



数据结构

Data Structure

张先宜

QQ群: **275437164** (XC数据结构交流群)

手机: 18056307221 13909696718

邮箱: zxianyi@163.com

QQ: 702190939

第1章 概论

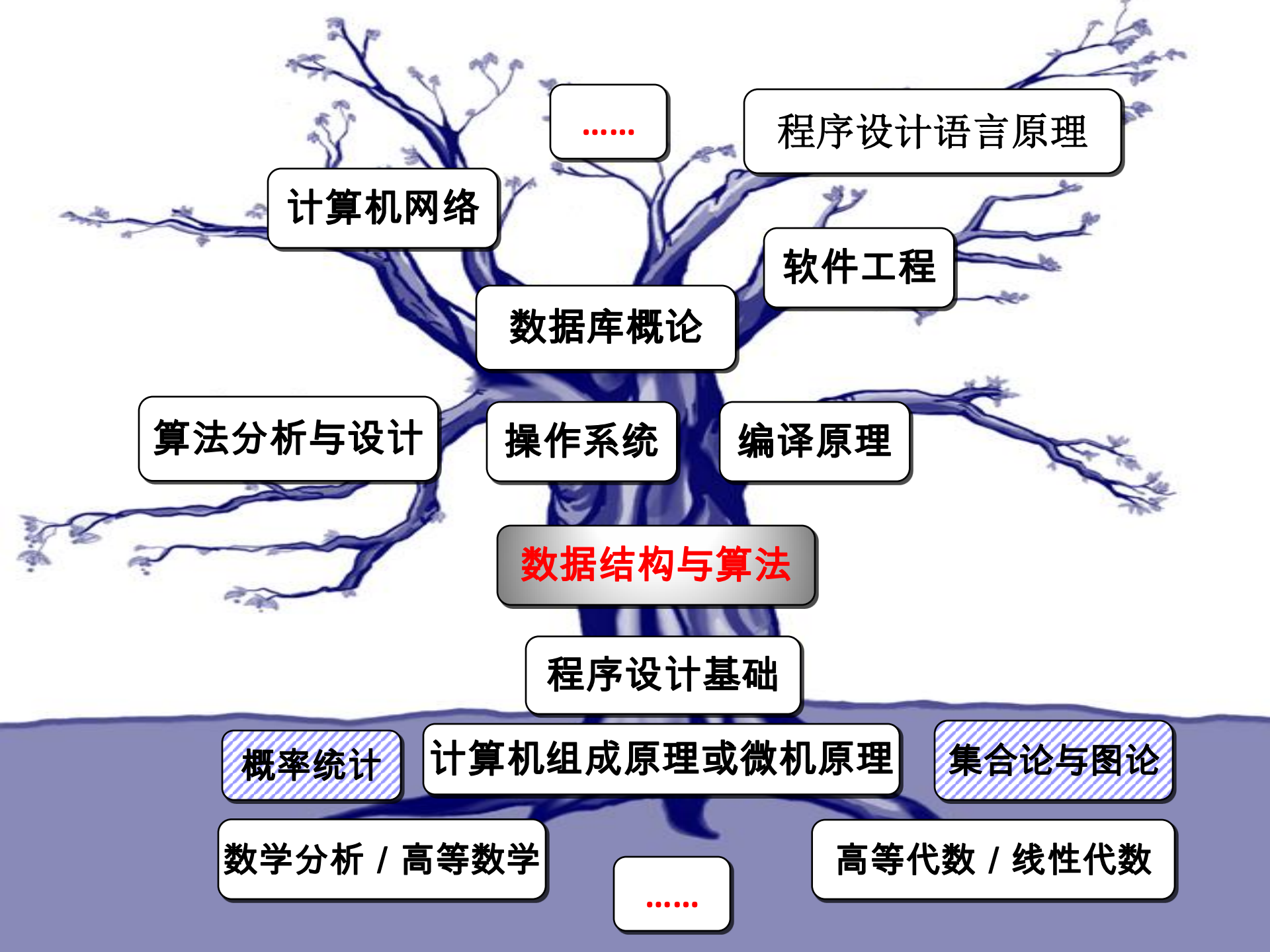
1.0 开场白

1.1 数据结构的研究内容

1.2 基本概念和术语

1.3 算法及其描述

1.4 算法分析



1.0 开场白

■ 1. 数据结构在计算机科学中的地位

👉 **最重要的专业主干基础课程**

★ 专业能力的核心课程，承前启后的重要作用

👉 **程序设计能力“质的飞跃”**

★ 算法设计与分析、操作系统、编译原理、数据库原理、网络、软件工程等的基础

👉 **计算思维能力提升 (computational thinking)**

■ 2.为什么要学习数据结构

👉 如果不学习数据结构，你充其量只能做**软件蓝领**，不能做**软件设计者**，更不可能成为**软件大师**。

■ 3. 你来听这门课的目的（摘自大话数据结构）

- ☞ 如果你的学习目的是为了将来要做一个优秀的程序员，向微软、Google的工程师们靠齐，那么你应该要努力学好它，不单是来听课、看看教科书，还需要下来做题、上机练习、看许多资料。不过话说回来，如果你真有这样的志向，就该早开始研究了，来听我的课，就更加有主动性，收获也会更大。
- ☞ 如果你的目的是为了考计算机、软件方面的研究生，那么这门必考课，你现在就可以准备起来——很多时候，考研玩的不是智商，其实就是一个投入时间问题而已。

👉 如果你只是为了混个学分，那么你至少应该要坚持来上课，在我的课堂上听懂了，学明白了，考前适当地复习，拿下这几个学分应该不在话下。

👉 如果你只是来打酱油的，当然也可以，我的课不妨碍你打酱油，但你也不要妨碍其他同学坐到好位子，所以请靠后坐，并且不说话，静心打酱油就好。


👉 如果，我是说真的如果，你是一个对编程无比爱好的人，你学数据结构的目的，既不是为了工作为了钱，也不是为了学位和考试，而**只是为了更好地去感受编程之美**。啊，你应该得到我的欣赏，我想我非常愿意与你成为朋友——因为我自己也没有做到如此纯粹地去学习和应用它。

■ 4. 学习数据结构的四种境界

- ☞ 能看懂各种数据结构的逻辑结构和存储结构，以及相应的运算（算法）；
- ☞ 能用算法描述语言描述各种数据结构；
- ☞ 能用一种熟悉的编程语言实现各种数据结构，验证简单应用；
- ☞ 能为应用系统自由的选择或设计合适的数据结构，不管什么语言和环境。

■ 5. 怎样才能学好数据结构？

- ☞ **上课认真听讲；**
- ☞ **认真看教材和参考书；**
- ☞ **与老师、同学、在线交流讨论；**
- ☞ **参与数据结构MOOC学习**
 - ✦ **北京大学 张铭**
 - ✦ **浙江大学 陈越**
 - ✦ **清华大学 邓俊辉**
- ☞ **网上搜索资料和例程；**
- ☞ **认真完成作业；**
- ☞ **独立思考课程相关问题、动手解决实际问题；**
- ☞ **撰写博客；**
- ☞ **上机、上机、上机.....** 我们的实验课时远远不够，加之同学们对C和C++熟练程度不足，必须利用平时时间加强实践训练。



👉 **学习新知识好比你来到一个新的城市，初来乍到，一切都是那么陌生，你可能有点紧张，有点焦虑，甚至有点恐惧，无所适从。只要待得时间长了，这里就是快乐老家了，闭着眼睛都可以走回去了。**

■ 6. 课程情况

☞ 理论课时 : 56

☞ 实验课时 : 24

☞ 学分 : 4.5

■ 7. 成绩计算

☞ 期末考试 : 50%

☞ 期中考试 : 20%

☞ 上机实验 : 15%

☞ 测验、作业、考勤、提问 : 15%

☞ 额外奖励 : 5% (作业上机调试、自选较难实验项目、课本找错、撰写博客、提供有效习题、积极讨论问题、帮助同学等 , 以100分为限)

■ 8.其它参考书和学习资源

- ☞ **数据结构与算法**，张铭、王腾蛟、赵海燕，高等教育出版社，2008年6月
- ☞ **数据结构(用面向对象方法与C++语言描述)第2版**，殷人昆主编，清华大学出版社,2007年6月
- ☞ **国内考试：清华大学严蔚敏教程**
- ☞ **国外深造：MIT教程**
- ☞ **谐趣入门：大话数据结构，程杰，清华大学出版社**

网上资源

★ MOOC

□ 浙江大学 陈越，北京大学 张铭，清华大学 邓俊辉.....

★ 王道论坛（数据结构考研网站）--
<http://www.cskaoayan.com/>

★

■ 9.编程工具

☞ **VC6.0**

☞ **CodeBlocks, devCPP**

☞ **Visual Studio 各种版本**

☞ **其它任何C或C++开发工具**


☞ **请大家今天就开始自行熟悉开发工具的使用吧！**

☞ **学会Java、C#你可以行走江湖啦！学会C++你就可以笑傲江湖啦，哈哈哈哈哈...！**

■ 10. 诚信

我真诚地保证：

- ☞ 我自己独立地完成了整个程序从分析、设计、编码到测试的所有工作。
- ☞ 如果在上述过程中，我遇到了什么困难而求教于人，那么，我将在程序实习报告中详细地列举我所遇到的问题，以及别人给我的提示——“在此，我感谢 XXX, ..., XXX对我的启发和帮助。”
- ☞ 我的程序中凡是引用到其他程序或文档之处，例如教材、课堂笔记、网上的源代码以及其他参考书上的代码段，我都已经在程序的注释里很清楚地注明了引用的出处。



✎ 我从未抄袭过别人的程序，也没有盗用别人的程序，不管是修改式的抄袭还是原封不动的抄袭。

✎ 我提交的作业都由自己独立完成，从未照抄别人的作业。

✎ <学生姓名>



I hear and I forget;
I see and I remember;
I do and I understand.

不闻不若闻之，闻之不若见之，见之不若知之，
知之不若行之。学至于行之而止矣。 【荀子·儒效】

1.1 数据结构的研究内容

■ 1.1.1 计算机解决实际问题的过程

■ 1. 问题建模

👉 通过**抽象**，建立实际问题求解的**数学（逻辑）模型**

👉 问题建模通常包括：

- ✦ 所描述问题中的数据对象的集合；
- ✦ 对象间关系及其描述；
- ✦ 问题求解的要求及方法等。

👉 建模的关键是抽象，即找出不同问题的共性，并把模型形式化描述出来。

☞ 例如：

- ✦ 计算机网络与电网、自来水管网、城市交通网本来没有关联，但是它们有许多共性的东西，都可以用抽象的图模型来表示。
- ✦ 磁盘目录结构、国家或组织的行政组织结构、家族的谱系等都可以抽象为树模型。
- ✦ 企业、组织、学校的事务信息管理等都可以抽象为线性表模型

☞ 实际问题的求解模型可以选择现有模型，或组合现有模型，更复杂的问题可能要创建新的模型。

☞ 建模过程涉及到数据结构。

■ 2. 构造求解算法

- ☞ **算法：根据建模抽象的问题模型，设计原问题的求解方法。**
- ☞ **借助模型的已有知识，可以相对简单地描述问题的求解方法。**
 - ✦ **例如：利用图结构的已有知识来分析和求解计算机网络问题变得更为简单、容易。**
- ☞ **当然，算法设计过程还涉及到更多的技术和知识，需要进一步学习和实践。**
 - ✦ **常见算法设计技术如：递归、分治法、回溯法、动态规划法、图搜索等。**

- ❏ 但我们日常遇到的实际问题都是经常出现的问题，人们已经对它的模型进行了大量的分析和研究，直接借用他人的成果就可以解决问题，所以在一开始你面对一些常见简单问题的处理时，你甚至感觉不到要去抽象模型、设计算法。
- ❏ 从某种意义上说，针对一个问题设计的算法不仅要能实现原问题的求解，而且还可能实现许多类似的具体问题的求解，尽管这些具体问题的背景及其描述形式可能存在较大的差异。
- ❏ 算法设计是计算机专业的核心能力，是区别于其他专业的最核心能力之一。

■ 3. 选择或设计存储结构

- ☞ 构造出求解算法之后，接着就需要考虑在计算机上实现求解了。计算机实现的首要工作就是为问题选择或设计合适的存储结构，以便将问题所涉及到的数据存储在计算机中。
- ☞ **存储结构包括数据中的基本对象及对象之间的关系**
- ☞ 不同的存储形式对问题的求解实现有较大的影响，所占用的存储空间也可能有较大的差异。通常要考虑时间和空间的综合性能最佳。
 - ✦ **例子：同一图片采用BMP和JPG格式存储空间的差异。**

- **BMP:2700KB , JPG:269KB , 相差10倍。**



■ 4. 编程实现

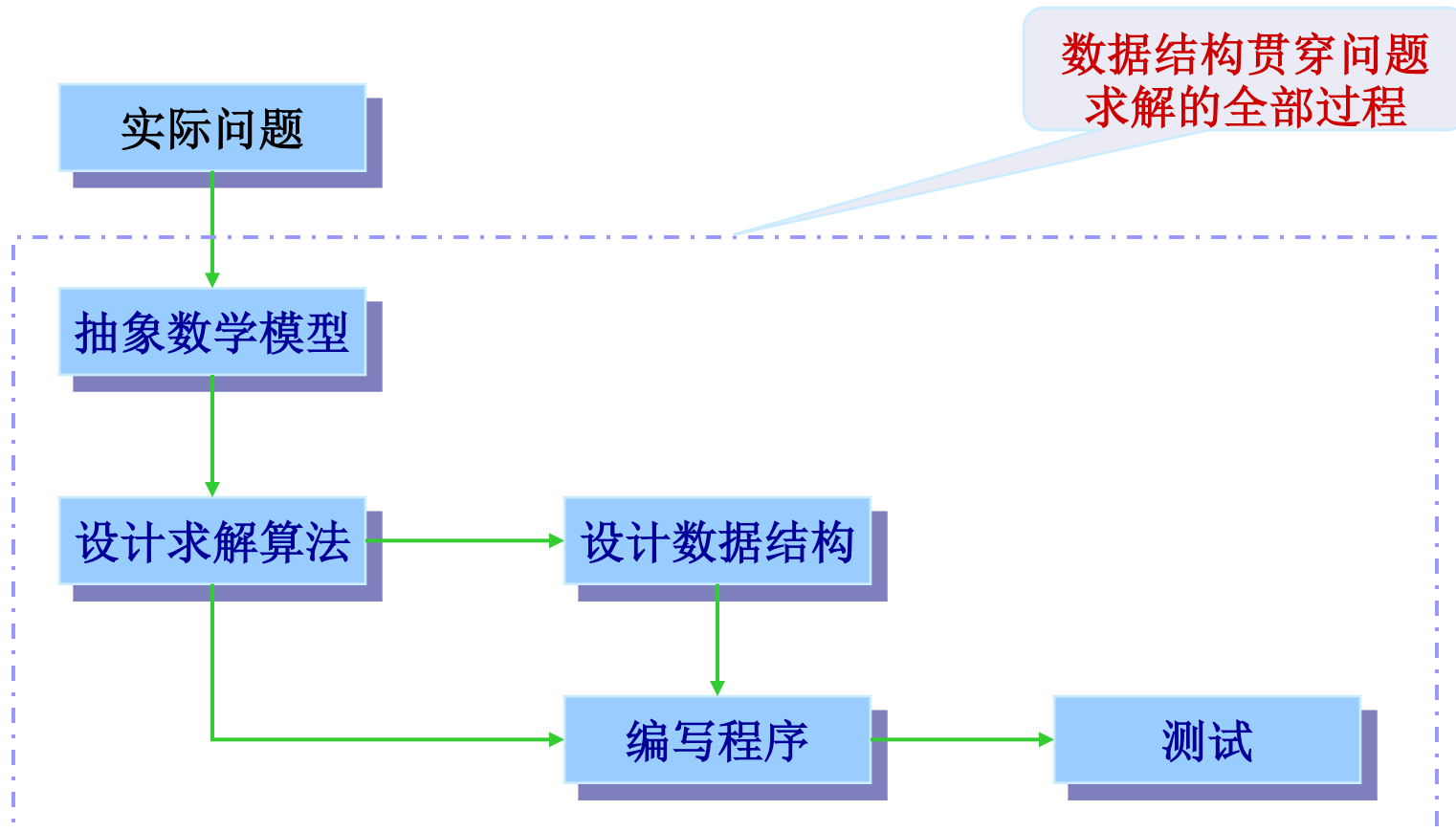
- ☞ 选择合适的设计语言和开发环境编写程序，实现既定的存储结构和算法。

■ 5. 测试

- ☞ 检测程序的功能和性能是否满足设计需求；
- ☞ 查找程序中的bug。

- **结论：数据结构贯穿计算机求解问题的全部过程。即前述各个阶段都可能用到数据结构和算法的知识和技术。可用下图来直观描述：**

■ 计算机解决问题流程



■ Niklaus Wirth

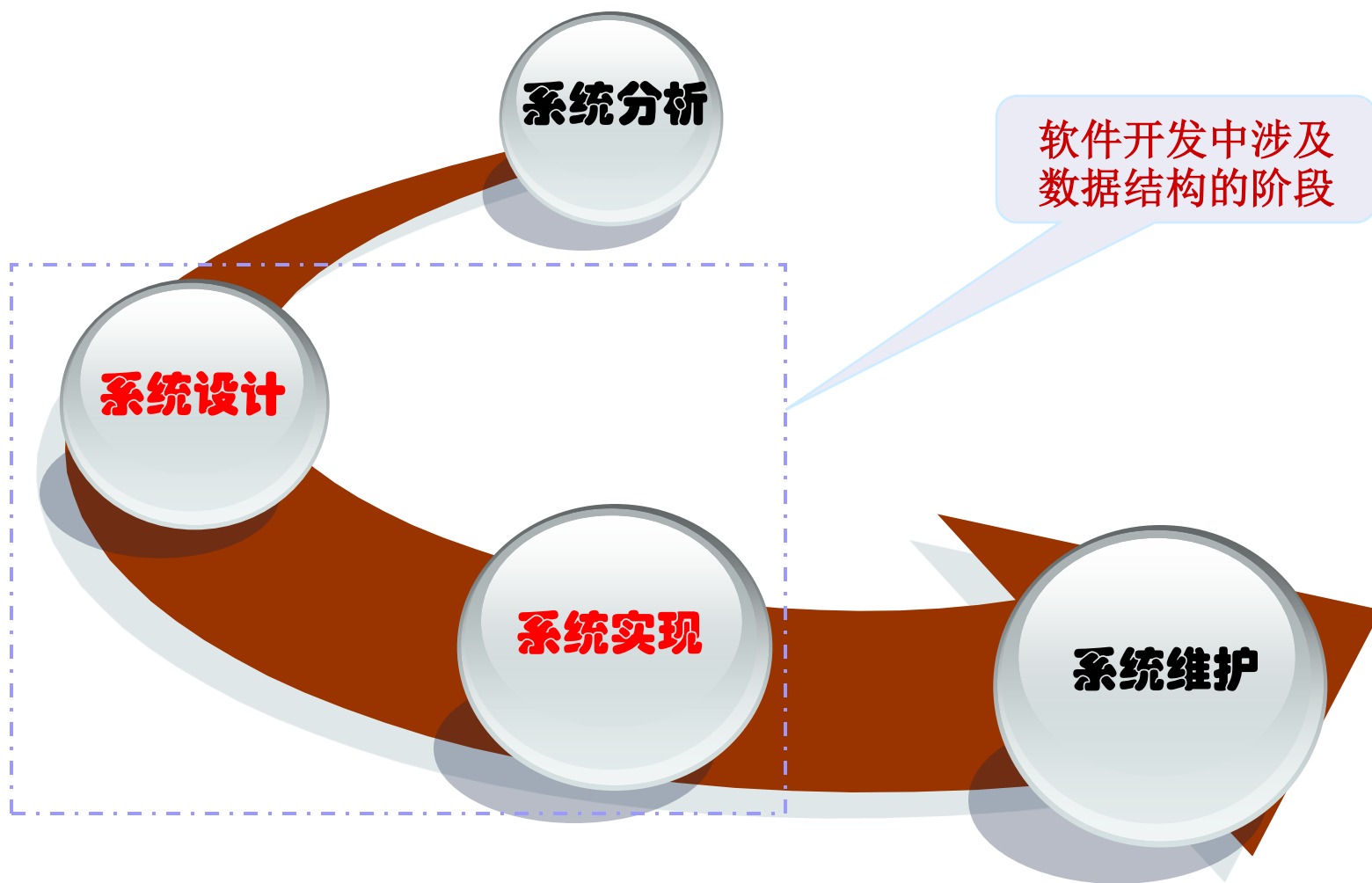


Algorithm + Data Structures = Programs

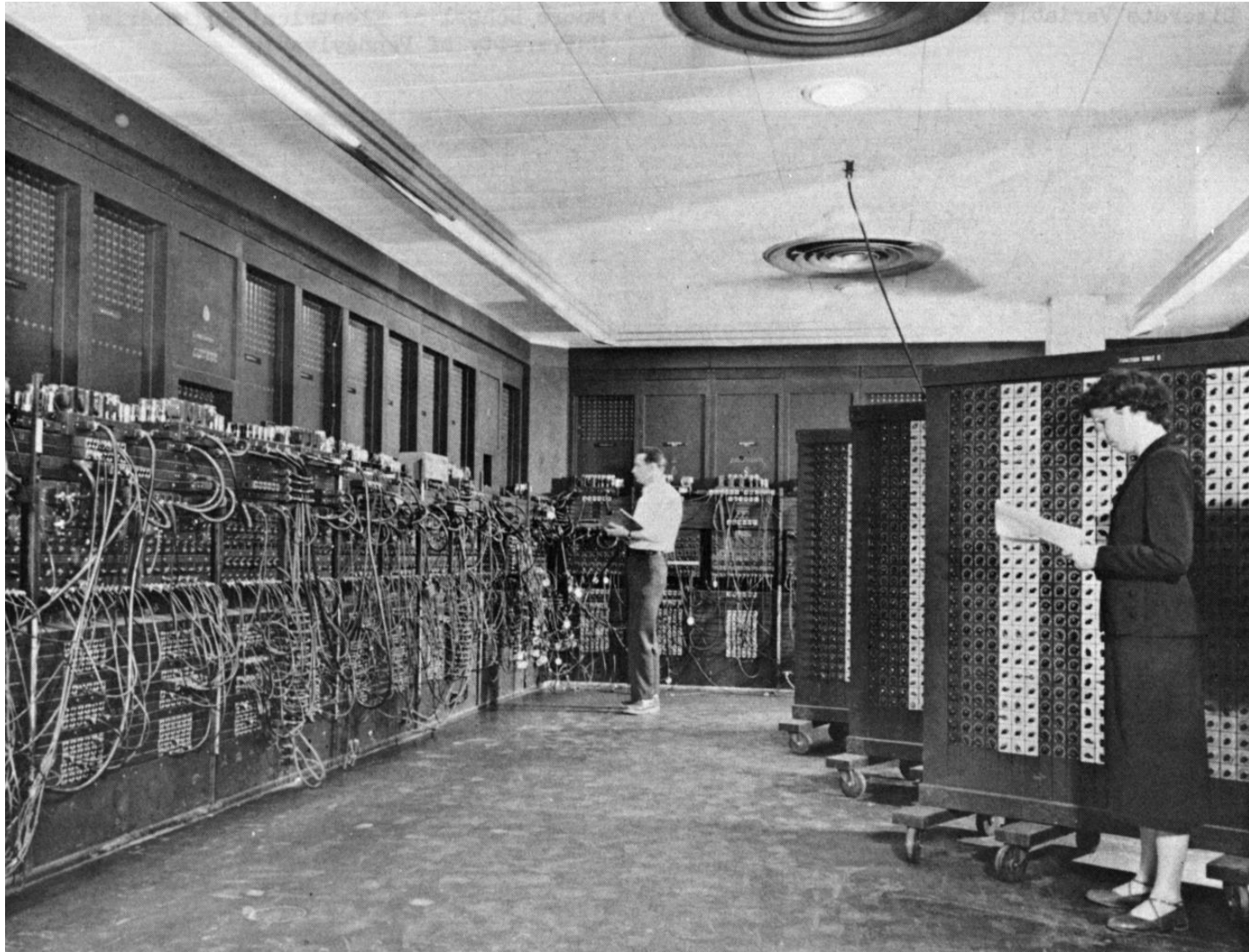
算法 + 数据结构 = 程序

👉 **程序：为计算机处理问题编制的一组指令集**

■ 1.1.2 软件开发中涉及数据结构的阶段



ENIAC



1.2 基本概念和术语

■ 1. 数据 (data)

☞ 数据 (data) —— 是描述客观事物的符号，是信息的载体。是能够输入到计算机中，并能被计算机处理的符号的集合。

✦ 例如，工资表，学生成绩表，电子通信录，电子字典，数字图像、声音、视频等

☞ 对数值类型数据可以进行计算；对字符型数据可以进行非数值处理；对图像、声音、视频等通过编码变为字符型数据来处理。

☞ 数据在不同学科理解上可能有一些细微的差别，在数据结构中一般指具有一定逻辑结构的数据。

■ 例如：集合类型数据

➡ { 12, 3, 6, 8, -5, 18 }

➡ { a, c, w, f, d }

➡ { 张三, 李四, 王五, 赵六 }

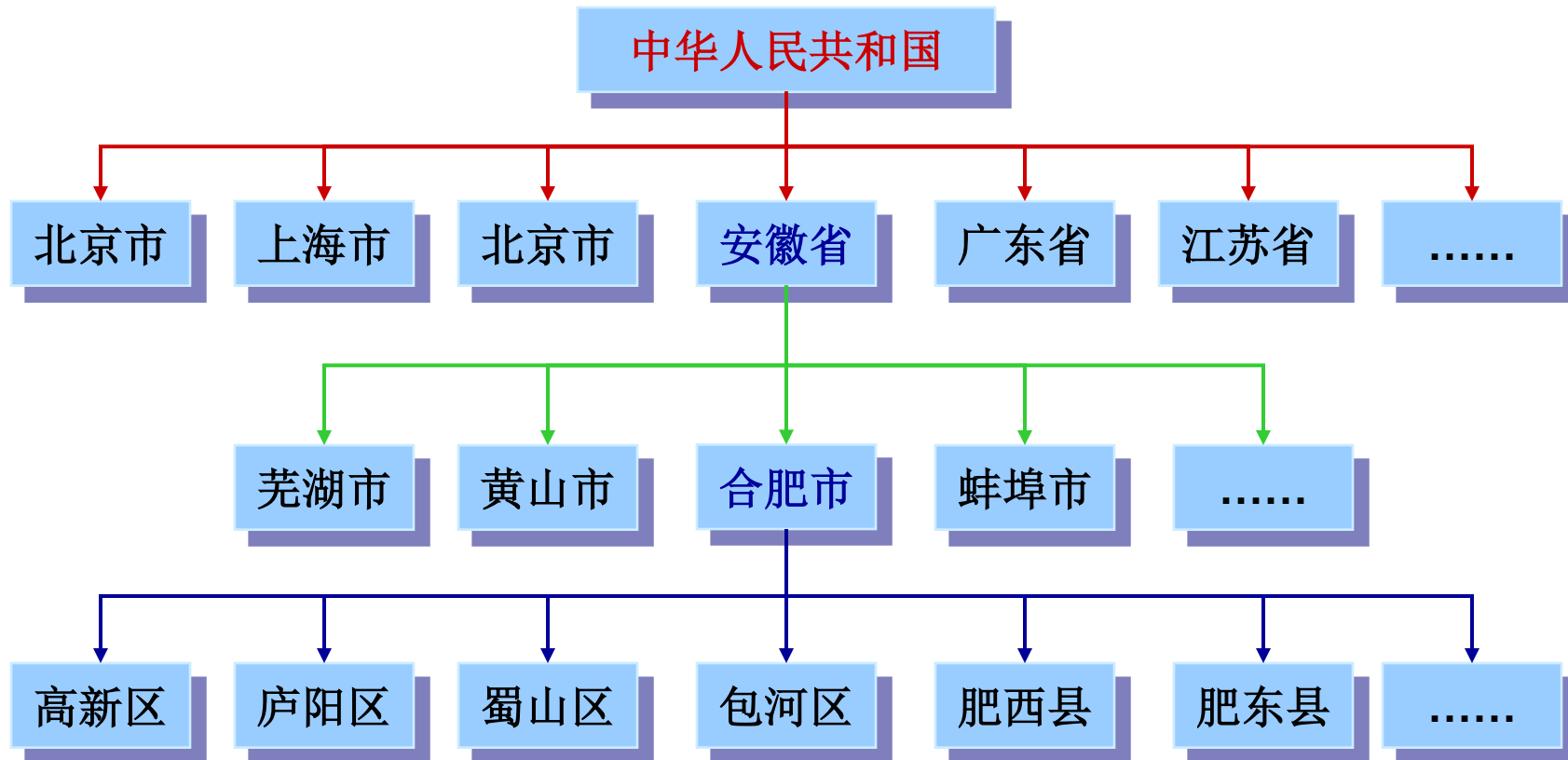
■例如：线性表类型数据—学生信息表

学号	姓名	性别	专 业	年 级
30880101	赵一	男	计算机科学与技术	3088级
30880208	钱二	女	信息安全	3088级
30890301	孙三	女	计算机应用	3089级
30890202	李四	男	信息安全	3089级
30890103	周五	男	计算机科学与技术	3089级
30900101	武六	女	计算机科学与技术	3090级
30900302	郑七	男	计算机应用	3090级
30900203	王八	男	信息安全	3090级
30910101	冯九	女	计算机科学与技术	3091级
30910302	陈十	男	计算机应用	3091级

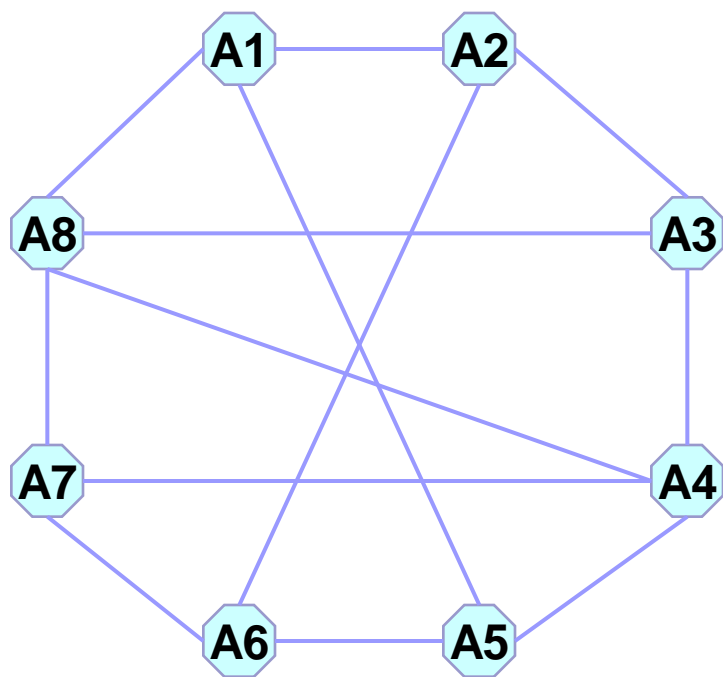
■例如：线性表类型数据—工资信息表

编号	姓名	基本工资	岗位津贴	奖金	...
05001	张三	8888.00	4588.00	988.00	
05002	李四	3500.00	2300.00	688.00	
05003	王五	5325.00	3487.00	888.00	
05004	赵六	2068.00	1898.00	588.00	
05006	刘七	4554.00	2786.00	788.00	

■ 例如：树类型数据—国家行政组织结构图



■ 例如：图（网络）型数据——“熟人”网络



■ 2. 数据元素 (data element)

- ☞ **描述数据对象**，也称为**元素、记录、结点、顶点等**。
- ☞ 构成数据的**基本单位**（具有完整的独立意义）。
- ☞ 在计算机程序中通常作为一个整体进行考虑和处理。
- ☞ **数据结构中考察数据之间的逻辑关系和运算都是以数据元素为基本单位的。**
- ☞ **例如：**
 - ✦ 学生信息表中的每个学生记录，国家行政组织结构图中的每个结点，熟人网的每个顶点都是一个数据元素。

■ 3. 数据项 (data item)

☞ 也称为字段 (field)、栏目 (column)。

☞ 数据元素 (对象) 各种属性的描述信息。

☞ 一个数据元素可以由若干个数据项构成。

☞ 数据项是数据不可分割的**最小单位**。

☞ 例子：

★ “人” 这个数据元素，可以有眼、耳、鼻、嘴、手、腿等数据项；

★ 也可有身高、体重、肤色等数据项；

★ 也可有姓名、性别、年龄、住址、电话等数据项。

★ 具体有哪些数据项，视应用而定。

■例如：学生信息表中的数据元素和数据项

学号	姓名	性别	专 业	年 级
30880101	赵一	男	计算机科学与技术	3088级
30880208	钱二	女	信息安全	3088级
30890301	孙三	女	计算机应用	3089级
30890202	李四	男	信息安全	3089级
30890103	周五	男	计算机科学与技术	3089级
30900101	武六	女	计算机科学与技术	3090级
30900302	郑七	男	计算机应用	3090级
30900203	王八	男	信息安全	3090级
30910101	冯九	女	计算机科学与技术	3091级
30910302	陈十	男	计算机应用	3091级

表中每一列对应一个
数据项（字段）
数据元素有**5**个数据项
构成

表中每一行对应一个
数据元素（记录）
这是一个线性表结构，
有**10**个数据元素

■ 4. 数据结构 (data structure)

☞ **构成数据的数据元素之间的结构关系。** (**数据元素及其关系的集合**)。

☞ **每种数据结构都包含三个方面的内容：**

✦ **逻辑结构**

✦ **存储结构 (物理结构)**

✦ **运算**

☞ **下面对这三个方面进行较详细的介绍：**

■ (1) 逻辑结构

☞ **表示数据元素之间的逻辑关系。**

☞ **数据结构按数据元素之间的内在逻辑关系分类：**

- ① **集合**--元素之间没有关系，元素不能重复。
- ② **线性结构**--元素之间具有一对一次序关系
- ③ **树形结构**（树型结构）--元素之间的关系类似于现实中的树，具有一对多的层次关系。
- ④ **图结构**（网状结构）--元素间的关系较复杂，呈多对多的网状关系。

☞ **数据的逻辑结构独立于计算机，是数据本身所固有的。**

☞ **参见本节开始的各个实例。**

■ (2) 存储结构

☞ **数据结构在内存中的存储实现形式。**

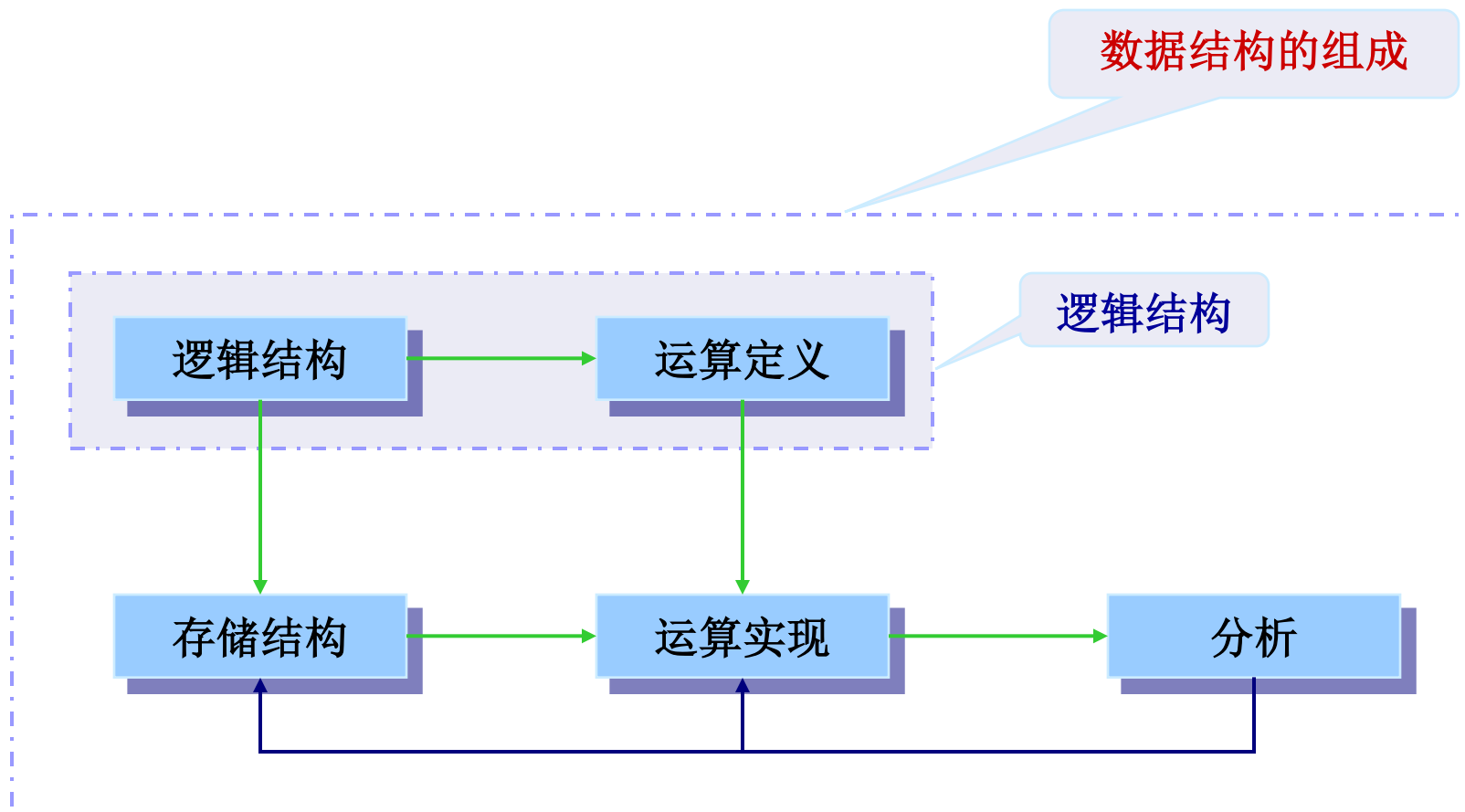
☞ **按存储形式分类：**

- ① **顺序存储结构**--所有元素存放在一片连续的存储单元中，逻辑上相邻的元素放到计算机内存仍然相邻。
- ② **链式存储结构**--所有元素存放在可以不连续的存储单元中，但元素之间的关系可以**通过地址确定**，逻辑上相邻的元素放到计算机内存后不一定是相邻的。即：每个元素上附加相邻元素的地址信息，通过此地址找到相邻元素。
- ③ **索引存储结构**
- ④ **散列存储结构**

■ (3) 运算

- ☞ **运算是指施加在数据结构上的一组操作总称，也称为算法。**
- ☞ **运算的定义依赖于逻辑结构，但运算的实现必依赖于存贮结构。**
- ☞ **每种数据结构都有插入、删除、修改、检索等共性操作（运算）；**
- ☞ **各种结构又可能有自己特有的操作，如树结构中找一个结点的父结点、子结点、兄弟结点等。**

■ 有关数据结构几个方面的联系图





ANALOG COMPUTERS

创新是进步的永恒动力！

1.3 算法及其描述

■ 1.3.1 算法 (algorithm)

- ☞ 通俗地讲，算法就是特定问题的求解方法。
- ☞ 更严格定义，**算法是由若干条指令组成的有穷序列**，必须满足（也称为算法的五大特性）：
 - ① **输入**：0或m个输入（算法开始前的初始量）
 - ② **输出**：1或n个输出，它们是算法执行完后的结果。
 - ③ **有穷性**：指令的执行次数必须是有限的。
 - ④ **确定性**：指令的描述是确定的，无二义性的。使得对相同的输入能产生相同的输出结果。
 - ⑤ **可行性**：每条指令的执行时间都是有限的。算法中每条指令可用计算机指令的有限次执行来实现。

■ 1.3.2 算法描述

- ☞ **用形式化方法表达算法。**
- ☞ **数据结构和算法独立于程序设计语言，可用多种手段进行算法描述：**

■ 1. 自然语言描述

- ☞ **用我们日常生活中的自然语言（可以是中文形式，也可以是英形式）也可以描述算法。**
- ☞ **特点：灵活、易用，但不严谨（一句话多种理解）。**

■ 2. 数学语言描述

- ☞ 数学语言或约定的符号语言来描述算法。

■ 3. 流程图描述

- ☞ 用图形符号描述算法，输入、输出、判断、处理分别用不同的框图表示，用箭头表示流程的流向。
- ☞ 特点：直观、易理解
- ☞ 本课程某些地方会用到流程图来辅助描述算法。

■ 4. 计算机语言描述

- ☞ 用一种计算机语言来表达算法，事实就是编写程序。
- ☞ 特点：准确、严格，但死板。

■ 5. 伪语言（类语言）描述

- ☞ 以一种计算机语言为基础，加上少量自然语言、数学语言等的描述方式。
- ☞ 类C、类C++、类Pascal、类Java、类C#等。
- ☞ 特点：计算机语言和自然语言的折中。
- ☞ 这样写出来的东西叫**伪代码（Pseudocode）**，不能直接到计算机上运行。
- ☞ **本课程采用C和C++结合的伪语言描述方法。**

■ 1.3.3 本课程使用的扩充伪语言说明

☞ C语言内容部分不再介绍。

■ 1. 输入

☞ `cin>>` : C++控制台键盘输入函数，用来从键盘输入一个数给同类型的变量。例：

✦ `cin>>x;` //键盘输入一个数给变量x。

☞ 可以同时输入多个数给不同变量。例：

✦ `cin>>x1>>x2>>x3>>x4>>x5;`

☞ 功能类似C语言中的scanf函数。

■ 2. 输出

☞ **cout<< : C++中控制台屏幕输出，将表达式的值输出到计算机屏幕。例：**

✦ **cout<<exp; //将表达式exp输出到屏幕**

☞ **可同时输出多个表达式的值**

✦ **cout<<exp1<<exp2<<exp3<<exp4<<endl; //endl为换行。**

☞ **功能类似C语言中的printf函数。**

■ 3. 引用

☞ C++中的一种新的函数参数传递方式，属于一种传址方式。

☞ 变量的引用事实是变量的一个别名，即：一个变量2个名字。像一个人名字叫“张三”，别名叫“三毛”，都是指同一个人。

☞ 用法：

✦ `int a=8;`

✦ `func(int & x);` //函数申明

✦ `func(a)` //函数调用，x为a的引用，别名

☞ 避免使用多重指针（指针的指针）。

■ 4. 最小和最大值函数

☞ `min()`和`max()` //伪代码

■ 5. 变量值交换

☞ `x1↔x2;` //交换变量`x1`和`x2`的值 , 伪代码

■ 6. 出错处理

☞ `error(“错误信息”);` //伪代码

☞ 相当于 :

```
cout<<“错误信息” <<endl;  
return;
```

- 为了便于听课、阅读教材、实验，请大家回去学习和练习C和C++语言重点相关内容：

- ☞ 指针

- ☞ 结构体

- ☞ 函数参数传递（传值、传址），特别是指针、引用作为参数传递

- ☞ 引用（C++）

- ☞ 文件输入/输出

- ☞ ...

■ 1.3.4 算法实例

【例1】 一百元钱买一百支笔，其中，钢笔3元一支，圆珠笔2元一支，铅笔0.5元一支。

☞ **解法一（最差方法）**：用3层循环

```
for(i=1;i<=100;i++)  
    for(j=0;j<=100;j++)  
        for(k=0;k<=100;k++)  
            if((i+j+k==100) && (3*i+2*j+0.5*k)==100)  
                printf(“%d,%d,%d”, i, j, k );
```

☞ **总循环次数**： $101*101*101=1030301$ 次。

■ 解法二：

☞ 分析：满足要求，最多买钢笔20支，圆珠笔34支；

☞ 且 $k=100-i-j$

```
for(i=1;i<=20;i++)  
    for(j=1;j<=34-i;j++)  
        if((3*i+2*j+0.5*(100-i-j))==100)  
            printf("%d,%d,%d", i, j, 100-i-j);
```

☞ 只需运行500次左右。

■ 解法三：

☞ 是否可以进一步优化呢？

☞ 由方程

$$+ i+j+k=100$$

$$+ 3*i+2*j+0.5*k=100$$

☞ 解得：

$$+ j=(100-5i)/3$$

$$+ k=100-i-j=(200+2i)/3$$

```
for(i=1;i<=20;i++)
```

```
    if((3*i+2*(100-5i)/3+0.5*(200+2i)/3)==100)
```

```
        printf(“%d,%d,%d”, i, (100-5i)/3, (200+2i)/3);
```

☞ 循环不到20次。

【例2】求最大公因子

求任意两个整数 M ， N 的最大公因子(M,N)

■ 解法一：-- 直接试探法

➡ 最大公因子应为 1 到 $\min(M,N)$ 之间的一个数。如果最大公因子为1，则 M 、 N 互质。

➡ 从小到大试探

✦ 试探次数 $\min(M,N)$ 次

➡ 从大到小试探

✦ 公因子较大时试探次数较少，公因子较小时试探次数较多，互质时试探 $\min(M,N)$ 次。

■ 解法二：-- 质因子分解法

- ➡ 从2开始寻找当前M和N的公因子；
 - ➡ 每当找到一个公因子 h_i ，就做 $M=M/h_i$ ， $N=N/h_i$ ；
 - ➡ 重复这个过程，直到M和N没有公因子为止；
 - ➡ 所有 h_i 的乘积即为最大公因子。
- ➡ h_i 的值从2到 $\min(M,N)$ ，加1尝试，但 $\min(M,N)$ 是变化的，越变越小。

➡ 例如(1000,550)

✦ 最大公因子为 $2*5*5=50$

2	1000	550
5	500	275
5	100	55
	20	11

解法三：-- 辗转相除法

若 $M=N*Q+R$

其中: R 为**余数**, 满足 $0 \leq R \leq N-1$

则 $(M,N)=(N,R)$

且当 $R=0$ 时, $(M,N)=N$

按照这种方法, 可以快速而方便地求出任意两个整数的最大公因子。

例如, $(1500, 550)$ 的求解过程如下:

$$1500 \% 550 = > 400$$

$$\begin{aligned} (1500, 550) &= (550, 400) \\ &= (400, 150) = (150, 100) \\ &= (100, 50) = (50, 0) = 50 \end{aligned}$$

最终求得1500和550的最大公因子为50。

辗转相除法算法C语言实现：

```
int hcf(int m, int n)
{
    while (n!=0)
    {
        r=m % n;
        m=n; n=r;
    }
    return m;
}
```

其对应的递归函数如下：

```
int hcf(int m, int n)
{
    if (n==0) return m;
    else return hcf(n, m % n);
}
```



发现美，享受美，创造美！

1.4 算法分析

- **由前面例题可知，对同一问题，不同的算法花费时间和空间是有差异的，某些方法难以实际应用。**
- **那么除了时间性能，还需要考虑哪些性能呢？**
 - ① **时间性能——运行算法所需的时间开销。**
 - ② **空间性能——运行算法所需的辅助空间的规模。**
 - ③ **其它性能——如可读性/可移植性等。**

1.4.1 时间复杂度 (Time Complexity)

■ 1. 时间复杂度描述方法讨论

- ① 以算法运行的机器时间开销来度量
问题是：与具体机器相关，难以比较
- ② 以算法中语句的执行次数来衡量
问题是：计算麻烦，难以实现
- ③ 以算法中语句的执行次数的数量级来替代。

■ 2. 数量级

☞ 如果变量 n 的函数 $f(n)$ 和 $g(n)$ 满足：

$$\lim_{n \rightarrow \infty} \left(\frac{f(n)}{g(n)} \right) = k$$

其中： k 为常数，且 $k \neq \infty$, $k \neq 0$

则称 $f(n)$ 和 $g(n)$ 是同一数量级的。

■ 3. 数量级的大O记法

👉 如果 $f(n)$ 和 $g(n)$ 是同一数量级的，记为 $f(n)=O(g(n))$ ，称此形式为大 O 记法。

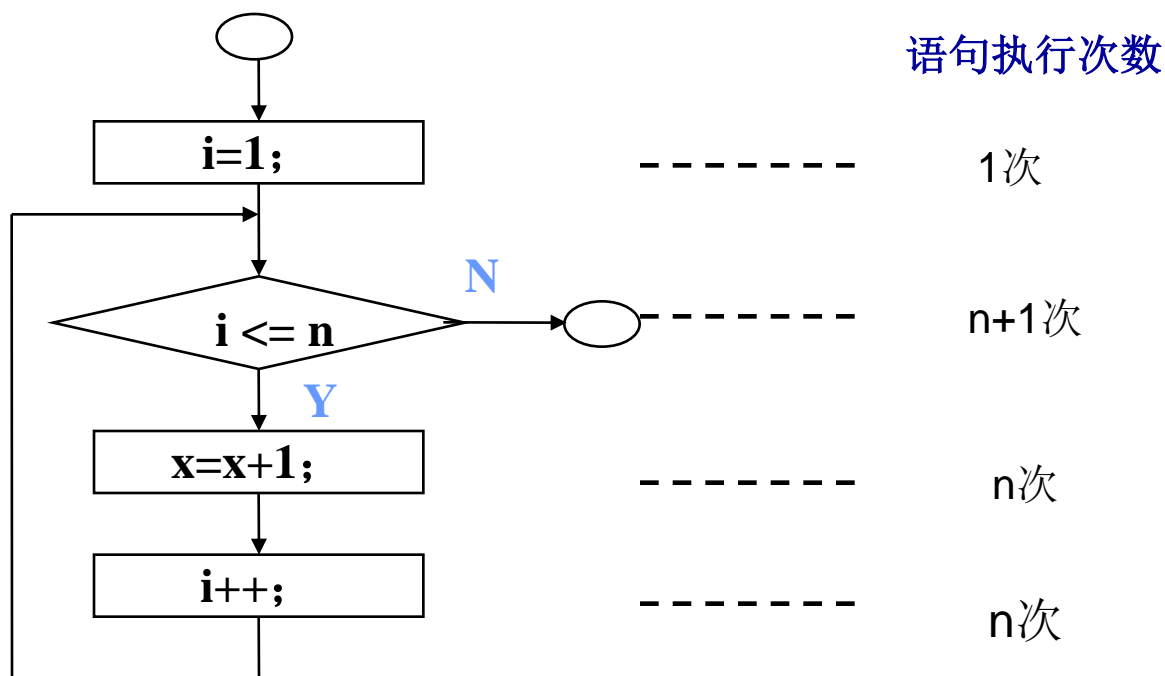
👉 常见时间复杂度从小到大依次为：

$$\begin{array}{ccccccc} O(1) & < & O(\log_2 n) & < & O(n) & < & O(n \log_2 n) \\ & & & & & & \\ & < & O(n^2) & < & O(n^3) & < & O(2^n) & < & O(n!) \end{array}$$

难以实际应用!

- 【例3】：求解以下程序段的时间复杂度：
`for (i=1; i<=n; i++) x=x+1;`

该语句的流程图如下:



共：3n+2次

由此可知，数量级为： $\lim f(n)/g(n) = \lim (3n+2)/n = 3$ ，
相应的时间复杂度为： **$O(n)$**

【例4】 `for(i=1; i<n; i++)`
`for(j=1; j<=i; j++) {x++;}`

【分析】 双重循环，且内层循环次数变化。时间性能取决于内层循环次数的数量级。估算内层循环次数：

☞ $i=1$, 内层循环执行1次 ($j:1--1$)

☞ $i=2$, 内层循环执行2次 ($j:1--2$)

☞ $i=3$, 内层循环执行3次 ($j:1--3$)

☞ ...

☞ $i=n-1$, 内层循环执行 $n-1$ 次 ($j:1--n-1$)

■ 内层循环执行总次数： $n(n-1)/2$

■ 时间复杂度： $O(n^2)$

【例5】 $i=1;$
 $\text{while}(i<n) \ i*=2;$

【分析】 设循环执行次数为 k ，则 k 与 i 值如下表：

次数 k	1	2	3	4	k
i 值	2^1	2^2	2^3	2^4	2^k

- 假设执行最后一次（第 k 次）循环时有 $2^k=n$ ，则 $k=\log_2 n$ 。
- 即使 2^k 不是正好等于 n ， k 也接近 $\log_2 n$ 。
- 所以时间复杂度为： $O(\log_2 n)$ 。

小结

- 数据结构研究的内容，在软件开发中的地位；
- 数据元素：一个元素有1个，或多个数据项构成。
- 数据项
- 数据结构（3个方面）
 - ☞ 逻辑结构
 - ☞ 存储结构
 - ☞ 运算
- 算法
- 时间复杂度和空间复杂度

逻辑 结构

- ① 集合
- ② 线性结构
- ③ 树结构
- ④ 图（网）结构

存储 结构

- ① 顺序结构
- ② 链式结构
- ③ 索引结构
- ④ 散列结构

作业

■ P10

👉 1.5

推荐一本书

- **How to stop worrying and start living.**

👉 **USA, Dale Carnegie**

- **中译名：人性的优点**

Thank you !

