NCR
Examination
教育考试研究中心 编审

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

上机指导

模拟训练

历年真题

四合一

精编本

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

二级

笔试教程

上机指导

四合一精编本

历年真题

模拟训练

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10套上机试卷及答案以供读者模拟训练；历年真题中收集了最近的两套考试真题及答案。

本书重点突出、内容全面、讲解精辟，适合应试人员考前复习使用。

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 7-80183-620-0

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062693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：21.375
字数：547千字 定价：32.00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 !

目　录

第一部分　笔试部分

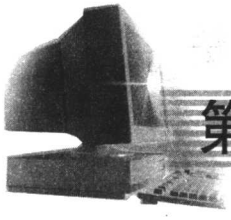
上篇　公共基础知识

下篇 C++语言程序设计

第二部分 上机部分

第三部分 模拟考场

第四部分 历年真题



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

$$算法的工作量 = f(n)$$

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,是指对数据集合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。而集合结构和非集合结构、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 $a_1,a_2,\cdots,a_n$ 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

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

其中，$a_i(i=1,2,\cdots,n)$ 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

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

在计算机中的顺序存储结构如图 1－1 所示。

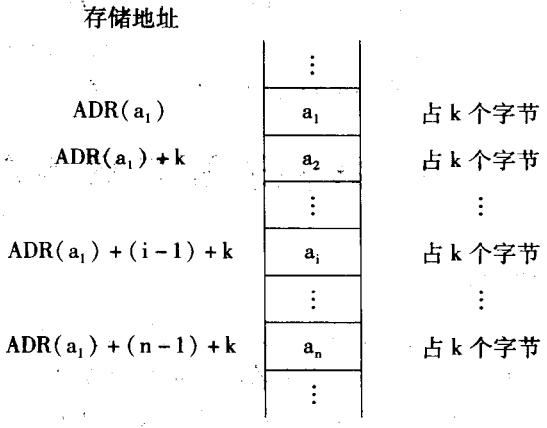


图1-1 线性表的顺序存储结构

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

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

变成长度为n+1的线性表

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

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($1\leqslant i\leqslant n$)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

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

变为长度为 n－1 的线性表

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

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 $i(1\leqslant i\leqslant n)$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

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

其中，$LOC(a_1)$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×(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。

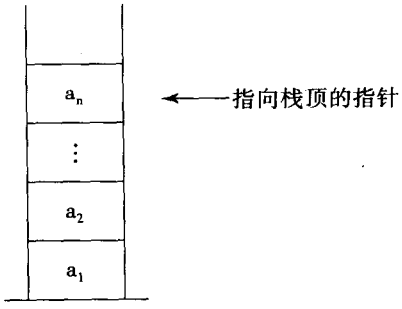


图 1－2　栈的顺序存储结构示意图

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 $a_1,a_2,\cdots,a_n$ 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。

(1)入栈运算

入栈运算是指在栈顶位置插入一个新元素,可分为两个基本操作:首先将栈顶指针进一,然后将新元素插入到栈顶指针指向的位置。

当栈顶指针已经指向存储空间的最后一个位置时,说明栈空间已满,不可能再进行入栈操作。这种情况称为栈"上溢"错误。

(2)退栈运算

退栈运算是指取出栈顶元素并赋予一个指定的变量,可分为两个基本操作:首先将栈顶元素(栈顶指针指向的元素)赋予一个指定的变量,然后将栈顶指针退一。

当栈顶指针为零时,说明栈空,不可能进行退栈操作。这种情况称为栈"下溢"错误。

(3)读栈顶元素

读栈顶元素是指将栈顶元素赋予一个指定的变量。必须注意,这个运算不删除栈顶元素,只是将它的值赋予一个变量,因此,在这个运算中,栈顶指针不会改变。

当栈顶指针为零时,说明栈空,读不到栈顶元素。

1.4.2 队列及其基本操作

1. 什么是队列

队列(Queue)是指允许在一端进行插入,而在另一端进行删除的线性表。允许插入的一端称为队尾,通常用一个称为队尾指针(rear)的指针指向队尾元素,即队尾指针总是指向最后被插入的元素;允许删除的一端称为排头(也称为队头),通常也用一个排头指针(front)指向排头元素的前一个位置。显然,在队列这种数据结构中,最先插入的元素将最先被删除,而最后插入的元素最后才能被删除。因此,队列又称为"先进先出"或"后进后出"的线性表,它体现了"先来先服务"的原则。

2. 循环队列及其运算

在实际应用中,队列的顺序存储结构一般采用循环队列的形式。

所谓循环队列,就是将队列存储空间的最后一个位置绕到第一个位置,形成逻辑上的环状空间,供队列循环使用,如图1-3所示。

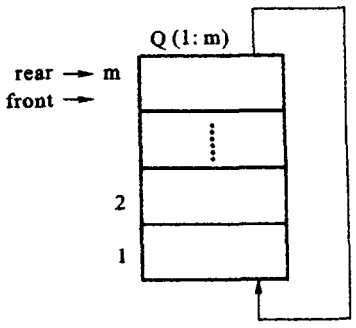


图1-3 循环队列存储空间示意图

在循环队列中,用队尾指针 rear 指向队列中的队尾元素,用排头指针 front 指向排头元素的前一个位置,因此,从排头指针 front 指向的后一个位置到队尾指针 rear 指向的位置之间所有的