

# 数据结构实验指导书

## 一 实验步骤

随之计算机性能的提高，它所面临的软件开发的复杂度也日趋增加。然而，编制一个 10,000 行的程序的难度绝不仅仅是一个 5,000 行的程序两倍，因此软件开发需要系统的方法。一种常用的软件开发方法，是将软件开发过程划分为分析、设计、实现和维护四个阶段。虽然数据结构课程中的实习题的复杂度远不如(从实际问题中提出来的)一个“真正的，，软件，但为了培养一个软件工作者所应具备的科学工作的方法和作风，我们制订了如下所述完成实习的五个步骤：

### (一)问题分析和任务定义

通常，实习题目的陈述比较简洁，或者说是模棱两可的含义。因此，在进行设计之前，首先应该充分地分析和理解问题，明确问题要求做什么?限制条件是什么。注意：本步骤强调的是做什么?而不是怎么做。对问题的描述应避开算法和所涉及的数据类型，而是对所需完成的任务作出明确的回答。例如：输入数据的类型、值的范围以及输入的形式；输出数据的类型、值的范围及输出的形式；若是会话式的输入，则结束标志是什么?是否接受非法的输入?对非法输入的回答方式是什么等。这一步还应该为调试程序准备好测试数据，包括合法的输入数据和非法形式的输入数据。

### (二)数据类型和系统设计

在设计这一步骤中需分逻辑设计和详细设计两步实现。逻辑设计指的是，对问题描述中涉及的操作对象定义相应的数据类型，并按照以数据结构为中心的原则划分模块，定义主程序模块和各抽象数据类型；详细设计则为定义相应的存储结构并写出各函数的伪码算法。在这个过程中，要综合考虑系统功能，使得系统结构清晰、合理、简单和易于调试，抽象数据类型的实现尽可能做到数据封装，基本操作的规格说明尽可能明确具体。作为逻辑设计的结果，应写出每个抽象数据类型的定义(包括数据结构的描述和每个基本操作的规格说明)，各个主要模块的算法，并画出模块之间的调用关系图。详细设计的结果是对数据结构和基本操作的规格说明作出进一步的求精，写出数据存储结构的类型定义，按照算法书写规范用类 c 语言写出函数形式的算法框架。在求精的过程中，应尽量避免陷入语言细节，不必过早表述辅助数据结构和局部变量。

### (三)编码实现和静态检查

编码是把详细设计的结果进一步求精为程序设计语言程序。程序的每行不要超过 60 个字符。每个函数体，即不计首部和规格说明部分，一般不要超过 40 行，最长不得超过 60 行，否则应该分割成较小的函数。要控制 if 语句连续嵌套的深度。其他要求参见第一篇的

算法书写规范。如何编写程序才能较快地完成调试是特别要注意的问题。对于编程很熟练的读者,如果基于详细设计的伪码算法就能直接在键盘上输入程序的话,则可以不必要用笔在纸上写出编码,而将这一步的工作放在上机准备之后进行,即在上机调试之前直接用键盘输入。然而,不管你是否写出编码的程序,在上机之前,认真的静态检查是必不可少的。多数初学者在编好程序后处于以下两种状态之一:一种是对自己的“精心作品”的正确性确信不疑;另一种是认为上机前的任务已经完成,纠查错误是上机的工作。这两种态度是极为有害的。事实上,非训练有素的程序设计者编写的程序长度超过 50 行时,极少不含有除语法错误以外的错误。上机动态调试决不能代替静态检查,否则调试效率将是极低的。

静态检查主要有两种方法,一是用一组测试数据手工执行程序(通常应先分模块检查);二是通过阅读或给别人讲解自己的程序而深入全面地理解程序逻辑,在这个过程中再加入一些注解和断言。如果程序中逻辑概念清楚,后者将比前者有效。

## (四)上机准备和上机调试

上机准备包括以下几个方面:

(1)高级语言文本(体现于编译程序用户手册)的扩充和限制。例如,常用的 Borland C(C++)和 Microsoft C(c++)与标准 c(C++)的差别,以及相互之间的差别。

(2)如果使用 C 或 c++语言,要特别注意与教科书的类 c 语言之间的细微差别。

(3)熟悉机器的操作系统和语言集成环境的用户手册,尤其是最常用的命令操作,以便顺利进行上机的基本活动。

(4)掌握调试工具,考虑调试方案,设计测试数据并手工得出正确结果。“磨刀不误砍柴工”。计算机各专业的学生应该能够熟练运用高级语言的程序调试器 DEBUG 调试程序。

上机调试程序时要带一本高级语言教材或手册。调试最好分模块进行,自底向上,即先调试低层函数。必要时可以另写一个调用驱动程序。这种表面上麻烦的工作实际上可以大大降低调试所面临的复杂性,提高调试工作效率。

在调试过程中可以不断借助 DEBUG 的各种功能,提高调试效率。调试中遇到的各种异常现象往往是预料不到的,此时不应“冥思苦想”,而应动手确定疑点,通过修改程序来证实它或绕过它。调试正确后,认真整理源程序及其注释,印出带有完整注释的且格式良好的源程序清单和结果。

## (五)总结和整理实习报告

## 二、实习报告规范

实习报告的开头应给出题目、班级、姓名、学号和完成日期,并包括以下七个内容:

### 1. 需求分析

以无歧义的陈述说明程序设计的任务,强调的是程序要做什么?明确规定:

(1)输入的形式和输入值的范围;

- (2)输出的形式;
- (3)程序所能达到的功能;
- (4)测试数据: 包括正确的输入及其输出结果和含有错误的输入及其输出结果。

## 2. 概要设计

说明本程序中用到的所有抽象数据类型的定义、主程序的流程以及各程序模块之间的层次(调用)关系。

## 3. 详细设计

实现概要设计中定义的所有数据类型, 对每个操作只需要写出伪码算法; 对主程序和其他模块也都需要写出伪码算法(伪码算法达到的详细程度建议为: 按照伪码算法可以在计算机键盘直接输入高级程序设计语言程序); 画出函数的调用关系图。

## 4. 调试分析

内容包括:

- (1)调试过程中遇到的问题是怎样解决的以及对设计与实现的回顾讨论和分析;
- (2)算法的时空分析(包括基本操作和其他算法的时间复杂度和空间复杂度的分析)和改进设想;
- (3)经验和体会等。

## 5. 用户使用说明

说明如何使用你编写的程序, 详细列出每一步的操作步骤。

## 6. 测试结果

列出你的测试结果, 包括输入和输出。这里的测试数据应该完整和严格, 最好多于需求分析中所列。

## 7. 附录

带注释的源程序。如果提交源程序软盘, 可以只列出程序文件名的清单。

在实验一中提供了实习报告实例。值得注意的是, 实习报告的各种文档资料, 如: 上述中的前三部分要在程序开发的过程中逐渐充实形成, 而不是最后补写(当然也可以应该最后用实验报告纸誊清或打印)。

## 三、实验内容

### 实验一 线性表及其应用

本次实习的主要目的在于帮助学生熟练掌握线性表的基本操作在两种存储结构上的实现，其中以各种链表的操作和应用作为重点内容。

#### 1. 1 运动会分数统计

##### 【问题描述】

参加运动会的  $n$  个学校编号为  $1 \sim n$ 。比赛分成  $m$  个男子项目和  $w$  个女子项目，项目编号分别为  $1 \sim m$  和  $m+1 \sim m+W$ 。由于各项目参加人数差别较大，有些项目取前五名，得分顺序为 7, 5, 3, 2, 1；还有些项目只取前三名，得分顺序为 5, 3, 2。写一个统计程序产生各种成绩单和得分报表。

##### 【基本要求】

产生各学校的成绩单，内容包括各校所取得的每项成绩的项目号、名次(成绩)、姓名和得分；产生团体总分报表，内容包括校号、男子团体总分、女子团体总分和团体总分。

##### 【测试数据】

对于  $n=4$ ,  $m=3$ ,  $w=2$ , 编号为奇数的项目取前五名，编号为偶数的项目取前三名，设计一组实例数据。

##### 【实现提示】

可以假设,  $n \leq 20$ ,  $m \leq 30$ ,  $w \leq 20$ , 姓名长度不超过 20 个字符。每个项目结束时，将其编号、类型符(区分取前五名还是前三名)输入，并按名次顺序输入运动员姓名、校名(和成绩)。

##### 【选作内容】

允许用户指定某项目采取其他名次取法。

#### ◆1. 2 约瑟夫环

##### 【问题描述】

约瑟夫(Joseph)问题的一种描述是：编号为  $1, 2, \dots, n$  的  $n$  个人按顺时针方向围坐一圈，每人持有一个密码(正整数)。一开始任选一个正整数作为报数上限值  $m$ ，从第一个人开始按顺时针方向自 1 开始顺序报数，报到  $m$  时停止报数。报  $m$  的人出列，将他的密码作为新的  $m$  值，从他在顺时针方向上的下一个人开始重新从 1 报数，如此下去，直至所有人全部出列为止。试设计一个程序求出出列顺序。

##### 【基本要求】

利用单向循环链表存储结构模拟此过程，按照出列的顺序印出各人的编号。

##### 【测试数据】

$m$  的初值为 20;  $n=7$ , 7 个人的密码依次为: 3, 1, 7, 2, 4, 8, 4, 首先  $m$  值为 6(正确的列顺序应为 6, 1, 4, 7, 2, 3, 5)。

##### 【实现提示】

程序运行后，首先要求用户指定初始报数上限值，然后读取各人的密码。可设  $n \leq 30$ 。此题所用的循环链表中不需要“头结点”，请注意空表和非空表的界限。

##### 【选作内容】

向上述程序中添加在顺序结构上实现的部分。

## 实验二 栈和队列及其应用

仅仅认识到栈和队列是两种特殊的线性表是远远不够的，本次实验的目的在于使读者深入了解栈和队列的特性，以便在实际问题背景下灵活运用他们；同时还将巩固对这两种结构的构造方法的理解。

### 2.1 迷宫问题

#### 【问题描述】

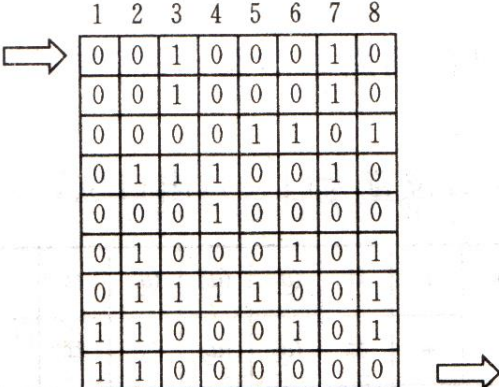
以一个  $m \times n$  的长方阵表示迷宫，0 和 1 分别表示迷宫中的通路和障碍。设计一个程序，对任意设定的迷宫，求出一条从入口到出口的通路，或得出没有通路的结论。

#### 【基本要求】

首先实现一个以链表作存储结构的栈类型，然后编写一个求解迷宫的非递归程序；求得的通路以三元组  $(i, j, d)$  的形式输出，其中：  $(i, j)$  指示迷宫中的一个坐标，  $d$  表示走到下一坐标的方向。如：对于下列数据的迷宫，输出的一条通路为：  $(1, 1, 1)$ ，  $(1, 2, 2)$ ，  $(2, 2, 2)$ ，  $(3, 2, 3)$ ，  $(3, 1, 2)$ ， ...。

#### 【测试数据】

迷宫的测试数据如下：左上角  $(1, 1)$  为入口右下角  $(8, 9)$  为出口。



	1	2	3	4	5	6	7	8
1	0	0	1	0	0	0	1	0
2	0	0	1	0	0	0	1	0
3	0	0	0	0	1	1	0	1
4	0	1	1	1	0	0	1	0
5	0	0	0	1	0	0	0	0
6	0	1	0	0	0	1	0	1
7	0	1	1	1	1	0	0	1
8	1	1	0	0	0	1	0	1
9	1	1	0	0	0	0	0	0

#### 【实现提示】

计算机解迷宫通常用的是“穷举求解”方法，即从入口出发，顺着某一个方向进行探索，若能走通，则继续往前走；否则沿着原路退回，换一个方向继续探索，直至出口位置，求得一条通路。假如所有可能的通路都探索到而未能到达出口，则所设定的迷宫没有通路。

可以二维数组存储迷宫数据，通常设定入口点的下标为  $(1, 1)$ ，出口点的下标为  $(n, n)$ 。为处理方便起见，可在迷宫的四周加一圈障碍。对于迷宫中任一位置，均可约定有东、南、西、北四个方向可通。

#### 【选作内容】

- (1) 编写递归形式的算法，求得迷宫中所有可能的通路；
- (2) 以方阵形式输出迷宫及其通路。

## 实验三 数组和广义表

本实习单元是作为从线性结构到非线性结构的过渡来安排的。数组和广义表可以看成其元素本身也是自身结构(递归结构)的线性表。广义表本质上是一种层次结构，自顶向下识

别并建立一个广义表的操作，可视为某种树的遍历操作：遍历逻辑的(或符号形式的)结构，访问动作是建立一个结点。稀疏矩阵的十字链表存储结构也是图的一种存储结构。由此可见，这个实习单元的训练具有承上启下的作用。希望读者能深入研究数组的存储表示和实现技术，熟悉广义表的存储结构的特性。

### 3.1 稀疏矩阵运算器

#### 【问题描述】

稀疏矩阵是指那些多数元素为零的矩阵。利用“稀疏”特点进行存储和计算可以大大节省存储空间，提高计算效率。实现一个能进行稀疏矩阵基本运算的运算器。

#### 【基本要求】

以“带行逻辑链接信息”的三元组顺序表表示稀疏矩阵，实现两个矩阵相加、相减和相乘的运算。稀疏矩阵的输入形式采用三元组表示，而运算结果的矩阵则以通常的阵列形式列出。

#### 【测试数据】

$$\begin{bmatrix} 10 & 0 & 0 \\ 0 & 0 & 9 \\ -1 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 1 & 0 & -3 \end{bmatrix} = \begin{bmatrix} 10 & 0 & 0 \\ 0 & 0 & 8 \\ 0 & 0 & -3 \end{bmatrix}$$

$$\begin{bmatrix} 10 & 0 \\ 0 & 9 \\ -1 & 0 \end{bmatrix} - \begin{bmatrix} 0 & 0 \\ 0 & -1 \\ 1 & -3 \end{bmatrix} = \begin{bmatrix} 10 & 0 \\ 0 & 10 \\ -2 & 3 \end{bmatrix}$$

$$\begin{bmatrix} 4 & -3 & 0 & 0 & 1 \\ 0 & 0 & 0 & 8 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 70 \end{bmatrix} \times \begin{bmatrix} 3 & 0 & 0 \\ 4 & 2 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & -6 & 0 \\ 8 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

#### 【实现提示】

1. 首先应输入矩阵的行数和列数，并判别给出的两个矩阵的行、列数对于所要求作的运算是否相匹配。可设矩阵的行数和列数均不超过 20。
2. 程序可以对三元组的输入顺序加以限制，例如，按行优先。注意研究教科书 5.3.2 节中的算法，以便提高计算效率。
3. 在用三元组表示稀疏矩阵时，相加或相减所得结果矩阵应该另生成，乘积矩阵可用二维数组存放。

#### 【选作内容】

1. 按教科书 5.3.2 节中的描述方法，以十字链表表示稀疏矩阵。
2. 增添矩阵求逆的运算，包括不可求逆的情况。在求逆之前，先将稀疏矩阵的内部表示改为十字链表。

## 实验四 树、图及其应用

树和图是两种应用极为广泛的数据结构，也是这门课程的重点。它们的特点在于非线性。本实习单元继续突出了数据结构加操作的程序设计观点，但根据这两种结构的非线性特点，将操作进一步集中在遍历操作上，因为遍历操作是其他众多操作的基础。本实习单元还希望达到熟悉各种存储结构的特性，以及如何应用树和图结构解决具体问题(即原理与应用

的结合)等目的。

◆4. 0 建立二叉树（参见第 6 章课件要求）

◆4. 1 哈夫曼编 / 译码器（问题描述参见第 6 章课件）

【问题描述】

利用哈夫曼编码进行通信可以大大提高信道利用率，缩短信息传输时间，降低传输成本。但是，这要求在发送端通过一个编码系统对待传数据预先编码，在接收端将传来的数据进行译码(复原)。对于双工信道(即可以双向传输信息的信道)，每端都需要一个完整的编 / 译码系统。试为这样的信息收发站写一个哈夫曼码的编 / 译码系统。

【基本要求】

一个完整的系统应具有以下功能：

(1)I：初始化(Initialization)。从终端读入字符集大小  $n$ ，以及  $n$  个字符和  $n$  个权值，建立哈夫曼树，并将它存于文件 `hfmTree` 中。

(2)E：编码(Encoding)。利用已建好的哈夫曼树(如不在内存，则从文件 `hfmTree` 中读入)，对文件 `ToBeTran` 中的正文进行编码，然后将结果存入文件 `CodeFile` 中。

(3)D：译码(Decoding)。利用已建好的哈夫曼树将文件 `CodeFile` 中的代码进行译码，结果存入文件 `TextFile` 中。

(4)P：印代码文件(Print)。将文件 `CodeFile` 以紧凑格式显示在终端上，每行 50 个代码。同时将此字符形式的编码文件写入文件 `CodePrin` 中。

(5)T：印哈夫曼树(Tree printing)。将已在内存中的哈夫曼树以直观的方式(树或凹入表形式)显示在终端上，同时将此字符形式的哈夫曼树写入文件 `TreePrint` 中。

【测试数据】

(1)利用教科书例 6-2 中的数据调试程序。

(2)用下表给出的字符集和频度的实际统计数据建立哈夫曼树，并实现以下报文的编码和译码：“THIS PROGRAM IS MY FAVORITE”。

字符		A	B	C	D	E	F	G	H	I	J	K	L
频度	186	64	13	22	32	103	21	15	47	57	1	5	32
字符	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
频度	57	63	15	1	48	51	80	23	8	18	1	16	1

【实现提示】

(1)编码结果以文本方式存储在文件 `CodeFile` 中。

(2)用户界面可以设计为“菜单”方式：显示上述功能符号，再加上“Q”，表示退出运行 Quit。请用户键入一个选择功能符。此功能执行完毕后再显示此菜单，直至某次用户选择了“Q”为止。

(3)在程序的一次执行过程中，第一次执行 I，D 或 C 命令之后，哈夫曼树已经在内存了，不必再读入。每次执行中不一定执行 I 命令，因为文件 `himTree` 可能早已建好。

【选作内容】

(1)上述文件 `CodeFile` 中的每个“O”或“1”实际上占用了一个字节的空间，只起到示意或模拟的作用。为最大限度地利用码点存储能力，试改写你的系统，将编码结果以二进制形式存放在文件 `codeFile` 中。

(2)修改你的系统，实现对你的系统的原程序的编码和译码(主要是将行尾符编 / 译码问题)。

(3)实现各个转换操作的源 / 目文件，均由用户在选择此操作时指定。



#### 4. 2 最小生成树问题（此处改用 Prim 算法实现，问题描述参见第 7 章课件）

##### 【问题描述】

若要在  $z$  个城市之间建设通信网络，只需要架设  $n - 1$  条线路即可。如何以最低的经济代价建设这个通信网，是一个网的最小生成树问题。

##### 【基本要求】

(1)利用克鲁斯卡尔算法求网的最小生成树。

(2)实现教科书 6. 5 节中定义的抽象数据类型 MFSet。以此表示构造生成树过程中的连通分量。

(3)以文本形式输出生成树中各条边以及他们的权值。

##### 【测试数据】

参见本题集中的习题 7. 7。

##### 【实现提示】

通信线路一旦建立，必然是双向的。因此，构造最小生成树的网一定是无向网。设图的顶点数不超过 30 个，并为简单起见，网中边的权值设成小于 100 的整数，可利用 C 语言提供的随机数函数产生。

图的存储结构的选取应和所作操作相适应。为了便于选择权值最小的边，此题的存储结构既不选用邻接矩阵的数组表示法，也不选用邻接表，而是以存储边(带权)的数组表示图。

##### 【选作内容】

利用堆排序(参见教科书 10. 4. 3 节)实现选择权值最小的边。

## 实验五 查找和排序

#### 5. 1 哈希表设计（问题描述参见第 9 章课件）

##### 【问题描述】

针对某个集体(比如你所在的班级)中的“人名”设计一个哈希表，使得平均查找长度不超过  $R$ ，完成相应的建表和查表程序。

##### 【基本要求】

假设人名为中国人姓名的汉语拼音形式。待填入哈希表的人名共有 30 个，取平均查找长度的上限为 2。哈希函数用除留余数法构造，用伪随机探测再散列法处理冲突。

##### 【测试数据】

取读者周围较熟悉的 30 个人的姓名。

##### 【实现提示】

如果随机函数自行构造，则应首先调整好随机函数，使其分布均匀。人名的长度均不超过 19 个字符(最长的人名如：庄双双(zhang ShUangshuang)。字符的取码方法可直接利用 C 语言中的 toascii 函数，并可对过长的人名先作折叠处理。

##### 【选作内容】

(1)从教科书上介绍的几种哈希函数构造方法中选出适用者并设计几个不同的哈希函数，比较它们的地址冲突率(可以用更大的名字集合作试验)。

(2)研究这 30 个人名的特点，努力找一个哈希函数，使得对于不同的拼音名一定不发生地址冲突。

(3)在哈希函数确定的前提下尝试各种不同处理冲突的方法，考查平均查找长度的变化和造好的哈希表中关键字的聚簇性。



## 5.2 内部排序算法比较（不做!）

### 【问题描述】

在教科书中，各种内部排序算法的时间复杂度分析结果只给出了算法执行时间的阶，或大概执行时间。试通过随机数据比较各算法的关键字比较次数和关键字移动次数，以取得直观感受。

### 【基本要求】

(1)对以下 6 种常用的内部排序算法进行比较：起泡排序、直接插入排序、简单选择排序、快速排序、希尔排序、堆排序。

(2)待排序表的表长不小于 100；其中的数据要用伪随机数产生程序产生；至少要用 5 组不同的输入数据作比较；比较的指标为有关键字参加的比较次数和关键字的移动次数(关键字交换计为 3 次移动)。

(3)最后要对结果作出简单分析，包括对各组数据得出结果波动大小的解释。

### 【测试数据】

由随机数产生器生成。

### 【实现提示】

主要工作是设法在已知算法中的适当位置插入对关键字的比较次数和移动次数的计数操作。程序还可以考虑几组数据的典型性，如，正序、逆序和不同程度的乱序。注意采用分块调试的方法。

### 【选作内容】

(1)增加折半插入排序、二路插入排序、归并排序、基数排序等。

(2)对不同的输入表长作试验，观察检查两个指标相对于表长的变化关系。还可以对稳定性作验证。