Software Architecture
软件体系结构

Lecture 3. 主程序–子程序系统与面向对象系统
Main Program and Subroutine &
Object Oriented Systems

Professor:

Yushan (Michael) Sun

Fall 2020

Content of This Lecture

1. Evolution of Program Design
2. Call and return Architecture
3. Main program and subroutine
4. Object-oriented Design
5. Case studies

Evolution of Program Design

1. Evolution of Program Design

Year period		Examples
1950s-1960s	simple program models 无结构化编程	COBOL (1959), Algol (1958), Basic(1964)
1970s - 1980s	structured programming 结构化设计 (Top-Down Design)	Pascal(1968), C(1972), Fortran(1957 published), Perl
	First object-oriented languages (esp C++) OO Design 面向对象设计	Ada, SmallTalk C++(1983),

1. Evolution of Program Design

- Business programming language COBOL
- In 1959, COBOL, for business use.
  - 非结构化编程语言代表 61 years old
- 1972, C was developed, at Bell Labs in New Jersey.
  - 结构化程序设计与编程语言代表 48 years old
- In 1983, C++ was released. 37 years old
  - 面向对象语言，面向对象编程
  - Note: Java was published in1995

25 years old

Unstructured Programming
非结构化编程

(1950’s-1960’s)

2. Unstructured programming

非结构化编程

- 所有的代码都在同一个程序文件中
- 每行包含一条语句，有行标识；允许程序从一行跳到另外一行 (use GOTO)
- 非结构化编程使用：迭代，分支，GOTO (iteration, branch and GOTO)
- 典型的非结构化语言: early BASIC, COBOL

2. Unstructured programming

- 非结构化编程的缺点：
- Disadvantages of Unstructured programming:

a) It is difficult to follow program logic.

b) It is hard to incorporate other code;

c) It is not easy to modify code (都是goto惹的祸)

d) It is difficult to test particular portions of the code

Structured Programming
结构化编程

(1965年提出概念，盛行于1970年代)

3. Structured programming

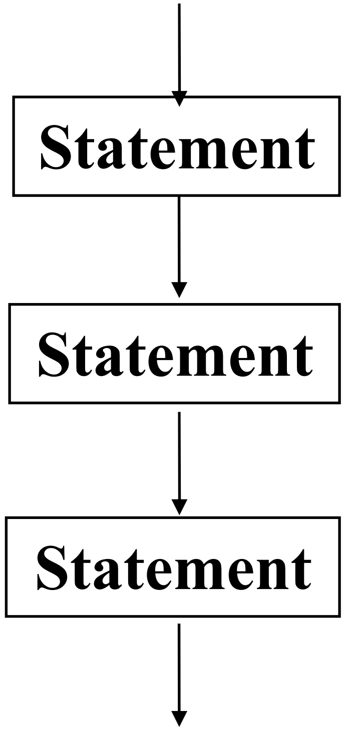
- 结构化编程是一个编程模式，其目标是改善计算机程序的清晰度、质量与节省开发时间，通过大量地使用
- **Structured programming** is a programming paradigm aimed on improving the clarity, quality, and development time of a computer program by making extensive use of
  - 子程序：subroutines,
  - 块状结构：block s tructures and
  - 循环结构：for and while loops.

- 结构化编程抛弃了GOTO语句。
- Language that support structured programming: started from ALGOL68, C

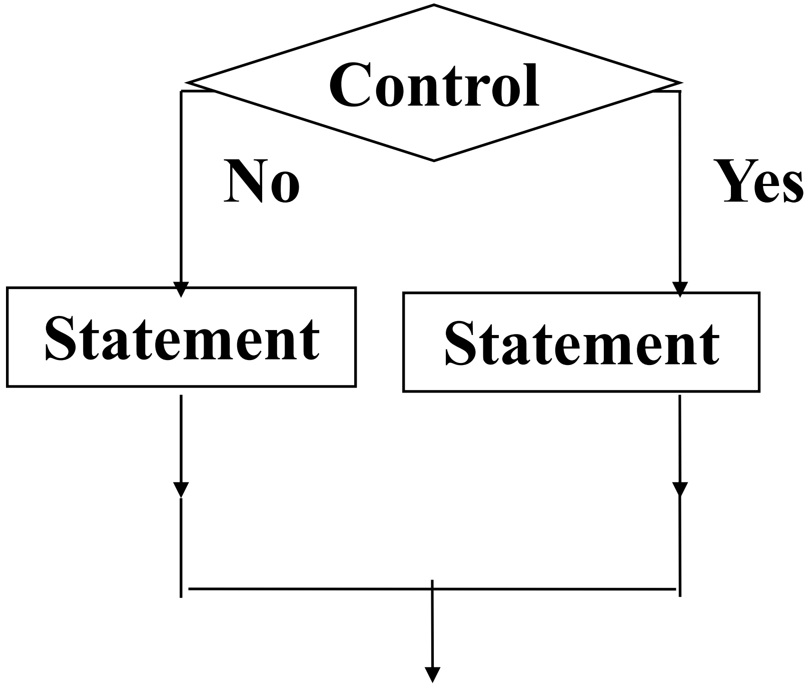
3. Structured programming

- **结构化程序定理**：只要一种编程语言可以依三个方式组合其子程序及调整控制流程，则每个可计算函数都可以用此种编程语言来表示（实现任何算法）。三个调整控制流程的方式为：
  - ◆**序列结构**：按照顺序执行一个子程序，然后再执行另一个子程序
  - ◆**选择结构**：根据布尔表达式的值，执行两个子程序中的一个子程序（if, else if, else）；
  - ◆**迭代结构**(for, while)：执行子程序，直到一个布尔表达式为true

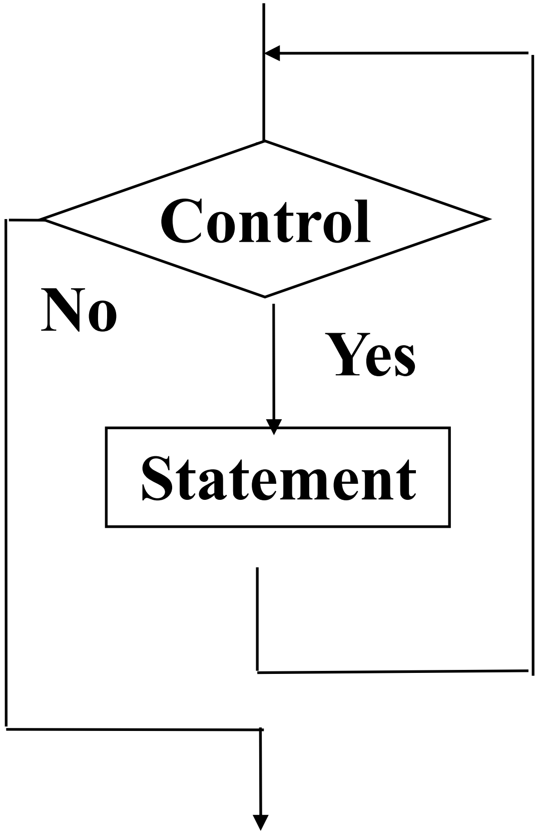
3. Structured programming


Statement
Statement
Statement

序列结构


Control
No
Yes
Statement
Statement

选择结构


Control
No
Yes
Statement

循环结构

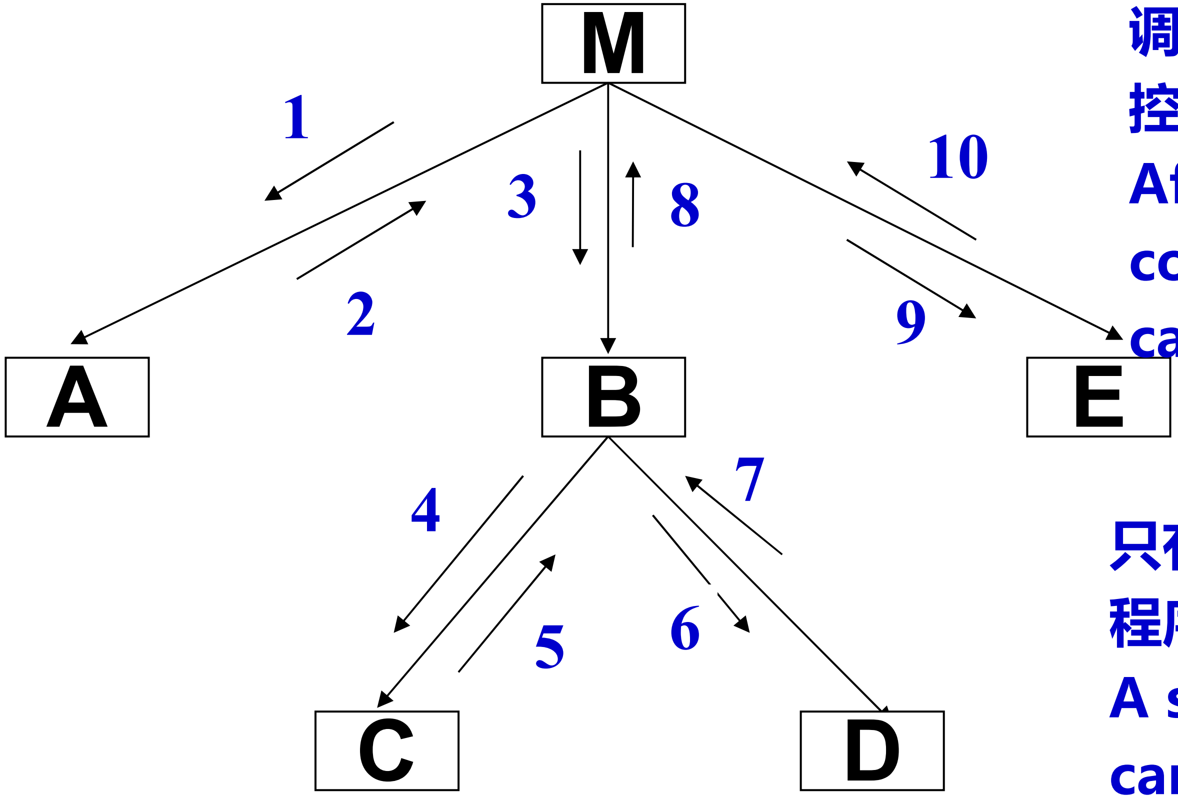
Back

Call and Return Architecture
调用-返回架构

4. Call and Return Architecture

- **调用-返回架构使用分割-征服策略 A system designed in call and return architecture uses divide and conquer strategy.**
- **将整个系统分为更小的子系统，以减少复杂度。Its main idea is to divide the whole system into some smaller blocks to reduce the complexity and enhance the modifiability**
- **调用-返回的执行顺序被一个单线程控制 The execution order of the Call and return system is usually controlled by a single thread**

4. Call and Return Architecture



调用子程序之后，程序执行控制权被返回给调用者

After calling a subroutine, control is returned to the calling sunroutine

只有获得控制权，一个软件组件程序才能被执行

A software component can be executed only if it gets the control

The control flow of call and return architecture

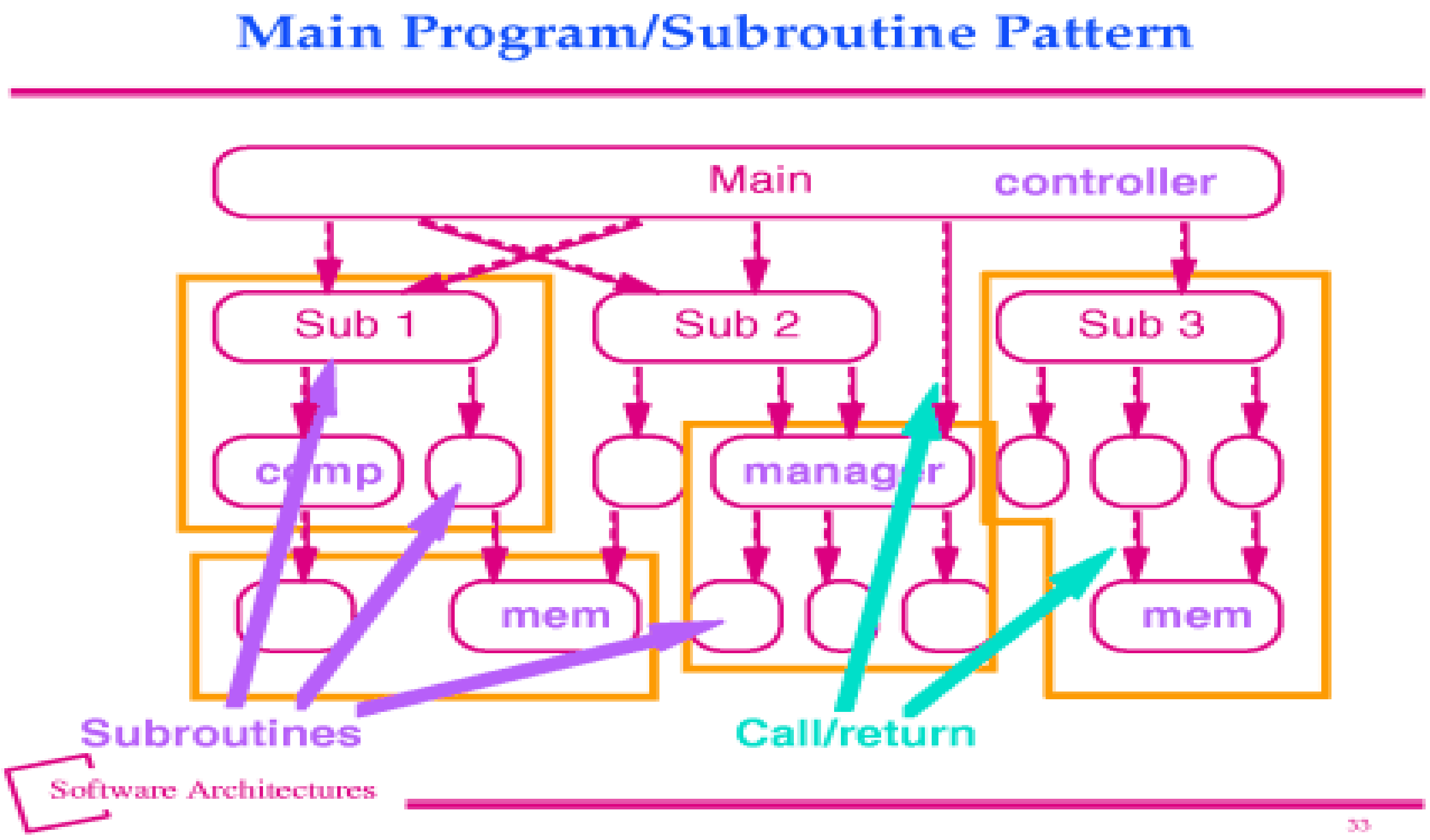
Back

Main Program and Subroutine

是一种Call and return 架构

代表结构化编程

5. Main Program and Subroutine



Main Program/Subroutine Pattern

Software Architectures

33

The program is composed of a main program and some subroutines

5. Main Program and Subroutine

- **架构：系统由主程序与形成层次状的子程序组成**
- **Architecture**: the system is composed of a main program and some hierarchically organized subroutines
  - **主程序调用高层子程序** Main program call higher level subroutines
  - **高层子程序调用较低层子程序**Higher level subroutines call lower level subroutines
  - **一个子程序的正确性依赖于其所调用的子程序。** The correctness of a subroutine often depends on the subroutine it calls

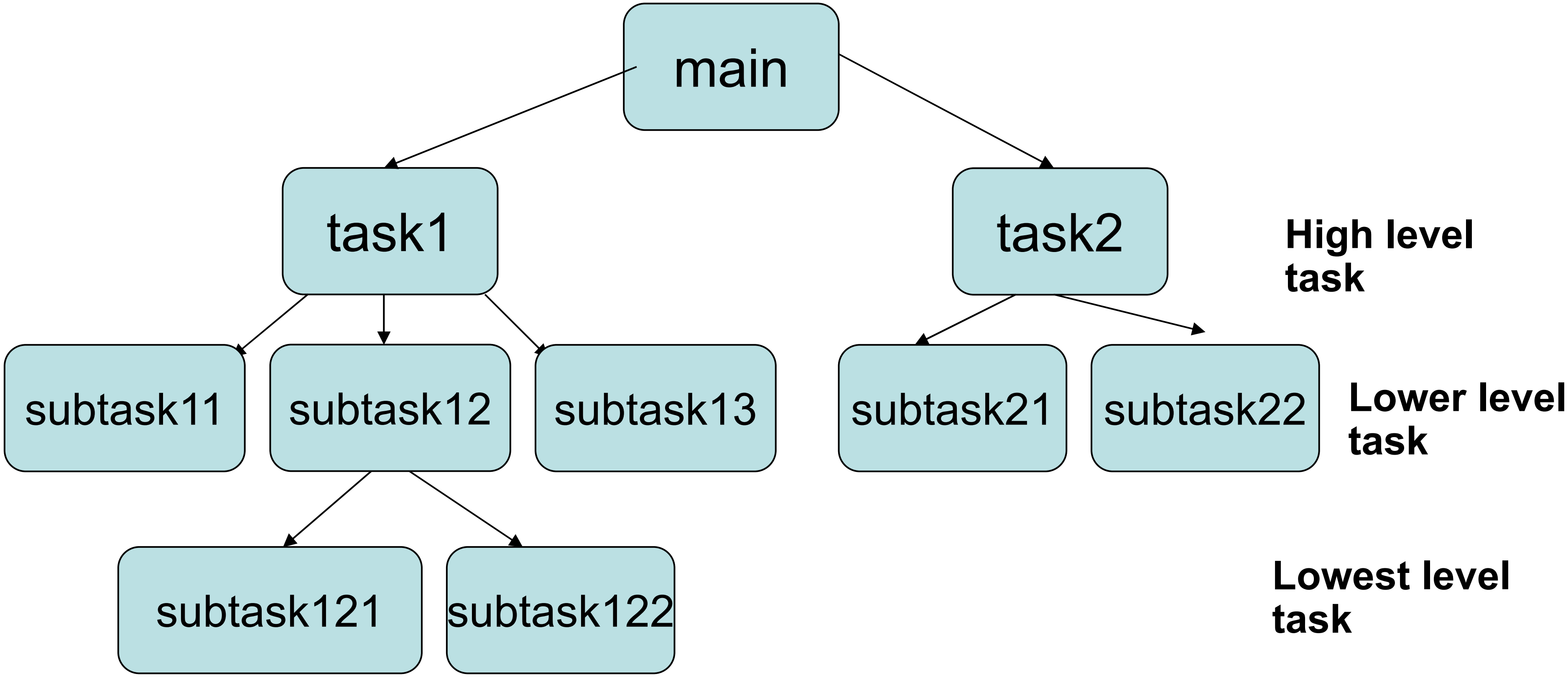
【注】A **subroutine** or **subprogram** (also called **procedure**, **method**, **function**, or **routine**) is a portion of code within a larger program, which performs a specific task and is relatively independent of the remaining code.

5. Main Program and Subroutine

主程序-子程序使用结构化设计

- 结构化的设计使用自顶向下的功能化设计
  Top-down functional design :
- 系统根据功能的视觉进行设计
  The system is designed from a functional viewpoint, starting with a high-level view and progressively refining this into a more detailed design.

5. Main Program and Subroutine



main

task1

task2

subtask11

subtask12

subtask13

subtask21

subtask22

subtask121

subtask122

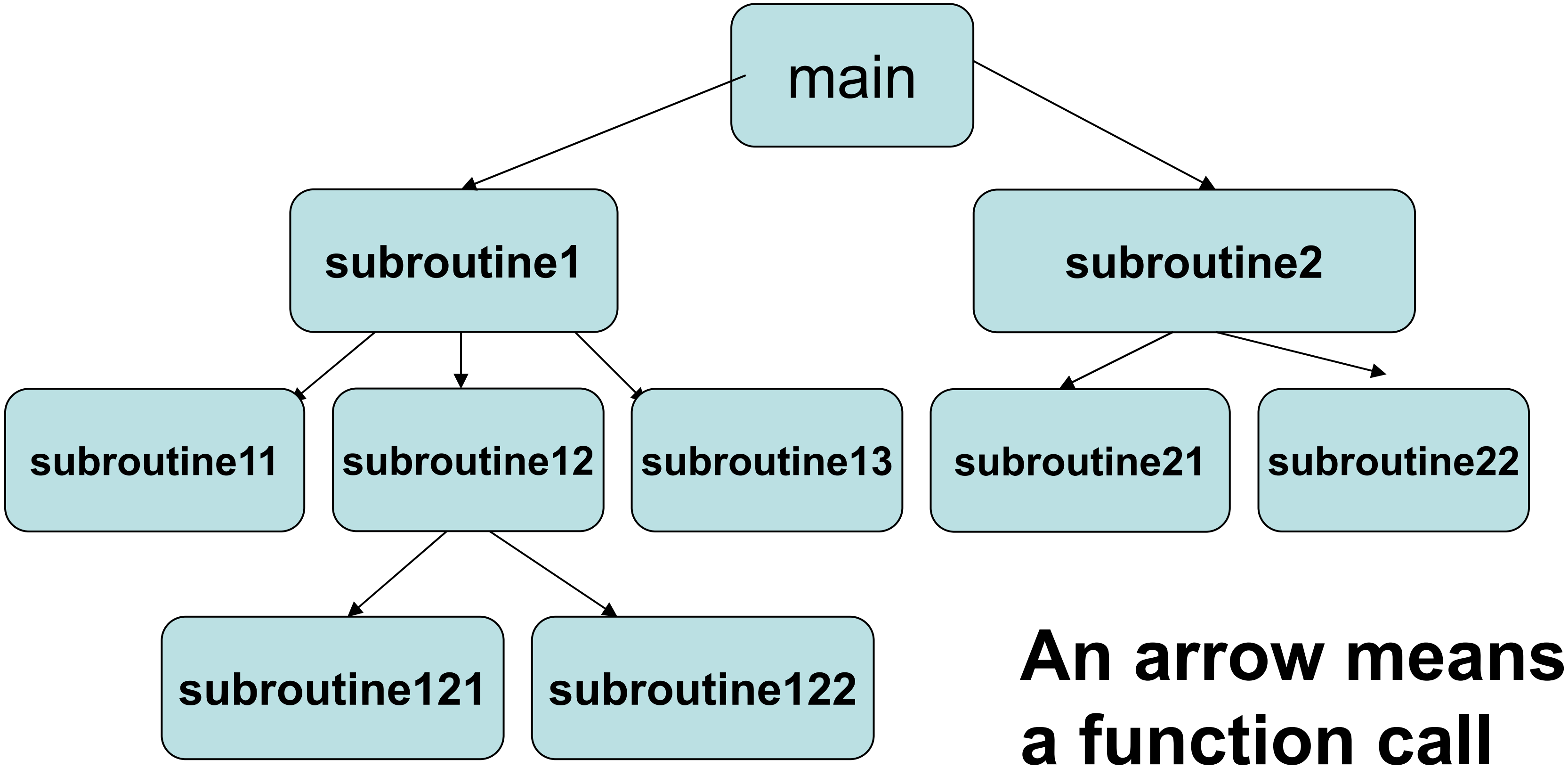
High level task

Lower level task

Lowest level task

The system depends on a collection of subroutines (functions), and no concept of object

5. Main Program and Subroutine



An arrow means a function call

The system's functionalities are achieved by many subroutines, with higher level subroutines calling lower level subroutines and so on

5. Main Program and Subroutine

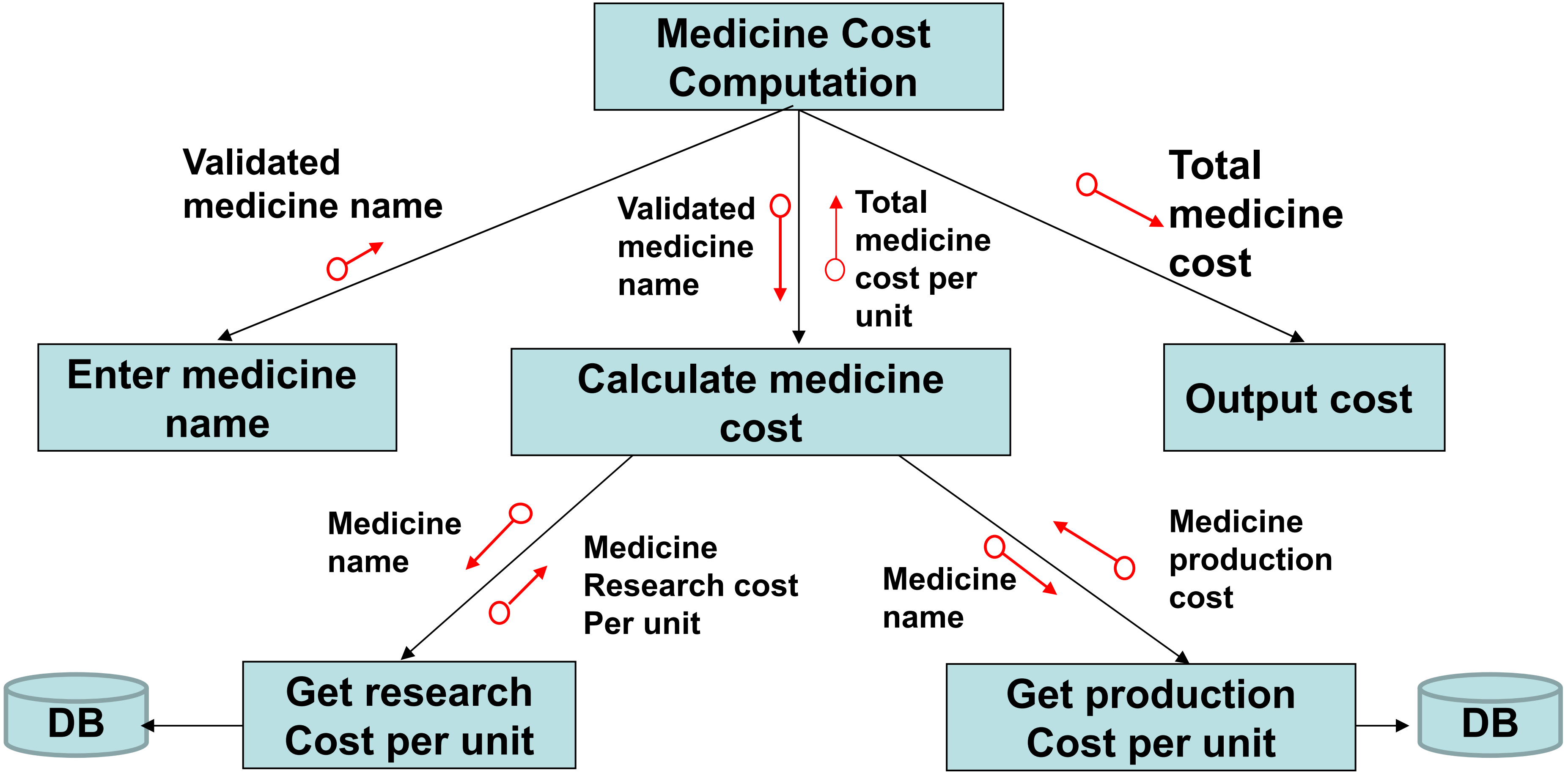
用C语言设计的医药成本计算问题

Example: a medicine cost computation problem

- **Requirement specifications**:
- Medicine Cost Computation program inputs medicine name, and the program will return the cost per unit amount of the given medicine
- The medicine cost include two part
  - **Research cost, and (研究成本)**
  - **Production cost (生产成本)**

5. Main Program and Subroutine

医药成本计算问题的程序结构图：



structure chart of the medicine cost program

5. Main Program and Subroutine

优点 (Advantage):

- 自顶向下的方法是非常成功的设计方法
  This has proved to be a highly successful design methodology.
- 对于10万行以下的代码的程序设计没有问题。
  It has allowed the development of large programs (100,000 lines or more of source code).

5. Main Program and Subroutine

缺点(Disadvantage):

- **However, as program size increases beyond this point (100,000 lines), the approach performs poorly.**
- **Reason:**
  - ➢**Code development becomes too slow and**
  - ➢**It becomes increasingly difficult to test the software and guarantee its reliability.**

Back

Object-oriented Design

是一种Call and return 架构

6. Object-oriented Design

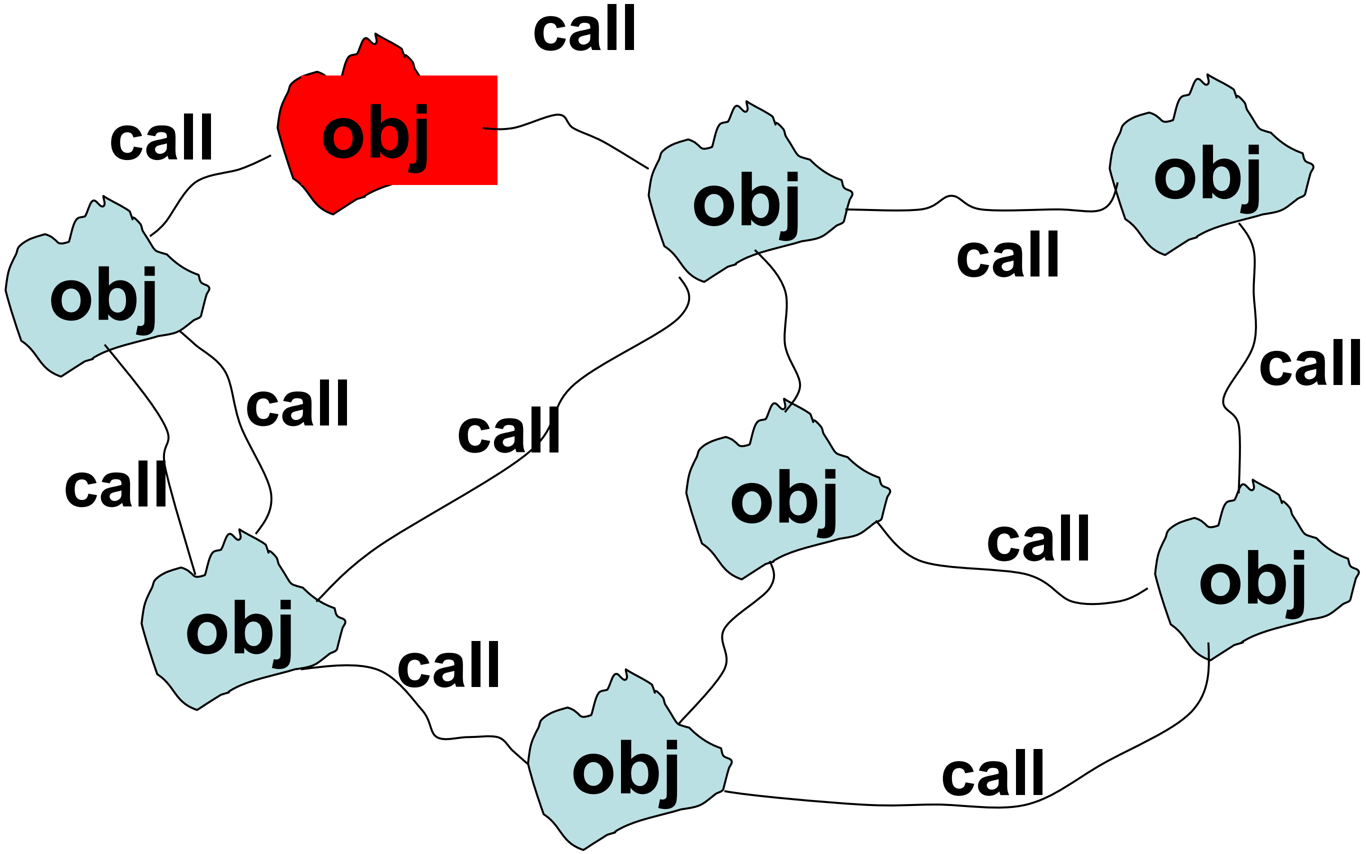
系统设计：系统可以看作是由一群对象组成

System design:

- 系统被视为一些对象的集合，而不是函数的集合。
  The system is viewed as a collection of objects rather than as functions with messages passed from object to object.
- 每个对象都拥有自己的操作
  Each object has its own set of associated operations.

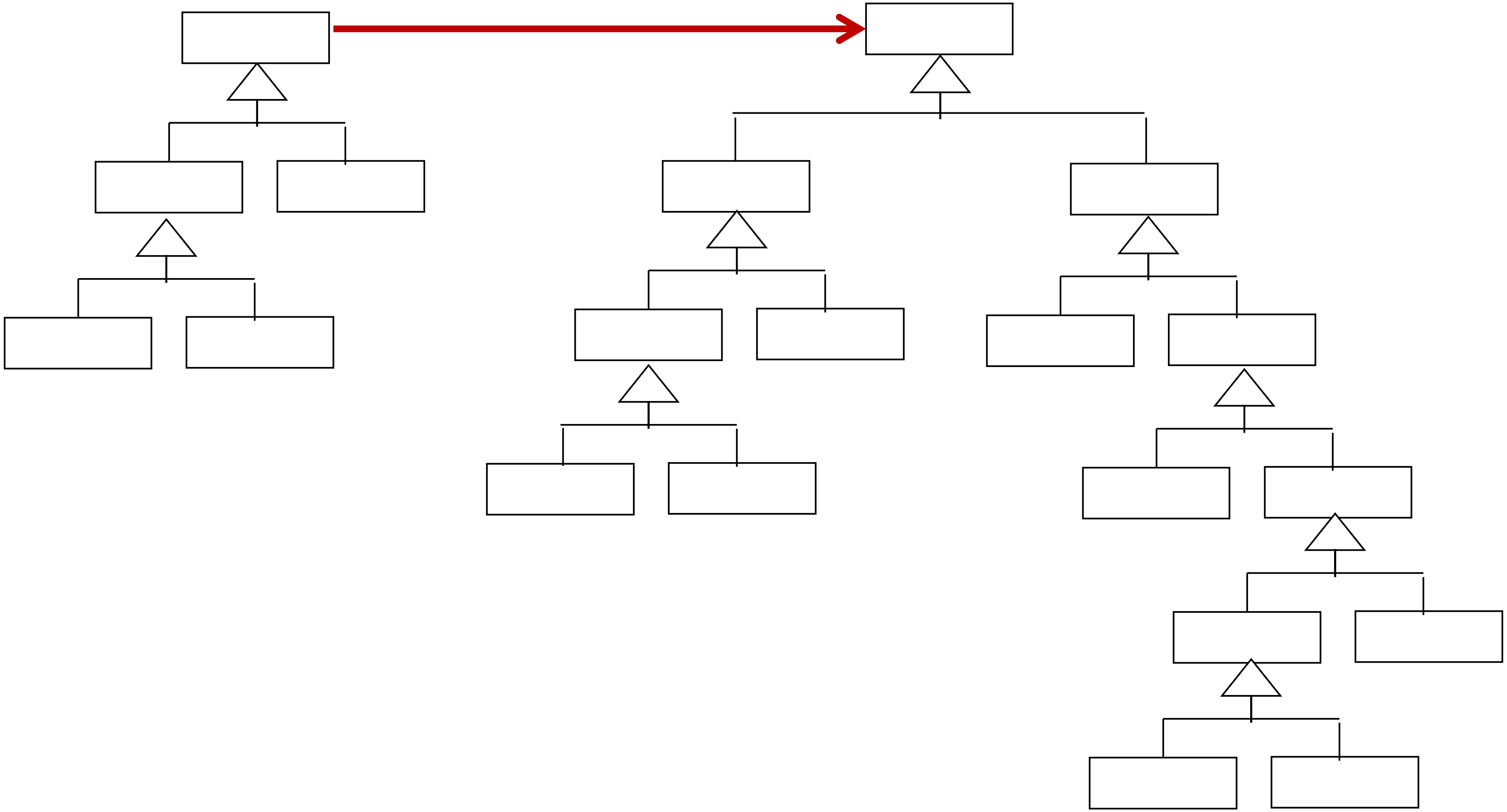
6. Object-oriented Design

A program with OO structure



Object oriented architecture

6. Object-oriented Design



面向对象设计产生类图

6. Object-oriented Design

Characteristics of Object oriented design

1) 封装(Encapsulation)
   - Restrict access to certain information
     - private variables; code inside a method

2) 继承(Inheritance)
   - Share one definition of shared functionality
     - Java class inheritance

3) 动态绑定(Dynamic binding)
   - Determine actual operation to call at runtime
   - Don’t know which object to invoke before the program runs, in the run-time, will dynamically decide which object to use.

4) 复用与维护(Reuse and maintenance)
   - Exploit capsulation and locality

Other Characteristics of OOD

a) 对象是现实世界实体的抽象. Objects are abstractions of real-world or system entities and manage themselves

b) 对象之间互相独立. Objects are independent (from encapsulation point of view)

c) 系统功能由对象服务表达. System functionality is expressed in terms of object services (functions)

d) 取消了共享数据区. Shared data areas are eliminated

e) 对象可以是分布式的. Objects may be distributed

f) 对象可以按顺序执行或者并行执行. Objects may execute sequentially or in parallel

6. Object-oriented Design

Advantages of OOD

a) 程序容易维护 (Easier maintenance)

Because an object hides its representation from its clients, it is possible to change the implementation without affecting those clients Objects may be understood as stand-alone entities

b) 对象可复用(Reusable)

Objects are appropriate reusable components

6. Object-oriented Design

Advantages of OOD

c) 现实世界映射 (Real world mapping)

For some systems, there may be an obvious mapping from real world entities to system objects

d) 容易分解系统(Easy decomposition of a system)

the bundling of a set of accessing routines with the data they manipulate allows designers to decompose problems into collections of interacting agents.

6. Object-oriented Design

结构化设计与面向对象设计的区别

- **结构化设计将软件分解为树形，其每个节点都是函数**
  Structured design decomposes a piece of software into various *functional components*. i.e., program is organized into a collection of functions, which will be executed in a defined order to produce the desired result.

- **面向对象设计将软件组织成一群互相调用的类**
  OO design organizes a piece of software into a collection of classes, that group together
  – Related items of data (*properties)*
  – Operations that are to be performed on t
  (*methods)*

Back

Case Study
案例分析

7. Case Study

项目描述: 要求设计一个关键词查找程序，算法如下

In 1974, Parnas proposed the following problem: key words in Context (KWIC) index System. Requirement specification:

a) Accepts an ordered set of lines;
b) Each Line is an ordered set of words,
c) Each word is an ordered set of characters.
d) Any line is circularly shifted by repeatedly removing the first word and appending it at the end of the line
e) The KWIC index system outputs a listing of all circular shifts of all lines in alphabetical order.

7. Case Study

查找关键词算法描述

- 算法描述：
- 1）输入一些行
- Input

  The sun is rising in the east

  Flowers are blooming

7. Case Study

2）进行圆圈移位 (产生多行)

After circular shift
The sun is rising in the east
sun is rising in the east the
is rising in the east the sun
rising in the east the sun is
in the east the sun is rising
the east the sun is rising in
east the sun is rising in the
Flowers are blooming
are blooming Flowers
blooming Flowers are

7. Case Study

3)进行按照字母排序

After alphabetizer

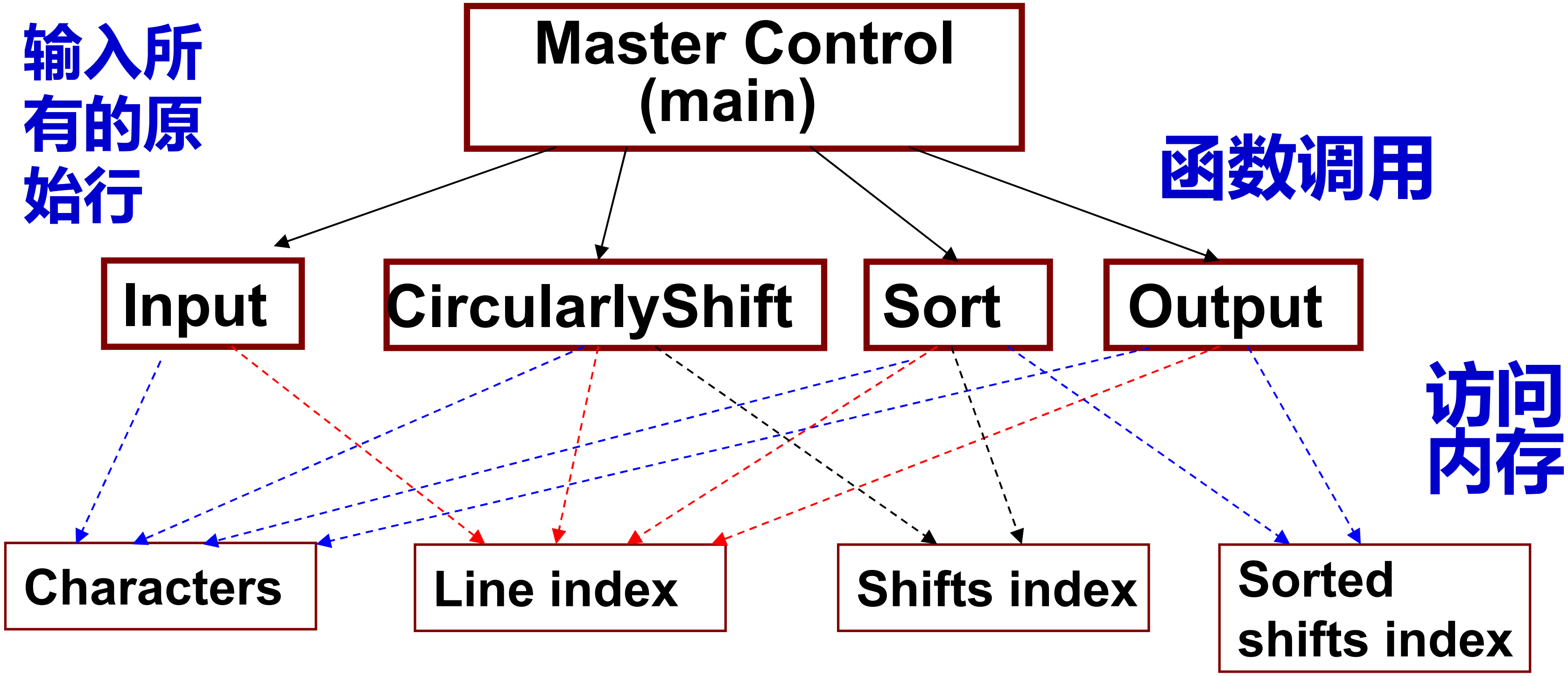
are blooming Flowers
blooming Flowers are
east the sun is rising in the
flowers are blooming
in the east the sun is rising
is rising in the east the sun
rising in the east the sun is
sun is rising in the east the
the east the sun is rising in
The sun is rising in the east

7. Case Study

设计1：主程序-子程序设计 (结构化设计)

Design of KWIC Problem
in Main Program and Subroutine
Architecture

7. Case Study



- Input：读入所有的行，将这些文字存储在内存Characters 中；同时产生行索引， 并且存入内存Line index之中
- CircularlyShift：参照 Characters和 Line index，而产生Shift索引Shifts index
- Sort：参照Characters，Line index，Shifts index而产生Sorted shifts index
- Output：输出的时候，必须参照Characters，Line index 和Sorted shifts index

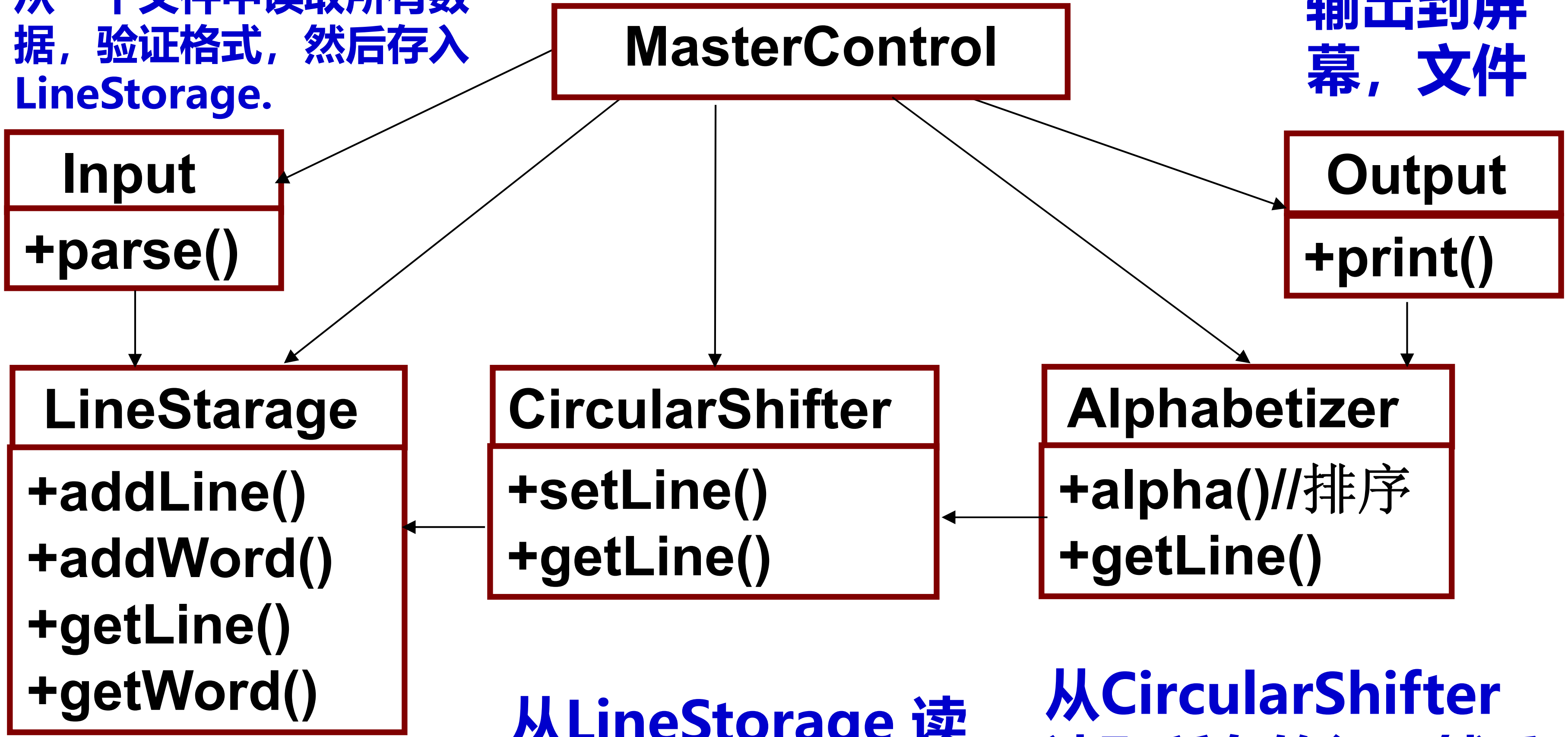
Main program and subroutine Design of the KWIC Program

7. Case Study

设计2：面向对象设计

Design of KWIC Problem
in Object Oriented Architecture

7. Case Study



从一个文件中读取所有数据，验证格式，然后存入LineStorage.

MasterControl

输出到屏幕，文件

Input

+parse()

Output

+print()

LineStarage

+addLine()
+addWord()
+getLine()
+getWord()

CircularShifter

+setLine()
+getLine()

Alphabetizer

+alpha()//排序
+getLine()

LineStorage存储所有的行的文字

从LineStorage 读取所有的行，然后，进行圆圈移位 (产生多行)；然后存储。

从CircularShifter读取所有的行，然后，进行排序；然后存储。

Back