4 The Programming Process
Extreme Programming

- Two types of participants
  - *customers*: who define the features using stories, describe detailed tests and assign priorities
  - *programmers*: who implement the stories

7.4 The Programming Process
Pair Programming

- The driver or pilot: controlling the computer and writing the code
- The navigator: reviewing the driver's code and providing feedback

7.4 The Programming Process

- Documentation is still essential in agile-methods
  - Assist the developers in planning, as a roadmap
  - Helps describe key abstractions and defines system boundaries
  - Assists in communicating among team members

What This Chapter Means for You

- Things to consider when writing a code
  - organizational standards and guidelines
  - reusing code from other projects
  - writing code to make it reusable on future projects
  - using the low-level design as an initial framework, and moving in several iterations from design to code

作业

- 分析一套源代码的代码规范和风格并讨论如何改进优化代码
  - 结合工程实践选题相关的一套源代码，根据其编程语言或项目特点，分析其在源代码目录结构、文件名/类名/函数名/变量名等命名、接口定义规范和单元测试组织形式等方面的做法和特点；
  - 列举哪些做法符合代码规范和风格一般要求；
  - 列举哪些做法有悖于“代码的简洁、清晰、无歧义”的基本原则，及如何进一步优化改进；
  - 总结同类编程语言或项目在代码规范和风格的一般要求。

中国科学技术大学软件学院
SCHOOL OF SOFTWARE ENGINEERING OF USTC

谢谢大家！

References

软件工程 - 理论与实践（第四版 影印版） **Software Engineering: Theory and Practice (Fourth Edition),Shari Lawrence Pfleeger,Joanne M. Atlee ,**高等教育出版社

软件工程 - 理论与实践（第四版） **Software Engineering: Theory and Practice (Fourth Edition),Shari Lawrence Pfleeger,Joanne M. Atlee,**杨卫东译, 人民邮电出版社

软件工程—实践者的研究方法（**Software Engineering-A Practitioner's Approach**）; (美) **Roger S. Pressman** 著； 机械工业出版社**ISBN**：**7-111-07282-0**

http://code.google.com/p/advancedsoftwareengineering/