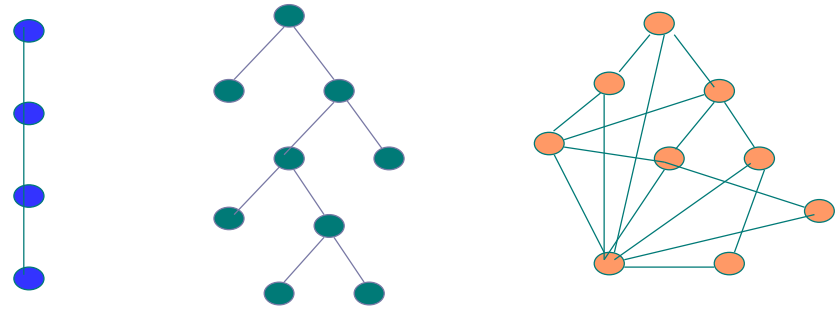
数据结构

Data Structures

华中科技大学计算机学院



1

华中科技大学计算机学院

主　讲: 许贵平

Email: gpxu@hust.edu.cn

助　教: 杨洋洋

QQ 群: HUST计卓-本硕博DS学习群（700757632）

微助教: 数据结构（2023级计卓本硕博）（MG989）

群名称:HUST_CS_DS_计卓/本硕博...
群　号:700757632

2

数据结构课程的意义与特征

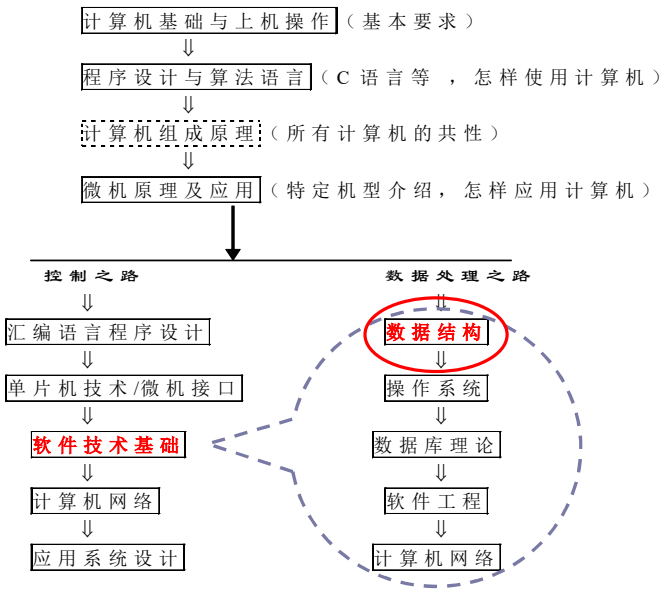
- 传统数据结构与算法课程的意义
  - ✓ 算法和数据结构是计算机科学的两大支柱, 是程序设计的基础
  - ✓ **Key abstractions** computer scientists and engineers use almost every day
  - ✓ **A big piece of what separates us from others**
  - ✓ **This is the class where you begin to think like a computer scientist**

——Prof. Dan Grossman @ UW

- **Parallelism is important**
- 数据结构与算法课程的特征
  - ✓ 计算机专业的一门综合性专业基础课
  - ✓ 是介于数学、计算机硬件、计算机软件的一门核心课程

3

计 算 机 系 列 课 程 之 间 的 联 系



Niklaus Wirth: (1934—2024) is a Swiss computer scientist, best known for designing several programming languages, including Pascal, and for pioneering several classic topics in software engineering. In 1984 he won the Turing Award for developing a sequence of innovative computer languages. In 1975 he wrote the book:

Data Structures + Algorithms = Programs

Niklaus Wirth

程序设计的实质

数据表示：将数据如何存储在计算机中

数据处理：处理数据，设计算法

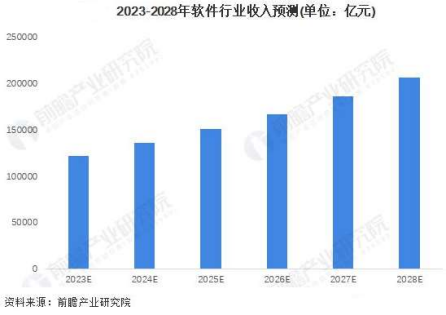
数据结构是程序设计的基础

程序设计是软件产业的基础

2023年中国软件行业全景图谱

—https://www.163.com/dy/article/IC6K2UM3051480KF.html

- **2022年中国软件业务收入10.8万亿元（同比增长11.2%）**
  - ✓ 近几年来，我国软件和信息技术服务业运行态势良好，收入和效益保持较快增长，吸纳就业人数稳步增加;产业向高质量方向发展步伐加快，结构持续调整优化，新的增长点不断涌现。
- **东部地区成为软件行业主力军，中西部地区增势突出。**
- **我国软件行业正处于成长期，市场规模增长较快，2028年市场规模预计超20万亿。**



国内软件行业发展的特点

- 市场巨大。
- 绿色产业。
- 国家大力支持。
- 高速增长。
- 高素质软件人才欠缺、待遇优厚。

国内软件行业存在的问题

- 资金投资相比美国依然不足
- 软件人才流失严重
- 软件盗版现象曾经比较严重
- 软件开发的工程化、规模化程度比较低
- 科技成果产业化的能力差，国产软件市场占有率不高
- 具有自主知识版权的系统软件，核心软件不足

（操作系统，数据库系统，企业ERP系统，…）

中国软件人才展望

- IT队伍结构失衡：既缺乏高级IT人才，也缺乏技能型、应用型信息技术人才，以及一大批能从事基础性工作的技术人员；
- 软件人才的“金字塔”型合理结构并未实现：既懂技术又懂管理的软件高级人才、系统分析及设计人员（软件工程师）、熟练程序员（软件蓝领）；
- 人才的地区分布差异也比较明显：一些发达城市人才有供过于求现象，而边远城市人才则普遍匮乏；
- 同学们应以学好数据结构为起点，打好坚实学科基础，从熟练程序员做起，最终成为具有创新能力的软件高级人才，为中国软件业添砖加瓦！

从软件到人工智能

- 信息化使软件融入各行各业，方方面面
- 智能化使人工智能时代加速到来
- 人工智能产业已上升为国家战略：
  - ✓ 十九大提出：推动互联网、大数据、人工智能和实体经济深度融合。
  - ✓ 二十大提出：推动战略性新兴产业融合集群发展，构建新一代信息技术、人工智能、生物技术、新能源、新材料、高端装备等一批新的增长引擎。
  - ✓ 2017年，国家发布了《新一代人工智能发展规划》和《新一代人工智能产业三年行动计划(2018-2020)》。

国家新一代人工智能发展规划

- 我国人工智能产业三步走战略目标
  - ✓ 第一步，到2020年人工智能总体技术和应用与世界先进水平同步。
  - ✓ 第二步，到2025年人工智能基础理论实现重大突破，部分技术与应用达到世界领先水平。
  - ✓ 第三步，到2030年人工智能理论、技术与应用总体达到世界领先水平，成为世界主要人工智能创新中心。

2025年湖北人工智能产业要实现千亿规模

凤凰網 湖北 湖北 > 正文

2025年湖北人工智能产业要实现千亿规模

2022年03月23日 20:44:17
来源：极目新闻

0人参与 0评论

极目新闻记者 陈倩

今日，极目新闻记者从湖北省经信厅获悉，《湖北省人工智能产业“十四五”发展规划》近日由该厅发布，规划提出，2025年湖北人工智能产业规模将达到1000亿元，在智能软硬件、自动驾驶汽车等多项重点项目上具有国际竞争力。

2025年湖北人工智能产业要进入全国第一方阵

《规划》提出，湖北省人工智能产业的发展目标是，到2025年，人工智能产业总体发展水平进入全国第一方阵，打造形成国内有重要影响力的人工智能创新核心区、应用先导区、产业集聚区。产业实力不断增强。基本形成涵盖核心技术、关键系统、支撑平台和智能应用的完备产业链和高端产业群，人工智能产业规模达到1000亿元。建设高水平武汉国家人工智能创新应用先导区、武汉国家新一代人工智能创新发展试验区。在智能制造、智慧医疗、智慧教育、智慧城市等重点领域打造100个以上应用示范场景，推广应用一批典型行业解决方案。培育引进5 - 10家龙头企业，培育100家以上专精特新“小巨人”企业，集聚500家以上创新企业。

数字中国建设整体布局规划——中共中央、国务院 2023.2.27

- 加快数字中国建设，对全面建设社会主义现代化国家、全面推进中华民族伟大复兴具有重要意义和深远影响。
- 到2025年，基本形成横向打通、纵向贯通、协调有力的一体化推进格局，数字中国建设取得重要进展。
- 到2035年，数字化发展水平进入世界前列，数字中国建设取得重大成就。
  - ✓ 构筑自立自强的数字技术创新体系。
  - ✓ 统筹布局一批数字领域学科专业点，培养创新型、应用型、复合型人才。
  - ✓ 推动高等学校、研究机构、企业等共同参与数字中国建设，建立一批数字中国研究基地。

数据结构课程工程认证目标

目标1

深刻理解四类基本数据结构，特别是线性类结构、树形类结构和图形类结构的逻辑结构、逻辑结构上定义的运算、物理结构、逻辑结构与物理结构对应关系、基于物理结构实现逻辑结构上定义的运算，建立基本的程序设计思维，能运用上述知识对实际应用问题设计方案和模型进行推理和验证。

目标2

深刻理解数据结构的逻辑表示、数据结构的物理表示、基于优化数据结构的算法设计，能利用上述知识和相关模型对计算机软件功能模块和计算机软件系统设计方案进行对比并选择合适的方案。

目标3

深刻理解数据结构逻辑结构、逻辑结构上定义的运算、物理结构、逻辑结构与物理结构对应关系、基于物理结构实现逻辑结构上定义的运算，具备运用上述知识解决实际问题，构建设计方案和模型的能力。

目标4

掌握算法时间复杂度和空间复杂度的衡量方法，并深刻理解查找、排序等经典算法的平均查找长度、稳定性等基本量化手段，能运用科学方法对计算机复杂工程问题解决过程中的关键影响因素进行分析，具备验证解决方案的合理性和对方案优化的能力。

目标5

掌握数据模型基本运算的实现方法，特别是数据结构与算法之间的内在联系，算法效率基本分析原理与基本分析方法以及算法效率与物理结构之间内在关系，　掌握递归程序设计技术，初步掌握高效算法的设计策略，能运用科学方法对计算机复杂工程问题的多种备选方案，根据约束条件并结合文献进行分析和研究，给出最优的解决模型和方法。

课程目标对毕业要求的支撑关系

目标1

1.3 能综合应用数学、自然科学、工程基础和专业知识，基于模型方法推演和分析计算机复杂工程问题。

目标2

1.4 能将软硬件知识、相关工程知识和模型方法用于计算机复杂工程问题解决方案进行比较和综合。

目标3

2.1 能综合运用数学、自然科学、工程科学以及计算机科学的基本原理，识别、判断和表达计算机复杂工程问题的关键环节，具有问题抽象能力。

目标4

2.2 对计算机复杂工程问题的一个系统或者过程，选择或建立一种模型，对关键影响因素进行分析。

目标5

2.3 对计算机复杂工程的多种可选方案，根据约束条件并结合文献进行分析和研究，并得出有效结论。

本课程的主要内容

注：第8章的内容不作要求。

1. 基本数据结构的定义、特性、运算与算法

   线性结构：线性表；栈，队列，双队列；数组，串。

   非线性结构：树，二叉树；图，网络。

2. 数据结构的存储结构与实现

   选择存储结构，设计算法。

3. 排序算法：内部排序，外部排序

4. 查找算法：顺序，折半，分块，哈希，二叉排序树等

5. 支持并行计算的数据结构与算法(Data structures & algorithms for problem solving with parallel computers)

6. 基本应用与综合应用

16

基本要求

1. 认真听课，参与课堂讨论；
2. 阅读教材与参考书；
3. 完成一定数量的书面作业；

成绩

期末考试（65％）+ 平时作业、考勤和课堂表现（35 ％）

平时作业（选择、判断、填空、简答、以及算法编写等题型）

1. Educoder平台线上完成，请注册：www.educouder.net
2. 注册后用邀请码YM3TXG进入课堂，注意每章作业截止期，不提供补交。

17

教材与参考书

《数据结构（C语言版）》

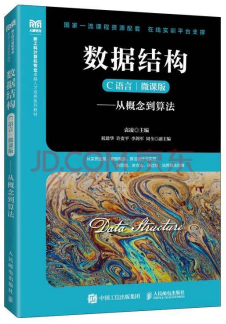
严蔚敏 吴伟民

清华大学出版社 2018

《数据结构（C语言|微课版）—从概念到算法》

袁凌 祝建华 许贵平 李剑军 周全

人民邮电出版社 2023.1



18

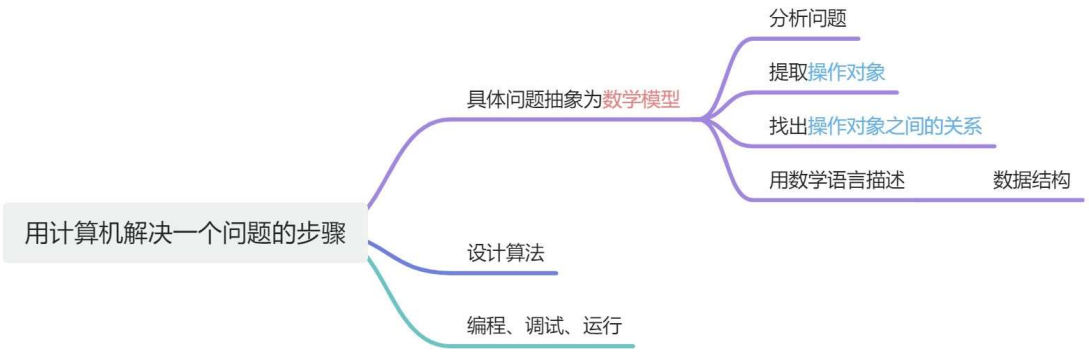
教材与参考书

1. 《数据结构题集》
   严蔚敏等　清华大学出版社 2021
2. **Parallel and Sequential Data Structures and Algorithms, CMU course 15-210**
   http://www.cs.cmu.edu/~15210/
3. **Data Structures and Algorithm Analysis in C,**
   Mark Allen Weiss, 机械工业出版社，2019

19

第一章　绪　　论

如何用计算机解决技术或应用问题？



20

第一章　绪　　论

1.1 什么是数据、结构(关系)、数据结构？

建立数学模型是分析具体问题的过程，包括：

- 分析具体问题中操作对象
- 找出这些对象间的关系，并用数学语言描述

数学模型分两类：

1）数值计算类：

例：

Love model

$$\begin{cases} \dfrac{dR}{dt} = aR + bJ \\ \dfrac{dJ}{dt} = cR + dJ \end{cases}$$

where

- ✓ *R* is Romeo's love for Juliet
- ✓ *J* is Juliet's love for Romeo (or hate if negative)
- ✓ *a, b, c, d* are constants that determine the "Romantic styles"

21

2）非数值计算类：

例1：5个整数组成的集合：

D={20, -5，66，15, 44}

其中:20, -5, 66等称为数据元素(元素)，

元素与元素之间关系是它们同属于集合D。

元素与元素间无直接关系。

例2：一列整数：(线性结构)

L=(20, -5，66，15, 44)

其中:元素与元素之间在L中是前后关系或线性关系。

L=(20, -5, 66, 15, 44)是一个线性表。

22

例3 一张登记表DL,也是一个线性表。

序号	姓　名	性 别	年 龄	
1	李　刚	男	25	记录1
2	王　霞	女	29	记录2
3	刘大海	男	40	记录3
4	李爱林	男	44	记录4

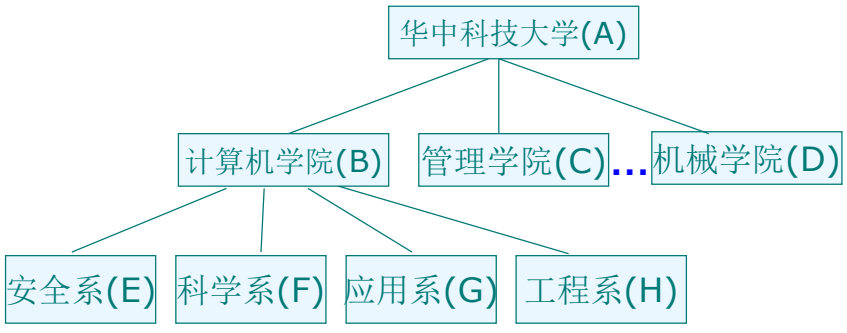
其中：姓名、性别、年龄是数据项(item)、数据域(field)；

(姓名，性别，年龄)是记录(record)，

C语言将“记录”(record)定义为”结构”(struct)。

23

例4　树状结构



树有许多应用：

- ✓ 家族谱系
- ✓ 索引结构
- ✓ 查询表达式
- ✓ 对弈树
- ✓ HTML DOM
- ✓ ……

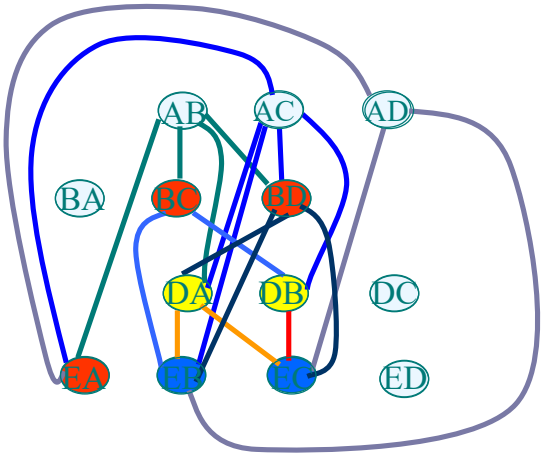
其中:A、B、C等是结点(node)；

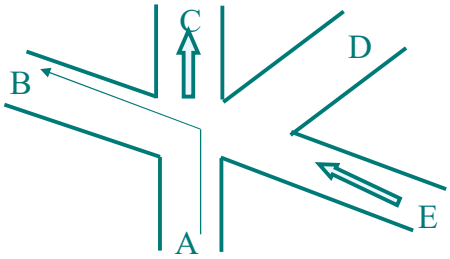
A与B，B与E，A与C等之间是层次关系或父子关系。

24

多叉路口交通灯管理问题

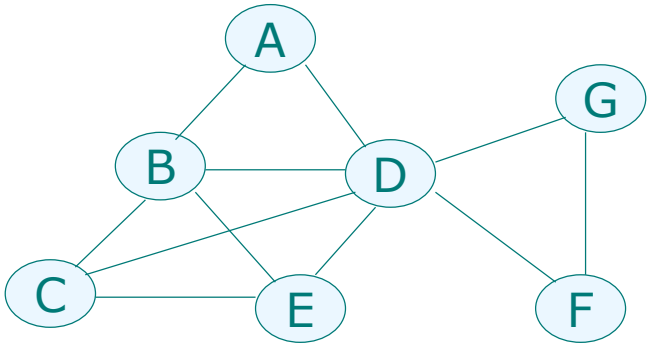
图





25

例5 图状结构



其中：A、B、C 等是顶点(vertex)，
图中任意两个顶点之间都可能有关系。

图可以表示许多应用：

- ✓ 交通路网
- ✓ 城市供水网
- ✓ 网络路由
- ✓ 社会关系网 (Social network)
- ✓ 概率因果关系推理
- ✓ ……

26

数据结构的学科定义:

是一门研究非数值计算程序设计问题中计算机操作对象以及它们之间关系和操作等的学科。

Data structures

"Clever" ways to organize information in order to enable efficient computation over that information.

27

1.2 基本概念和术语

1. 数据（data):

所有能输入到计算机中并被计算机程序加工、处理的符号的总称。

如：整数、实数、字符、声音、图象、图形等。

2. 数据元素(data element):

数据的基本单位。（元素、记录、结点、顶点）

在计算机程序中通常作为一个逻辑整体进行考虑和处理。

3. 数据项(data item):

是数据的不可分割的最小单位。如：姓名、年龄等。

一个数据元素可由一个或多个数据项组成。如:（姓名、年龄）

28

4. 数据对象(data object)

由性质相同(类型相同)的数据元素组成的集合。

数据对象是数据的一个子集。

例1 由4个整数组成的数据对象

D1={20, -30, 88, 45}

例2 由正整数组成的数据对象

D2={1, 2, 3, . . . }

例3 由26个字母组成的数据对象

D3={A, B, C, . . . , Z}

其中：D1，D3是有穷集，D2是无穷集。

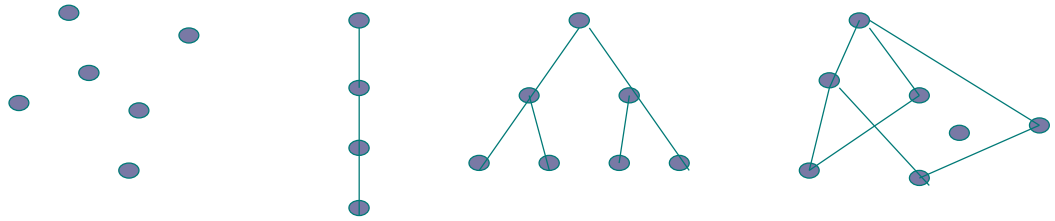
29

5. 数据结构 Data_Structure=（D, S）

相互之间存在一种或多种特定关系的数据元素的集合。

数据元素之间的关系称为结构。

四类基本结构：



集合
仅同属一个集合

线性结构
一对一

树形结构
一对多

图状结构
多对多

6. 数据的逻辑结构

抽象地反映各数据元素之间的逻辑关系。

数据结构(逻辑结构)分类:

- 数据结构
  - 线性结构
    1. 线性表
    2. 栈
    3. 队列, 双队列
    4. 数组
    5. 字符串
  - 非线性结构
    1. 树, 二叉树
    2. 图

31

7. 数据的存储结构

数据（逻辑）结构在计算机存储器中的实现或映象(mapping)。

存储结构也称为：存储表示, 物理结构, 物理表示。

存储结构分为：

- 顺序存储结构——借助元素在存储器中的相对位置来表示数据元素间的逻辑关系
- 链式存储结构——借助指示元素存储地址的指针表示数据元素间的逻辑关系

数据的逻辑结构与存储结构密切相关

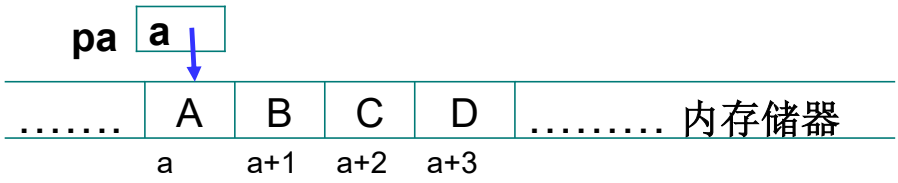
算法设计 ⟶ 逻辑结构

算法实现 ⟶ 存储结构

32

(1) 顺序存储结构(向量, 一维数组)

例. 线性表L＝（'A','B','C','D'）；



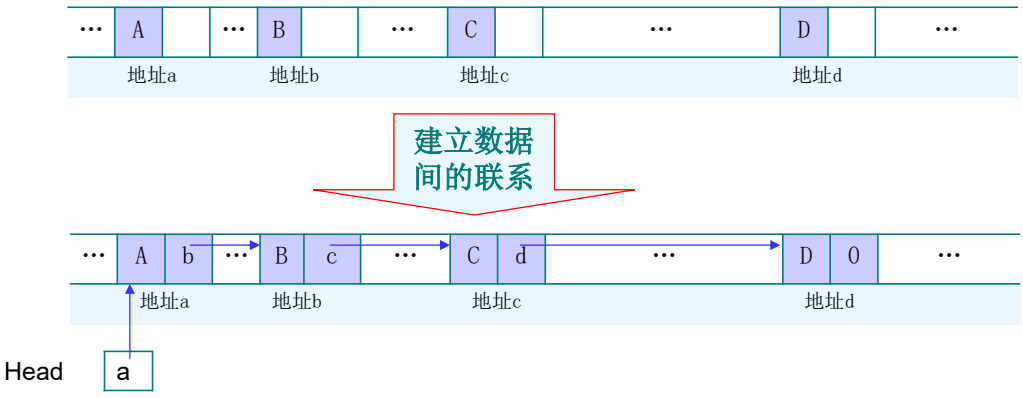
C语言实现方法：char a[4]={'A','B','C','D'};

'A'	'B'	'C'	'D'
a[0]	a[1]	a[2]	a[3]

33

(2) 非顺序存储结构(链接表)

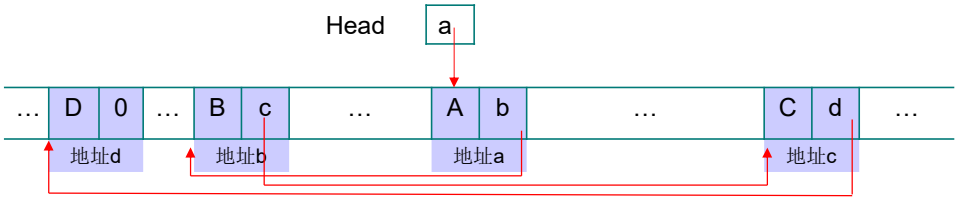
例. 单链表：分配不一定连续的空间



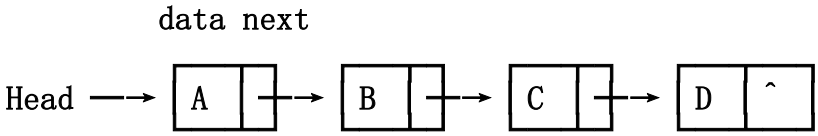
34

(2)非顺序存储结构(链接表)

例. 单链表：存放数据的空间顺序可任意



一般表达形式：



4个结点的单链表

35

8.数据类型（data type）

是一个值的集合(确定了数据的取值范围)、存储格式和定义在这个值上的一组操作(相关联的合法运算)的总称，编译系统支持的数据类型可认为是实现了的数据结构。

(1)原子类型(如：int,char,float,指针类型等)

(2)结构类型(如：数组,结构,联合体等)

9.抽象数据类型(Abstract Data Type)

用户以数学抽象的方式描述应用问题中的数据对象、数据的逻辑结构及其一组相关的操作（应用问题的数据模型）。

- ✓ 不考虑其在计算机的存储结构与操作的具体实现；
- ✓ 由基本的数据类型构成；
- ✓ 具有数据抽象与数据封装的特征。

36

抽象数据类型的描述方法

抽象数据类型可以用以下的三元组来表示：

ADT = (D, R, P)

数据对象　D上的关系集　D上的操作集

ADT 常用定义格式

```
ADT 抽象数据类型名{
        数据对象：<数据对象的定义>
        数据关系：<数据关系的定义>
        基本操作 ：<基本操作的定义>
    } ADT 抽象数据类型名
```

37

其中基本操作的定义格式为：

基本操作名（参数表）

初始条件：〈初始条件描述〉
操作结果：〈操作结果描述〉

赋值参数	只为操作提供输入值；
引用参数	以&打头，除可提供输入值外，还将返回操作结果。
初始条件	描述操作执行之前数据结构和参数应满足的条件，若不满足，则操作失败，并返回相应出错信息。
操作结果	说明了操作正常完成之后，数据结构的变化状况和应返回的结果。

38

例:定义抽象数据类型“复数”

ADT Complex {

数据对象: D = {e_1,e_2 | e_1,e_2∈RealSet }

数据关系: R = {<e_1,e_2> | e_1是复数的实数部分，e_2是复数的虚数部分}

基本操作:

AssignComplex(&Z, v_1, v_2)

操作结果: 分别用v_1、v_2为实部和虚部构造复数 Z。

DestroyComplex(&Z)

操作结果: 复数Z被销毁。

GetReal(Z, &realPart)

初始条件: 复数已存在。

操作结果: 用realPart返回复数Z的实部值。

……

} ADT Complex

自学：例1-6和例1-7关于抽象数据类型“三元组”的定义、表示和实现。

39

数据结构的三个方面：

- 数据的逻辑结构
  - 线性结构
    - 线性表
    - 栈
    - 队
  - 非线性结构
    - 树结构
    - 图结构
- 数据的存储结构
  - 顺序存储
  - 链式存储
- 数据的运算：查找、排序、插入、删除、修改等

什么是数据的逻辑结构，存储结构（物理结构）与数据的运算？

40

1.3 算法和算法分析

什么是算法？如何评判算法的好坏？

时间复杂度和空间复杂度如何表示？

41

1. 算法：是对特定问题求解步骤的一种描述，是指令的有限序列。

例. 求a[0]..a[n-1]中n个数的平均值(假定n>0)。

```
float average(float a[ ],int n)
{ int i; float s=0.0;    //累加器赋初值
  for (i=0; i<n; i++)
     s=s+a[i];           //a[i]累加到s中
  s=s/n;                 //计算平均值
  printf("ave=%f",s);    //输出
  return s;              //返回
 }                       //其中：a,n为输入量；s为输出量。
```

42

2. 算法的基本特征：有穷性、确定性、可行性、必有输出

(1) 有穷性：在有限步(或有限时间)之后算法终止。

```
例. { i=0; s=0;
      while (i<=10){ s+=i; i++; }
    }
```

(2) 确定性：无二义性。

(3) 可行性：算法中的操作是由已经实现的基本运算的有限次执行来实现的。

(4) 输入：有0或多个输入量。

(5) 输出：至少有一个输出量。

43

3. 算法设计要求（评价准则）：

(1) 正确性（四个层次）

a. 无语法错误；

b. 对n组输入产生正确结果；

c. 对特殊输入产生正确结果；

d. 对所有输入产生正确结果。

(2) 可读性：“算法主要是为了人的阅读与交流”。

(3) 健壮性

```
scanf("%d",&x);   y/=x; ? ? ?
```

(4) 高效与低存储量

44

下列描述不符合算法的什么特征和要求？

例1
```
void suanfa1( )
    { int i,s=0;
      for (i=0; i>=0; i++)        //死循环
        s++;                      //不能终止
     }
```

例2
```
float suanfa2( )
    { int x; float y;
      scanf("%d", &x);
      y=sqrt(x);                  //当x<0时,出错
      return(y);
     }
```

45

4. 算法的描述工具：

（1） 自然语言；

（2） 程序设计语言；

（3） 流程图（框图）；

（4） 伪码语言：是一种包括高级程序设计语言的三种基本结构（顺序、选择、循环）和自然语言成分的“面向读者”的语言。

（5） 类C：介于伪码语言和程序设计语言之间的一种表示形式。
保留了C语言的精华；不拘泥于C语言的语法细节；
同时也添加了一些C++的成分。

特点：便于理解、阅读；能方便的转换成C语言。

目的：便于简明扼要的描述算法；突出算法的思路。

46

〈1〉 预定义变量和类型：

```
#define TRUE     1
#define FALSE    0
#define OK       1
#define ERROR    0
#define INFEASIBLE     -1
#define OVERFLOW       -2
typedef int Status;      //一般用于算法函数的返回类型
typedef char ElemType;   // 此时char等价于Elemtype;
```

注意：抽象数据元素类型约定表示为ElemType，由用户自行定义。

47

```
typedef struct node { ElemType     elem;
                      struct node  *next;
                    } NODE, *LINK;
struct node m,*pm;
NODE  n,*pn;
LINK  p;                // 等价于 struct node *p;
```

48

〈2〉函数： 用以表示算法

```
函数类型  函数名（函数参数表）｛
          //算法说明
          语句序列
          ｝//函数名
```

注：

算法说明应包括功能说明，输入，输出；

为提高算法可读性，关键位置加以注释；

明确实参和形参的匹配规则，以便能正确使用算法函数。

〈3〉其他：Page 10

49

算法描述举例：

问题：设一维数组a[0..n-1]中已有n个整数,其中n为常数,
试设计算法:求a[]中的最大值。

算法基本思想：

1. maxai=a[0];
2. i=1;
3. 若i<=n-1,则：
   3.1 若a[i]>maxai,则 maxai=a[i];
   3.2 i++;
   3.3 转3
4. maxai为最大值。

50

C 函数(算法)

```
int a_maxint(int a[],int n)
  //求数组a中n个元素的最大值
{ int j,maxai=a[0];                    //最大值的初值
  for (j=1; j<=n-1; j++)
    if (a[j]>maxai)                    //比较元素大小
      maxai＝a[j];                     //新的最大值
  printf("maxai＝%d\n",maxai);         //输出
  return maxai;
}
```

51

算法效率：用据该算法编的程序在计算机上执行所消耗的时间来度量。

1.事后统计：利用计算机内记时功能

缺点：①必须先运行依据算法编制的程序

②所得时间统计量依赖于硬件、软件等环境因素

2.事前分析：一个高级语言程序在计算机上运行所耗的时间取决于：

①算法策略　②问题的规模　③程序语言

④编译程序产生机器代码质量　⑤机器执行指令速度

同一个算法分别用不同的语言、编译程序、计算机上运行，效率均不同，使用绝对时间单位衡量算法效率不合适。

52

5. 算法的时间复杂度：

基于算法中基本操作(或语句)重复执行次数总和所表示的算法执行时间度量。　　(设n为求解的问题的规模)

基本操作(或语句)执行次数总和称为语句频度，记作f(n)；

(渐近)时间复杂度记作T(n)，则：　T(n)=O(f(n))

渐进符号（O）的定义：

当且仅当存在一个正的常数 c，使得对所有的 $n \geq n_0$ ，有 $T(n) \leq cf(n)$，则：$T(n) = O(f(n))$

常数阶	对数阶	线性阶	线性对数阶	平方阶	立方阶	…	K次方阶	指数阶
O(1)	$O(\log_2 n)$	O(n)	$O(n \log_2 n)$	$O(n^2)$	$O(n^3)$		$O(n^k)$	$O(2^n)$

53

Asymptotic Notations

$T(N) = O(f(N))$ if there are positive constants c and n_0 such that

$$T(N) \leq c \cdot f(N) \text{ for all } N \geq n_0.$$

$T(N) = \Omega(g(N))$ if there are positive constants c and n_0 such that

$$T(N) \geq c \cdot g(N) \text{ for all } N \geq n_0.$$

$T(N) = \Theta(h(N))$ if and only if $T(N) = O(h(N))$ and $T(N) = \Omega(h(N))$.

$T(N) = o(p(N))$ if $T(N) = O(p(N))$ and $T(N) \neq \Theta(p(N))$.

Note:

- $2N + 3 = O(N) = O(N^{k \geq 1}) = O(2^N) = \cdots$, We shall always take the smallest $f(N)$.
- $2^N + N^2 = \Omega(2^N) = \Omega(N^2) = \Omega(N) = \Omega(1) = \cdots$, We shall always take the largest $g(N)$.

Rules of Asymptotic Notation

- If $T_1(N) = O(f(N))$ and $T_2(N) = O(g(N))$, then
  - (a) $T_1(N) + T_2(N) = \max(O(f(N)), O(g(N)))$,
  - (b) $T_1(N) * T_2(N) = O(f(N) * g(N))$.
- If $T(N)$ is a polynomial of degree k, then $T(N) = \Theta(N^k)$.
- $\log^k N = O(N)$ for any constant k. (logarithms grow very slowly.)

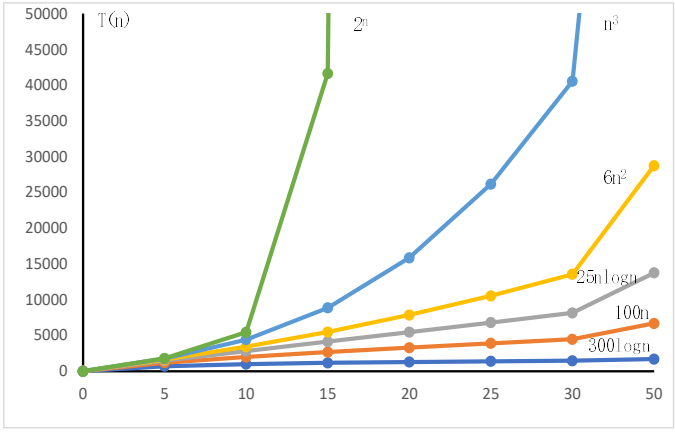
Note: When compare the complexities of two programs asymptotically, make sure that N is sufficiently large.

For example, suppose that $T_{p1}(N) = 10^6 N$ and $T_{p2}(N) = N^2$. Although it seems that $\Theta(N^2)$ grows faster than $\Theta(N)$, but if $N < 10^6$, P2 is still faster than P1.

		Input size n					
Time	Name	1	2	4	8	16	32
1	constant	1	1	1	1	1	1
$\log n$	logarithmic	0	1	2	3	4	5
n	linear	1	2	4	8	16	32
$n \log n$	log linear	0	2	8	24	64	160
n^2	quadratic	1	4	16	64	256	1024
n^3	cubic	1	8	64	512	4096	32768
2^n	exponential	2	4	16	256	65536	4294967296
$n!$	factorial	1	2	24	40326	2092278988000	26313×10^{33}

难解问题

算法不同阶的时间复杂度



常数阶	对数阶	线性阶	线性对数阶	平方阶	立方阶	…	K次方阶	指数阶
O(1)	$O(\log_2 n)$	O(n)	$O(n \log_2 n)$	$O(n^2)$	$O(n^3)$		$O(n^k)$	$O(2^n)$

57

	Time for $f(n)$ instructions on a 10^9 instr/sec computer						
n	$f(n)=n$	$n\log_2 n$	n^2	n^3	n^4	n^{10}	2^n
10	.01μs	.03μs	.1μs	1μs	10μs	10sec	1μs
20	.02μs	.09μs	.4μs	8μs	160μs	2.84hr	1ms
30	.03μs	.15μs	.9μs	27μs	810μs	6.83d	1sec
40	.04μs	.21μs	1.6μs	64μs	2.56ms	121.36d	18.3min
50	.05μs	.28μs	2.5μs	125μs	6.25ms	3.1yr	13d
100	.10μs	.66μs	10μs	1ms	100ms	3171yr	$4*10^{13}$yr
1,000	1.00μs	9.96μs	1ms	1sec	16.67min	$3.17*10^{13}$yr	$32*10^{283}$yr
10,000	10μs	130.03μs	100ms	16.67min	115.7d	$3.17*10^{23}$yr	
100,000	100μs	1.66ms	10sec	11.57d	3171yr	$3.17*10^{33}$yr	
1,000,000	1.0ms	19.92ms	16.67min	31.71yr	$3.17*10^{7}$yr	$3.17*10^{43}$yr	

μs = microsecond = 10^{-6} seconds
ms = millisecond = 10^{-3} seconds
sec = seconds
min = minutes
hr = hours
d = days
yr = years

Worst-Case Analysis

- In general, we are interested in three types of performance
  - Best-case / Fastest
  - Average-case
  - Worst-case / Slowest
- For some algorithms, worst case occurs often, average case is often roughly as bad as the worst case. So generally, worse case running time.
- When determining worst-case, we tend to be pessimistic
  - If there is a conditional, count the branch that will run the slowest
  - This will give a loose bound on how slow the algorithm may run

59

例1 { int s;

scanf(“%d”,&s);

s++;

printf(“%d”,s);

}

其中，语句频度为：f(n)=f(1)=3

时间复杂度为：T(n)=O(f(n))=O(3)=O(1)

O(1)称为常量阶/常量数量级

60

例2 分析下面的算法

```
void sum(int a[],int n)
{ int s=0,i;              // 1次
  for(i=0; i<n; i++)      // n次(严格为2*(n+1)次)
    s=s+a[i];             // n次
  printf("%d",s);         // 1次
 }
```

其中，语句频度为：f(n)=1+n+n+1

时间复杂度为：T(n)=O(f(n))=O(2n+2)=O(n)

O(n)称为线性阶/线性数量级

61

例3 分析下面的算法

```
1. void sum(int m,int n)
2. { int i,j,s=0;                  // 1次
3.   for(i=1; i<=m; i++)           // m次
4.     { for(j=1; j<=n; j++)       // m*n次
5.         s++;                    // m*n次
6.       printf("%d",s);           // m次
7.     }
8.  }
```

其中：f(m,n)=1+m+2*m*n+m=2mn+2m+1

当m=n时，$f(n)=2n^2+2n+1$

$T(n)=O(f(n))=O(2n^2+2n+1)=O(n^2)$

$O(n^2)$ 称为平方阶/平方数量级

62

例4 分析下面的算法：当n=5,指出输出结果

```
1.  void sum(int n)
2.  { int i,j,s=0;                    // 1次
3.    for(i=1; i<=n; i++)             // n次
4.     { for(j=1; j<=i; j++)          // ?次
5.        s++;                        // ?次
6.       printf("%d",s);             // n次
7.     }
8.  }
```

其中：第4行的次数为 1+2+...+n=n(1+n)/2

第5行的次数为 1+2+...+n=n(1+n)/2

$f(n)=1+n+n(n+1)+n=n^2+3n+1$

$T(n)=O(f(n))=O(n^2)$ 为平方阶

63

RECURSIONS

How about iterative Fibonacci?

Fibonacci number: Fib(0) = Fib(1) = 1, Fib(*n*) = Fib(*n*–1) + Fib(*n*–2)

```
long int  Fib ( int  N )        /* T ( N ) */
{
        if  ( N <= 1 )          /* O( 1 ) */
           return  1;           /* O( 1 ) */
        else
           return  Fib( N – 1 )  +  Fib( N – 2 );
}
                    /*T(N –1)*/  /*O(1)*/   /*T(N –2)*/
```

$T(N) = T(N-1) + T(N-2) + 2 \geq \text{Fib}(N)$

Proof by induction

$\left(\frac{3}{2}\right)^N \leq \text{Fib}(N) \leq \left(\frac{5}{3}\right)^N \Rightarrow T(N)$ grows exponentially

Big-Oh Analysis of Recursive Algorithms

❑ **Common recurrence relations in analysis of algorithms**

- ✓ $T(N) = T(N-1) + \Theta(1) \Rightarrow T(N) = O(N)$
- ✓ $T(N) = T(N-1) + \Theta(N) \Rightarrow T(N) = O(N^2)$
- ✓ $T(N) = T(N/2) + \Theta(1) \Rightarrow T(N) = O(\log_2 N)$
- ✓ $T(N) = 2T(N/2) + \Theta(N) \Rightarrow T(N) = O(N\log_2 N)$
- ✓ **How do you get these? Just expand the right side and count!**

❑ **Note: Multiplicative constants matter in recurrence relations**

- ✓ $T(N) = 4T(N/2) + \Theta(N)$ is $O(N^2)$, not $O(N\log_2 N)$!

6. 算法的空间复杂度：

执行算法所需存储空间的大小，记作：

$$s(n)=o(f(n))$$

算法的存储量包括：

（1）输入数据所占空间；

（2）程序本身所占空间；

（3）辅助变量所占空间（一般分析只针对此项）。

原地工作：

辅助变量所占的额外空间相对于输入数据而言为常数。

Compare the Algorithms: MCSS Problem

Given (possibly negative) integers $A_1, A_2, \ldots, A_N$, find the maximum value of $\sum_{k=i}^{j} A_k$.

Max sum is 0 if all the integers are negative.

Algorithm 1

```
int  MaxSubsequenceSum ( const int A[ ],  int  N ) {
        int  ThisSum,  MaxSum,  i,  j,  k;
/* 1*/  MaxSum = 0;                    /* initialize the maximum sum */
/* 2*/  for( i = 0; i < N; i++ )       /* start from A[ i ] */
/* 3*/      for( j = i; j < N; j++ ) {     /* end at A[ j ] */
/* 4*/          ThisSum = 0;
/* 5*/          for( k = i; k <= j; k++ )
/* 6*/              ThisSum += A[ k ];     /* sum from A[ i ] to A[ j ] */
/* 7*/          if ( ThisSum > MaxSum )
/* 8*/              MaxSum = ThisSum;    /* update max sum */
          } /* end for-j and for-i */
/* 9*/  return  MaxSum;
}
```

Brute Force

$T(N) = O(N^3)$

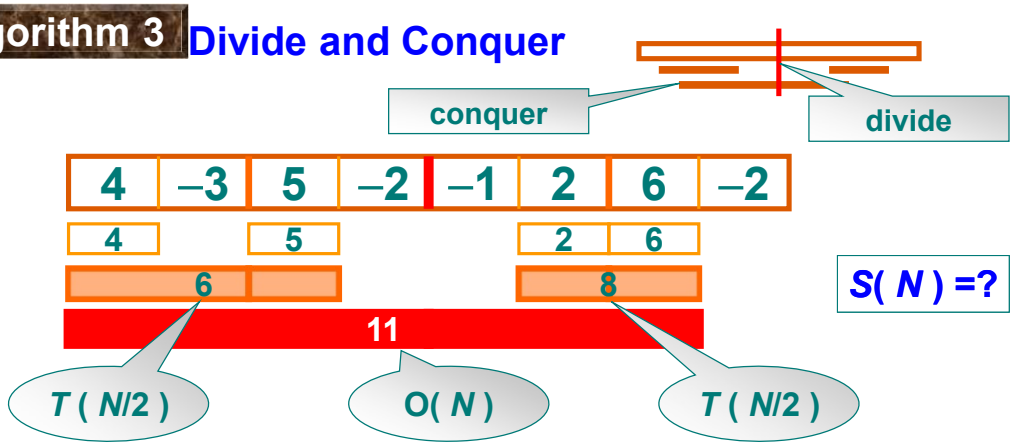
$S(N) = ?$

Algorithm 2

```
int  MaxSubsequenceSum ( const int A[ ],  int  N ) {
        int  ThisSum,  MaxSum,  i,  j;
/* 1*/  MaxSum = 0;                    /* initialize the maximum sum */
/* 2*/  for( i = 0; i < N; i++ ) {     /* start from A[ i ] */
/* 3*/      ThisSum = 0;
/* 4*/      for( j = i; j < N; j++ ) {     /* end at A[ j ] */
/* 5*/          ThisSum += A[ j ];     /* sum from A[ i ] to A[ j ] */
/* 6*/          if ( ThisSum > MaxSum )
/* 7*/              MaxSum = ThisSum;      /* update max sum */
          } /* end for-j */
        } /* end for-i */
/* 8*/  return  MaxSum;
}
```

$T(N) = O(N^2)$

Algorithm 3 Divide and Conquer



$T(N) = 2\,T(N/2) + c\,N, \quad T(1) = O(1)$

$= 2\,[2\,T(N/2^2) + c\,N/2] + c\,N = 2^2 T(N/2^2) + c2N = \ldots$

$= 2^k\,O(1) + c\,k\,N \quad$ where $N/2^k = 1$

$= O(N \log_2 N)$

Also true for $N \neq 2^k$

Algorithm 4 On-line Algorithm

```
int MaxSubsequenceSum( const int  A[ ],  int  N )
{
          int  ThisSum, MaxSum, j;
/* 1*/    ThisSum = MaxSum = 0;
/* 2*/    for ( j = 0; j < N; j++ ) {
/* 3*/        ThisSum += A[ j ];
/* 4*/        if  ( ThisSum > MaxSum )
/* 5*/            MaxSum = ThisSum;
/* 6*/        else if ( ThisSum < 0 )        // !!!
/* 7*/            ThisSum = 0;
          } /* end for-j */
/* 8*/    return MaxSum;
}
```

At any point in time, the algorithm can correctly give an answer to the subsequence problem for the data it has already read.

$T(N) = O(N)$ $S(N) = O(1)$

A[ ] is scanned once only.

Running times of several algorithms for maximum subsequence sum (in seconds)

Algorithm		1	2	3	4
Time		$O(N^3)$	$O(N^2)$	$O(N \log N)$	$O(N)$
Input Size	N =10	0.00103	0.00045	0.00066	0.00034
	N =100	0.47015	0.01112	0.00486	0.00063
	N =1,000	448.77	1.1233	0.05843	0.00333
	N =10,000	NA	111.13	0.68631	0.03042

Note: The time required to read the input is not included.

讨论：

- n个整数存于一维数组R中，设计一个时空高效算法使R中保存的序列循环左移p(0<p<n)个位置，即

$$(x_0, x_1, \cdots, x_{n-1}) \Rightarrow (x_p, x_{p+1}, \cdots, x_{n-1}, x_0, x_1, \cdots, x_{p-1})$$

$$\Uparrow \qquad \Uparrow$$

$$(x_0, x_1, \cdots, x_{n-1}) \Rightarrow (x_{n-1}, x_{n-2}, \cdots, x_p, x_{p-1} \cdots, x_1, x_0)$$

- ✓ 算法一：每次移一位，循环p次，平均T(n)=O(n²)；S(n)=O(1)
- ✓ 算法二：构造等长辅助数组，T(n)=O(n)；S(n)=O(n)
- ✓ 算法三：逆置法，T(n)=O(n)；S(n)=O(1)

72

本章小结

- 基本概念和术语：
  - ✓ 数据、数据元素、数据项、数据结构。特别是数据结构的逻辑结构、存储结构及数据运算的含义及其相互关系。数据结构的两大类逻辑结构和两种常用的存储表示方法。
- 算法及算法分析
  - ✓ 算法、算法的时间复杂度和空间复杂度、最坏的和平均时间复杂度等概念，算法描述和算法分析的方法、对一般的算法要能分析出时间复杂度与空间复杂度。

Note: Shorter code does not always mean better code. ---M. A. Weiss

73

- 教学的基本框架

方面 / 层次	数据表示	数据处理
抽象	逻辑结构	基本运算
实现	存储结构	算法
评价	不同数据结构的比较及算法分析	

74

本章作业： Educoder平台线上完成

课外思考：本章习题

一: 1, 5-13, 15

二

三: 2-10, 14

Readings:

(1) 教材第一章及本讲义；

(2) 复习C语言，重点理解指针、结构类型和递归等概念。

75