

飞思考试中心

全国计算机等级考试考点分析、题解与模拟： 二级 C(最新版)

全国计算机等级考试命题研究中心

编著

飞思教育产品研发中心

联合监制

未来教育教学与研究中心

电子工业出版社

Publishing House of Electronics Industry

北京·BEIJING

内 容 简 介

本书依据教育部考试中心最新发布的《全国计算机等级考试考试大纲》,在《全国计算机等级考试考点分析、题解与模拟(2012版)》的基础上修订而成。在编写过程中,一方面结合最新大纲和数套真题,对重要考点进行了分析、讲解,并选取经典考题进行了深入剖析;另一方面配有同步练习、无纸化模拟考试试题,以逐步向考生详尽透析考试中的所有知识要点。“一书在手,通关无忧”。

本书配有“全国计算机等级考试模拟软件”。其中智能化的答题系统按照本书的顺序循序渐进、逐步编排;完全模拟真实考试,考试步骤、考试界面、考试方式、题目形式与真实考试完全一致,并可以自动评分。“书+光盘,物超所值”。

本书适合作为全国计算机等级考试考前培训班辅导用书,也可作为应试人员的自学用书。

未经许可,不得以任何方式复制或抄袭本书之部分或全部内容。

版权所有,侵权必究。

图书在版编目(CIP)数据

全国计算机等级考试考点分析、题解与模拟. 二级 C/全国计算机等级考试命题研究中心编著. —北京:电子工业出版社, 2007. 11

(飞思考试中心)

ISBN 978-7-121-05206-4

I. 全... II. 全... III. ①电子计算机—水平考试—自学参考资料②C 语言—程序设计—水平考试—自学参考资料
IV. TP3

中国版本图书馆 CIP 数据核字(2007)第 160320 号

责任编辑:王树伟

特约编辑:赵树刚

印 刷: 北京富博印刷有限公司

装 订:

出版发行:电子工业出版社

北京海淀区万寿路 173 信箱 邮编:100036

开 本:880×1230 1/16 印张:14.75 字数:566.4 千字

印 次:2013 年 1 月第 1 次印刷

定 价:29.80 元(含光盘 1 张)

凡所购买电子工业出版社图书有缺损问题,请向购买书店调换。若书店售缺,请与本社发行部联系,联系及邮购电话:(010)88254888。

质量投诉请发邮件至 zlt@phei.com.cn,盗版侵权举报请发邮件至 dbqq@phei.com.cn。

服务热线:(010)88258888。

全国计算机等级考试自 1994 年由国家教育部考试中心推出以来,为评测全社会非计算机专业人员的计算机知识与技能,培养各行业的计算机应用人才开辟了一条新的道路,受到了用人单位和学习人员的热烈欢迎。全国计算机等级考试通过数年的发展,已经成为我国最大型的计算机类考试。

为了帮助更多的学习者顺利地通过考试,并掌握相应的操作技能,我们在深入调研、详尽分析考试大纲的基础上,组织国内著名高校的计算机专家和一线教师编写了本书。

本书共分为三大部分,同时配有一张学习软件光盘。

※ 考点分析/真题题解/同步练习

“考点分析”结合最新考试大纲、教材,对教材中考核的重点和难点进行了讲解,内容涵盖了大纲中所有的选择题和操作题的考点。

“真题题解”选取极具代表性的经典例题。例题符合考试命题规律的特征,对题目的讲解深入、透彻,循序渐进,极有条理。

“同步练习”提供了大量习题,对前面所学的理论知识进行温习和巩固,以练促学、学练结合。

※ 无纸化考试试题

结合最新考试大纲,筛选与演绎出的典型试题集,不论在形式上还是难度上,都与真题类似,使学习者熟悉整个考试过程,了解考试的题型、题量,并配有详细的解析,使学习者既能知其然,也能知其所以然。

※ 配套学习软件

本书配套光盘具有如下特色:

- 超大量仿真考试模拟试卷,自动组卷,即时评分,由专家对答题结果进行“现场指导”。
- 从抽题、答题到交卷完全模拟真实考试,唯一不同之处是可以对上机作答进行评分。

本书所有操作题都经过上机调试通过。由于时间仓促,书中难免有不当之处,敬请指正。

联系方式

电 话: (010)82552266 68134545 88254160

电子邮件: support@fecit.com.cn eduwin@sina.com

未来教育考试网: <http://www.eduexam.cn>

飞思在线: <http://www.fecit.com.cn> <http://www.fecit.net>

通用网址: 计算机图书、飞思、飞思教育、飞思科技、FECIT

全国计算机等级考试命题研究中心

飞思教育产品研发中心

未来教育教学与研究中心

第 0 章 公共基础知识

0.1 数据结构与算法	1	0.4 数据库设计基础	20
0.2 程序设计基础	11	0.5 同步练习	26
0.3 软件工程基础	13	0.6 同步练习答案	27

第 1 章 C 语言概述

1.1 C 语言基础知识	28	1.3 同步练习	35
1.2 常量、变量和数据类型	30	1.4 同步练习答案	36

第 2 章 运算符与表达式

2.1 C 语言运算符简介	37	2.4 位运算	42
2.2 算术运算符和算术表达式	38	2.5 同步练习	43
2.3 赋值运算符和赋值表达式	40	2.6 同步练习答案	45

第 3 章 基本语句

3.1 C 语句概述	46	3.5 数据格式的输入与输出	49
3.2 赋值语句	47	3.6 同步练习	53
3.3 输入/输出的概念及其实现	48	3.7 同步练习答案	56
3.4 单个字符的输入/输出	48		

第 4 章 选择结构

4.1 关系运算符和关系表达式	57	4.4 switch 语句和 goto 语句	62
4.2 逻辑运算符和逻辑表达式	58	4.5 同步练习	64
4.3 if 语句和用 if 语句构成的选择结构	59	4.6 同步练习答案	68

第 5 章 循环结构

5.1 概述	69	5.6 几种循环的比较	75
5.2 while 语句	69	5.7 break 语句和 continue 语句	75
5.3 do...while 语句	70	5.8 同步练习	77
5.4 for 语句	71	5.9 同步练习答案	80
5.5 循环的嵌套	74		

第 6 章 数 组

6.1 一维数组的定义和引用	81	6.4 同步练习	87
6.2 二维数组的定义和引用	83	6.5 同步练习答案	90
6.3 字符数组	84		

第 7 章 函 数

7.1 概述	91	7.3 函数定义的一般形式	92
7.2 库函数	92	7.4 函数参数和函数返回值	94

7.5	函数的调用	95	7.9	变量的存储类别	100
7.6	函数的嵌套调用与递归调用	97	7.10	内部函数和外部函数	103
7.7	数组作为函数参数	99	7.11	同步练习	103
7.8	全局变量和局部变量	99	7.12	同步练习答案	107

第 8 章 指 针

8.1	关于地址和指针	108	8.6	返回指针值的函数	116
8.2	变量的指针和指向变量的指针变量	109	8.7	指针数组和指向指针的指针	117
8.3	数组与指针	110	8.8	同步练习	118
8.4	字符串与指针	113	8.9	同步练习答案	121
8.5	指向函数的指针	115			

第 9 章 编译预处理和动态存储分配

9.1	宏定义	122	9.4	关于动态存储的函数	126
9.2	文件包含	123	9.5	同步练习	127
9.3	条件编译	125	9.6	同步练习答案	129

第 10 章 结构体与共用体

10.1	用 typedef 说明一种新类型名	130	10.6	指向结构体类型数据的指针	134
10.2	结构体类型	130	10.7	链表	135
10.3	结构体类型变量的定义	131	10.8	共用体	139
10.4	结构体变量的引用	132	10.9	同步练习	141
10.5	结构体数组	133	10.10	同步练习答案	143

第 11 章 文 件

11.1	C 语言文件的概念	144	11.5	文件的定位	148
11.2	文件类型指针	145	11.6	同步练习	149
11.3	文件的打开与关闭	145	11.7	同步练习答案	151
11.4	文件的读写	147			

第 12 章 操作题高频考点

12.1	C 程序设计基础	152	12.5	数组	159
12.2	C 语言的基本结构	154	12.6	字符串	162
12.3	函数	157	12.7	结构体、共用体 and 用户定义类型	164
12.4	指针	158	12.8	文件	166

第 13 章 无纸化考试试题

13.1	无纸化考试试题(1)	168	13.5	无纸化考试试题(5)	200
13.2	无纸化考试试题(2)	176	13.6	参考答案及解析	208
13.3	无纸化考试试题(3)	184	13.7	无纸化考试试题及解析(6)~(105)(见光盘)	224
13.4	无纸化考试试题(4)	192			

附 录

附录 A	常用字符与 ASCII 码对照表	225	附录 C	运算符的优先级与结合性	227
附录 B	C 语言关键字	226			

第 0 章 公共基础知识

考核知识点

重要考点提示

- 数据结构与算法
- 程序设计基础
- 软件工程基础
- 数据库设计基础

- 栈和队列
- 树与二叉树
- 结构化分析方法
- 数据库系统的基本概念
- 数据模型
- 关系代数

0.1 数据结构与算法

考核概率 100%

考点 1 算 法

1. 算法的基本概念

算法是指一系列解决问题的清晰指令。

(1) 算法的基本特征。

- 可行性: 针对实际问题而设计的算法, 执行后能够得到满意的结果, 即必须有一个或多个输出, 即使在数学理论上是正确的, 如果在实际的计算工具上不能执行, 则该算法也是不具有可行性的。
- 确定性: 是指算法中每一步骤都必须是有明确定义的。
- 有穷性: 是指算法必须能在有限的时间内做完。
- 拥有足够的情报: 一个算法是否有效, 还取决于为算法所提供的情报是否足够。

(2) 算法的基本要素。

算法一般由两种基本要素构成:

- 对数据对象的运算和操作。
- 算法的控制结构, 即运算和操作时间的顺序。

算法中对数据的运算和操作: 算法就是按解题要求从指令系统中选择合适的指令组成的指令序列。因此计算机算法就是计算机能执行的操作所组成的指令序列。不同的计算机系统, 指令系统是有差异的, 但一般的计算机系统中包括的运算和操作有 4 类: 算术运算、逻辑运算、关系运算和数据传输。

算法的控制结构: 算法中各操作之间的执行顺序称为算法的控制结构。算法的功能不仅取决于所选用的操作, 还与各操作之间的执行顺序有关。基本的控制结构包括顺序结构、选择结构和循环结构。

(3) 算法设计的基本方法。

算法设计的基本方法有列举法、归纳法、递推法、递归法、减半递推技术和回溯法。

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。

一般情况下,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 表示问题的规模。该表达式表示随着问题规模 n 的增大,算法执行时间的增长率和 $f(n)$ 的增长率相同。

在同一个问题规模下,如果算法执行所需的基本运算次数取决于某一特定输入时,可以用两种方法来分析算法的工作量:平均性态分析和最坏情况分析。

(2)算法的空间复杂度。

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。算法执行期间所需要的存储空间包括以下 3 个部分:

- 算法程序所占的空间。
- 输入的初始数据所占的存储空间。
- 算法执行过程中所需要的额外空间。

在实际操作中,为了减少算法所占的存储空间,通常采用压缩存储的技术,用于减少不必要的额外空间。

考点 2 数据结构的基本概念

1. 数据结构的定义

数据结构是指相互有关联的数据元素的集合,即数据的组织形式。

(1)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(即前后件关系)的数据结构。它包括两个要素,数据元素的集合和数据元素之间的关系。

(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(也称为数据的物理结构)。数据结构的存储方式包括顺序存储方法、链式存储方法、索引存储方法和散列存储方法。而采用不同的存储结构,其数据处理的效率是不同的。因此,在进行数据处理时,选择合适的存储结构是很重要的。

数据结构研究的内容主要包括 3 个方面:

- 数据集合中各数据元素之间的逻辑关系,即数据的逻辑结构。
- 在对数据进行处理时,各数据元素在计算机中的存储关系,即数据的存储结构。
- 对各种数据结构进行的运算。

2. 数据结构的图形表示

数据元素之间最基本的关系是前后件关系。前后件关系,即每一个二元组,都可以用图形来表示。用中间标有元素值的方框表示数据元素,一般称为数据结点,简称为结点。对于每一个二元组,用一条有向线段从前件指向后件。

图形表示数据结构具有直观易懂的特点,在不引起歧义的情况下,前件结点到后件结点连线上的箭头可以省略。例如,树型结构中,通常都是用无向线段来表示前后件关系的。

3. 线性结构与非线性结构

根据数据结构中各数据元素之间前后件关系的复杂程度,一般将数据结构分为两大类型,即线性结构和非线性结构。

如果一个非空的数据结构满足有且只有一个根结点,并且每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,又称线性表。如果不满足上述条件的数据结构则称为非线性结构。

考点 3 线性表及其顺序存储结构**1. 线性表的基本概念**

在数据结构中,线性结构又称为线性表,线性表是最简单也是最常用的一种数据结构。

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,除表中的第一个元素外,有且只有一个前件;除了最后一个元素外,有且只有一个后件。

线性表或者是个空表,或者可以表示为:

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

其中, $a_i (i = 1, 2, \dots, n)$ 是线性表的数据元素,也称为线性表的一个结点。

每个数据元素在不同情况下其具体含义各不相同,它可以是一个数或一个字符,也可以是一个具体的事物,甚至是更复杂的信息。但是需要注意的是,同一线性表中的数据元素必定具有相同的特性,即属于同一数据对象。

TIPS 小提示

非空线性表具有以下一些结构特征:

- 只有一个根结点,即头结点,它无前件;
- 有且只有一个终结点,即尾结点,它无后件;
- 除头结点与尾结点外,其他所有结点有且只有一个前件,也有且只有一个后件。结点个数为 n 称为线性表的长度,当 $n = 0$ 时,称为空表。

2. 线性表的顺序存储结构

将线性表中的元素逐个存储在一片相邻的存储区域中。这种顺序表示的线性表也称为顺序表。

线性表的顺序存储结构具有以下两个基本特点:

- 元素所占的存储空间必须是连续的。
- 元素在存储空间的位置是按逻辑顺序存放的。

从上述特点也可以看出,线性表是用元素在计算机内物理位置上的相邻关系来表示元素之间逻辑上的相邻关系。只要确定了首地址,线性表内任意元素的地址都可以方便地计算出来。

3. 线性表的插入运算

在线性表的插入运算中,在第 i 个元素之前插入一个新元素,完成插入操作主要有以下 3 个步骤。

- (1) 把原来第 n 个结点至第 i 个结点依次往后移一个元素位置。
- (2) 把新结点放在第 i 个位置上。
- (3) 修正线性表的结点个数为 $n+1$ 。

TIPS 小提示

一般会为线性表开辟一个大于线性表长度的存储空间,经过多次插入运算,可能出现存储空间已满的情况,如果此时仍继续插入运算,将会产生错误,此类错误称为“上溢”。

如果需要在线性表末尾进行插入运算,则只需在表的末尾增加一个元素即可,不需要移动线性表中的元素。

如果在第一个位置插入新的元素,则需要移动表中所有的元素。

4. 线性表的删除运算

在线性表的删除运算中,删除第 i 个位置的元素,则要从第 $i+1$ 个元素开始,直到第 n 个元素之间共

$n-i$ 个元素依次向前移一个位置,完成删除运算主要有以下几个步骤。

(1)把第 i 个元素之后(不包括第 i 个元素)的 $n-i$ 个元素依次前移一个位置。

(2)修正线性表的结点个数。

综上所述,如果删除运算在线性表的末尾进行,即删除第 n 个元素,则不需要移动线性表中的元素。如果要删除第 1 个元素,则需要移动表中的所有数据。

TIPS

小提示

由线性表的上述性质可以看出,线性表的顺序存储结构适用于小线性表,或者建立之后其中元素不常变动的线性表,而不适用于需要经常进行插入和删除运算的线性表和长度较大的线性表。

考点 4 栈和队列

1. 栈及其基本运算

(1)栈的基本概念。

栈实际上也是线性表,只不过是一种特殊的线性表。在这种特殊的线性表中,其插入运算与删除运算都只在线性表的一端进行。

在栈中,允许插入与删除的一端称为栈顶(top),另一端称为栈底(bottom)。若栈中没有元素称为空栈,栈也被称为“先进后出”表,或“后进先出”表。

(2)栈的特点。

根据栈的上述定义,栈具有以下特点:

- 栈顶元素总是最后被插入的元素,也是最早被删除的元素。
- 栈底元素总是最早被插入的元素,也是最后才能被删除的元素。
- 栈具有记忆作用。
- 在顺序存储结构下,栈的插入和删除运算都不需要移动表中其他数据元素。
- 栈顶指针 top 动态反映了栈中元素的变化情况。

(3)栈的顺序存储及其运算。

栈的状态如图 0-1 所示。

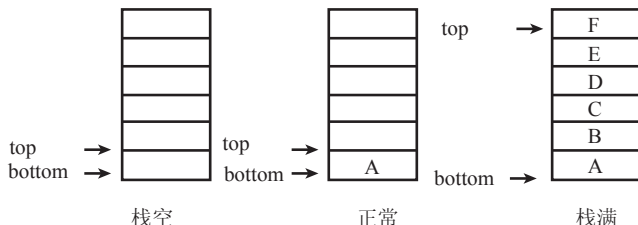


图 0-1 栈的状态

根据栈的状态,可以得知栈的基本运算有以下 3 种。

- 入栈运算:在栈顶位置插入一个新元素。
- 退栈运算:取出栈顶元素并赋给一个指定的变量。
- 读栈顶元素:将栈顶元素赋给一个指定的变量。

2. 队列及其基本运算

(1)队列的基本概念。

队列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

个称为尾指针 (rear) 的指针指向队尾元素;允许删除的一端称为排头,通常用一个头指针 (front) 指向头元素的前一个位置。

因此,队列又称为“先进先出”(First In First Out, FIFO) 的线性表。插入元素称为入队运算,删除元素称为退队运算。

队列的基本结构如图 0-2 所示。

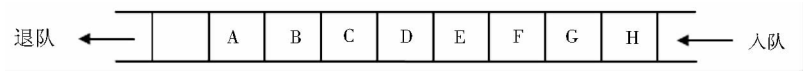


图 0-2 队列

(2) 循环队列及其运算。

所谓循环队列,就是将队列存储空间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。

在循环队列中,用尾指针 (rear) 指向队列的尾元素,用头指针 (front) 指向头元素的前一个位置。因此,从头指针 (front) 指向的后一个位置,直到尾指针 (rear) 指向的位置之间所有的元素均为队列中的元素。循环队列的初始状态为空,即 $rear = front$ 。

循环队列的基本运算主要有两种:入队运算与退队运算。

- 入队运算是指在循环队列的队尾加入一个新的元素。
- 退队运算是指在循环队列的排头位置退出一个元素,并赋给指定的变量。

TIPS 小提示

栈是按照“先进后出”或“后进先出”的原则组织数据的,而队列是按照“先进先出”或“后进后出”的原则组织数据。这就是栈和队列的不同。

考点 5 线性链表

1. 线性链表的基本概念

线性表的链式存储结构称为线性链表。

为了存储线性表中的每一个元素,一方面要存储数据元素的值;另一方面要存储各数据元素之间的前后件关系。为此,在链式存储方式中,每个结点由两部分组成:一部分称为数据域,用于存放数据元素值;另一部分称为指针域,用于存放下一个数据元素的存储序号,即指向后件结点。链式存储结构既可以表示线性结构,也可以表示非线性结构。

线性表链式存储结构的特点是用一组不连续的存储单元存储线性表中的各个元素。因为存储单元不连续,所以数据元素之间的逻辑关系就不能依靠数据元素的存储单元之间的物理关系来表示。

2. 线性链表的基本运算

线性链表主要包括以下几种运算:

- 在线性链表中包含指定元素的结点之前插入一个新元素。
- 在线性链表中删除包含指定元素的结点。
- 将两个线性链表按要求合并成一个线性链表。
- 将一个线性链表按要求进行分解。
- 逆转线性链表。
- 复制线性链表。
- 线性链表的排序。
- 线性链表的查找。

3. 循环链表及其基本运算

(1) 循环链表的定义。

在单链表的第一个结点前增加一个表头结点,队头指针指向表头结点,在最后一个结点的指针域的值由 NULL 改为指向表头结点,这样的链表称为循环链表。在循环链表中,所有结点的指针构成了一个环状链。

(2) 循环链表与单链表的比较。

对单链表的访问是一种顺序访问,从其中某一个结点出发,只能找到它的直接后继,但无法找到它的直接前驱。而且对于空表和第一个结点的处理必须单独考虑,空表与非空表的操作不统一。

在循环链表中,只要指出表中任何一个结点的位置,就可以从它出发访问到表中其他所有的结点。由于表头结点是循环链表所固有的结点,因此,即使在表中没有数据元素的情况下,表中也至少有一个结点存在,从而使空表和非空表的运算统一。

考点 6 树和二叉树

1. 树的基本概念

树是一种简单的非线性结构,直观地来看树是以分支关系定义的层次结构。树是由 $n(n \geq 0)$ 个结点构成的有限集合,当 $n=0$ 时,树称为空树。当 $n \neq 0$ 时,树中的结点应该满足以下两个条件:

- 有且仅有一个没有前驱的结点称之为根。
- 其余结点分成 $m(m > 0)$ 个互不相交的有限集合 T_1, T_2, \dots, T_m , 其中每一个集合又都是一棵树,称 T_1, T_2, \dots, T_m 为根结点的子树。

在树的结构中主要涉及下面几个概念。

- 每一个结点只有一个前件,称为父结点,没有前件的结点只有一个,称为树的根结点,简称树的根。
- 每一个结点可以有多个后件,称为该结点的子结点。没有后件的结点称为叶子结点。
- 一个结点所拥有的后继个数称为该结点的度。
- 所有结点最大的度称为树的度。
- 树的最大层次称为树的深度。

2. 二叉树及其基本性质

(1) 二叉树的定义。

二叉树是一种非线性结构,是一个有限的结点集合,该集合或者为空,或者由一个根结点及其两棵互不相交的左右二叉子树组成。当集合为空时,称该二叉树为空二叉树。

二叉树具有以下特点。

- 二叉树可以为空,空的二叉树没有结点,非空二叉树有且只有一个根结点。
- 每一个结点最多有两棵子树,且分别称为该结点的左子树与右子树。

(2) 满二叉树和完全二叉树。

满二叉树:除最后一层外,每一层上的所有结点都有两个子结点,即在满二叉树的第 k 层上有 2^{k-1} 个结点,且深度为 m 的满二叉树中有 $2^m - 1$ 个结点。

完全二叉树:除最后一层外,每一层上的结点数都达到最大值;在最后一层上只缺少右边的若干结点。

满二叉树与完全二叉树的关系:满二叉树一定是完全二叉树,但完全二叉树不一定是满二叉树。

(3) 二叉树的主要性质。

- 一棵非空二叉树的第 k 层上最多有 2^{k-1} 个结点($k \geq 1$)。
- 深度为 m 的满二叉树中有 $2^m - 1$ 个结点。
- 对于任何一棵二叉树,度为 0 的结点(即叶子结点)总是比度为 2 的结点多一个。
- 具有 n 个结点的完全二叉树的深度 k 为 $\lceil \log_2 n \rceil + 1$ 。

3. 二叉树的存储结构

在计算机中,二叉树通常采用链式存储结构,用于存储二叉树中各元素的存储结点由数据域和指针域组成。由于每一个元素可以有两个后件(即两个子结点),所以用于存储二叉树的存储结点的指针域有两个:一个指向该结点的左子结点的存储地址,称为左指针域;另一个指向该结点的右子结点的存储地址,称为右指针域。因此,二叉树的链式存储结构也称为二叉链表。

对于满二叉树与完全二叉树可以按层次进行顺序存储。

4. 二叉树的遍历

二叉树的遍历是指不重复地访问二叉树中的所有结点。二叉树的遍历主要是针对非空二叉树的,对于空二叉树,则结束返回。

二叉树的遍历有前序遍历、中序遍历和后序遍历。

(1)前序遍历(DLR)。

首先访问根结点,然后遍历左子树,最后遍历右子树。

(2)中序遍历(LDR)。

首先遍历左子树,然后访问根结点,最后遍历右子树。

(3)后序遍历(LRD)。

首先遍历左子树,然后遍历右子树,最后访问根结点。

TIPS 小提示

已知一棵二叉树的前序遍历序列和中序遍历序列,可以唯一确定这棵二叉树。已知一棵二叉树的后序遍历序列和中序遍历序列,也可以唯一确定这棵二叉树。已知一棵二叉树的前序遍历序列和后序遍历序列,不能唯一确定这棵二叉树。

考点 7 查找技术

1. 顺序查找

顺序查找一般是指在线性表中查找指定的元素。其基本思路是:从表中的第一个元素开始,依次将线性表中的元素与被查找元素进行比较,直到两者相符,查到所要找的元素为止。否则,表中没有要找的元素,查找不成功。

在最好的情况下,第一个元素就是要查找的元素,则比较次数为1次。

在最坏的情况下,顺序查找需要比较 n 次。

在平均情况下,需要比较 $n/2$ 次。因此查找算法的时间复杂度为 $O(n)$ 。

在下列两种情况下只能够采取顺序查找:

- 如果线性表中元素的排列是无序的,则无论是顺序存储结构还是链式存储结构,都只能用顺序查找;
- 即便是有序线性表,若采用链式存储结构,只能进行顺序查找。

2. 二分查找

使用二分法查找的线性表必须满足以下两个条件:

- 顺序存储结构。
- 线性表是有序表。

所谓有序表,是指线性表中的元素按值非递减排列(即从小到大,但允许相邻元素值相等)。

对于长度为 n 的有序线性表,利用二分法查找元素 x 的过程如下:

(1)将 x 与线性表的中间项进行比较。

(2)若中间项的值等于 x ,则查找成功,结束查找。

(3)若 x 小于中间项的值,则在线性表的前半部分以二分法继续查找。

(4)若 x 大于中间项的值,则在线性表的后半部分以二分法继续查找。

这样反复进行查找,直到查找成功或子表长度为 0(说明线性表中没有这个元素)为止。

当有序线性表为顺序存储时采用二分查找的效率要比顺序查找高得多。对于长度为 n 的有序线性表,在最坏的情况下,二分查找只需比较 $\log_2 n$ 次,而顺序查找则需要比较 n 次。

考点 8 排序技术

1. 交换类排序法

交换类排序法是指借助数据元素的“交换”进行排序的一种方法。本节介绍的冒泡排序法和快速排序法就属于交换类排序法。

(1)冒泡排序法。

冒泡排序的基本思想:在线性表中依次查找相邻的数据元素,将表中最大的元素不断往后移动,反复操作直到消除所有逆序。此时,该表已经排序结束。

冒泡排序法的基本过程如下。

①从表头开始往后查找线性表,在查找过程中逐次比较相邻两个元素的大小。若在相邻两个元素中,前面的元素大于后面的元素,则将它们交换。

②从后向前查找剩下的线性表(除去最后一个元素),同样,在查找过程中逐次比较相邻两个元素的大小。若在相邻两个元素中,后面的元素小于前面的元素,则将它们交换。

③对剩下的线性表重复上述过程,直到剩下的线性表变空为止,线性表排序完成。

假设线性表的长度为 n ,则在最坏的情况下,冒泡排序需要经过 $n/2$ 遍从前往后的扫描和 $n/2$ 遍从后往前扫描,需要比较 $n(n-1)/2$ 次,其数量级为 n^2 。

(2)快速排序法。

快速排序法的基本思想:在线性表中逐个选取元素,将线性表进行分割,直到所有元素全部选取完毕,此时线性表已经排序结束。

快速排序法的基本过程如下。

①从线性表中选取一个元素,设为 T ,将线性表后面小于 T 的元素移动到前面,而将大于 T 的元素移到后面,这样就将线性表分成了两部分(称为两个子表)。 T 就是处于分界线的位置,将线性表分成了前后两个子表,且前面子表中的所有元素均不大于 T ,而后子表中的所有元素均不小于 T ,此过程称为线性表的分割。

②对分割后的子表再按上述原则进行反复分割,直到所有子表为空结束,则此时的线性表就变成有序。

2. 插入类排序法

插入排序是指将无序序列中的各元素依次插入已经有序的线性表中。下面将主要介绍简单插入排序法和希尔排序法。

(1)简单插入排序法。

简单插入排序是把 n 个待排序的元素看成一个有序表和一个无序表,开始时,有序表只包含一个元素,而无序表包含 $n-1$ 个元素,每次取无序表中的第一个元素插入到有序表中的正确位置,使之成为增加一个元素的新的有序表。插入元素时,插入位置及其后的记录依次向后移动。最后有序表的长度为 n ,而无序表为空,此时排序完成。

在简单插入排序中,每一次比较后最多移除一个逆序,因此,该排序方法的效率与冒泡排序法相同。一般简单插入排序需要 $n(n-1)/2$ 次比较。

(2) 希尔排序法。

希尔排序法是将整个无序序列分割成若干个小的子序列并分别进行插入排序。

分割方法如下：

- ①将相隔某个增量 h 的元素构成一个子序列。
- ②在排序过程中, 逐次减少这个增量, 直到 h 减到 1 时, 进行一次插入排序, 排序即可完成。

希尔排序的效率与所选取的增量序列有关。

3. 选择类排序法

选择排序是通过每次从待排序序列中选出的最小值元素, 顺序放在已排好序的有序子表的后面, 直到全部序列满足排序要求为止。下面就介绍选择类排序法中的简单选择排序法和堆排序法。

(1) 简单选择排序法。

进行简单选择排序, 首先从所有 n 个待排序的数据元素中选择最小的元素, 将该元素与第一个元素交换, 再从剩下的 $n-1$ 个元素中选出最小的元素与第二个元素交换。重复这样的操作直到所有的元素有序为止。

简单选择排序需要比较 $n(n-1)/2$ 次。

(2) 堆排序法。

堆排序的方法如下：

- ①将一个无序序列建成堆。
- ②将堆顶元素与堆中最后一个元素交换。忽略已经交换到最后的那个元素, 考虑前 $n-1$ 个元素构成的子序列, 只有左、右子树是堆, 可以将该子树调整为堆。这样反复做第二步, 直到剩下的子序列空为止。

堆排序需要比较的次数为 $O(n\log_2 n)$ 。

真题题解

【例1】下列叙述中正确的是()。

- A) 程序执行的效率与数据的存储结构密切相关
- B) 程序执行的效率只取决于程序的控制结构
- C) 程序执行的效率只取决于所处理的数据量
- D) 以上三种说法都不对

答案:A)

解析: 在计算机中, 数据的存储结构对数据的执行效率有较大影响, 如在有序存储的表中, 查找某个数值比在无序存储的表中查找的效率要高很多。

【例2】下列有关顺序存储结构的叙述, 不正确的是()。

- A) 存储密度大
- B) 逻辑上相邻的结点物理上不必邻接
- C) 可以通过计算机直接确定第 i 个结点的存储地址
- D) 插入、删除操作不方便

答案:B)

解析: 顺序存储结构要求逻辑上相邻的元素物理地址上也相邻, 所以只有选项 B 叙述错误。

【例3】在一个长度为 n 的顺序表中, 向第 i 个元素 ($1 \leq i \leq n+1$) 的位置插入一个新元素, 需要从后向前依次移动() 个元素。

- A) $n-i$
- B) i
- C) $n-i-1$
- D) $n-i+1$

答案:D)

解析: 根据顺序表的插入运算的定义知道, 在第 i 个位置上插入 x , 从 a_i 到 a_n 都要向后移动一个位置, 所以共需要移动 $n-i+1$ 个元素。

【例4】下列对队列的叙述正确的是()。

- A) 队列属于非线性表
C) 队列在队尾删除数据

- B) 队列按“先进后出”原则组织数据
D) 队列按“先进先出”原则组织数据

答案:D)

解析:队列是一种特殊的线性表,它只能在一端进行插入,在另一端进行删除。允许插入的一端称为队尾,允许删除的另一端称为队头。队列又称为“先进先出”或“后进后出”的线性表,体现了“先到先服务”的原则。

【例 5】下列关于栈的描述正确的是()。

- A) 在栈中只能插入元素而不能删除元素
B) 在栈中只能删除元素而不能插入元素
C) 栈是特殊的线性表,只能在一端插入或删除元素
D) 栈是特殊的线性表,只能在一端插入元素,而在另一端删除元素

答案:C)

解析:栈是一种特殊的线性表。在这种特殊的线性表中,其插入和删除操作只能在线性表的一端进行。

【例 6】下列叙述中正确的是()。

- A) 线性链表是线性表的链式存储结构
B) 栈与队列是非线性结构
C) 双向链表是非线性结构
D) 只有根结点的二叉树是线性结构

答案:A)

解析:根据数据结构中各数据元素之间前后关系的复杂程度,可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下列两个条件:① 有且只有一个根结点;② 每个结点最多有一个前驱,也最多有一个后继,则称该数据结构为线性结构,也称线性表。若不满足上述条件,则称之为非线性结构。线性表、栈与队列、线性链表都是线性结构,而二叉树是非线性结构。

【例 7】对图 0-3 中的二叉树进行后序遍历的结果为()。

- A) ABCDEF
B) DBE AFC
C) ABDECF
D) DEBFCA

答案:D)

解析:执行后序遍历,依次执行如下操作:

- ① 首先按照后序遍历的顺序遍历根结点的左子树。
- ② 然后按照后序遍历的顺序遍历根结点的右子树。
- ③ 最后访问根结点。

【例 8】下列数据结构中,能用二分法进行查找的是()。

- A) 顺序存储的有序线性表
B) 线性链表
C) 二叉链表
D) 有序线性链表

答案:A)

解析:二分法查找只适用于顺序存储的有序表。所谓有序表是指线性表中的元素按值非递减排列(即从小到大,但允许相邻元素值相等)。

【例 9】对于长度为 n 的线性表,下列各排序法所对应的比较次数中正确的是()。

- A) 冒泡排序为 $n/2$
B) 冒泡排序为 n
C) 快速排序为 n
D) 快速排序为 $n(n-1)/2$

答案:D)

解析:假设线性表的长度为 n ,则冒泡排序需要经过 $n/2$ 遍的从前往后扫描和 $n/2$ 遍的从后往前扫描,需要比较次数为 $n(n-1)/2$ 。快速排序法在最坏的情况下,比较次数也是 $n(n-1)/2$ 。

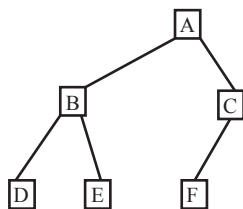


图 0-3

0.2 程序设计基础

考核概率 65%

考点 9 程序设计方法与风格

1. 程序设计方法

程序设计是指设计、编制、调试程序的方法和过程。

程序设计方法是研究问题求解如何进行系统构造的软件方法学。常用的程序设计方法有结构化程序设计方法、软件工程方法和面向对象方法。

2. 程序设计风格

程序设计风格是指编写程序时所表现出的特点、习惯和逻辑思路。良好的程序设计风格可以使程序结构清晰合理,程序代码便于维护。因此,程序设计风格深深地影响着软件的质量和维持。要形成良好的程序设计风格,主要应注重和考虑的因素包括以下方面:

- 源程序文档化。
- 数据说明方法。
- 语句的结构。
- 输入和输出。

考点 10 结构化程序设计

1. 结构化程序设计的原则

结构化程序设计方法的主要原则可以概括为自顶向下、逐步求精、模块化及限制使用 goto 语句。

- 自顶向下:程序设计时,应先考虑总体,后考虑细节;先考虑全局目标,后考虑具体问题。
- 逐步求精:将复杂问题细化,细分为逐个小问题依次求解。
- 模块化:是把程序要解决的总目标分解为若干目标,再进一步分解为具体的小目标,把每个小目标称为一个模块。
- 限制使用 goto 语句。

2. 结构化程序设计的基本结构

结构化程序设计有 3 种基本结构,即顺序结构、选择结构和循环结构,其基本形式如图 0-4 所示。

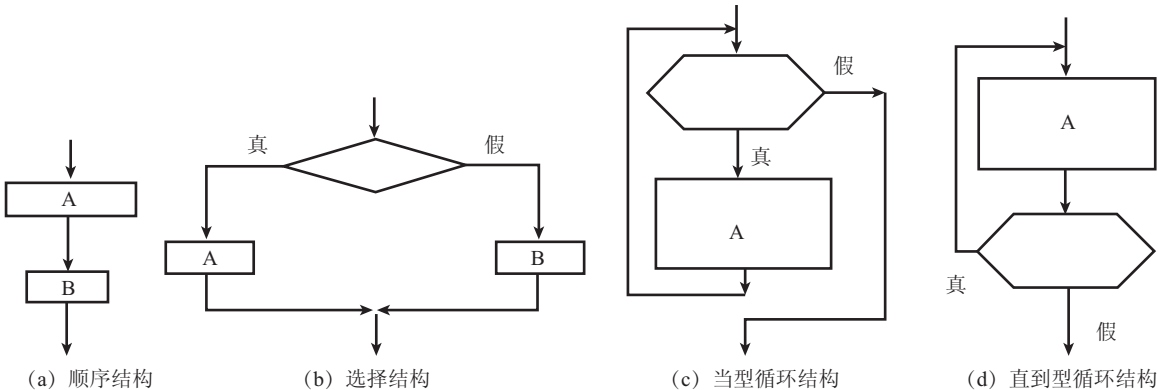


图 0-4 结构化程序设计的基本结构

3. 结构化程序设计的原则、方法和应用

结构化程序设计是一种面向过程的程序设计方法。在结构化程序设计的具体实施中,需要注意以下问题。

- 使用程序设计语言的顺序、选择、循环等有限的控制结构表示程序的控制逻辑。
- 选用的控制结构只准许有一个入口和一个出口。
- 程序语句组成容易识别的块,每块只有一个入口和一个出口。
- 复杂结构应该使用嵌套的基本控制结构进行组合嵌套来实现。
- 语言中没有的控制结构,应该采用前后一致的方法来模拟。
- 严格控制 goto 语句的使用。

考点 11 面向对象的程序设计

1. 面向对象方法的本质

面向对象方法的本质就是主张从客观世界固有的事物出发来构造系统,提倡用人类在现实生活中常用的思维方法来认识、理解和描述客观事物,强调最终建立的系统能够映射问题域。

2. 面向对象方法的优点

面向对象方法有以下主要优点:

- 与人类习惯的思维方法一致。
- 稳定性好。
- 可重用性好。
- 易于开发大型软件产品。
- 可维护性好。

3. 面向对象方法的基本概念

(1) 对象。

对象是面向对象方法中最基本的概念。对象可以用来表示客观世界中任何实体,它既可以是具体的物理实体的抽象,也可以是人为概念,或者是任何有明确边界和意义的东西。

(2) 类。

类是具有共同属性、共同方法的对象的集合,是关于对象的抽象描述,反映属于该对象类型的所有对象的性质。

(3) 实例。

实例是指一个具体对象则是其对应类的一个实例。

(4) 消息。

消息是一个实例与另一个实例之间传递的信息,它请求对象执行某一处理或回答某一要求的信息,它统一了数据流和控制流。

(5) 继承。

继承是使用已有的类定义作为基础建立新类的定义技术。在面向对象技术中,类组成为具有层次结构的系统:一个类的上层可有父类,下层可有子类;一个类直接继承其父类的描述(数据和操作)或特性,子类自动地共享基类中定义的数据和方法。

(6) 多态性。

对象根据所接受的信息而做出动作,同样的消息被不同的对象接受时可以有完全不同的行动,该现象称为多态性。

TIPS 小提示

当使用“对象”这个术语时,既可以指一个具体的对象,也可以泛指一般的对象,但是当使用“实例”这个术语时,必须是指一个具体的对象。

真题题解

【例1】在下列叙述中,不属于良好程序设计风格要求的是()。

- A) 程序的效率第一,清晰第二
- B) 程序的可读性好
- C) 程序中要有必要的注释
- D) 输入数据前要有提示信息

答案:A)

解析:著名的“清晰第一,效率第二”的论点已经成为主导的程序设计风格,所以选项A是错误的,其余选项都是良好程序设计风格的要求。

【例2】下列选项中不符合良好程序设计风格的是()。

- A) 源程序要文档化
- B) 数据说明的次序要规范化
- C) 避免滥用 goto 语句
- D) 模块设计要保证高耦合、高内聚

答案:D)

解析:良好的程序设计风格使程序结构清晰合理,使程序代码便于维护。应注意和考虑的因素主要有:① 源程序要文档化;② 数据说明的次序要规范化;③ 语句的结构应简单直接,不应该为提高效率而把语句复杂化,避免滥用 goto 语句;④ 模块设计要保证低耦合、高内聚。

【例3】下列选项中不属于结构化程序设计方法的是()。

- A) 自顶向下
- B) 逐步求精
- C) 模块化
- D) 可复用

答案:D)

解析:20世纪70年代以来,提出了许多软件设计方法,主要包括① 逐步求精。对复杂的问题,应设计一些子目标作过渡,逐步细化。② 自顶向下。程序设计时,应先考虑总体,后考虑细节;先考虑全局目标,后考虑局部目标。一开始不要过多追求细节,先从最上层总目标开始设计,逐步使问题具体化。③ 模块化。一个复杂问题肯定是由若干相对简单的问题构成。模块化是把程序要解决的总目标分解为分目标,再进一步分解为具体的小目标,把每个小目标称为一个模块。而可复用是面向对象程序设计的一个优点,不是结构化程序设计方法。

【例4】在面向对象方法中,实现信息隐蔽是依靠()。

- A) 对象的继承
- B) 对象的多态
- C) 对象的封装
- D) 对象的分类

答案:C)

解析:对象是由数据和操作组成的封装体,与客观实体有直接的对应关系。对象之间通过传递消息互相联系,以模拟现实世界中不同事物彼此之间的关系。面向对象技术的3个重要特性为封装性、继承性和多态性。

0.3 软件工程基础

考核概率 85%

考点 12 软件工程基本概念

1. 软件定义与软件特点

(1) 软件的定义。

软件(Software)是与计算机系统的操作有关的计算机程序、规程、规则,以及可能有的文件、文档及数据。计算机软件由两部分组成:一是机器可执行的程序和数据;二是机器不可执行的,与软件开发、运行、维

护、使用等有关的文档。

(2) 软件的特点。

软件主要包括以下几个特点:

- 软件是一种逻辑实体,具有抽象性。
- 软件的生产与硬件不同,它没有明显的制作过程。
- 软件在运行、使用期间,不存在磨损、老化问题。
- 软件的开发、运行对计算机系统具有依赖性,受计算机系统的限制,这就导致了软件移植的问题。
- 软件复杂性高、成本昂贵。
- 软件开发涉及诸多的社会因素。

2. 软件危机与软件工程

(1) 软件危机。

软件危机泛指在计算机软件的开发和维护中所遇到的一系列严重问题。具体地说,在软件开发和维护过程中,软件危机主要表现在以下几个方面:

- 软件需求的增长得不到满足。
- 软件的开发成本和进度无法控制。
- 软件质量难以保证。
- 软件不可维护或维护程度非常低。
- 软件的成本不断提高。
- 软件开发生产率的提高赶不上硬件的发展和应用需求的增长。

总之,可以将软件危机归结为成本、质量、生产率等问题。

(2) 软件工程。

软件工程是应用于计算机软件的定义、开发和维护的一整套方法、工具、文档、实践标准和工序。

软件工程包括软件开发技术和软件工程管理。软件工程包括 3 个要素,即方法、工具和过程。软件的核心思想是把软件产品看做是一个工程产品来处理。

3. 软件工程过程与软件生命周期

(1) 软件工程过程。

软件工程过程是把输入转化成为输出的一组彼此相关的资源和活动。

(2) 软件生命周期。

通常,将软件产品从提出、实现、使用维护到停止使用的过程称为软件生命周期。

软件生命周期主要包括软件定义、软件开发及软件运行维护三个阶段。其中软件生命周期的主要活动阶段包括可行性研究与计划制定、需求分析、软件设计、软件实现、软件测试和运行维护。

4. 软件工程的目标与原则

(1) 软件工程的目标。

软件工程需达到的目标是:在给定成本、进度的前提下,开发出具有有效性、可靠性、可理解性、可维护性、可重用性、可适应性、可移植性、可追踪性和可互操作性且满足用户需求的产品。

(2) 软件工程的原则。

为了实现上述软件工程目标,在软件开发过程中,必须遵循软件工程的基本原则。这些原则适用于所有的软件项目。这些基本原则包括抽象、信息隐蔽、模块化、局部化、确定性、一致性、完备性和可验证性。

5. 软件开发工具与软件开发环境

软件开发工具与软件开发环境的使用提高了软件的开发效率、维护效率和软件质量。

(1) 软件开发工具。

软件开发工具的产生、发展和完善促进了软件的开发速度和质量的提高。软件开发工具从初期的单项

工具逐步向集成工具发展。与此同时,软件开发的各种方法也必须得到相应的软件工具的支持,否则方法就很难有效地实施。

(2)软件开发环境。

软件开发环境是全面支持软件开发过程的软件工具集合。这些软件工具按照一定的方法或模式组合起来,支持软件生命周期的各个阶段和各项任务的完成。

计算机辅助软件工程(CASE)是当前软件开发环境中富有特色的研究工作和发展方向。CASE 将各种软件工具、开发机器和一个存放过程信息的中心数据库组合起来,形成软件工程环境。一个良好的工程环境将最大限度地降低软件开发的技术难度并使软件开发的质量得到保证。

考点 13 结构化分析方法

1. 需求分析和需求分析方法

(1)需求分析。

软件需求是指用户对目标软件系统在功能、行为、性能、设计约束等方面的期望。

需求分析的任务是发现需求、求精、建模和定义需求的过程。需求分析将创建所需的数据模型、功能模型和控制模型。

需求分析阶段的工作可以概括为 4 个方面,即需求获取、需求分析、编写需求规格说明书和需求评审。

(2)需求分析方法。

常用的需求分析方法有结构化分析方法和面向对象分析方法。

2. 结构化分析方法

(1)结构化分析方法介绍。

结构化分析方法是结构化程序设计理论在软件需求分析阶段的应用。

结构化分析方法的实质是着眼于数据流,自顶向下,逐层分解,建立系统的处理流程,以数据流图和数字字典为主要工具,建立系统的逻辑模型。




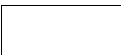
(2)结构化分析方法的常用工具。

常用工具包括数据流图(DFD)、数字字典(DD)、判断树和判断表。下面主要介绍数据流图和数字字典。

数据流图(Data Flow Diagram,DFD)是描述数据处理的工具,是需求理解的逻辑模型的图形表示,它直接支持系统的功能建模。

数据流图从数据传递和加工的角度,来刻画数据流从输入到输出的移动变换过程。数据流图中的主要图形元素及说明如表 0-1 所示。

表 0-1 数据流程图中主要图形元素及说明

图形	说明
	加工(转换):输入数据经加工产生输出
	数据流:沿箭头方向传送数据,一般在旁边标注数据流名
	存储文件:表示处理过程中存放各种数据的文件
	数据的源点/终点:表示系统和环境的接口,属系统之外的实体

数据字典是结构化分析方法的核心。数据字典是对所有与系统相关的数据元素的一个有组织的列表,以及明确的、严格的定义,使得用户和系统分析员对于输入、输出、存储成分和中间计算结果有共同的理解。通常数据字典包含的信息有名称、别名、何处使用/如何使用、内容描述、补充信息等。在数据字典中有 4 种

类型的条目,即数据流、数据项、数据存储和加工。

TIPS 小提示

数据流图与程序流程图中用箭头表示的控制流有本质的不同,千万不要混淆。此外,数据存储和数据流都是数据,仅仅是所处的状态不同。数据存储是处于静止状态的数据,数据流是处于运动中的数据。

3. 软件需求规格说明书

软件需求规格说明书是需求分析阶段的最后结果,是软件开发中的重要文档之一。

软件需求规格说明书的标准主要有正确性、无歧义性、完整性、可验证性、一致性、可理解性、可修改性和可追踪性。

考点 14 结构化设计方法

1. 软件设计的基本概念及方法

(1) 软件设计的基础。

软件设计是软件工程的重要阶段,是一个把软件需求转换为软件表示的过程。软件设计的基本目标是用比较抽象概括的方式确定目标系统如何完成预定的任务,即软件设计是确定系统的物理模型。

(2) 软件设计的基本原理。

软件设计遵循软件工程的基本目标和原则,建立了适用于在软件设计中应该遵循的基本原理和与软件设计有关的概念,主要包括抽象、模块化、信息隐藏及模块的独立性。下面主要介绍模块独立性的一些度量标准。

模块的独立程度是评价设计好坏的重要度量标准。衡量软件的模块独立性的定性度量标准是使用耦合性和内聚性。

耦合性是模块间互相联结的紧密程序的度量。内聚性是一个模块内部各个元素间彼此结合的紧密程度的度量。通常较优秀的软件设计,应尽量做到高内聚、低耦合。

(3) 结构化设计方法。

结构化设计就是采用最佳可能方法设计系统的各个组成部分及各成分之间的内部联系的技术。也就是说,结构化设计是这样一个过程,它决定用哪些方法把哪些部分联系起来,才能解决好某个具体的、有清楚定义的问题。

结构化设计方法的基本思想是将软件设计成由相对独立、单一功能的模块组成的结构。

TIPS 小提示

一般来说,要求模块之间的耦合尽可能弱,即模块尽可能独立,且要求模块的内聚程度尽可能高。内聚性和耦合性是一个问题的两个方面,耦合性程度弱的模块,其内聚程度一定高。

2. 概要设计

(1) 概要设计的任务。

软件概要设计的任务主要包括以下几个方面:

- 设计软件系统结构。
- 数据结构及数据库设计。
- 编写概要设计文档。
- 概要设计文档评审。

(2) 面向数据流的设计方法。

在需求分析设计阶段,产生了数据流图。面向数据流的设计方法定义了一些不同的映射方法,利用这些映射方法可以把数据流图变换成结构图表示的软件结构。DFD 从系统的输入数据流到系统的输出数据流的一连串连续加工形成了一条信息流。下面首先介绍数据流图的不同类型。

数据流图的信息流可分为两种类型:变换流和事务流。相应的,数据流图有两种典型的结构形式:变换型和事物型。

面向数据流的结构化设计过程如下:

- ①确认数据流图的类型(是事务型还是交换型)。
- ②说明数据流的边界。
- ③把数据流图映射为程序结构。
- ④根据设计准则对产生的结构进行优化。

(3) 结构化设计的准则。

大量的实践表明,以下的设计准则可以借鉴为设计的指导和对软件结构图进行优化的条件。

- 提高模块独立性。
- 模块规模应该适中。
- 深度、宽度、扇入和扇出都应适当。
- 模块的作用域应该在控制域之内。
- 降低模块之间接口的复杂程度。
- 设计单入口、单出口的模块。
- 模块功能应该可以预测。

TIPS 小提示

扇出过大意味着模块过分复杂,需要控制和协调过多的下级模块;扇出过小时可以把下级模块进一步分解成若干个子功能模块,或者合并到它的上级模块中。扇入越大则共享该模块的上级模块数目越多,这是有好处的,但是,不能牺牲模块的独立性而单纯追求高扇入。大量实践表明,设计得好的软件结构通常顶层扇出比较高,中层扇出较少,底层模块有高扇入。

3. 详细设计

(1) 详细设计的任务。

详细设计的任务是为软件结构图中的每一个模块确定实现算法和局部数据结构,用某种选定的表达工具表示算法和数据结构的细节。

(2) 详细设计的工具。

常见的过程设计工具包括以下内容:

- 图形工具:程序流程图、N-S、PAD 及 HIPO。
- 表格工具:判定表。
- 语言工具:PDL(伪码)。

考点 15 软件测试

软件测试是保证软件质量的重要手段,其主要过程涵盖了整个软件生命期的过程,包括需求定义阶段的需求测试、编码阶段的单元测试、集成测试,以及其后的确认测试、系统测试,验证软件是否合格、能否交付用户使用等。

1. 软件测试的目的及准则

(1) 软件测试的目的。

软件测试是为了发现错误而执行程序的过程。

一个好的测试用例是指很可能找到迄今为止尚未发现的错误的用例。

一个成功的测试是发现了至今尚未发现的错误的测试。

(2) 软件测试的准则。

鉴于软件测试的重要性,要做好软件测试,设计出有效的测试方案和好的测试用例,软件测试人员需要充分理解和运用软件测试的一些基本准则:

- 所有测试都应追溯到用户需求。
- 严格执行测试计划,排除测试的随意性。
- 充分注意测试中的群集现象。
- 程序员应避免检查自己的程序。
- 穷举测试不可能。
- 妥善保存测试计划、测试用例、出错统计和最终分析报告,为日后的维护提供方便。

2. 软件测试技术和方法综述

软件测试的方法是多种多样的,对于软件测试方法和技术,可以从不同角度加以分类。

若从是否需要执行被测软件的角度,可以分为静态测试和动态测试方法。若按照功能划分,可以分为白盒测试和黑盒测试。

(1) 静态测试与动态测试。

静态测试不实际运行软件,主要通过人工进行分析,包括代码检查、静态结构分析、代码质量度量等。其中代码检查分为代码审查、代码走查、桌面检查、静态分析等具体形式。

动态测试是基于计算机的测试,是为了发现错误而执行程序的过程。设计高效、合理的测试用例是动态测试的关键。

测试用例就是为测试设计的数据,由测试输入数据和预期的输出结果两部分组成。测试用例的设计方法一般分为黑盒测试方法和白盒测试方法两类。

(2) 白盒测试方法与测试用例设计。

白盒测试也称结构测试或逻辑驱动测试,它根据程序的内部逻辑来设计测试用例,检查程序中的逻辑通路是否都按预定的要求正确的工作。

白盒测试的主要方法有逻辑覆盖测试、基本路径测试等。

(3) 黑盒测试方法与测试用例设计。

黑盒测试也称功能测试或数据驱动测试,它根据规格说明书的功能来设计测试用例,检查程序的功能是否符合规格说明的要求。

黑盒测试的主要诊断方法有等价类划分法、边界值分析法、错误推测法、因果图法等,主要用于软件确认测试。

3. 软件测试的实施

软件测试的实施过程主要有 4 个步骤,即单元测试、集成测试、确认测试(验收测试)和系统测试。

(1) 单元测试。

单元测试也称模块测试,模块是软件设计的最小单位,单元测试是对模块进行正确性的检验,以期能尽早发现各模块内部可能存在的各种错误。

(2) 集成测试。

集成测试也称组装测试,它是对各模块按照设计要求组装成的程序进行测试,主要目的是发现与接口有关的错误。

(3) 确认测试。

确认测试的任务是用户根据合同进行,确定系统功能和性能是否可接受。确认测试需要用户积极参

与,或者以用户为主进行。

(4) 系统测试。

系统测试由将软件系统与硬件、外设或其他元素结合在一起,对整个软件系统进行测试。

系统测试的内容包括功能测试、操作测试、配置测试、性能测试、安全测试和外部接口测试等。

考点 16 程序的调试

在对程序进行了成功的测试之后将进行程序的调试。程序调试的任务是诊断和改正程序中的错误。

本节主要讲解了程序调试的概念和方法。

1. 程序调试的基本概念

调试是作为成功测试之后出现的步骤,也就是说,调试是在测试发现错误之后排除错误的过程。软件测试贯穿整个软件生命期,而调试主要在开发阶段。

程序调试活动由两部分组成:

- 根据错误的迹象确定程序中错误的确切性质、原因和位置。
- 对程序进行修改,排除这个错误。

(1) 调试的基本步骤。

- ① 错误定位。
- ② 修改设计和代码,以排除错误。
- ③ 进行回归测试,防止引入新的错误。

(2) 调试的原则。

调试活动由对程序中错误的定性、定位和排错两部分组成,因此调试原则也从以下两个方面考虑:

- ① 确定错误的性质和位置的原则。
- ② 修改错误的原则。

2. 软件调试方法

调试的关键在于推断程序内部的错误位置及原因。从是否跟踪和执行程序的角度,类似于软件测试,分为静态调试和动态调试。静态调试主要是指通过人的思维来分析源程序代码和排错,是主要的调试手段,而动态调试是辅助静态调试的。

主要的软件调试方法有强行排错法、回溯法和原因排除法。其中强行排错法是传统的调试方法,回溯法适用于小规模程序的排错,因为排除法是通过演绎和归纳以及二分法来实现的。

真题题解

【例1】下列描述中正确的是()。

- | | |
|--------------------|---------------------|
| A) 程序就是软件 | B) 软件开发不受计算机系统的限制 |
| C) 软件既是逻辑实体,又是物理实体 | D) 软件是程序、数据与相关文档的集合 |

答案:D)

解析:计算机软件是计算机系统中与硬件相互依存的另一部分,包括程序、数据及相关文档的完整集合。软件具有如下特点:① 软件是一种逻辑实体,而不是物理实体,具有抽象性;② 软件的生产过程与硬件不同,没有明显的制作过程;③ 软件在运行、使用期间,不存在磨损、老化问题;④ 软件的开发、运行对计算机系统具有不同程度的依赖性,这导致软件移植的问题;⑤ 软件复杂性高,成本昂贵;⑥ 软件开发涉及诸多的社会因素。

【例2】从工程管理角度,软件设计一般分两步完成,它们是()。

- | | |
|----------------|--------------|
| A) 概要设计与详细设计 | B) 数据设计与接口设计 |
| C) 软件结构设计与数据设计 | D) 过程设计与数据设计 |

答案:A)

解析:从工程管理角度看,软件设计分两步完成:概要设计与详细设计。概要设计将软件需求转化为软件体系结构、确定

系统级接口、全局数据结构或数据库模式;详细设计确立每个模块的实现算法和局部数据结构,用适当方法表示算法和数据结构的细节。

【例 3】下列叙述中正确的是()。

- A) 软件测试应该由程序开发者来完成 B) 程序经调试后一般不需要再测试
C) 软件维护只包括对程序代码的维护 D) 以上三种说法都不对

答案:D)

解析:程序调试的任务是诊断和改正程序中的错误。它与软件测试不同,软件测试是尽可能多地发现软件中的错误。先要发现软件的错误,然后借助于一定的调试工具去找出软件错误的具体位置。软件测试贯穿整个软件生命周期,调试主要在开发阶段。为了实现更好的测试效果,应该由独立的第三方来构造测试。软件的运行和维护是指将已交付的软件投入运行,并在运行使用中不断地维护,根据提出的新需求进行必要而且可能的扩充和删改。

【例 4】软件调试的目的是()。

- A) 发现错误 B) 更正错误 C) 改善软件性能 D) 验证软件的正确性

答案:B)

解析:软件调试的目的是诊断和改正程序中的错误,改正以后还需要进行测试。

0.4 数据库设计基础

考核概率 100%

考点 17 数据库系统的基本概念

1. 数据、数据库、数据库管理系统、数据库系统

(1) 数据。

数据(Data)是描述事物的符号记录。

(2) 数据库。

数据库(DataBase, 简称 DB)是指长期存储在计算机内的、有组织的、可共享的数据集合。

(3) 数据库管理系统。

数据库管理系统(DataBase Management System, DBMS)是数据库的机构,它是一个系统软件,负责数据库中的数据组织、数据操纵、数据维护、控制及保护和数据服务等。

数据库管理系统的主要类型有 4 种,即文件管理系统、层次数据库系统、网状数据库系统和关系数据库系统,其中关系数据库系统的应用最广泛。

(4) 数据库系统。

数据库系统(DataBase System, DBS),是指引进数据库技术后的整个计算机系统,能实现有组织地、动态地存储大量相关数据,提供数据处理和信息资源共享的便利手段。

TIPS 小提示

在数据库系统、数据库管理系统和数据库三者之间,数据库管理系统是数据库系统的组成部分,数据库又是数据库管理系统的管理对象,因此可以说数据库系统包括数据库管理系统,而数据库管理系统又包括数据库。

2. 数据库系统的发展

数据管理发展至今已经经历了人工管理阶段、文件系统阶段和数据库系统阶段三个阶段。

一般认为,未来的数据库系统应支持数据管理、对象管理和知识管理,应该具有面向对象的基本特征。在关于数据库的诸多新技术中,面向对象数据库系统、知识库系统和关系数据库系统的扩充是比较重要的。

(1) 面向数据库系统。

用面向对象方法构筑面向对象数据库模型,使其具有比关系数据库系统更为通用的能力。

(2) 知识库系统。

用人工智能中的方法特别是用逻辑知识表示方法构筑数据模型,使其模型具有特别通用的能力。

(3) 关系数据库系统的扩充。

利用关系数据库作进一步扩展,使其在模型的表达能力与功能上有进一步的加强,如与网络技术相结合的 Web 数据库、数据仓库及嵌入式数据库等。

3. 数据库系统的基本特点

数据库系统具有数据的集成性、数据的高共享性与低冗余性、数据独立性、数据统一管理与控制等特点。

4. 数据库系统的内部机构体系

数据模式是数据库系统中数据结构的一种表示形式,具有不同的层次与结构方式。

数据库系统在其内部具有三级模式及二级映射,三级模式分别是概念模式、内模式与外模式;二级映射是外模式/概念模式的映射和概念模式/内模式的映射。三级模式与二级映射构成了数据库系统内部的抽象结构体系。

模式的三个级别层次反映了模式的三个不同环境及它们的不同要求,其中内模式处于底层,它反映了数据在计算机物理结构中的实际存储形式;概念模式处于中层,它反映了设计者的数据全局逻辑要求,而外模式位于最外层,它反映了用户对数据的要求。

TIPS 小提示

一个数据库只有一个概念模式和一个内模式,但是有多个外模式。

考点 18 数据模型

1. 数据模型的基本概念

数据是现实世界符号的抽象,而数据模型是数据特征的抽象。它从抽象层次上描述了系统的静态特征、动态行为和约束条件,为数据库系统的信息表示与操作提供一个抽象的框架。数据模型所描述的内容包括数据结构、数据操作及数据约束。

数据模型按不同的应用层次分为概念数据模型、逻辑数据模型和物理数据模型。

目前,逻辑数据模型也有很多种,较为成熟并先后被人们大量使用过的有层次模型、网状模型、关系模型、面向对象模型等。

2. E-R 模型

E-R 模型(实体联系模型)将现实世界的要求转化成实体、联系、属性等几个基本概念,以及它们间的两种基本连接关系,并且可以用 E-R 图非常直观地表示出来。

E-R 图提供了表示实体、属性和联系的方法。

- 实体:客观存在并且可以相互区别的事物,用矩形表示,矩形框内写明实体名。
- 属性:描述实体的特性,用椭圆形表示,并用无向边将其与相应的实体连接起来。
- 联系:实体之间的对应关系,它反映现实世界事物之间的相互联系,用菱形表示,菱形框内写明联系名。

在现实世界中,实体之间的联系可分为“一对一”的联系(简记为 1:1)、“一对多”的联系(简记为 1:n)、“多对多”的联系(简记为 M:N 或 m:n)三种类型。

3. 层次模型

层次模型是用树型结构表示实体及其之间联系的模式。在层次模型中,结点是实体,树枝是联系,从上

到下是一对多的关系。

层次模型的基本结构是树型结构,自顶向下,层次分明。其缺点是:受文件系统影响大,模型受限制多,物理成分复杂,操作与使用均不理想,且不适用于表示非层次性的联系。

4. 网状模型

网状模型是用网状结构表示实体及其之间联系的模型。可以说,网状模型是层次模型的扩展,表示多个从属关系的层次结构,呈现一种交叉关系。

网状模型是以记录型为结点的网络,它反映现实世界中较为复杂的事物间的联系。

网状模型结构如图 0-5 所示。

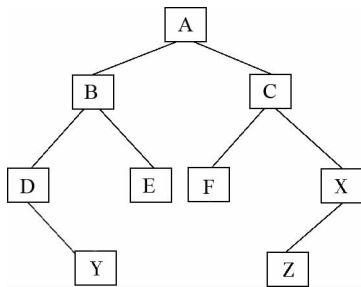


图 0-5 网状模型结构示意图

5. 关系模型

(1) 关系的数据结构。

关系模型采用二维表来表示,简称表。二维表由表框架及表的元组组成。表框架是由 n 个命名的属性组成, n 称为属性元数。每个属性都有一个取值范围称为值域。表框架对应了关系的模式,即类型的概念。在表框架中按行可以存放数据,每行数据称为元组。

在二维表中唯一能标识元组的最小属性集称为该表的键(或码)。二维表中可能有若干个键,它们称为该表的候选码(或候选键)。从二维表的候选键中选取一个作为用户使用的键称为主键(或主码)。如表 A 中的某属性集是某表 B 的键,则称该属性集为 A 的外键(或外码)。

关系是由若干个不同的元组所组成的,因此关系可视为元组的集合。

(2) 关系的操纵。

关系模型的数据操纵即是建立在关系上的数据操纵,一般有数据查询、增加、删除及修改 4 种操作。

(3) 关系中的数据约束。

关系模型允许定义三类数据约束:实体完整性约束、参照完整性约束和用户定义的完整性约束。其中前两种完整性约束由关系数据库系统自动支持。对于用户定义的完整性约束,则由关系数据库系统提供完整性约束语言,用户利用该语言写出约束条件,运行时由系统自动检查。

考点 19 关系代数

1. 传统的集合运算

(1) 关系并运算。

若关系 R 和关系 S 具有相同的结构,则关系 R 和关系 S 的并运算记为 $R \cup S$,表示由属于 R 的元组或属于 S 的元组组成。

(2) 关系交运算。

若关系 R 和关系 S 具有相同结构,则关系 R 和关系 S 的交运算记为 $R \cap S$,表示由既属于 R 的元组又属于 S 的元组组成。

(3) 关系差运算。

若关系 R 和关系 S 具有相同的结构,则关系 R 和关系 S 的差运算记为 $R - S$,表示由属于 R 的元组且不

属于 S 的元组组成。

(4) 广义笛卡儿积。

分别为 n 元和 m 元的两个关系 R 和 S 的广义笛卡儿积 $R \times S$ 是一个 $(n \times m)$ 元组的集合。其中的两个运算对象 R 和 S 的关系可以是同类型也可以是不同类型。

2. 专门的关系运算

专门的关系运算有选择、投影、连接等。

(1) 选择。

从关系中找到满足给定条件元组的操作称为选择。选择的条件以逻辑表达式给出,使得逻辑表达式为真的元组将被选取。选择又称为限制。它是在关系 R 中选择满足给定选择条件 F 的诸元组,记作:

$$\sigma_F(R) = \{t | t \in R \wedge F(t) = \text{'真'}\}$$

其中,选择文件 F 是一个逻辑表达式,取逻辑值“真”或“假”。

(2) 投影。

从关系模式中指定若干个属性组成新的关系称为投影。

关系 R 上的投影是从关系 R 中选择出若干属性列组成新的关系,记作:

$$\pi_A(R) = \{t[A] | t \in R\}$$

其中 A 为 R 中的属性列。

(3) 连接。

也称为 θ 连接,它是从两个关系的笛卡儿积中选取满足条件的元组,记作:

$$R \bowtie_{A\theta B} S = \{t_r t_s | t_r \in R \wedge t_s \in S \wedge t_r[A] \theta t_s[B]\}$$

其中, A 和 B 分别为 R 和 S 上度数相等且可比的属性组。 θ 是比较运算符。

连接运算是从广义笛卡儿积 $R \times S$ 中选取 R 关系在 A 属性组上的值与 S 关系在 B 属性组上值满足关系 θ 的元组。

连接运算中有两种最为重要且常用的连接,一种是等值连接;另一种是自然连接。

θ 为“=”的连接运算称为等值连接,是从关系 R 与关系 S 的广义笛卡儿积中选取 A 、 B 属性值相等的元组,则等值连接为:

$$R \bowtie_{A=B} S = \{t_r t_s | t_r \in R \wedge t_s \in S \wedge t_r[A] = t_s[B]\}$$

自然连接(Natural-join)是一种特殊的等值连接,它要求两个关系中进行比较的分量必须是相同的属性组,并且在结果中去掉重复的属性列,则自然连接可记作:

$$R \bowtie S = \{t_r t_s | t_r \in R \wedge t_s \in S \wedge t_r[B] = t_s[B]\}$$

考点 20 数据库设计与管理的

数据库设计是数据库应用的核心。

1. 数据库设计概述

数据库设计的基本任务是根据用户对象的信息需求、处理需求和数据库的支持环境设计出数据模型。

数据库设计的基本思想是过程迭代和逐步求精。数据库设计的根本目标是要解决数据共享问题。

在数据库设计中有两种方法:

- 面向数据的方法,是以处理信息需求为主,兼顾处理需求。
- 面向过程的方法,是以处理需求为主,兼顾信息需求。

其中,面向数据的方法是主流的设计方法。

目前,数据库设计一般采用生命周期法,即将整个数据库应用系统的开发分解成目标独立的若干阶段,包括需求分析阶段、概念设计阶段、逻辑设计阶段、物理设计阶段、编码阶段、测试阶段、运行阶段、进一步修改阶段。

2. 数据库设计的需求分析

需求收集和分析是数据库设计的第一阶段,这一阶段收集到的基础数据和一组数据流图是下一步设计概念结构的基础。需求分析的主要工作有绘制数据流程图、数据分析、功能分析、确定功能处理模块和数据之间的关系。

需求分析和表达经常采用的方法有结构化分析方法和面向对象的方法。结构化分析方法用自顶向下、逐层分解的方式分析系统。数据流图表达了数据和处理过程的关系,数据字典对系统中数据的详尽描述,是各类数据属性的清单。

数据字典是各类数据描述的集合,它通常包括 5 个部分,即数据项,是数据的最小单位;数据结构,是若干数据项有意义的集合;数据流,可以是数据项,也可以是数据结构,表示某一处理过程的输入和输出;数据存储,处理过程中存取的数据,常常是手工凭证、手工文档或计算机文件;处理过程。

数据字典是在需求分析阶段建立,在数据库设计过程中不断修改、充实、完善的。

3. 数据库的概念设计

(1)数据库概念设计。

数据库概念设计的目的是分析数据间内在的语义关联,在此基础上建立一个数据的抽象模型。

数据库概念设计的方法主要有两种:

- 集中式模式设计法。
- 视图集成设计法。

(2)数据库概念设计的过程。

使用 E - R 模型与视图集成法进行设计时,需要按以下步骤进行:

- ①选择局部应用。
- ②视图设计。
- ③视图集成。

4. 数据库的逻辑设计

(1)从 E - R 图向关系模式转换。

从 E - R 图到关系模式的转换是比较直接的,实体与联系都可以表示成关系,在 E - R 图中属性也可转换成关系的属性。实体集也可转换成关系,如表 0-2 所示。

表 0-2 在 E - R 模型与关系间的比较

E - R 模型	关系	E - R 模型	关系
属性	属性	实体集	关系
实体	元组	联系	关系

如联系类型为 1:1,则每个实体的码均是该关系的候选码。

如联系类型为 1: N,则关系的码为 N 端实体的码。

如联系类型为 M: N,则关系的码为诸实体的组合。具有相同码的关系模式可合并。

(2)逻辑模式规范化。

在关系数据库设计中存在的问题有数据冗余、插入异常、删除异常和更新异常。

数据库规范化的目的在于消除数据冗余和插入/删除/更新异常。规范化理论有 4 种范式,从第一范式到第四范式的规范化程度逐渐升高。

(3)关系视图设计。

关系视图是在关系模式的基础上所设计的直接面向操作用户的视图,它可以根据用户需求随时创建。

5. 数据库的物理设计

(1)数据库物理设计的概念。

数据库在物理设备上的存储结构与存取方法称为数据库的物理结构,它依赖于给定的计算机系统。为一个给定的逻辑模型选取一个最符合应用要求的物理结构的过程,就是数据库的物理设计。

(2)数据库物理设计的主要目标。

数据库物理设计的主要目标是对数据库内部物理结构作调整并选择合理的存取路径,以提高数据库访问速度及有效利用存储空间。

6. 数据库管理

数据库是一种共享资源,它需要维护与管理,这种工作称为数据库管理,而实施此项管理的人称为数据库管理员(DBA)。

数据库管理包括数据库的建立、数据库的调整、数据库的重组、数据库安全性与完整性控制、数据库故障恢复和数据库监控。

真题题解

【例1】下列叙述中正确的是()。

- A) 数据库系统是一个独立的系统,不需要操作系统的支持
- B) 数据库技术的根本目标是要解决数据的共享问题
- C) 数据库管理系统就是数据库系统
- D) 以上说法都不对

答案:B)

解析:数据库系统(DataBase System, DBS),是由数据库(数据)、数据库管理系统(软件)、计算机硬件、操作系统及数据库管理员组成。作为处理数据的系统,数据库技术的主要目的就是解决数据的共享问题。

【例2】在数据库系统中,用户所见的数据模式为()。

- A) 概念模式
- B) 外模式
- C) 内模式
- D) 物理模式

答案:B)

解析:概念模式是数据库系统中对全局数据逻辑结构的描述,是全体用户(应用)公共数据视图,它主要描述数据的记录类型及数据间关系,还包括数据间的语义关系等。数据库管理系统的三级模式结构由外模式、模式、内模式组成。数据库的外模式也叫做用户级数据库,是用户所看到和理解的数据库,是从概念模式导出的子模式,用户可以通过子模式描述语言来描述用户级数据库的记录,还可以利用数据语言对这些记录进行操作。内模式(或存储模式、物理模式)是指数据在数据库系统内的存储介质上的表示,是对数据的物理结构和存取方式的描述。

【例3】下列说法中正确的是()。

- A) 为了建立一个关系,首先要构造数据的逻辑关系
- B) 表示关系的二维表中各元组的每一个分量还可以分成若干个数据项
- C) 一个关系的属性名称为关系模式
- D) 一个关系可以包含多个二维表

答案:A)

解析:元组已经是数据的最小单位,不可再分;关系的框架称为关系模式;关系框架与关系元组一起构成了关系,即一个关系对应一张二维表。选项A)中指出,在建立关系前,需要先构造数据的逻辑关系是正确的。

【例4】用树型结构表示实体之间联系的模型是()。

- A) 关系模型
- B) 网状模型
- C) 层次模型
- D) 以上都是

答案:C)

解析:数据模型是指反映实体及其实体间联系的数据组织的结构和形式,有关系模型、网状模型和层次模型等。其中层次模型实际上是以记录型为结点构成的树,它把客观问题抽象为一个严格的、自上而下的层次关系,所以,它的基本结构是树型结构。

【例5】设有如下3个关系表:

R		
A	B	C
1	1	2
2	2	3

S		
A	B	C
3	1	3

T		
A	B	C
1	1	2
2	2	3
3	1	3

下列操作中正确的是()。

A) $T = R \cap S$

B) $T = R \cup S$

C) $T = R \times S$

D) $T = R / S$

答案:C)

解析:集合的并、交、差、广义笛卡儿积:设有两个关系为 R 和 S ,它们具有相同的结构, R 和 S 的并是由属于 R 和 S ,或者同时属于 R 和 S 的所有元组组成,记作 $R \cup S$; R 和 S 的交是由既属于 R 又属于 S 的所有元组组成,记作 $R \cap S$; R 和 S 的差是由属于 R 但不属于 S 的所有元组组成,记作 $R - S$; 元组的前 n 个分量是 R 的一个元组,后 m 个分量是 S 的一个元组,若 R 有 $K1$ 个元组, S 有 $K2$ 个元组,则 $R \times S$ 有 $K1 \times K2$ 个元组,记为 $R \times S$ 。从图中可见,关系 T 是关系 R 和关系 S 的简单扩充,而扩充的符号为“ \times ”,故答案为 $T = R \times S$ 。

【例 6】在下列关系运算中,不改变关系表中的属性个数但能减少元组个数的是()。

A) 并

B) 交

C) 投影

D) 笛卡儿乘积

答案:B)

解析:关系的基本运算有两类:传统的集合运算(并、交、差)和专门的关系运算(选择、投影、连接)。集合的并、交、差:设有两个关系为 R 和 S ,它们具有相同的结构, R 和 S 的并是由属于 R 和 S ,或同时属于 R 和 S 的所有元组组成,记作 $R \cup S$; R 和 S 的交是由既属于 R 又属于 S 的所有元组组成,记作 $R \cap S$; R 和 S 的差是由属于 R 但不属于 S 的所有元组组成,记作 $R - S$ 。因此,在关系运算中,不改变关系表中的属性个数但能减少元组(关系)个数的只能是集合的交。

【例 7】在 E - R 图中,用来表示实体之间联系的图形是()。

A) 矩形

B) 椭圆形

C) 菱形

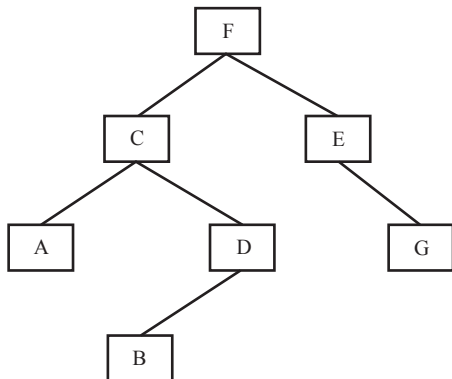
D) 平行四边形

答案:C)

解析:E - R 图中规定,用矩形表示实体,椭圆形表示实体属性,菱形表示实体关系。

0.5 同步练习

1. 对下列二叉树进行中序遍历的结果是()。



A) ACBDFEG

B) ACBDFGE

C) ABDCGEF

D) FCADBEG

2. 按照“后进先出”原则组织数据的数据结构是()。

A) 队列

B) 栈

C) 双向链表

D) 二叉树

3. 下列叙述中正确的是()。

- A) 一个逻辑数据结构只能有一种存储结构
B) 数据的逻辑结构属于线性结构,存储结构属于非线性结构
C) 一个逻辑数据结构可以有多种存储结构,且各种存储结构不影响数据处理的效率
D) 一个逻辑数据结构可以有多种存储结构,且各种存储结构影响数据处理的效率
4. 下面选项中不属于面向对象程序设计特征的是()。
A) 继承性 B) 多态性 C) 类比性 D) 封装性
5. 下列叙述中正确的是()。
A) 软件交付使用后还需要进行维护 B) 软件一旦交付使用就不需要再进行维护
C) 软件交付使用后其生命周期就结束 D) 软件维护是指修复程序中被破坏的指令
6. 下列描述中正确的是()。
A) 软件工程只是解决软件项目的管理问题
B) 软件工程主要解决软件产品的生产率问题
C) 软件工程的主要思想是强调在软件开发过程中需要应用工程化原则
D) 软件工程只是解决软件开发中的技术问题
7. 在软件设计中,不属于过程设计工具的是()。
A) PDL(过程设计语言) B) PAD 图 C) N-S 图 D) DFD 图
8. 数据库设计的4个阶段是需求分析、概念设计、逻辑设计和()。
A) 编码设计 B) 测试阶段 C) 运行阶段 D) 物理设计
9. 数据库技术的根本目标是要解决数据的()。
A) 存储问题 B) 共享问题 C) 安全问题 D) 保护问题
10. 数据库独立性是数据库技术的重要特点之一。所谓数据独立性是指()。
A) 数据与程序独立存放 B) 不同的数据被存放在不同的文件中
C) 不同的数据只能被对应的应用程序所使用 D) 以上3种说法都不对
11. 下列关于栈的叙述正确的是()。
A) 栈是非线性结构 B) 栈是一种树状结构
C) 栈具有“先进先出”的特征 D) 栈具有“后进先出”的特征
12. 结构化程序设计所规定的三种基本控制结构是()。
A) 输入、处理、输出 B) 树型、网型、环型
C) 顺序、选择、循环 D) 主程序、子程序、函数
13. 下列叙述正确的是()。
A) 算法的效率只与问题的规模有关,而与数据的存储结构无关
B) 算法的时间复杂度是指执行算法所需要的计算工作量
C) 数据的逻辑结构与存储结构是一一对应的
D) 算法的时间复杂度与空间复杂度一定相关
14. 在结构化程序设计中,模块划分的原则是()。
A) 各模块应包括尽量多的功能 B) 各模块的规模尽量大
C) 各模块之间的联系应尽量紧密 D) 模块内具有高内聚度,模块间具有低耦合度
15. 某二叉树中有 n 个度为2的结点,则该二叉树中的叶子结点数为()。
A) $n+1$ B) $n-1$ C) $2n$ D) $n/2$

0.6 同步练习答案

1. A) 2. B) 3. D) 4. C) 5. A) 6. C) 7. D) 8. D)
9. B) 10. D) 11. D) 12. C) 13. B) 14. D) 15. A)

第 1 章 C 语言概述

考核知识点

- 程序的组成、main() 函数和其他函数
- 头文件、数据说明、函数的开始和结束标志
- 源程序的书写格式
- C 语言的风格
- C 语言的数据类型及其定义方法
- 不同类型数据间的转换与运算

重要考点提示

- C 程序的源文件、目标文件和可执行文件的生成过程
- 标识符的命名规则
- 数据类型的转换及取值范围

1.1 C 语言基础知识

考核概率 80%

考点 1 C 语言概述

C 语言是一种结构紧凑、使用方便、程序执行效率高的编程语言,它有 9 种控制语句、32 个关键字(见表 1-1)和 34 种运算符。C 语言的数据结构也非常丰富,它的多种数据类型可以实现如链表、树、栈等复杂的运算,并且用结构化控制语句(if...else,for 语句等)来实现函数的模块化。C 语言的语法不太严格,程序设计自由度大,它可以直接访问物理地址,还可以直接对硬件操作。C 语言也是一种移植性比较好的语言。

表 1-1 C 语言关键字

auto	break	case	char	const	continue	default
double	else	enum	extern	float	for	goto
int	long	register	return	short	signed	sizeof
do	if	static	struct	switch	typedef	union
unsigned	void	volatile	while			

考点 2 C 语言程序的构成

- (1)C 语言的源程序是由函数构成的,每一个函数完成相对独立的功能,其中至少包括一个主函数(main()函数)。
- (2)C 程序总是从 main()函数开始执行。
- (3)C 语言规定每个语句以分号(;)结束,分号是语句组成不可缺少的部分,它在每条语句的最后出现。
- (4)程序的注释部分应括在“/*”与“*/”之间,“/”和“*”之间不能有空格,注释部分允许出现在程序的任何位置。

【例 1】显示“How are you !”的 C 语言程序。

```
#include <stdio.h>

main()                                /* 主函数 */
{
    printf(" How are you ! \n ");    /* 调用库函数 printf()显示字符串 */
}
```

运行结果是在屏幕上显示一行英文：“How are you !”。

例题说明：

(1)本程序是由一个 main()函数构成的。main 是函数名,函数名后面圆括号内是填写参数的,由于本程序主函数没有参数,所以是空的,但括号不能省略。main()后面有一对花括号,花括号内是由语句组成的函数体,本程序只有一个语句。

(2)printf()函数是 C 语言的库函数,它的功能是在屏幕上输出指定的内容,“\n”是转义字符,它代表回车换行。

(3)关于转义字符见表 1-2。

表 1-2 C 语言的转义字符及功能

字 符 形 式	功 能
\n	换行
\t	横向跳格 (代表【Tab】键)
\v	竖向跳格
\b	退格符(代表【Backspace】键)
\r	回车符号
\f	走纸换页符
\\	反斜杠字符“\”
\'	单引号(撇号)字符
\ddd	1~3 位八进制数所代表的一个 ASCII 字符
\xhh	1~2 位十六进制数所代表的一个 ASCII 字符
\0	空值
\"	双引号(撇号)字符

考点 3 C 程序的生成过程

C 程序是先由源文件经编译生成目标文件,然后经过连接生成可执行文件,如图 1-1 所示。

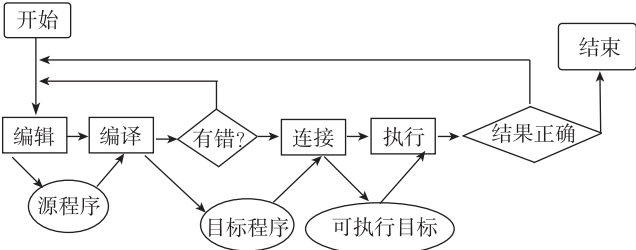


图 1-1 编译过程

源程序的扩展名为 .c,目标程序的扩展名为 .obj,可执行程序的扩展名为 .exe。

真题题解

【例 1】C 语言程序的基本单位是()。

- A) 函数 B) 过程 C) 子程序 D) 子例程

答案:A)

解析:在 C 语言中任何程序都是由一个或多个函数构成的。而过程、子程序和子例程都不是 C 语言中的概念。

【例 2】下列各选项中,合法的 C 语言关键字是()。

- A) integer B) sin C) string D) void

答案:D)

解析:参考表 1-1,选项 A)是在其他语言中的关键字,C 语言为了简化语法,则使用了“int”。选项 B)虽然是一个被库函数使用的特殊标识符,但并不是 C 语言的关键字。选项 C)不是 C 语言的关键字。选项 D)是表示一个“空”的 C 语言关键字。

【例 3】下列选项中,是 C 语言提供的合法的关键字的是()。

- A) swith B) cher C) default D) Case

答案:C)

解析:选项 A)和选项 B)为拼写错误,选项 D)中出现了大写字母。

【例 4】C 语言的程序一行写不下时,应该()。

- A) 用回车符换行 B) 在任意一个空格处换行
C) 用分号换行 D) 用逗号换行

答案:B)

解析:C 语言可以在任何一个分隔符或空格处换行。

【例 5】以下说法正确的是()。

- A) C 语言程序是从第一个定义的函数开始执行
B) 在 C 语言程序中,要调用的函数必须在 main() 函数中定义
C) C 语言程序是从 main() 函数开始执行
D) C 语言程序中的 main() 函数必须放在程序的开始部分

答案:C)

解析:C 语言程序总是从程序的 main() 函数开始执行。main() 函数可以放在 C 程序的任何位置,包括最前面和最后面。C 程序中的函数可以任意地相互调用,它们之间的关系是平等的。

1.2 常量、变量和数据类型

考核概率 100%

C 语言提供的数据结构是以数据类型的形式出现的,且有常量与变量之分,如图 1-2 所示。

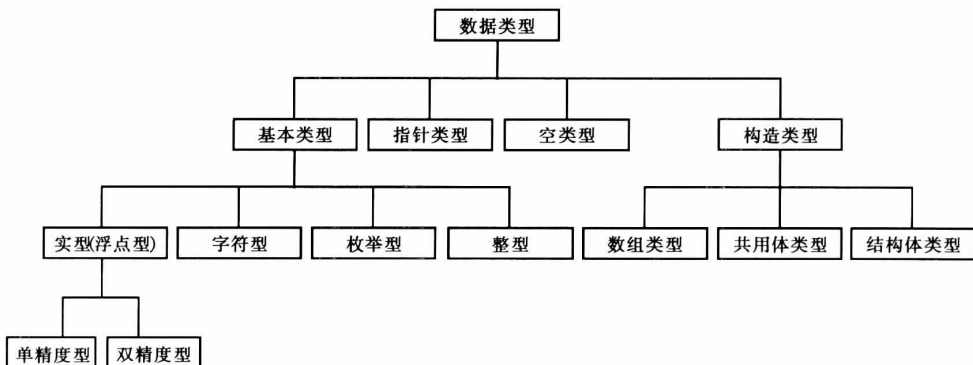


图 1-2 数据类型

考点 4 标识符

在编写程序时,必须为函数、变量等命名,这个名字称为标识符。标识符可以是一个或多个字符,标识符的第一个字符必须是字母或下画线,随后的字符只能是字母、数字或下画线。要区分字母的大小写,例如,t和T是两个不同的变量。下面的标识符是合法的:

a5 , Data , _str

以下都是非法的标识符:

#sum , 8m , str3:2 , i-j

标识符不能与程序中具有特殊意义的关键字相同,不能与用户编制的函数名、C语言库函数相同,在程序中各种标识符尽量不要重复,以便区分。选择变量名和其他标识符时,应注意做到“见名知义”。

考点 5 常量

在程序运行中,其值不能被改变的量称为常量。常量有5种类型:整型常量、实型常量、字符常量、字符串常量和符号常量。

1. 整型常量

整型常量有3种形式:十进制整型常量、八进制整型常量和十六进制整型常量。

下面举例说明几种常量的书写方式。

十进制整型常量:564 , 0 , -23 , 85L 等,基本数字范围为0~9。

八进制整型常量:061 ,037L , -026 ,0773 等,基本数字范围为0~7。

十六进制整型常量:0x66 , 0x1101 , 0x , 0x5AC0 , -0xFF,基本数字范围为0~9,从10~15写为A~F或a~f。

其中L为长整型。

2. 实型常量

实型常量有两种表示形式:小数形式和指数形式。

小数形式: 45.3 0.000744 -623.0

指数形式: 45.3e0 4.53e-3 -4.53e2

[2]4.53e1 0.453e-2 -453e0

[2]453.0e-1 453e-5 -0.453e3

TIPS 小提示

(1) 小数部分为0的实型常量,可以写为453.0或依照人们日常习惯写为453。

(2) 用小数表示时,小数点的两边必须有数,不能写成“.453”和“453.”,而应该写成“0.453”和“453.0”。

(3) 用指数写法时,e前必须有数字,e后面的指数必须为整数。

3. 字符常量

一个字符常量代表ASCII码字符集里的一个字符,在程序中用单撇号括起来,以便区分,如'a'、'p'、'w'。

注意: 'a' 和 'A' 是两个不同的字符常量。

除了形式上的字符常量外,C语言还有特殊的字符常量,如转义字符常量'\n'。其中“\”是转义的意思,后面跟不同的字符表示不同的意思,具体请参阅表1-2。

4. 字符串常量

字符串常量是用双撇号括起来的一个或一串字符。注意其与字符常量的区别。书写形式如"china"、

"How are you"、" "、"@ shou"、"342mono"。

5. 符号常量

符号常量是由宏定义“#define”定义的常量,在 C 程序中可用标识符代表一个常量。

【例 2】定义一个常量并输出。

```
#include <stdio. h>
#define PI 3. 14159
main()
{
    float a;
    a = PI;          /* PI 表示 3. 14159 */
    printf(" %f ",a);
}
```

运行结果是在屏幕上显示:3. 141590。

例题说明:

#define 是宏定义,有关宏定义在以后的章节中详细说明,此程序中所有出现 PI 的地方都代表 3. 14159,同时 PI 称为符号常量。习惯上我们用大写字母来表示符号常量,小写字母表示变量,这样比较容易区别。

考点 6 变量

变量就是其值可以改变的量。变量要有变量名,在内存中占据一定的存储单元,存储单元里存放的是该变量的值。不同类型的变量其存储单元的大小不同,变量在使用前必须定义。

1. 整型变量

整型变量分为 4 种:基本型(int)、短整型(short int 或 short)、长整型(long int 或 long)和无符号型(unsigned int ,unsigned short,unsigned long)。

C 标准没有具体规定各类数据所占内存的字节数,如基本型变量(int)在 IBM PC 上占 16 位,在 IBM 370 机型上占 32 位,而在 Honeywell 机上占 36 位。

现以 IBM PC 为例,说明各类整型变量所占的位数及可表达的数的范围,见表 1-3。

表 1-3 各类整型变量所表示数的范围

类 型	所 占 位 数	数 的 范 围	说 明
[signed] int	16	- 32768 ~ 32767	整型
[signed] short [int]	16	- 32768 ~ 32767	短整型
[signed] long int	32	- 2147483648 ~ 2147483647	长整型
unsigned [int]	16	0 ~ 65535	无符号整型
unsigned short [int]	16	0 ~ 65535	无符号短整型
unsigned long [int]	32	0 ~ 4294967295	无符号长整型

2. 实型变量

实型变量分为单精度类型(float)和双精度类型(double)两种。如:

```
float a , b ;
double m ;
```

在一般的系统中,float 型数据在内存中占 4 字节(32 位),double 型数据占 8 字节。单精度实数提供 7 位有效数字,双精度实数提供 15 ~ 16 位有效数字。实型常量不分 float 型和 double 型,一个实型常量可以赋给一个 float 型或 double 型变量,但变量根据其类型截取实型常量中相应的有效数字。

3. 字符变量

字符变量用来存放字符常量,字符变量用关键字 char 说明,每个字符变量中只能存放一个字符。

定义形式:

```
char cr1 , cr2 ;
```

赋值:

```
cr1 = 'm' , cr2 = 'n' ;
```

将一个字符赋给一个字符变量时,并不是将该字符本身存储到内存中,而是将该字符对应的 ASCII 码存储到内存单元中。例如,字符 'A' 的 ASCII 码为 65,在内存中的存放形式如下:

```
01000001
```

由于在内存中字符以 ASCII 码存放,它的存储形式和整数的存储形式类似,所以 C 语言中字符型数据与整型数据之间可以通用,一个字符能用字符的形式输出,也能用整数的形式输出,字符数据也能进行算术运算,此时相当于对它们的 ASCII 码进行运算(ASCII 码具体值详见附录 A)。

考点 7 类型的自动转换和强制转换

当同一表达式中各数据的类型不同时,编译程序会自动把它们转变成同一类型后再进行计算。转换优先级为:

```
char < int < float < double
```

即左边级别“低”的类型向右边转换。具体地说,若在表达式中优先级最高的数据是 double 型,则此表达式中的其他数据均被转换成 double 型,且计算结果也是 double 型;若在表达式中优先级最高的数据是 float 型,则此表达式中的其他数据均被转换成 float 型,且计算结果也是 float 型。

在做赋值运算时,若赋值号左右两边的类型不同,则赋值号右边的类型向左边的类型转换;当右边的类型高于左边的类型时,则在转换时对右边的数据进行截取。

除自动转换外,还有强制转换,表示形式是:

```
( 类型 )(表达式);
```

有关表达式及赋值知识将在下章做详细介绍。

真题题解

【例 1】下列叙述不正确的是()。

- A) 在 C 程序中,% 是只能用于整数运算的运算符
- B) 在 C 程序中,无论是整数还是实数,都能正确无误地表示
- C) 若 a 是实型变量,C 程序中 a=20 是正确的,因此实型变量允许被整型数赋值
- D) 在 C 程序中,语句之间必须要用分号“;”分隔

答案:B)

解析:选项 B) 只对了一半,整数可以无误地表示,而实数并不都是正确无误的。

【例 2】以下选项中正确的整型常量是()。

- A) 34.1
- B) -80
- C) 2,000
- D) 1 2 3

答案:B)

解析:本题是考查 C 语言的十进制整型常量。选项 A) 后边有小数点,所以不对。选项 C) 和选项 D) 在数字间有逗号和空格,也显然不对。

【例 3】在 C 程序中,可以作为用户标识符的一组标识符是()。

- A) void define WORD
- B) as_b3 _224 Else
- C) Switch -wer case
- D) 4b DO SIG

答案:B)

解析:选项 A) 中的 void 是 C 语言的关键字。选项 C) 中的 -wer 前边有一个字符是减号,而 case 是关键字。选项 D) 中的 4b 是以数字开头。

【例 4】TURBO C 中,int 类型变量所占字节数是()。

- A) 1
- B) 2
- C) 3
- D) 4

答案:B)

解析:TURBO C 中 int 类型变量占两个字节,数值范围是 $-32768 \sim +32767$ 。

【例 5】下列不合法的十六进制数是()。

- A) 0xff B) 0Xcde C) 0x11 D) 0x23

答案:A)

解析:十六进制的基本数字范围是 $0 \sim 9, a \sim f$ 或 $A \sim F$ 。十六进制中代表数字的字母也可以用大写字母。但开头必须以数字 0 和字母 x 或大写 X 开头。选项 A) 则是以字母 o 开头,所以是错误的。

【例 6】在 C 语言中,下列合法的字符常量是()。

- A) '\039' B) '\x76' C) 'ab' D) '\o'

答案:B)

解析:因选项 C) 和选项 D) 是字符串的形式,所以较易排除,关键是选项 A) 和选项 B) 的取舍,由于选项 A) 后既不是八进制数,也不是十六进制数,所以可以排除。验证一下选项 B) 的形式,可知其属于“\”后加十六进制数的形式,正确。

【例 7】C 语言中定义了一个变量,该变量代表内存中的一个()。

- A) 区域 B) 单元 C) 地址 D) 容量

答案:C)

解析:C 语言中定义的一个变量代表内存中的一个地址,也就是在内存中分配给这个变量一个单元,用来存放变量的值,这个内存单元的大小根据变量的类型不同而不同。

【例 8】若 int 类型数据占两个字节,则下列语句的输出结果为()。

```
int k = -1;
printf("%d,%u\n", k, k);
```

A) -1, -1 B) -1, 32767 C) -1, 32768 D) -1, 65535

答案:D)

解析:此题是考查 C 语言对有符号和无符号整型数据的处理,区别在于对数据最高位的解释上。对于一个有符号整数,C 编译程序将该值(二进制代码)的最高位作为符号标志位(符号标志位是 0,表示正数;若为 1,则表示负数);而对于一个无符号整数,该数的最高位将被作为数值位处理。设 int 型数据占两个字符,则 -1 在内存中应以 16 位全 1 来表示,因此当把 k 值按 %d 格式输出(即按有符号整数处理)时,仍为 -1;而按 %u 格式输出(即按无符号整数处理)时,为 65535(2 的 16 次方减 1)。

【例 9】已知字母 a 的 ASCII 码为 97,则执行下列语句后输出为()。

```
char a = 'a';
a--;
printf("%d,%c", a + '2' - '0', a + '3' - '0');
```

- A) a, c
B) a -- 运算不合法,故有语法错误
C) 98, c
D) 格式描述和输出项不匹配,输出无定值

答案:C)

解析:C 语言规定,所有的字符常量都可以作为整型常量来处理,因而字符常量也可参与算术运算。本例中字符变量 a 的初值为字符'a',其对应的整数值就是它的 ASCII 码 97,经过自减运算 a-- 后,变量 a 所对应的整数值为 96;虽然题中没有给出字符'2'和'0'的 ASCII 码值,但我们应该知道'2' - '0'是将字符 2 转换成整数 2 的表达式,所以 a + '2' - '0'的值是 98。同理可以推知 a + '3' - '0'的值是 99,按照格式描述符的输出应该是 c。

【例 10】若 ch 为 char 型变量,k 为 int 型变量(已知字符 a 的 ASCII 码是 97),则执行下列语句后的输出为()。

```
ch = 'a';
k = 12;
printf("%x,%o", ch, ch, k);
printf("k = %d\n", k);
```

- A) 因变量类型与格式描述符的类型不匹配,输出无定值
B) 输出项与格式描述符个数不符,输出为 0 或不定值
C) 61, 141, k = %d

D)61,141,k=%12

答案:C)

解析:在C语言中,字符数据既可以用字符形式输出,也可以用整数形式输出。本题的第一个printf语句的格式说明输出应当是61,141,多余的输出项k将不予输出;第二个printf语句,由于格式说明符中包含两个连续的%字符,根据C语言的规定,两个%将不再作为格式描述符使用,而是处理成字符“%”原样输出,因此在这里输出项k没有对应的格式描述符,将不予输出。

【例11】sizeof(double)是()。

A)一种函数调用

B)一个整型表达式

C)一个双精度表达式

D)一个不合法的表达式

答案:B)

解析:在C语言中,一个函数的调用格式是:函数名(参数列表)。虽然“sizeof(double)”与函数调用有相同的格式,但sizeof是一个C语言关键字,因此,它不是一个函数调用。sizeof在C语言中表示求一个变量或数据类型所占用的内存字节数的运算符,所以“sizeof(double)”表示求双精度浮点数据类型的内存占用字节数。显然,该表达式返回的是一个整数,而不是一个双精度数。

【例12】以下叙述中不正确的是()。

A)在C程序中所有的变量必须先定义后使用

B)在程序中,APH和aph是两个不同的变量

C)若a和b类型相同,在执行了赋值语句a=b后,b中的值将放入a中,b中的值不变

D)当输入数值时,对于整型变量只能输入整型值;对于实型变量只能输入实型值

答案:D)

解析:C语言是一种类型要求严格的语言,所以所有变量在使用之前都必须定义或说明,以便让编译程序知道该变量的类型。先定义变量还有一个用途,即让编译程序能够为该变量分配一定的存储空间。C语言对于语言中所有标识符都区分大小写,并且规定一定的长度。C语言中的数值型变量具有一定的自动匹配功能,即如果一个变量是实型的,也可以赋予它整型变量。

1.3 同步练习

1. 以下选项中合法的用户标识符是()。

A)int

B)a#

C)5mem

D)_243

2. C语言中的简单数据类型有()。

A)整型、实型、逻辑型

B)整型、字符型、逻辑型

C)整型、实型、字符型

D)整型、实型、字符型、逻辑型

3. 以下选项中正确的整型常量是()。

A)-37

B)32,758

C)3 2 6

D)6.

4. 以下选项中合法的C语言字符常量是()。

A)'\\t'

B)"A"

C)67

D)A

5. 以下选项中不正确的实型常量是()。

A).123

B)1e4

C)3.640E-1

D)0.35

6. 以下选项中合法的C语言赋值语句是()。

A)a=b=34

B)a=34,b=34

C)--i;

D)m=(int)(x+y);

7. 不合法的十六进制数是()。

A)0xff

B)0X11

C)0x1g

D)0Xabc

8. ()是构成C语言程序的基本单位。

A)函数

B)变量

C)子程序

D)语句

9. 在C语言中,char型数据在内存中是以()形式存储的。

A)原码

B)补码

C)ASCII码

D)反码

10. 设int类型的数据长度为两个字节,则unsigned int类型数据的取值范围是()。

- A) 0 ~ 255 B) 0 ~ 65535 C) - 32768 ~ 32767 D) - 256 ~ 255
11. 以下()是不正确的转义字符。
A) '\\\\' B) '\\'' C) '020' D) '\\0'
12. 一个 C 语言程序总是从()开始执行。
A) 主过程 B) 主函数 C) 子程序 D) 主程序
13. 以下叙述不正确的是()。
A) 一个 C 源程序可由一个或多个函数组成
B) 一个 C 源程序必须包含一个 main() 函数
C) C 程序的基本组成单位是函数
D) 在 C 程序中,注释说明只能位于一条语句的后面
14. C 语言规定:在一个源程序中,main() 函数的位置()。
A) 必须在最开始 B) 必须在系统调用的库函数的后面
C) 可以任意 D) 必须在最后
15. 以下叙述正确的是()。
A) 在 C 程序中,main() 函数必须位于程序的最前面
B) C 程序的每行中只能写一条语句
C) C 语言本身没有输入/输出语句
D) 在对一个 C 程序进行编译的过程中,可发现注释中的拼写错误
16. 在 C 语言中,数字 029 是一个()。
A) 八进制数 B) 十六进制数 C) 十进制数 D) 非法数
17. 为求出 return 语句返回计算 100! 的结果,此函数的类型说明应为()。
A) int B) long C) unsigned long D) 选项 A), B), C) 都不对
18. C 语言中的标识符只能由字母、数字和下画线 3 种字符组成,且第一个字符()。
A) 必须为字母 B) 必须为下画线
C) 必须为字母或下画线 D) 可以是字母、数字和下画线中任意一种字符
19. 若有代数式 $\frac{3ae}{bc}$, 则不正确的 C 语言表达式是()。
A) a/b/c * e * 3 B) 3 * a * e/b/c C) 3 * a * e/b * c D) a * e/c/b * 3

1.4 同步练习答案

1. D) 2. C) 3. A) 4. A) 5. A) 6. D) 7. C) 8. A)
9. C) 10. B) 11. C) 12. B) 13. D) 14. C) 15. C) 16. D)
17. D) 18. C) 19. C)

第 2 章 运算符与表达式

考核知识点

- C 语言运算符的种类、运算符优先级和结合性
- 不同类型数据间的转换与运算
- C 语言表达式的各种类型和求值规则
- 位运算符的含义及使用
- 简单的位运算

重要考点提示

- 运算符的优先级及表达式的求值规则
- 复合的赋值表达式的使用,自增和自减运算符及逻辑运算符的使用
- 简单的位运算操作

2.1 C 语言运算符简介

考核概率 90%

考点 1 C 运算符简介

C 语言的运算符范围很宽,几乎把所有的基本操作都作为运算符处理,具体运算符见表 2-1。

表 2-1 C 语言运算符表

名 称	运 算 符
算术运算符	+ , - , * , / , %
关系运算符	> , >= , == , != , < , <=
位运算符	>> , << , ~ , & , , ^
逻辑运算符	! , , &&
条件运算符	? :
指针运算符	& , *
赋值运算符	=
逗号运算符	,
字节运算符	sizeof
强制运算符	(类型名)(表达式)
其他	下标,分量,函数

另外,按参与运算的对象个数,C 语言运算符可分为:单目运算符(如 !)、双目运算符(如 + , -)和三目运算符(如 ? :)。本章只对其中的几种进行讲解,关系运算符和逻辑运算符将在以后的章节中讲解。

考点 2 运算符的结合性和优先级

(1)在 C 语言的运算符中,所有的单目运算符、条件运算符、赋值运算符及其扩展运算符,结合方向都是从右向左,其余运算符的结合方向是从左向右。

(2)各类运算符优先级的比较:

初等运算符 > 单目运算符 > 算术运算符(先乘除后加减) > 关系运算符 > 逻辑运算符(不包括“!”) > 条件运算

符 > 赋值运算符 > 逗号运算符。

说明:以上优先级别由左到右递减,初等运算符优先级最高,逗号运算符优先级最低。

考点 3 强制类型转换运算符

可以利用强制类型转换符将一个表达式转换成所需类型,其一般形式为:

(类型名)(表达式)

例如:

(char)(x + y); 将(x + y)的值强制转换为字符型。

(double)(m * n); 将(m * n)的值强制转换为 double 类型。

TIPS

小提示

表达式应用括号括起来,如果写成(char)x + y,则表示只将 x 转化为字符型,然后与 y 相加。

考点 4 逗号运算符和逗号表达式

用逗号运算符将几个表达式连接起来,例如 a = b + c, b = a * a, c = a + b, 称为逗号表达式。

一般形式为:

表达式 1, 表达式 2, 表达式 3, ..., 表达式 n

逗号表达式的求解过程是:先求解表达式 1,然后依次求解表达式 2,直到表达式 n 的值。整个逗号表达式的值就是表达式 n 的值。需要注意的是,逗号运算符是所有运算符中级别最低的。

2.2 算术运算符和算术表达式

考核概率 100%

考点 5 基本的算术运算符

(1) + (加法运算符或正值运算符,如 2 + 6)。

(2) - (减法运算符或负值运算符,如 6 - 3)。

(3) * (乘法运算符,如 2 * 8)。

(4) / (除法运算符,如 6 / 5)。

(5) % (模运算符或称求余运算符,% 两侧均应为整型数据,如 9 % 7 的值为 2)。

需要说明的是:两个整数相除的结果为整型,如 5 / 3 的结果值为 1,舍去小数部分,如果参加 +、-、*、/ 运算的两个数中有一个数为实数,则结果是 double 类型。

考点 6 算术表达式和运算符的优先级与结合性

算术表达式是用算术运算符和括号将运算量(也称操作数)连接起来的、符合 C 语言语法规则的表达式。运算对象包括函数、常量和变量等。

在计算机语言中,算术表达式的求值规律与数学中的四则运算的规律类似,其运算规则和要求如下。

(1)在算术表达式中,可使用多层圆括号,但括号必须配对。运算时从内层圆括号开始,由内向外依次计算各表达式的值。

(2)在算术表达式中,对于不同优先级的运算符,可按运算符的优先级由高到低进行运算,若表达式中运算符的优先级相同,则按运算符的结合方向进行运算。

(3) 如果一个运算符两侧的操作数类型不同,则先利用自动转换或强制类型转换,使两者具有相同类型,然后进行运算。

考点 7 自增自减运算符

作用:使变量的值增1或减1。

如: $++i$, $--i$ (在使用 i 之前,先使 i 的值加1、减1)。

$i++$, $i--$ (在使用 i 之后,使 i 的值加1、减1)。

TIPS 小提示

(1) 只有变量才能用自增运算符($++$)和自减运算符($--$),而常量或表达式不能用,如 $10++$ 或 $(x+y)++$ 都是不合法的。

(2) $++$ 和 $--$ 的结合方向是“自右向左”,如 $-i++$, i 的左边是负号运算符,右边是自增运算符,负号运算和自增运算都是“自右向左”结合的,相当于 $-(i++)$ 。

在循环语句中常用到自增(减)运算符,在指针中也常用到该运算符,考生要弄清楚“ $i++$ ”和“ $++i$ ”及“ $i--$ ”和“ $--i$ ”的区别,防止用错。

真题题解

【例1】以下程序的输出结果为()。

```
#include <stdio.h>
main()
{
    int i=4,a;
    a=i++;
    printf("a=%d,i=%d",a,i);
}
```

A) $a=4,i=4$

B) $a=5,i=4$

C) $a=4,i=5$

D) $a=5,i=5$

答案:C)

解析:本题考查的是自增运算符及赋值运算符的综合使用问题。自增运算符是一元运算符,其优先级比赋值运算符高,要先计算。把表达式 $i++$ 的值赋予 a ,由于 $i++$ 的结果为当前 i 的值(当前 i 的值为4),所以 $i++$ 的值为4,得到 a 的值为4。同时,计算了 $i++$ 后, i 由4变为5。

【例2】在C语言中,要求参加运算的数必须是整数的运算符是()。

A) %

B) /

C) !

D) **

答案:A)

解析:选项A)中,%是求余运算符,它的运算对象必须是整型。选项B)中,“/”是除法运算符,它的运算对象可以是整型也可以是实型。选项C)中,“!”是求逻辑“非”的运算符,它是一个单目运算符,在它右边可以是任意合法的表达式。选项D)中所示的“**”用于指针运算。

【例3】对于条件表达式 $(M)?(a++):(a--)$,其中的表达式 M 等价于()。

A) $M==0$

B) $M==1$

C) $M!=0$

D) $M!=1$

答案:C)

解析:因为条件表达式 $e1?e2:e3$ 的含义是 $e1$ 为真时,其值等于表达式 $e2$ 的值,否则为表达式 $e3$ 的值。“为真”就是“不等于假”,因此 M 等价于 $M!=0$ 。

2.3 赋值运算符和赋值表达式

考核概率 100%

考点 8 赋值运算符和赋值表达式

赋值符号“=”就是赋值运算符,作用是将一个数据赋给一个变量或将一个变量的值赋给另一个变量,由赋值运算符组成的表达式称为赋值表达式。一般形式为:

变量名 = 表达式

在程序中可以多次给一个变量赋值,每赋一次值,与它相应的存储单元中的数据就被更新一次,内存中当前的数据就是最后一次所赋值的那个数据。

考点 9 复合的赋值运算符

在赋值运算符之前加上其他运算符可以构成复合赋值运算符。其中与算术运算有关的复合运算符是:
+=, -=, *=, /=, %=。

TIPS 小提示

两个符号之间不可以有空格,复合赋值运算符的优先级与赋值运算符的相同。表达式 $n += 1$ 等价于 $n = n + 1$,作用是取变量 n 中的值增 1 再赋给变量 n ,其他复合的赋值运算符的运算规则依次类推。

如求表达式 $a += a -= a * a$ 的值,其中 a 的初值为 12。

步骤:(1)先进行“ $a -= a * a$ ”运算,相当于 $a = a - a * a = 12 - 144 = -132$ 。

(2)再进行“ $a += -132$ ”运算,相当于 $a = a + (-132) = -132 - 132 = -264$ 。

考点 10 赋值运算中的类型转换

如果赋值运算符两侧的类型不一致,在赋值前系统将自动先把右侧表达式求得的数值按赋值号左边变量的类型进行转换(也可以用强制类型转换的方式),但这种转换仅限于某些数据之间,通常称为“赋值兼容”。对于另一些数据,例如,后面将要讨论的地址值,就不能赋给一般的变量,称为“赋值不兼容”。常用的转换规则如下:

(1)当实型数据赋值给整型变量时,将实型数据的小数部分截断。

如 `int x;`,执行“`x = 5.21;`”后, x 的值为 5。

(2)当整型数据赋值给实型变量时,数值不变,但以浮点数形式存储到实型变量中。

如 `float x = 45;`

输出 x 的结果为 45.00000。

(3)当 `double` 类型数据赋值给 `float` 型变量时,取其前面 7 位的有效数字,存放到 `float` 型变量的存储单元中,这时数值可能溢出。

(4)当字符型数据赋值给整型变量时,由于整型变量占两个字节,而字符只占一个字节,只需将字符数据(8 位)放到整型变量低 8 位中,对该整型变量最高位进行符号扩展,其他位补零。

(5)当整型、短整型、长整型数据赋值给一个 `char` 类型变量时,将其低 8 位原封不动地送到 `char` 类型变量中(即截断)。

真题题解

【例1】若已定义 x 和 y 为 `double` 类型,则表达式“ $x = 1, y = x + 3/2$ ”的值是()。

A) 1

B) 2

C) 2.0

D) 2.5

答案:C)

解析:本题中的表达式为逗号表达式,此表达式的结果为 $y = x + 3/2$ 的值。 $y = x + 3/2$ 的运算次序为:先进行 $3/2$ 运算,两个运算数均为整型量,结果也为整型量,等于 1,此结果将与 `double` 类型数进行相加,要转换为 $1.00 \cdots 00$ 。最后将 x 的值 1 转换成 `double` 型,与 $1.00 \cdots 00$ 相加。

【例2】若变量已正确定义并赋值,符合 C 语言语法的表达式是()。

A) $a = a + 7;$ B) $a = 7 + b + c, a++$ C) $\text{int}(12.3/4)$ D) $a = a + 7 = c + b$

答案:B)

解析:选项 A) 中, $a = a + 7;$ 赋值表达式的最后有一个分号“;”, C 语言规定,语句用分号结束,所以 $a = a + 7;$ 是一条赋值语句,而不是表达式。选项 B) 中,“ $a = 7 + b + c, a++$ ”是一个逗号表达式,它由 $a = 7 + b + c$ 和 $a++$ 两个表达式组成,前者是一个赋值表达式,后者是一个自增 1 的赋值表达式,所以它是一个合法的表达式。选项 C) 中, $\text{int}(12.3/4)$ 看似是一个强制类型转换表达式,但语法规则,类型名应当放在一对圆括号内才构成强制类型转换运算符,因此写成 $(\text{int})(12.3/4)$ 才是正确的。在使用强制类型转换运算符时,需要注意运算符的优先级,例如, $(\text{int})(3.6 * 4)$ 和 $(\text{int})3.6 * 4$ 中因为 (int) 的优先级高于 $*$ 运算符,因此它们将有不同的计算结果。选项 D) 中, $a = a + 7 = c + b$ 看似是一个赋值表达式,但是在 $a + 7 = c + b$ 中,赋值号的左边是一个算术表达式 $a + 7$;按规定,赋值号的左边应当是一个变量或一个代表某个存储单元的表达式,以便把赋值号的右边的值存放在该存储单元中,因此赋值号的左边不可以是算术表达式,因为它不能代表内存中任何一个存储单元。

【例3】若 a 为整型变量,则以下语句()。

```
a = -2L;
```

```
printf("%d\n", a);
```

A) 赋值不合法

B) 输出值为 -2

C) 输出为不确定值

D) 输出值为 2

答案:B)

解析:本题的关键是要弄清楚 C 语言中常量的表示方法和有关的赋值规则。在一个整型常量后面加一个字母 `l` 或 `L`,则认为是 `long int` 型常量。一个整型常量,如果其值在 $-32768 \sim +32767$ 范围内,可以赋给一个 `int` 型或 `long int` 型变量;但如果整型常量的值超出了上述范围,而在 $-2147483648 \sim 2147483647$ 范围内,则应将其值赋给一个 `long int` 型变量。本例中 $-2L$ 虽然为 `long int` 型变量,但是其值为 -2 ,因此可以通过类型转换把长整型转换为短整型,然后赋给 `int` 型变量 a ,并按照 `%d` 格式输出该值。

【例4】若 k 为 `int` 型变量,则以下语句()。

```
k = 8567;
```

```
printf("| % -06d | \n", k);
```

A) 输出格式描述不合法

B) 输出为 |008567|

C) 输出为 |8567 |

D) 输出为 | -8567|

答案:C)

解析:因为输出格式符是 `% -06d`,含义是输出值占 6 个位置,左边对齐,右边不满 6 个补空格,其他的都原样输出。

【例5】已知字符 A 的 ASCII 码值是 65,以下程序()。

```
#include <stdio.h>
```

```
main( )
```

```
{
```

```
    char a = 'A';
```

```
    int b = 20;
```

```
    printf("%d,%o", (a = a + 1, a + b, b), a + 'a' - 'A', b);
```

```
}
```

A) 表达式非法,输出零或不确定值

B) 因输出项过多,无输出或输出不确定值

- C)输出结果为 20,141
- D)输出结果为 20,1541,20

答案:C)

解析:首先应该注意到 printf() 函数有 3 个实参数:(a=a+1,a+b,b)。a+ 'a' - 'A'和 b,并没有问题,可见选项 A)错误。由于格式控制字符串“%d,%o”中有两个描述符项,而后面又有表达式,因此,必定会产生输出,选项 B)也是错误的。既然控制字符串中只有两个格式描述符,输出必然只有两个数据,故选项 D)错误。

2.4 位运算

考核概率 100%

在计算机中,数据都是以二进制数形式存放的,位运算就是指对存储单元中二进制位的运算。

考点 11 位运算符和位运算

C 语言提供 6 种位运算符,见表 2-2。

表 2-2 位运算符

操 作 符	含 义	规 则
&	按位与	若两个相应的二进制位都为 1,则该位的结果为 1,否则为 0
	按位或	两个相应的二进制位中只要有一个为 1,则该位的结果为 1,否则为 0
^	按位异或	若两个二进制位同号,则结果为 0,异号则为 1
~	按位求反	按位取反,即 0 变 1,1 变 0
<<	左移	将一个数的二进制位全部左移若干位
>>	右移	将一个数的二进制位全部右移若干位

说明:(1)位运算中除“~”以外,均为双目运算符,要求两侧各有一个运算量。
(2)运算量只能是整型或字符型数据,不能为实型数据。

真题题解

【例 1】下述程序的输出结果是()。

```
#include <stdio. h >
void main( )
{
    char a = 3,b = 6;
    char c = a ^ b << 2;
    printf( " \n% d", c);
}
```

- A)27
- B)10
- C)20
- D)28

答案:A)

解析:本例中的关键是运算符的优先次序问题。因为 << 运算优先于 ^ 运算,即 $c = a ^ (b << 2) = 3 ^ (6 * 4) = 3 ^ 24 = 00000011 ^ 00011000 = 27$ 。

【例 2】下述语句的输出为()。

```
int m = - 1;
printf( " % d,% u,% o", m,m,m);
```

- A) - 1, - 1, - 1
- B) - 1, 32767, - 177777
- C) - 1, 32768, 177777
- D) - 1, 65535, 177777

答案:D)

解析:为了给出此题的正确答案,必须弄清-1在内存中的存储形式。在内存中-1是以补码的形式存储的:1111111111111111,即16个1的形式。如果将其视为有符号数,即按%d的格式输出,最高位是符号位,1表示负数,计算出原码,其值为-1;若将此数视为无符号数,即按%u的格式输出,则最高位被看做是数据位,直接计算出其值是65535;若按%o格式输出,即按八进制数输出,其值是177777。

【例3】在x值处于-2~2,4~8时值为“真”,否则为“假”的表达式是()。

- A) $(2 > x > -2) || (4 > x > 8)$
 B) $!(((x < -2) || (x > 2)) \&\&((x \leq 4) || (x > 8)))$
 C) $(x < 2) \&\& (x \geq -2) \&\& (x > 4) \&\& (x < 8)$
 D) $(x > -2) \&\& (x > 4) || (x < 8) \&\& (x < 2)$

答案:B)

解析:本题是考查关系运算和逻辑运算的混合运算。要给出此题的正确答案,首先需要了解数学上的区间在C语言中的表示方法,如x在[a,b]区间,其含义是x既大于等于a又小于等于b,相应的C语言表达式是 $x \geq a \&\& x \leq b$ 。本例中给出了两个区间,一个数只要属于其中一个区间即可,这是“逻辑或”的关系。在选项A)中,区间的描述不正确。选项B)把“!”去掉,剩下的表达式描述的是原题中给定的两个区间之外的部分,加上“!”否定正好是题中的两个区间的部分,是正确的。选项C)是恒假的,因为它的含义是x同时处于两个不同的区间内。选项D)所表达的也不是题中的区间。

【例4】已知小写字母a的ASCII码为97,大写字母A的ASCII码为65,以下程序的结果为()。

```
#include <stdio.h>
main()
{
    unsigned int a=32,b=66;
    printf("%c\n",a|b);
}
```

- A)66 B)98 C)b D)B

答案:C)

解析:先将a和b转化为二进制数,分别是:a=00100000,b=01000010。a|b的值为01100010,将结果转化为十进制数是98。由于C语言字符型数据和整型数据是通用的,因此98按字符型输出应该是小写字母b。

【例5】设有定义char a,b;若想通过a&b运算保留a的第3位和第6位的值,则b的二进制数应是()。

- A)00100100 B)11011011 C)00010010 D)01110010

答案:A)

解析:由“按位与”运算的功能可知:两个对应的二进制数只要有一个为0,“按位与”的结果就为0,只有它们均为1时结果才为1。因此,若想保留哪位上的数,就用1去“按位与”,其他位用0屏蔽掉。

2.5 同步练习

- 若有定义:int x=3,y=2; float a=2.5,b=3.5;则下面表达式的值为()。
 $(x+y)\%2+(int)a/(int)b$
 A)1.0 B)1 C)2.0 D)2
- 若x和n均是int型变量,且x的初值为12,n的初值为5,则执行下面表达式后x的值为()。
 $x\%=(n\%=2)$
 A)0 B)1 C)2 D)3
- 假设所有变量均为整型,则表达式(a=2,b=5,a++,b++,a+b)的值为()。
 A)7 B)8 C)9 D)10
- 下列程序的输出结果是()。

```
#include <stdio.h>
```

main()

```
{ double d=3.2; int x,y;
  x=1.2; y=(x+3.8)/5.0;
  printf("%d\n", d*y); }
```

A)3 B)3.2 C)0 D)3.07

5. 设 $\text{int } x=1, y=1$; 表达式 $(!x || y--)$ 的值是()。

A)0 B)1 C)2 D)-1

6. 若已定义 x 和 y 为 double 类型,则表达式 $x=1, y=x+3/2$ 的值是()。

A)1 B)2 C)2.0 D)2.5

7. 若变量 a, i 已正确定义,且 i 已正确赋值,则合法的语句是()。

A) $i = \text{int}(a)$ B) $++i$; C) $a = a++ = 5$; D) $a = \text{int}(i)$;

8. 若执行以下程序段后, $c3$ 的值为()。

```
int c1=1, c2=2, c3;
c3=1.0/c2*c1;
```

A)0 B)0.5 C)1 D)2

9. 如下程序的运行结果是()。

```
#include <stdio.h>
main( )
{ int y=3, x=3, z=1;
  printf("%d %d\n", (++x, y++), z+2);
}
```

A)3 4 B)4 2 C)4 3 D)3 3

10. 能正确表示逻辑关系“ $a \geq 10$ 或 $a \leq 0$ ”的 C 语言表达式是()。

A) $a >= 10 \text{ or } a <= 0$ B) $a >= 0 \text{ || } a <= 10$
C) $a >= 10 \text{ \&\& } a <= 0$ D) $a >= 10 \text{ || } a <= 0$

11. 若变量 x, y, z 均为 double 类型且已正确赋值,不能正确表示 $\frac{x}{y \times z}$ 的 C 语言表达式是()。

A) $x/y * z$ B) $x * (1/(y * z))$ C) $x/y * 1/z$ D) $x/y/z$

12. 设 x, y, t 均为 int 型变量,则执行语句: $x=y=3; t=++x || ++y$; 后, y 的值为()。

A)不定值 B)4 C)3 D)1

13. 设 a, b, c, d, m, n 均为 int 型变量,且 $a=5, b=6, c=7, d=8, m=2, n=2$, 则逻辑表达式 $(m=a>b) \&\& (n=c>d)$ 运算后, n 的值为()。

A)0 B)1 C)2 D)3

14. 假定 w, x, y, z, m 均为 int 型变量,有如下程序段:

```
w=1; x=2; y=3; z=4;
m=(w<x)? w:x; m=(m<y)? m:y; m=(m<z)? m:z;
则该程序运行后, m 的值是( )。
```

A)4 B)3 C)2 D)1

15. 下列程序的输出结果是()。

```
#include <stdio.h>
main( )
{
  int a=0, b=0, c=0;
  if( ++a>0 || ++b>0)
    ++c;
  printf("\na=%d, b=%d, c=%d", a, b, c);
}
```

A) $a=0, b=0, c=0$ B) $a=1, b=1, c=1$

C) a = 1, b = 0, c = 1

D) a = 0, b = 1, c = 1

16. 以下程序的输出结果是()。

```
#include <stdio.h>
main()
{
    int a=5,b=4,c=6,d;
    printf("%d\n",d=a>b?(a>c?a:c):(b));
}
```

A) 5

B) 4

C) 6

D) 不确定

17. 在 C 语言中,如果下面的变量都是 int 类型,则输出的结果是()。

```
sum = pad = 5; pad = sum ++ , pad ++ , ++ pad;
printf("%d\n",pad);
```

A) 7

B) 6

C) 5

D) 4

18. 以下程序的输出结果是()。

```
#include <stdio.h>
main()
{
    int i=010, j=10;
    printf("%d,%d\n", ++i, j--);
}
```

A) 11,10

B) 9,10

C) 010,9

D) 10,9

19. 已知 int i; float f; ,正确的语句是()。

A) (int f)%i;

B) int(f)%i;

C) int(f%i);

D) (int)f%i;

20. 已知 int j,i=1; ,执行语句 j=-i++; 后,j 的值是()。

A) 1

B) 2

C) -1

D) -2

21. 已知 int a=4,b=5; ,则执行表达式 a=a>b 后,变量 a 的值为()。

A) 0

B) 1

C) 4

D) 5

22. C 语言中,下列运算符的操作数必须是 int 类型的运算符是()。

A) %

B) /

C) --

D) ++

23. 判断 char 类型数据 c1 是否为大写字母的最简单且正确的表达式为()。

A) 'A' <= c1 <= 'Z'

B) (c1 >= 'A') & (c1 <= 'Z')

C) ('A' <= c1) AND ('Z' >= c1)

D) (c1 >= 'A') && (c1 <= 'Z')

24. 已知各变量的类型说明如下:

```
int k,a,b;
unsigned long w=5;
double x=1.42;
```

则以下不符合 C 语言语法的表达式是()。

A) x%(-3)

B) w+=-2

C) k=(a=2,b=3,a+b)

D) a+=a-=a*(a=3)

2.6 同步练习答案

- | | | | | | | | |
|--------|--------|--------|--------|--------|--------|--------|--------|
| 1. B) | 2. A) | 3. C) | 4. C) | 5. B) | 6. C) | 7. B) | 8. A) |
| 9. D) | 10. D) | 11. A) | 12. C) | 13. C) | 14. D) | 15. C) | 16. C) |
| 17. A) | 18. B) | 19. D) | 20. C) | 21. A) | 22. A) | 23. D) | 24. A) |

第3章 基本语句

考核知识点

- 表达式语句、空语句和复合语句
- 数据的输入与输出,输入/输出函数的调用
- 复合语句

重要考点提示

- 赋值语句的使用
- 格式输入/输出,原样输出的使用
- 正确使用 printf ()函数和 scanf()函数

3.1 C 语句概述

考核概率 10%

C 语言的语句用来向计算机系统发出指令,一个实际的源程序通常包含若干语句,这些语句用来完成一定的操作任务。

C 程序中的语句,按照它们在程序中出现的顺序依次执行,由这样的语句构成的程序结构称为顺序结构。

考点 1 C 语句分类

1. 控制语句

控制语句完成一定的控制功能,C 语言共有 9 种控制语句,见表 3-1。

表 3-1 控制语句

语 句	名 称
if()… else …	条件语句
switch	多分支选择语句
for()…	循环语句
while()…	循环语句
do…while()	循环语句
continue	结束本次循环语句
break	终止执行 switch 或循环语句
return	返回语句

说明:以上语句中“()”表示一个条件,“…”表示内嵌语句。

2. 其他类型语句

函数调用语句(由函数调用加一个分号构成),如 scanf("% d" ,&a) ;。

表达式语句(由一个表达式;构成一个语句),如 a = b ;。

3. 空语句

C 语言中所有语句都必须由一个分号(;)结束,如果只有一个分号如 `main(){};`,这个分号也是一条语句,称为空语句,程序执行时不产生任何动作,但表示存在着一条语句。

4. 复合语句

在 C 语言中花括号“{}”不仅可以用做函数体的开始和结束标志,同时也常用做复合语句的开始和结束标志,复合语句也可称为“语句体”。

TIPS 小提示

在 C 语言中,任何表达式都可以加上分号构成语句,如“`i++;`”。随意加“;”会导致很多逻辑上的错误,要慎用,不要乱用。复合语句中最后一个语句中最后的分号不能忽略不写。

以下我们将详细介绍几种顺序执行的语句。

真题题解

【例1】指出下列哪一个选项的程序是错误的()。

A) `#include <stdio.h>`

`main()`

{

`int x,y,z;`

`x=0;y=x+1;`

`z=x+y;`

}

C) `#include <stdio.h>`

`main()`

{

`int x,y,z;`

`x=0;y=x+1,`

`z=x+y;`

}

B) `#include <stdio.h>`

`main()`

{

`int x,y,z;`

`x=0,y=x+1;`

`z=x+y;`

}

D) `#include <stdio.h>`

`main()`

{

`int x,y,z;`

`x=0,y=x+1;`

`z=x+y,`

}

答案:D)

解析:在选项 A)中的语句没有一条是错误的。选项 B)和选项 C)中都有一个逗号表达式,在选项 B)中是:`x=0,y=x+1`;在选项 C)中是:`y=x+1,z=x+y`;所以选项 B)和选项 C)中也没有错误。选项 D)中的最后一条语句是以逗号结束的,而 C 语言中不能用逗号作为一个语句的结束标志,每一条语句最后应该是分号,所以选项 D)中的程序是错误的。

3.2 赋值语句

考核概率 10%

前面已经介绍赋值语句是由赋值表达式和末尾的分号(;)构成的。这里要提醒读者注意:“=”与“==”是两个不同的运算符,前者才是赋值运算符,而后者是关系运算符,用来进行条件判断,不能把二者混为一谈。如“`i=2;`”,功能是把数值 2 放到变量 i 中,而 `i==2` 是判断变量 i 的值是否为 2。“`j=j+1;`”在程序执行时,首先取出 j 中的值,执行加数值 1 的操作后再把新值放回到 j 中。

真题题解

【例 1】合法的 C 语言赋值语句是()。

A) $a = b = 58$

B) $k = \text{int}(a + b);$

C) $a = 58, b = 58$

D) $--i;$

答案:D)

解析:选项 A) 是一个合法的赋值表达式,但不是 C 语言的赋值语句,因为最后没有构成语句所必要的分号。选项 B) 中虽然最后有一个分号,但赋值号右边的强制类型转换符是错误的,应该写成 $(\text{int})(a + b)$,当使用强制类型转换符时,类型名一定要用圆括号括起来。选项 C) 是一个逗号表达式,但最后没有分号,因此不是语句。选项 D) 是一个由自减运算符构成的赋值表达式,且用分号结束,因此这是一个合法的赋值语句。

3.3 输入/输出的概念及其实现

考核概率 5%

(1) 数据从计算机内部向外部输出设备(如显示器、打印机等)输送的操作称为“输出”,数据从计算机外部向输入设备(如键盘、鼠标、扫描仪等)送入的操作称为“输入”。

(2) C 语言本身不提供输入/输出语句,可以通过函数来实现输入和输出的操作。

(3) 在使用 C 语言库函数时,首先要用预编译命令“`#include`”将有关的“头文件”包含到用户源文件中。这里需要用到编译预处理命令,在后面的章节中我们会详细讲到。

3.4 单个字符的输入/输出

考核概率 55%

这里介绍 C 标准 I/O 函数库中最简单的对单个字符进行输入/输出的函数 `putchar()` 和 `getchar()`。

考点 2 字符输出函数 `putchar()`

`putchar()` 函数的作用是向终端输出一个字符。

如:

```
putchar(a);
```

它输出字符变量 `a` 的值,`a` 也可以是字符型变量或整型变量。若 `a` 是整型变量,则输出的是 ASCII 码值为该变量值的那个字符。

考点 3 字符输入函数 `getchar()`

`getchar()` 函数的作用是从终端输入一个字符,`getchar()` 函数没有参数,函数值就是从输入设备得到的字符。

TIPS 小提示

`getchar()` 只能接收一个字符,`getchar()` 函数得到的字符可以赋给一个字符变量或整型变量,也可以不赋给任何变量,作为表达式的一部分。如果在一个函数中(今为 `main()` 函数)要调用 `getchar()` 和 `putchar()` 函数,在该主函数之前的包含命令“`#include <stdio.h>`”是必不可少的。

3.5 数据格式的输入与输出

考核概率 100%

考点 4 printf() 函数

printf() 函数是 C 语言提供的标准输出函数,它的作用是向终端(或系统隐含指定的输出设备)按指定格式输出若干个数据。

1. printf() 函数的一般形式

printf(格式控制,输出表列);

如:

```
printf(" %f, %d" ,x,y);
```

printf 是函数名,括号内由以下两部分组成:

(1)“格式控制”:用双引号括起来的字符串是“格式控制”字符串,它包括两种信息。

① 格式转换说明,由“%”和格式字符组成,如 %d、%s 等。上例中,当输出项为 int 型时,系统规定用 d 作为格式描述字符,因此,有“%d”。当输出项为 float 或 double 类型时,用 f 或 e 作为格式描述字符。格式描述符要与输出项一一对应且类型匹配。

② 需要原样输出的字符(通常指除了格式说明与一些转义字符外的那部分)也写在格式控制内。

(2)“输出表列”是需要输出的一些数据,可以是常量、变量或表达式。例如:

```
printf("x=%d y=%d",x,y);
```

其中,“x=%d y=%d”是格式说明;x,y 是输出表列。输出表列中的各输出项要用逗号隔开。若 x,y 的值为 7,8,以上两条输出结果为:

```
x=7 y=8
```

TIPS 小提示

在两数之间有空格,因为在两个格式说明符中间有一个空格。

2. 格式字符

可以根据需要在“%”与格式字符之间插入“宽度说明”、左对齐符号“-”、前导零符号“0”等。

(1)d 格式符,用来对十进制数进行输入/输出,其中“%d”按整型数据的实际长度输出,“%md”指定 m 为输出字段所占的宽度。

(2)o 格式符,以八进制数形式输出整数,同样可以通过如“%8o”的格式指定输出时所占的宽度。

(3)x 格式符,以十六进制数形式输出整数,同样可以通过如“%12x”的格式指定输出时所占的宽度。

(4)u 格式符,用来输出 unsigned 型数据,即输出无符号的十进制数。

(5)c 格式符,用来输出一个字符。

(6)s 格式符,用来输出一个字符串。

(7)f 格式符,用来输出实数(包括单、双精度),以小数形式输出,使整数部分全部如数输出。

(8)e 格式符,以指数形式输出实数。

(9)g 格式符,用来输出实数。

TIPS 小提示

对于 f、e、g 格式符可以用“整型数 1. 整型数 2”的形式,在指定宽度的同时来指定小数位的位数,其中,“整型数 1”用来指定输出数据所占的总宽度,“整型数 2”用来确定精度。精度对于不同的格式符有着不同的含义。当输出位数多于“整型数 2”指定的宽度时,截去右边多余的小数,并对截去的第一位小数做四舍五入处理。当输出数据的小数位数少于“整型数 2”指定的宽度时,在小数的最右边添 0,当输出的数据所占的宽度大于“整型数 1”指定的宽度时,小数位仍按上述规则处理,整数部分并不丢失。也可以用“.”整型数 2”的形式来指定小数位数,这时输出的数据所占宽度由系统决定。通常,系统对 float 类型提供 7 位有效位数,对于 double 类型提供 15 位有效位数。

3. 使用 printf() 函数时的注意事项

(1) 在格式控制字符串中,格式说明与输出项从左到右在类型上必须一一对应匹配,如不匹配将导致数据输出出现错误,如在输出 long 型数据时,一定要用 %ld 格式控制,而不能用 %d 格式控制。

(2) 在格式控制串中,格式说明与输出项的个数也要相等,如格式说明的个数多于输出项的个数,则对于多余的格式将输出不定值(或 0 值)。

(3) 在格式控制串中,除了合法的格式说明外,可以包含任意的合法字符(包括转义字符),这些字符在输出时将被“原样输出”。

(4) 如果要输出“%”,则应该在格式控制串中用两个连续的百分号“%%”来表示。

考点 5 scanf() 函数

1. scanf() 函数的一般形式

scanf(格式控制,地址表列);

其中 scanf 是函数名,“格式控制”的含义同 printf() 函数,“地址表列”由若干个变量地址组成,既可以是变量的地址,也可以是字符串的首地址(参见“字符数组”一节)。

例如:

```
scanf("%d",&a);
printf("%d",a);
```

运行时输入 123,按回车键后则会在屏幕上输出整型变量 a 的值 123。其中“%d”是格式控制字符串。&a 是输入项。其中的“&”是“取地址运算符”,&a 指 a 在内存中的地址,如需要多个输入项时,输入项之间要用逗号隔开。在实际输入时,若一次向计算机输入多个数据,则每两个数据间要以一个或多个空格间隔,也可以用回车键或跳格键【Tab】。

2. 格式说明

scanf() 函数中的格式说明也是以 % 开始,以一个格式字符结束,中间可以加入附加的字符。

说明:

(1) 对 unsigned 型变量的数据,可以用 %d、%o、%x 格式输入。

(2) 在 scanf() 函数中格式字符前可以用一个整数指定输入数据所占宽度,但对于实型数则不能指定其小数位的宽度。

(3) 在格式控制串中,格式说明的个数应该与输入项的个数相等,且要类型匹配,如不匹配,系统也不会给出出错信息,但有可能使程序的结果不正确。若格式说明的个数少于输入项的个数,scanf() 函数结束输入,多余的项继续从终端接收新的数据,若格式说明的个数多于输入项个数时,scanf() 函数同样也结束输入。

3. 使用 scanf() 函数时应注意的问题

(1) scanf() 函数中的输入项只能是地址表达式,而不能是变量名或其他内容,也就是说输入项必须是某

个存储单元的地址,这一点一定要掌握。

例如:

```
int m,n;
scanf("%d,%d",m,n);
```

是不对的,应将其中的“m,n”改为“&m,&n”。

(2)如果在“格式控制”字串中除了格式说明以外还有其他字符,则在输入数据时应输入与这些字符相同的字符。

(3)在用“%c”格式输入字符时,空格字符和转义字符都作为有效字符输入。

(4)在输入数据时,若实际输入数据少于输入项个数,scanf()函数会等待输入,直到满足条件或遇到非法字符才结束;若实际输入数据多于输入项个数,多余的数据将留在缓冲区备用,作为下一次输入操作的数据。

在输入数据时,遇到以下情况时认为输入结束。

遇空格、按“回车”或“跳格”(【Tab】)键,上述字符统一可称为“间隔符”。

TIPS 小提示

在程序运行到要求实际输入时,间隔符的数目不限,按指定的宽度结束,如“%3d”,只取3列。

scanf()函数中的格式控制串是为输入数据用的,其间的字符不能实现原样输出。若想在输入时出现提示性语言,则需要另外使用 printf 语句。

真题题解

【例1】若有说明:double a;则正确的输入语句为()。

- A) scanf("%lf",a);
C) scanf("%lf",&a)

- B) scanf("%f",&a);
D) scanf("%le",&a);

答案:D)

解析:选项 A)错,因为语句中使用的是变量 a 而不是变量 a 的地址。选项 B)错,因为 double 型应该用%lf 或%le 格式。选项 C)错,句末没有加分号。

【例2】阅读以下程序:

```
#include <stdio.h>
main()
{
    char str[10];
    scanf("%s",str);
    printf("%s\n",str);
}
```

运行上面的程序,输入字符串 HOW DO YOU DO,则程序的输出结果是()。

- A) HOW DO YOU DO
C) HOWDOYOU DO

- B) HOW
D) how do you do

答案:B)

解析:因为当从键盘输入字符串 HOW DO YOU DO 时,由于 scanf()函数输入时遇到空格结束,只将 HOW 3 个字符送到字符数组 str 中,并在其后自动加上结束符'\0'。

【例3】若变量已正确定义,以下程序段:

```
x = 5.16894;
printf("%f\n", (int)(x * 1000 + 0.5)/(float)1000);
```

的输出结果是()。

A)输出格式说明与输出项不匹配,输出无定值

B)5. 170000

C)5. 168000

D)5. 169000

答案:D)

解析:由输出语句看,输出项是一个算术表达式,分子和分母部分都采用了强制类型转换,分子部分得的是整型数,分母部分得的是实型数。按规则,若算术运算符两边类型不一致,一个是整型一个是实型时,整型数将首先转换成实型数,然后进行计算;因此,输出项是一个实型数,输出格式说明与输出项的值是匹配的,选项 A)的说法不成立。接着应当计算输出项的值,输出结果应当是 5. 169000。

【例 4】若有以下程序段:

```
int a=0,b=0,c=0;
```

```
c=(a--=a-5),(a=b,b+3);
```

```
printf("%d,%d,%d\n",a,b,c);
```

其输出结果是()。

A)3,0,-10

B)0,0,5

C)-10,3,-10

D)3,0,3

答案:B)

解析:由定义可见,a、b、c 都是整型变量,且都赋了初值 0。第二行的语句由逗号表达式组成,其中第一个:c=(a--=a-5)是赋值表达式,此赋值表达式等同于:c=(a=a-(a-5)),因为 a 的初值为 0,它首先给 a 赋 5,然后 a 把值赋给 c,所以 c 的值为 5。程序最后需要输出 a 的值,在语句 c=(a--=a-5),(a=b,b+3);中,a 的值在(a=b,b+3)中通过 a=b 被重新赋予 0,因此,a 最终的值为 0。程序最后还需输出 b 的值,但它的值与第二行的赋值语句无关,取决于初值 0。在输出语句中,每个格式说明之间都有一个逗号,应当原样输出,所以输出的结果应该如选项 B)所示。

【例 5】若有以下程序段:

```
#include <stdio.h>
```

```
main()
```

```
{
```

```
int a=2,b=5;
```

```
printf("a=%d,b=%d\n",a,b);
```

```
}
```

其输出结果是()。

A)a=%2,b=%5

B)a=2,b=5

C)a=%d,b=%d

D)a=%d,b=%d

答案:D)

解析:为了解答本题,首先需要了解 C 编译程序将如何解释格式串中的%d。C 语言规定,连续的两个百分号(%%)将按一个%字符处理,输出一个%号,所以%%d被解释为输出两个字符:%和d。根据以上分析,在格式串中没有用于整型数输出的格式说明符"%d",因此也就无法对整型变量 a 和 b 进行输出,格式串中的所有内容将按原样输出。

【例 6】以下程序段:

```
float a=3.1415;
```

```
printf("|%6.0f|\n",a);
```

其输出结果是()。

A)|3.1415|

B)|3.0|

C)|3|

D)|3.1

答案:C)

解析:在输出格式串中,最前的|号和\n之前的|号,按规定照原样输出。当在输出格式中指定输出的宽度时,输出的数据在指定宽度内右对齐。对于实型数,当指定小数位为 0 时,输出的实型数据将略去小数点和小数点后的数。

【例 7】以下程序段的输出结果是()。

```
float a=57.666;
```

```
printf(" *%010.2f*\n",a);
```

A) * 0000057. 66 * B) * 57. 66 * C) * 0000057. 67 * D) * 57. 57 *

答案:C)

解析:在输出格式串中,最前的 * 号和\n之前的 * 号,按规定照原样输出。输出格式%09. 2f 和%10. 2f 有以下共同之处:
(1)输出时,数据占有相同的宽度(10 位);(2)输出时,小数点后的小数占两位。若输出数据的小数多于两位,对小数点后的第3 位进行四舍五入处理。输出格式%010. 2f 和 %10. 2f 的不同之处是:输出格式%010. 2f 中,在宽度说明10 之前有一个0,其作用是:在数据前面的多余空格处填以数字0。例如,若 a 中的数据是 57. 666,用%10. 2f 格式输出时,输出结果是:_ _ _ _ _ 57. 67,此处_ 代表一个空格;而用%010. 2f 格式输出时,输出结果是:0000057. 67,空格用0 代替。

【例 8】若变量 c 定义为 float 类型,当从终端输入 283. 1900 后按回车键,能给变量 c 赋以 283. 19 的输入语句是()。

A) scanf("%f", c); B) scanf("%8. 4f", &c);
C) scanf("%6. 2f", &c); D) scanf("%8f", &c);

答案:D)

解析:选项 A) 中,变量 c 的前面没有求地址运算符 &,因此是错误的。选项 D) 中,在% 和格式字符 f 之间用 8 指定了数据所占的宽度,而输入的数据也占 8 位,因此可以正确输入。选项 B) 和选项 C) 中,在% 和格式字符 f 之间分别用 8. 4 和 6. 2 企图指定输入数据的宽度,这是错误的:C 语言规定,在输入时不可以在格式说明中对实型数指定小数位的宽度,只可以用一个整数来指定输入数据所占的总的宽度。

【例 9】若有以下定义语句:

```
int u = 010, v = 0x10, w = 10;  
printf( "%d, %d, %d\n", u, v, w );
```

则输出结果是()。

A) 8 ,16 ,10 B) 10 ,10 ,10
C) 8 ,8 ,10 D) 8 ,10 ,10

答案:A)

解析:本题考查两个知识点:一是整形常量的不同表示法;二是格式输出函数 printf() 的字符格式。题中“int u = 010, v = 0x10, w = 10;”语句中的变量 u、v、w 分别是八进制数、十六进制数和十进制数表示法,对应着十进制数的 8、16 和 10。而 printf() 函数中的“%d”是格式字符,表示以十进制整数形式输出输出项。

【例 10】下面程序的输出是()。

```
#include <stdio. h>  
main( )  
{  
    int k = 11;  
    printf( "%d, %o, %x\n", k, k, k );  
}
```

A) 11 ,12 ,11 B) 11 ,13 ,13
C) 11 ,013 ,0xb D) 11 ,13 ,b

答案:D)

解析:根据本章所学的格式符,可知本题为以带符号十进制数、八进制数及十六进制数的形式输出整数。在 C 语言中,常用前缀 0 表示八进制数,用前缀 0X 或 0x 表示十六进制数,但在用 printf() 函数输出时,并不输出前缀。

3.6 同步练习

1. 下面程序的输出结果是()。

```
#include <stdio. h>  
main( )  
{  
    int x = 5, y = 3;
```



```
printf( "% d\n" ,y = x/y );
```

- A)0 B)1 C)3 D)不确定的值

2. 若变量已正确定义,下面程序段的输出结果是()。

```
x = 5. 238794;
```

```
printf( "% f\n" ,(int)( x * 1000 + 0. 5)/(float)1000);
```

- A)5. 239000 B)输出格式说明与输出项不匹配,输出无定值
 C)5. 238000 D)5. 24

3. 下面语句:

```
printf( "% l% 8. 5f l\n" ,3461. 45);
```

的输出结果是()。

- A) |61. 45000| B) |3461. 450|
 C) |3461. 45000| D) |3461. 4500|

4. 与数学公式 $\sqrt{|\cos(x)|}$ 等价的 C 语言表达式是(),假定其中的 x 的单位是度数且不考虑 π 值的精度。

- A) sqrt(cos(x)) B) sqrt(abs(cos(x * 3. 14/180)))
 C) sqrt(abs(cos(x * (/180)))) D) sqrt(fabs(cos(x * 3. 14/180)))

5. 下面的程序()。

```
#include <stdio. h>
```

```
main( )
```

```
{
    int x = 3, y = 0, z = 0;
    if( x = y + z) printf( " * * * * " );
    else printf( " # # # # " );
}
```

- A)有语法错误不能通过编译
 B)输出 * * * *
 C)可以通过编译,但是不能通过连接,因而不能运行
 D)输出 # # # #

6. 执行下面程序中的输出语句后,a 的值是()。

```
#include <stdio. h>
```

```
main( )
```

```
{
    int a;
    printf( "% d\n" ,( a = 3 * 5, a * 4, a + 5 ) );
}
```

- A)65 B)20 C)15 D)10

7. 以下程序不用第三个变量,实现将两个数进行对调的操作。请填空()。

```
#include <stdio. h>
```

```
main( )
```

```
{
    int a, b;
    scanf( "% d% d" ,&a, &b);
    printf( " a = % d b = % d" , a, b );
    a = a + b; b = a - b; a = _____;
    printf( " a = % d b = % d\n" , a, b );
}
```

- A) a + b B) a - b C) b * a D) a/b