



教育部考试中心 **新** **大** **纲** 配套3导丛书

全国计算机等级考试

National Computer Rank Examination

笔试教程

四合一

精编本

上机指导

模拟训练

历年真题

二 级

C 语言程序设计

航空工业出版社

National Computer Rank Examination

全国计算机等级考试

C 语言程序设计

教育考试研究中心 编审

笔
试
教
程

上
机
指
导

二 级

四合一精编本

模
拟
训
练

历
年
真
题

航空工业出版社

内 容 提 要

本书包括笔试部分、上机部分、模拟考场、历年真题4个部分的内容。笔试部分按最新考试大纲的考试要点进行详解,并通过典型例题对考试重点、难点进行分析,各章后均配有本章考点自测及答案;上机部分除了介绍上机考试系统的使用外,还提供了上机例题详解及答案;模拟考场中设置了5套全真笔试试卷和5套上机试卷及答案以供读者模拟训练;历年真题中收集了近年的4套考试真题及答案。

本书内容精练、结构合理、便于自学,适合作为等级考试的参考用书,是应试人员必备的辅导教材。

图书在版编目(CIP)数据

二级C语言程序设计/计算机等级考试命题研究组,
《全国计算机等级考试四合一精编本》编委会编. —北
京:航空工业出版社,2006.1

(全国计算机等级考试四合一精编本)

ISBN 7-80183-615-4

I. 二... II. ①计... ②全... III. C语言—程序设
计—水平考试—自学参考资料 IV. TP312

中国版本图书馆CIP数据核字(2005)第062698号

二级C语言程序设计 Erji C Yuyan Chengxu Sheji

航空工业出版社出版发行

(北京市安定门外小关东里14号 100029)

发行部电话:010-64919539 010-64978486

三河市燕山印刷有限公司印刷

全国各地新华书店经售

2006年1月第1版

2006年1月第1次印刷

开本:787×1092 1/16

印张:25.25

字数:646千字

定价:36.00元



前言



全国计算机等级考试(National Computer Rank Examination,简称 NCRE),是经原国家教育委员会(现教育部)批准,由教育部考试中心主办,面向社会,用于考查应试人员计算机应用知识与能力的全国性计算机水平考试体系,从 1994 年至今已开考二十余次。

为了适应新形势下我国市场经济发展的需要,进一步满足人们学习计算机应用技术的需求,全国计算机等级考试(NCRE)在科目设置、考核内容、考试形式上进行了大规模的调整,启用新大纲,并于 2005 年上半年在全国推广。

调整后的全国计算机等级考试(NCRE)一级开设一级 MS Office(原称一级)、一级 B、一级 WPS Office 三个科目,完全采取上机考试形式;二级新增二级 Java、二级 Access、二级 C++ 三个科目,考试形式分笔试和上机;三级上机考试环境改在 Windows 2000 下进行。

为了适应全国和各地区计算机等级考试的需要,帮助考生做好考前复习工作,根据国家教育部考试中心新制定的《全国计算机等级考试大纲》,我们编写了本套全国计算机等级考试四合一精编本丛书。本套丛书有如下特点:

本套丛书集教程考点分析、上机指导、操作题解、模拟训练和历年真题于一身,从大纲出发,研究考试要点,并通过典型例题对考试重点内容进行详解,按照章节讲解、典型例题分析、同步训练及全真模拟的组合方式让考生全面深入地掌握考试内容,全方位提高整体应试能力。

本套丛书考试要点全面,例题具有代表性,模拟试卷针对性强,可作为教材使用,又比教材集中、精练、易学,是考生短期强化提高笔试、上机两项能力的得力助手。而且,本套丛书进一步体现了导教、导学、导考的“3 导”理念,是将“教、学、考”优化集合的精品。

本套丛书提供书中程序或者相应操作的源文件下载。读者朋友可以上网登录特为本套丛书专设的页面 <http://www.study-book.cn/ncre/> 获取相关的 NCRE 考试信息及图书源文件。考生在上机练习书中的例题时不需要将书上的源代码一一录入电脑,在涉及相关操作时也不需要自己重新架设操作所需的上一级步骤,避免了时间上的浪费和录入出错。

通过本丛书网站我们还将免费赠送最新真题和更新内容,解决考生为了一套真题不得不重新购买新资料的烦恼。您的满意,我们的心意,需要帮助请致电:(010)68425475

由于时间仓促,书中难免有不当之处,敬请指正。

编者

一书在手 考试无忧

祝考生学习进步 轻松过关!



第一部分 笔试部分

上篇 公共基础知识

第1章 数据结构与算法	(1)
1.1 算法	(2)
1.1.1 算法的基本概念	(2)
1.1.2 算法复杂度	(2)
1.2 数据结构的基本概念	(3)
1.2.1 什么是数据结构	(3)
1.2.2 数据结构的图形表示	(4)
1.2.3 线性结构与非线性结构	(4)
1.3 线性表及其顺序存储结构	(5)
1.3.1 线性表的基本概念	(5)
1.3.2 线性表的顺序存储结构	(5)
1.3.3 顺序表的插入运算	(6)
1.3.4 顺序表的删除运算	(6)
1.4 栈和队列	(7)
1.4.1 栈及其基本运算	(7)
1.4.2 队列及其基本操作	(8)
1.5 线性链表	(9)
1.5.1 线性链表的基本概念	(9)
1.5.2 线性链表的基本运算	(11)
1.5.3 循环链表及其基本运算	(12)
1.6 树与二叉树	(12)
1.6.1 树的基本概念	(12)

1.6.2 二叉树及其基本性质	(12)
1.6.3 二叉树的存储结构	(13)
1.6.4 二叉树的遍历	(13)
1.7 查找技术	(14)
1.7.1 顺序查找	(15)
1.7.2 二分法查找	(15)
1.8 排序技术	(15)
1.8.1 交换类排序法	(15)
1.8.2 插入类排序	(16)
1.8.3 选择类排序	(16)
第2章 程序设计基础	(21)
2.1 程序设计方法与风格	(21)
2.2 结构化程序设计	(22)
2.2.1 结构化程序设计的原则	(22)
2.2.2 结构化程序的基本结构与特点	(23)
2.2.3 结构化程序设计原则和方法的应用	(24)
2.3 面向对象的程序设计	(24)
第3章 软件工程基础	(27)
3.1 软件工程基本概念	(27)
3.1.1 软件定义与软件特点	(27)
3.1.2 软件危机与软件工程	(27)
3.1.3 软件工程过程与软件生命周期	(28)

3.1.4	软件工程的目标与原则	(28)
3.1.5	软件开发工具与软件开发环境	(29)
3.2	结构化分析方法	(30)
3.2.1	需求分析与需求分析方法	(30)
3.2.2	结构化分析方法	(30)
3.2.3	软件需求规格说明书	(32)
3.3	结构化设计方法	(33)
3.3.1	软件设计的基本概念	(33)
3.3.2	概要设计	(34)
3.3.3	详细设计	(36)
3.4	软件测试	(37)
3.4.1	软件测试的目的	(37)
3.4.2	软件测试的准则	(38)
3.4.3	软件测试技术与方法综述	(38)
3.4.4	软件测试的实施	(40)
3.5	程序的调试	(42)
3.5.1	基本概念	(42)
3.5.2	软件调试方法	(42)
第4章	数据库设计基础	(46)
4.1	数据库系统的基本概念	(46)
4.1.1	数据、数据库、数据库管理系统	(46)
4.1.2	数据库系统的发展	(47)
4.1.3	数据库系统的基本特点	(47)
4.1.4	数据库系统的内部结构体系	(48)
4.2	数据模型	(49)
4.2.1	数据模型的基本概念	(49)
4.2.2	E-R模型	(49)
4.2.3	层次模型	(51)
4.2.4	网状模型	(51)
4.2.5	关系模型	(51)
4.3	关系代数	(53)
4.4	数据库设计与管理	(55)
4.4.1	数据库设计概述	(55)
4.4.2	数据库设计的需求分析	(56)
4.4.3	数据库概念设计	(56)
4.4.4	数据库的逻辑设计	(57)

4.4.5	数据库的物理设计	(57)
-------	----------	------

下篇 C 语言程序设计

第1章	程序设计基本概念	(61)
1.1	程序和程序设计	(61)
1.1.1	C程序	(61)
1.1.2	程序设计	(62)
1.2	算法	(62)
1.3	结构化程序设计和模块化结构	(63)
1.3.1	结构化程序	(63)
1.3.2	模块化结构	(64)
第2章	C程序设计的初步知识	(66)
2.1	简单C语言程序的构成和格式	(66)
2.2	常量、变量和标识符	(67)
2.2.1	标识符	(67)
2.2.2	常量	(68)
2.2.3	用定义一个符号名的方法来代表一个常量	(68)
2.2.4	变量	(68)
2.3	整型数据	(69)
2.3.1	整型常量	(69)
2.3.2	整型变量	(69)
2.3.3	整型数据的分类	(69)
2.3.4	整数在内存中的存储形式	(70)
2.4	实型数据	(71)
2.4.1	实型常量	(71)
2.4.2	实型变量	(71)
2.5	算术表达式	(72)
2.5.1	基本的算术运算符	(72)
2.5.2	运算符的优先级、结合性和算术表达式	(72)
2.5.3	强制类型转换表达式	(73)
2.6	赋值表达式	(73)
2.6.1	赋值运算符和赋值表达式	(73)
2.6.2	复合的赋值表达式	(74)
2.6.3	赋值运算中的类型转换	(74)
2.7	自加、自减运算符和逗号运算符	(75)

2.7.1	自加运算符(++)和自减运算符(--)	(75)	4.4.3	在 switch 语句体中使用 break 语句	(92)
2.7.2	逗号运算符和逗号表达式	(75)	4.5	语句标号和 goto 语句	(94)
第3章	顺序结构	(78)	4.5.1	语句标号	(94)
3.1	赋值语句	(78)	4.5.2	goto 语句	(94)
3.2	数据输出	(78)	第5章	循环结构	(99)
3.2.1	printf 函数的一般调用形式	(78)	5.1	while 语句和用 while 语句构成的循环结构	(99)
3.2.2	printf 函数中常用的格式说明	(79)	5.1.1	while 循环的一般形式	(99)
3.2.3	调用 printf 函数时的注意事项	(80)	5.1.2	while 循环的执行过程	(99)
3.3	数据输入	(81)	5.2	do-while 语句和用 do-while 语句构成的循环结构	(100)
3.3.1	scanf 函数的一般调用形式	(81)	5.2.1	do-while 语句构成的循环结构	(100)
3.3.2	scanf 函数中常用的格式说明	(81)	5.2.2	do-while 循环的执行过程	(101)
3.3.3	通过 scanf 函数从键盘输入数据	(82)	5.3	for 语句和用 for 语句构成的循环结构	(102)
3.4	复合语句和空语句	(83)	5.3.1	for 语句构成的循环结构	(102)
3.4.1	复合语句	(83)	5.3.2	for 循环的执行过程	(102)
3.4.2	空语句	(83)	5.3.3	有关 for 语句的说明	(102)
第4章	选择结构	(87)	5.4	循环结构的嵌套	(103)
4.1	关系运算和逻辑运算	(87)	5.5	break 和 continue 语句在循环体中的作用	(104)
4.1.1	C 语言中的逻辑值	(87)	5.5.1	break 语句	(104)
4.1.2	关系运算符和关系表达式	(87)	5.5.2	continue 语句	(105)
4.1.3	逻辑运算符和逻辑表达式	(88)	第6章	字符型数据	(110)
4.2	if 语句和用 if 语句构成的选择结构	(89)	6.1	字符型常量	(110)
4.2.1	if 语句	(89)	6.1.1	字符常量	(110)
4.2.2	嵌套的 if 语句	(89)	6.1.2	转义字符常量	(110)
4.3	条件表达式构成的选择结构	(91)	6.1.3	字符串常量	(111)
4.3.1	条件运算符	(91)	6.1.4	可对字符型量进行的运算	(111)
4.3.2	由条件运算符构成的条件表达式	(91)	6.2	字符变量	(112)
4.3.3	条件表达式的运算功能	(91)	6.3	字符的输入和输出	(112)
4.3.4	条件运算符的优先级	(91)	6.3.1	调用 printf 和 scanf 函数输出和输入字符	(112)
4.4	switch 语句以及用 switch 语句和 break 语句构成的选择结构	(92)	6.3.2	调用 putchar 和 getchar 函数输出和输入字符	(113)
4.4.1	switch 语句	(92)	第7章	函数	(118)
4.4.2	switch 语句的执行过程	(92)			

7.1 库函数	(118)	9.2 一维数组和指针	(142)
7.2 函数的定义和返回值	(119)	9.2.1 一维数组和数组元素的 地址	(142)
7.2.1 函数定义的语法	(119)	9.2.2 通过数组和首地址引用 数组元素	(142)
7.2.2 函数的返回值	(120)	9.2.3 通过指针来引用一维数组 元素	(143)
7.3 函数的调用	(121)	9.2.4 用带下标的指针变量引用一维数 组元素	(143)
7.3.1 函数的两种调用方式	(121)	9.3 函数之间对一维数组和数组元素 的引用	(143)
7.3.2 函数调用时的语法要求	(121)	9.3.1 数组元素作实参	(143)
7.4 函数的说明	(122)	9.3.2 数组名作实参	(144)
7.4.1 函数说明的形式	(122)	9.3.3 数组元素地址作为实参	(144)
7.4.2 函数说明的位置	(122)	9.3.4 函数的指针形参和函数体中 数组的区别	(144)
7.5 调用函数和被调用函数之间的 数据传递	(123)	9.4 二维数组的定义和二维数组元素 的引用	(146)
第8章 指针	(129)	9.4.1 二维数组的定义	(146)
8.1 变量的地址和指针	(129)	9.4.2 二维数组元素的引用	(146)
8.2 指针变量的定义和指针变量的基 类型	(129)	9.4.3 二维数组的初始化	(147)
8.3 给指针变量赋值	(130)	9.4.4 通过赋初值定义二维数组 的大小	(147)
8.3.1 给指针变量赋地址值	(130)	9.5 二维数组和指针	(148)
8.3.2 给指针变量赋“空”值	(130)	9.5.1 二维数组和数组元素的 地址	(148)
8.4 对指针变量的操作	(131)	9.5.2 通过地址来引用二维数组 元素	(149)
8.4.1 通过指针来引用一个存储 单元	(131)	9.5.3 通过建立一个指针数组来 引用二维数组元素	(149)
8.4.2 移动指针	(132)	9.5.4 通过建立一个行指针来引用 二维数组元素	(150)
8.4.3 指针比较	(133)	9.6 二维数组名和指针数组作为实参	(151)
8.5 函数之间地址值的传递	(133)	9.6.1 二维数组名作为实参时,实参 和形参之间的数据传递	(151)
8.5.1 形参为指针变量时,实参和 形参之间的数据传递	(133)	9.6.2 指针数组作为实参时,实参和 形参之间的数据传递	(151)
8.5.2 通过传送地址值,在被调用 函数中直接改变调用函数中 的变量的值	(134)	第10章 字符串	(159)
8.5.3 函数返回地址值	(134)	10.1 用一个一维字符数组来存放 字符串	(159)
第9章 数组	(140)		
9.1 一维数组的定义和一维数组 元素的引用	(140)		
9.1.1 一维数组的定义	(140)		
9.1.2 一维数组元素的引用	(141)		
9.1.3 一维数组的初始化	(141)		
9.1.4 通过赋初值定义数组的 大小	(141)		

10.1.1 通过赋初值的方式给一维 字符数组赋字符串	(159)	12.3.1 全局变量的作用域和 生存期	(182)
10.1.2 在 C 程序执行过程中给一维 字符数组赋字符串	(160)	12.3.2 在同一编译单位内用 extern 说明符来扩展全局变量的 作用域	(183)
10.2 使指针指向一个字符串	(161)	12.3.3 在不同编译单位内用 extern 说明符来扩展全局变量的 作用域	(183)
10.2.1 通过赋初值的方式使指针 指向一个字符串	(161)	12.3.4 静态全局变量	(183)
10.2.2 通过赋值运算使指针指向 一个字符串	(161)	12.4 函数的存储分类	(184)
10.2.3 用字符数组作为字符串和 用指针指向的一个字符串 之间的区别	(162)	12.4.1 用 extern 说明函数	(184)
10.3 字符串的输入和输出	(162)	12.4.2 用 static 说明函数	(184)
10.3.1 输入和输出字符串时的必要 条件	(162)	第 13 章 编译预处理和动态存储 分配	(187)
10.3.2 用格式说明符 %s 进行整串 输入和输出	(162)	13.1 编译预处理	(187)
10.3.3 调用 gets、puts 函数,从终端 进行字符串行的输入和 输出	(163)	13.1.1 宏替换	(187)
10.4 字符串数组	(164)	13.1.2 文件包含	(189)
10.5 用于字符串处理的函数	(165)	13.2 动态存储分配	(190)
第 11 章 对函数的进一步讨论	(172)	13.2.1 malloc 函数和 free 函数	(190)
11.1 传给 main 函数的参数	(172)	13.2.2 calloc 函数	(190)
11.2 通过实参向函数传递函数名或 指向函数的指针变量	(174)	第 14 章 结构体、共用体 and 用户定义 类型	(194)
11.3 函数的递归调用	(174)	14.1 用 typedef 说明一种新类型名	(194)
第 12 章 C 语言中用户标识符的 作用域和存储类	(180)	14.2 结构体类型	(195)
12.1 局部变量、全局变量和存储分类	(180)	14.2.1 结构体类型的说明	(195)
12.1.1 用户标识符的作用域	(180)	14.2.2 结构体类型的变量、数组和 指针变量的定义	(196)
12.1.2 局部变量、全局变量和 存储分类	(180)	14.2.3 给结构体变量、数组赋 初值	(197)
12.2 局部变量及其作用域和生存期	(181)	14.2.4 引用结构体变量中的 数据	(197)
12.2.1 auto 变量	(181)	14.2.5 函数之间结构体变量的 数据传递	(199)
12.2.2 register 变量	(181)	14.2.6 利用结构体变量构成 链表	(199)
12.2.3 静态存储类的局部变量	(181)	14.3 共用体	(203)
12.3 全局变量及其作用域和生存期	(182)	14.3.1 共用体类型的说明和变量 定义	(203)
		14.3.2 共用体变量的引用	(204)

第 15 章 位运算	(210)	函数进行输入和输出	(217)
15.1 位运算符	(210)	16.6 判文件结束函数 feof	(218)
15.2 位运算符的运算功能	(210)	16.7 fscanf 函数和 fprintf 函数	(218)
第 16 章 文件	(214)	16.8 fgets 函数和 fputs 函数	(219)
16.1 C 语言文件的概念	(214)	16.9 fread 函数和 fwrite 函数	(220)
16.2 文件指针	(215)	16.10 文件定位函数	(220)
16.3 打开文件	(215)	16.10.1 fseek 函数	(221)
16.4 关闭文件	(216)	16.10.2 ftell 函数	(221)
16.5 调用 getc(fgetc) 和 putc(fputc)		16.10.3 rewind 函数	(221)

第二部分 上机部分

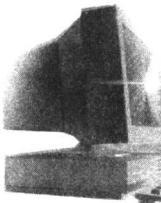
上机指导	(225)	上机例题详解	(233)
------------	-------	--------------	-------

第三部分 模拟考场

全真笔试模拟试卷(一)	(255)	上机模拟试卷(二)	(318)
全真笔试模拟试卷(二)	(267)	上机模拟试卷(三)	(321)
全真笔试模拟试卷(三)	(278)	上机模拟试卷(四)	(324)
全真笔试模拟试卷(四)	(289)	上机模拟试卷(五)	(327)
全真笔试模拟试卷(五)	(302)	全真笔试模拟试卷参考答案	(330)
上机模拟试卷(一)	(315)	上机模拟试卷参考答案	(333)

第四部分 历年真题

2004 年 4 月全国计算机等级考试二级笔试试卷 C 语言程序设计	(336)
2004 年 9 月全国计算机等级考试二级笔试试卷 C 语言程序设计	(352)
2005 年 4 月全国计算机等级考试二级笔试试卷 C 语言程序设计	(364)
2005 年 9 月全国计算机等级考试二级笔试试卷 C 语言程序设计	(378)
历年真题参考答案	(392)



第一部分 笔试部分

上篇 公共基础知识

考试大纲要求

1. 掌握算法的基本概念。
2. 掌握基本数据结构及其操作。
3. 掌握基本排序和查找算法。
4. 掌握逐步求精的结构化程序设计方法。
5. 掌握软件工程的基本方法,具有初步应用相关技术进行软件开发的能力。
6. 掌握数据的基本知识,了解关系数据库的设计。



第1章 数据结构与算法

【本章考试要点】

1. 算法的基本概念;算法复杂度的概念和意义(时间复杂度与空间复杂度)。
2. 数据结构的定义;数据的逻辑结构与存储结构;数据结构的图形表示;线性结构与非线性结构的概念。
3. 线性表的定义;线性表的顺序存储结构及其插入与删除运算。
4. 栈和队列的定义;栈和队列的顺序存储结构及其基本运算。
5. 线性单链表、双向链表与循环链表的结构及其基本运算。
6. 树的基本概念;二叉树的定义及其存储结构;二叉树的前序、中序和后序遍历。
7. 顺序查找与二分法查找算法;基本排序算法(交换类排序,选择类排序,插入类排序)。



【考试内容详解】

▶▶ 1.1 算法

1.1.1 算法的基本概念

所谓算法是指解题方案的准确而完整的描述。

1. 算法的基本特征

- (1) 可行性;
- (2) 确定性;
- (3) 有穷性;
- (4) 拥有足够的情报。

2. 算法的基本要素

一个算法通常由两种基本要素组成:一是对数据对象的运算和操作;二是算法的控制结构。

(1) 算法中对数据对象的运算和操作

- ① 算术运算:主要包括加、减、乘、除等运算。
- ② 逻辑运算:主要包括“与”、“或”、“非”等运算。
- ③ 关系运算:主要包括“大于”、“小于”、“等于”、“不等于”等运算。
- ④ 数据传输:主要包括赋值、输入、输出等操作。

(2) 算法的控制结构

一个算法一般都可以用顺序、选择、循环三种基本控制结构组合而成。

3. 算法设计基本方法

- (1) 列举法;
- (2) 归纳法;
- (3) 递推;
- (4) 递归;
- (5) 减半递推技术。

1.1.2 算法复杂度

算法的复杂度主要包括时间复杂度和空间复杂度。

1. 算法的时间复杂度

所谓算法的时间复杂度,是指执行算法所需要的计算工作量。算法的工作量用算法所执行的基本运算次数来度量,而算法所执行的基本运算次数是问题规模的函数,即

$$\text{算法的工作量} = f(n)$$

其中, n 是问题的规模。

在同一个问题的规模下,当算法执行所需的基本运算次数取决于某一特定输入时,可以用以下两种方法来分析算法的工作量。



(1) 平均性态 (Average Behavior)

所谓平均性态分析,是指用各种特定输入下的基本运算次数的加权平均值来度量算法的工作量。

设 x 是所有可能输入中的某个特定输入, $p(x)$ 是 x 出现的概率(即输入为 x 的概率), $t(x)$ 是算法在输入为 x 时所执行的基本运算次数,则算法的平均性态定义为

$$A(n) = \sum_{x \in D_n} p(x)t(x)$$

其中 D_n 表示当规模为 n 时,算法执行时所有可能输入的集合。

(2) 最坏情况复杂性 (Worst-Case Complexity)

所谓最坏情况分析,是指在规模为 n 时,算法所执行的基本运算的最大次数。它定义为

$$W(n) = \max_{x \in D_n} \{t(x)\}$$

显然, $W(n)$ 的计算要比 $A(n)$ 的计算方便得多。

2. 算法的空间复杂度

一个算法的空间复杂度,一般是指执行这个算法所需要的内存空间。一个算法所占用的存储空间包括算法程序所占的空间、输入的初始数据所占的存储空间,以及算法执行过程中所需要的额外空间。

【例 1】 下面这个程序段的时间复杂度是 ()

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

```
{
```

```
    y = y + 1;
```

```
    for(j=0; j<=2*n; j++)
```

```
        x++;
```

```
}
```

A. $O(\log_2 n)$

B. $O(n)$

C. $O(2\log_2 n)$

D. $O(n^2)$

【解析】语句的频度指的是该语句重复执行的次数。一个算法中所有语句的频度之和构成了该算法的运行时间。在本例算法中,其中语句 $y = y + 1$ 的频度是 $n - 1$,语句 $x++$ 的频度是 $(n - 1)(2n + 1) = 2n^2 - n - 1$,则该程序段的时间复杂度是 $T(n) = n - 1 + 2n^2 - n - 1 = O(n^2)$ 。故答案选 D。

1.2 数据结构的基本概念

1.2.1 什么是数据结构

简单地说,数据结构是指相互有关联的数据元素的集合。计算机已被广泛用于数据处理,实际问题中的各数据元素之间总是相互关联的。所谓数据处理,是指对数据集中的各元素以各种方式进行处理,包括插入、删除、查找、更改等运算,也包括对数据元素进行分析。

1. 数据的逻辑结构

数据结构是指反映数据元素之间关系的数据元素集合的表示。

一个数据结构应包含以下两方面的信息:





- (1) 表示数据元素的信息;
- (2) 表示各数据元素之间的前后件关系。

其中数据元素之间的前后件关系是指它们的逻辑关系,而与它们在计算机中的存储位置无关。因此,上面所述的数据结构实际上是数据的逻辑结构。

所谓数据的逻辑结构,是指反映数据元素之间逻辑关系的数据结构。

数据的逻辑结构有两个要素:一是数据元素的集合,通常记为 D ;二是 D 上的关系,它反映了 D 中各数据元素之间的前后件关系,通常记为 R 。即一个数据结构可以表示为

$$B = (D, R)$$

其中, B 表示数据结构。为了反映 D 中各数据元素之间的前后件关系,一般用二元组来表示。

2. 数据的存储结构

数据的逻辑结构在计算机存储空间中的存放形式称为数据的存储结构(也称数据的物理结构)。

一般来说,一种数据的逻辑结构根据需要可以表示成多种存储结构,常用的存储结构有顺序、链接、索引等存储结构。而采用不同的存储结构,其数据处理的效率是不同的。

1.2.2 数据结构的图形表示

一个数据结构除了用二元关系表示外,还可以直观地用图形表示。在数据结构的图形表示中,对于数据集合 D 中的每一个数据元素用中间标有元素值的方框表示,一般称为数据结点,并简称为结点。为了进一步表示各数据元素之间的前后件关系,对于关系 R 中的每一个二元组,用一条有向线段从前件结点指向后件结点。

通常,一个数据结构中的元素结点可能是在动态变化的。根据需要或在处理过程中,可以在一个数据结构中增加一个新结点(称为插入运算),也可以删除数据结构中的某个结点(称为删除运算)。插入与删除是对数据结构的两种基本运算。除此之外,对数据结构的运算还有查找、分类、合并、分解、复制和修改等。不仅数据结构中的结点(即数据元素)个数在动态地变化,而且,各数据元素之间的关系也有可能在动态地变化。

1.2.3 线性结构与非线性结构

如果在一个数据结构中一个数据元素都没有,则称该数据结构为空的数据结构。根据数据结构中各类数据元素之间前后件关系的复杂度,一般将数据结构分为两大类型:线性结构与非线性结构。

如果一个非空的数据结构满足两个条件:一是有且只有一个根结点,二是每一个结点最多有一个前件,也最多有一个后件,则称该数据结构为线性结构。线性结构又称为线性表。

需要特别说明的是,在一个线性结构中插入或删除任何一个结点后还应是线性结构。如果一个数据结构不是线性结构,则称为非线性结构。线性结构与非线性结构都可以是空的数据结构。一个空的数据结构究竟是属于线性结构还是非线性结构,这要根据具体的情况来确定。如果对该数据结构的运算是按线性结构的规则来处理的,则属于线性结构;否则属于非线性结构。

【例 2】在数据结构中,从逻辑上可以把数据结构分成 ()

- A. 动态结构和静态结构
- B. 线性结构和非线性结构
- C. 集合结构和非集合结构
- D. 树形结构和图状结构



【解析】逻辑结构即数据元素之间的逻辑关系,是从逻辑关系上描述数据,与数据的存储无关。因此,根据数据元素之间的关系,逻辑结构被分为两大类:线性结构和非线性结构。而集合结构、非集合结构、树形结构指的都是特定的数据结构类型。故答案选 B。

1.3 线性表及其顺序存储结构

1.3.1 线性表的基本概念

线性表(Linear List)是最简单、最常用的一种数据结构。线性表由一组数据元素构成,数据元素可以是简单项,一个数据元素还可以由若干个数据项组成。由若干数据项组成的数据元素称为记录(Record),而由多个记录构成的线性表称为文件(File)。

综上所述,线性表是由 $n(n \geq 0)$ 个数据元素 $a_1, a_2, \dots, a_n$ 组成的一个有限序列,表中的每一个数据元素,除了第一个外,有且只有一个前件,除了最后一个外,有且只有一个后件,即线性表或是一个空表,或可以表示为

$$(a_1, a_2, \dots, a_i, \dots, a_n)$$

其中, $a_i (i = 1, 2, \dots, n)$ 是属于数据对象的元素,通常也称其为线性表中的一个结点。显然,线性表是一种线性结构。

非空线性表有以下一些结构特征:

- (1) 有且只有一个根结点 a_1 , 它无前件;
 - (2) 有且只有一个终端结点 a_n , 它无后件;
 - (3) 除根结点与终端结点外,其他所有结点有且只有一个前件,也有且只有一个后件。
- 线性表中结点的个数 n 称为线性表的长度。当 $n = 0$ 时,称为空表。

1.3.2 线性表的顺序存储结构

线性表的顺序存储结构具有以下两个基本特点:

- (1) 线性表中所有元素所占的存储空间是连续的;
- (2) 线性表中各元素在存储结构中是按逻辑顺序依次存放的。

在线性表的顺序存储结构中,如果线性表中各数据元素所占的存储空间(字节数)相等,则要在该线性表中查找某一个元素是很方便的。

假设线性表中的第一个数据元素的存储地址(指第一个字节的地址,即首地址)为 $ADR(a_1)$, 每一个数据元素占 k 个字节,则线性表中第 i 个元素 a_i 在计算机存储空间中的存储地址为

$$ADR(a_i) = ADR(a_1) + (i - 1)k$$

即在顺序存储结构中,线性表中每一个数据元素在计算机存储空间中的存储地址由该元素在线性表中的位置序号惟一确定。一般来说,长度为 n 的线性表

$$(a_1, a_2, \dots, a_i, \dots, a_n)$$

在计算机中的顺序存储结构如图 1-1 所示。

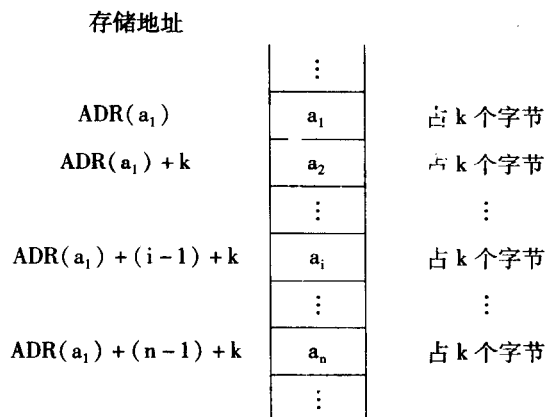


图 1-1 线性表的顺序存储结构

在线性表的顺序存储结构下,可以对线性表进行各种处理。主要的运算有以下几种:

- (1)在线性表的指定位置处加入一个新的元素(即线性表的插入);
- (2)在线性表中删除指定的元素(即线性表的删除);
- (3)在线性表中查找某个(或某些)特定的元素(即线性表的查找);
- (4)对线性表中的元素进行排序(即线性表的排序);
- (5)按要求将一个线性表分解成多个线性表(即线性表的分解);
- (6)按要求将多个线性表合并成一个线性表(即线性表的合并);
- (7)复制一个线性表(即线性表的复制);
- (8)逆转一个线性表(即线性表的逆转)等。

1.3.3 顺序表的插入运算

所谓插入运算是指在结构中的某指定位置上增加一个新结点。

线性表的插入操作是指在线性表的第 $i-1$ 个数据元素和第 i 个数据元素之间插入一个新的数据元素,就是要使长度为 n 的线性表

$$(a_1, \dots, a_{i-1}, a_i, \dots, a_n)$$

变成长度为 $n+1$ 的线性表

$$(a_1, \dots, a_{i-1}, b, a_i, \dots, a_n)$$

数据元素 a_{i-1} 和 a_i 之间的逻辑关系发生了变化。

一般情况下,要在第 $i(1 \leq i \leq n)$ 个元素之前插入一个新元素时,首先要从最后一个(即第 n 个)元素开始,直到第 i 个元素之间共 $n-i+1$ 个元素依次向后移动一个位置后,这样第 i 个位置就被空出,然后将新元素插入到第 i 项。插入结束后,线性表的长度就增加了 1。

1.3.4 顺序表的删除运算

所谓删除运算是指撤销结构中某结点的内容。

与插入相反,线性表的删除操作是使长度为 n 的线性表

$$(a_1, \dots, a_{i-1}, a_i, a_{i+1}, \dots, a_n)$$



变为长度为 $n-1$ 的线性表

$$(a_1, \dots, a_{i-1}, a_{i+1}, \dots, a_n)$$

数据元素 a_{i-1} , a_i 和 a_{i+1} 之间的逻辑关系发生了变化。

一般情况下,要删除第 i ($1 \leq i \leq n$) 个元素时,要从第 $i+1$ 个元素开始,直到第 n 个元素之间共 $n-i$ 个元素依次向前移动一个位置。删除结束后,线性表的长度就减小了 1。

【例 3】一个数据表第 1 个元素的存储地址是 100,每个元素的长度为 2,则第 5 个元素的地址是 ()

A. 100

B. 108

C. 110

D. 120

【解析】数据元素的存储位置均取决于第 1 个数据元素的存储位置,即

$$\text{LOC}(a_i) = \text{LOC}(a_1) + (i-1) \times C$$

其中, $\text{LOC}(a_1)$ 为基地址, C 为一个数据元素所占的字节数,所以第 5 个元素的地址 $= 100 + 2 \times (5-1) = 108$ 。故答案选 B。

1.4 栈和队列

1.4.1 栈及其基本运算

1. 什么是栈

栈(Stack)是限定仅在表尾进行插入和删除操作的线性表。在栈中允许插入与删除的一端称为栈顶,而不允许插入与删除的另一端称为栈底。栈顶元素总是最后被插入的元素,从而也是最先被删除的元素;栈底元素总是最先被插入的元素,从而也是最后才能被删除的元素。栈的修改是按“后进先出”或“先进后出”的原则进行的,因此,栈又称先进后出表或后进先出表,通常用指针 top 来指示栈顶的位置,用指针 bottom 指向栈底。

2. 栈的顺序存储及其运算

栈的顺序存储结构是指利用一组地址连续的存储单元依次存放自栈底到栈顶的数据元素,同时附设指针指示栈顶元素在顺序栈中的位置,如图 1-2 所示。

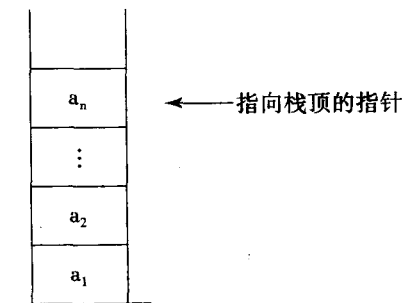


图 1-2 栈的顺序存储结构示意图

在图 1-2 中, a_1 为栈底元素, a_n 为栈顶元素。栈中的元素按照 $a_1, a_2, \dots, a_n$ 的顺序进栈,退栈的第一个元素为栈顶元素 a_n 。

栈的基本运算有三种:入栈、退栈与读栈顶元素。