全国计算机等级考试应试辅导丛书（新大纲）

# 新编二级C语言
# 题眼分析与全真训练

计算机等级考试试题研究组　主编

人 民 邮 电 出 版 社

**图书在版编目（CIP）数据**

**新编二级 C 语言题眼分析与全真训练／计算机等级考试试题研究组主编．**
**—北京：人民邮电出版社，**2005.6
**（全国计算机等级考试应试辅导丛书：新大纲）**

ISBN 7-115-13451-0

**Ⅰ．新… Ⅱ．计… 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** 052652 **号**

## 内 容 提 要

**本书根据教育部考试中心颁发的新大纲和指定的教程，以对考生进行综合指导为原则，综合了历年考试题（常考题）及考前培训班教师的实际教学经验编写而成。**

**本书抓住** 3 **个重点：考点精讲、题眼分析与全真训练，目的是让考生在较短时间内能快速提高应试能力，顺利过关。本书配有上机盘，上机盘的登录、抽题、答题和交卷等与真实的上机考试完全一致，并且具有自动生成试卷、自动计时和试题评析的功能，便于考生自学与提高。盘中提供** 5 **套全真上机模拟题，供考生上机实战。**

**本书内容包括：程序设计基本概念、C 程序设计的初步知识、顺序结构、选择结构、循环结构、字符型数据、函数、指针、数组、字符串、用户标识符的作用域和存储类、编译预处理和动态存储分配、结构体、共用体和用户定义类型、位运算、文件、上机考试专题辅导、笔试全真模拟试题及参考答案、上机考试全真模拟试题及参考答案。**

**本书适合准备参加全国计算机等级考试（二级** C **语言）的考生考前自学，也可作为普通高校、成人高等教育及各类培训学校举办的考前辅导班的培训教材。**

**全国计算机等级考试应试辅导丛书（新大纲）**

**新编二级 C 语言题眼分析与全真训练**

- **主　　编　计算机等级考试试题研究组**
  **责任编辑　刘建章**
- **人民邮电出版社出版发行**　　**北京市崇文区夕照寺街** 14 **号**
  **邮编**　100061　　**电子函件**　315@ptpress.com.cn
  **网址**　http://www.ptpress.com.cn
  **北京隆昌伟业印刷有限公司印刷**
  **新华书店总店北京发行所经销**
- **开本：**787×1092　1/16
  **印张：**17.5
  **字数：**427 **千字**　　2005 **年** 6 **月第** 1 **版**
  **印数：**1 – 8 000 **册**　　2005 **年** 6 **月北京第** 1 **次印刷**

ISBN 7-115-13451-0/TP・4680

**定价：**28.80 **元（附光盘）**

**读者服务热线：(010)67132692　印装质量热线：(010)67129223**

# 前　言

全国计算机等级考试是目前国内影响最大、参加人数最多的计算机类水平考试。为了帮助广大考生顺利通过计算机等级考试，并全面提高考生的计算机应用水平，我们在深入剖析新考试大纲和历年考题的基础上，编写了本丛书。

本丛书具有以下特色。

- ✧ **名师执笔，权威严谨：**丛书由从事全国计算机等级考试试题研究人员及在等级考试第一线从事命题研究、教学、辅导和培训的老师分工编写，层次清晰，结构严谨，导向准确。
- ✧ **一点一练，高效实用：**书的章名、节名与教育部考试中心指定教程同步，每节中分为“考点提炼”和“题眼分析”两个板块。
  - 考点提炼：将指定的考试内容进行浓缩，精讲考试要点、重点与难点。
  - 题眼分析：精选历年真题（常考题）进行解析，题型丰富，分析透彻。
- ✧ **书盘结合，上机无忧：**在书中特别提供一章将上机常考题进行分类，提炼出题型，按类型进行解析，便于考生专项攻克，提高复习效率。在本书附送光盘中，提供5套全真上机模拟题供考生上机实战。上机盘特点如下：
  - 登录、抽题、答题、交卷等与真实上机考试完全一致，营造逼真的考试氛围。
  - 自动生成试卷、自动计时，特别增加了试题评析功能，便于考生自学与提高。
- ✧ **全真模拟，实战提高：**根据新大纲、新考点、新题型进行命题，提供5套笔试与5套上机全真模拟题，供考生考前实战，感受全真训练。

本套丛书以对考生进行综合指导为原则，具有极强的针对性，特别适合希望在较短时间内取得较大收获的广大应试考生，也可作为全国计算机等级考试各类培训班的教材，以及大、中专院校师生的教学参考书。

丛书由计算机等级考试试题研究组主编，本书由张伍荣、于新豹、陶安、张碧编写。另外，参与本书编排和校对的还有卢晓峰、李文龙、郑家琴、叶雪清、黄奕铭、丁院云、俞顺霖、凌明强、李海、丁善祥、许勇、王健、汪伟、许明亚、骆坚、骆健、张凌云、李曼、毛红梅，在此一并致以衷心地感谢！

尽管我们精益求精，但书中难免存在错漏或不妥之处，敬请读者批评指正。联系邮箱：NCREservice@126.com。

计算机等级考试试题研究组

# 全真模拟上机盘使用说明

◆ **本软件的安装密码**：ptpress2401　　◆ **本软件的准考证号码**：2421999999000101

◆ **配套光盘使用方法**

1．安装完毕后，单击“程序”子菜单中的“新编二级 C 语言题眼分析与全真训练”中的“上机”选项，打开“二级 C 语言考试模拟软件”对话框，如图 1 所示。

2．单击“设置”按钮，根据需要设置“考生文件夹位置”和“试卷抽取方式”。设置好后，单击“进入”按钮，再单击“开始登录”按钮，打开“考生登录”界面，如图 2 所示。

图 1

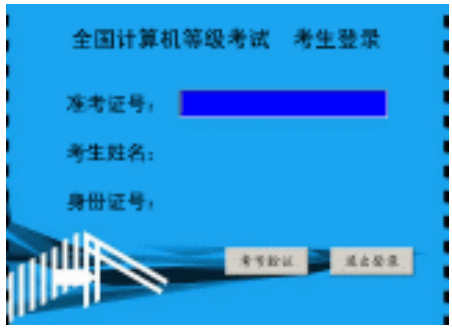



图 2

3．输入正确的准考证号，然后单击“考号验证”按钮，在弹出的询问准考证号是否正确的提示框中单击“是”按钮，再单击“开始考试”按钮，便可进入“考试须知”界面，如图 3 所示。

4．单击“开始答题并计时”按钮，进入考试窗口，如图 4 所示。

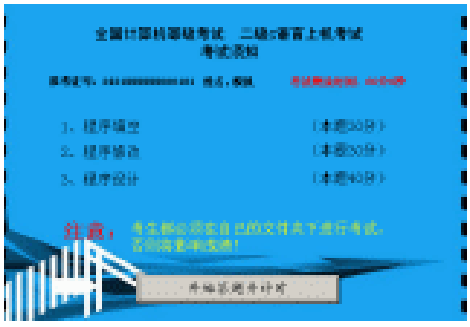


图 3

图 4

5．考试窗口由位于屏幕顶部的“考生状态栏”和“考试主界面”组成。其中，考生状态栏用于显示考生的准考证号、姓名、考试剩余时间等信息。并且，考生随时单击“考生状态栏”中的“显示窗口”字符，将显示考试主界面，且“显示窗口”4 个字会变成“隐藏窗口”；此时，若单击“隐藏窗口”字符，考试主界面就会被隐藏。

6．查看试题：单击“考试主界面”中的试题按钮，窗口中就会显示相应的试题内容。例如：单击“程序填空题”按钮，窗口中就会显示程序填空题的试题内容。

7．查看试题评析：单击“服务项目”菜单中的“试题评析”命令，可阅读试题评析。

8．答题：单击“考试项目”中的“进入考生文件夹”命令，可打开 DOS 窗口。

9．直接输入调用 TC 的命令，进入 TC 窗口，然后调进试题程序（本光盘所带软件，可直接从考生文件夹下调进试题程序，无需设置路径）。

# 目　录


第 1 章　程序设计基本概念 …… 1
1.1　程序和程序设计 …… 1
1.1.1　考点提炼 …… 1
1.1.2　题眼分析 …… 1
1.2　算法 …… 2
1.2.1　考点提炼 …… 2
1.2.2　题眼分析 …… 3
1.3　结构化程序设计和模块化结构 …… 3
1.3.1　考点提炼 …… 3
1.3.2　题眼分析 …… 4
1.4　单元训练及参考答案 …… 4
1.4.1　单元训练 …… 4
1.4.2　参考答案 …… 4
第 2 章　C 程序设计的初步知识 …… 5
2.1　简单 C 语言程序的构成和格式 …… 5
2.1.1　考点提炼 …… 5
2.1.2　题眼分析 …… 5
2.2　常量、变量和标识符 …… 6
2.2.1　考点提炼 …… 6
2.2.2　题眼分析 …… 7
2.3　整型数据 …… 8
2.3.1　考点提炼 …… 8
2.3.2　题眼分析 …… 9
2.4　实型数据 …… 10
2.4.1　考点提炼 …… 10
2.4.2　题眼分析 …… 10
2.5　算术表达式 …… 11
2.5.1　考点提炼 …… 11
2.5.2　题眼分析 …… 12
2.6　赋值表达式 …… 13
2.6.1　考点提炼 …… 13
2.6.2　题眼分析 …… 13
2.7　自加、自减运算符和逗号运算符 …… 15
2.7.1　考点提炼 …… 15
2.7.2　题眼分析 …… 15
2.8　单元训练及参考答案 …… 16
2.8.1　单元训练 …… 16
2.8.2　参考答案 …… 18
第 3 章　顺序结构 …… 19
3.1　赋值语句 …… 19
3.1.1　考点提炼 …… 19
3.1.2　题眼分析 …… 19
3.2　数据输出 …… 19
3.2.1　考点提炼 …… 19
3.2.2　题眼分析 …… 21
3.3　数据输入 …… 22
3.3.1　考点提炼 …… 22
3.3.2　题眼分析 …… 24
3.4　复合语句和空语句 …… 25
3.4.1　考点提炼 …… 25
3.4.2　题眼分析 …… 25
3.5　单元训练及参考答案 …… 25
3.5.1　单元训练 …… 25
3.5.2　参考答案 …… 27
第 4 章　选择结构 …… 28
4.1　关系运算和逻辑运算 …… 28
4.1.1　考点提炼 …… 28
4.1.2　题眼分析 …… 29
4.2　if 语句和用 if 语句构成的选择结构 …… 31
4.2.1　考点提炼 …… 31
4.2.2　题眼分析 …… 31
4.3　条件表达式构成的选择结构 …… 34
4.3.1　考点提炼 …… 34
4.3.2　题眼分析 …… 34
4.4　switch 语句 …… 36
4.4.1　考点提炼 …… 36
4.4.2　题眼分析 …… 36
4.5　语句标号和 goto 语句 …… 38
4.5.1　考点提炼 …… 38
4.5.2　题眼分析 …… 38

4.6 单元训练及参考答案……38
4.6.1 单元训练……38
4.6.2 参考答案……40
第 5 章 循环结构……41
5.1 while 语句和用 while 语句构成的循环结构……41
5.1.1 考点提炼……41
5.1.2 题眼分析……41
5.2 do-while 语句和用 do-while 语句构成的循环结构……43
5.2.1 考点提炼……43
5.2.2 题眼分析……44
5.3 for 语句和用 for 语句构成的循环结构…45
5.3.1 考点提炼……45
5.3.2 题眼分析……46
5.4 循环结构的嵌套……49
5.4.1 考点提炼……49
5.4.2 题眼分析……49
5.5 break 和 continue 语句在循环体中的作用……51
5.5.1 考点提炼……51
5.5.2 题眼分析……52
5.6 单元训练及参考答案……55
5.6.1 单元训练……55
5.6.2 参考答案……58
第 6 章 字符型数据……59
6.1 字符型常量……59
6.1.1 考点提炼……59
6.1.2 题眼分析……60
6.2 字符变量……61
6.2.1 考点提炼……61
6.2.2 题眼分析……61
6.3 字符的输入和输出……63
6.3.1 考点提炼……63
6.3.2 题眼分析……64
6.4 单元训练及参考答案……66
6.4.1 单元训练……66
6.4.2 参考答案……67
第 7 章 函数……68
7.1 库函数……68
7.1.1 考点提炼……68
7.1.2 题眼分析……68
7.2 函数的定义和返回值……69
7.2.1 考点提炼……69
7.2.2 题眼分析……70
7.3 函数的调用……71
7.3.1 考点提炼……71
7.3.2 题眼分析……72
7.4 函数的说明……74
7.4.1 考点提炼……74
7.4.2 题眼分析……75
7.5 调用函数和被调用函数之间的数据传递……76
7.5.1 考点提炼……76
7.5.2 题眼分析……76
7.6 单元训练及参考答案……78
7.6.1 单元训练……78
7.6.2 参考答案……80
第 8 章 指针……81
8.1 变量的地址和指针……81
8.1.1 考点提炼……81
8.1.2 题眼分析……81
8.2 指针变量的定义和引用……82
8.2.1 考点提炼……82
8.2.2 题眼分析……84
8.3 函数之间地址值的传递……86
8.3.1 考点提炼……86
8.3.2 题眼分析……88
8.4 单元训练及参考答案……90
8.4.1 单元训练……90
8.4.2 参考答案……92
第 9 章 数组……93
9.1 一维数组的定义和元素的引用……93
9.1.1 考点提炼……93
9.1.2 题眼分析……94
9.2 一维数组和指针……97

9.2.1 考点提炼 …… 97
9.2.2 题眼分析 …… 98
9.3 函数之间对一维数组和元素的引用 …… 100
9.3.1 考点提炼 …… 100
9.3.2 题眼分析 …… 101
9.4 二维数组的定义和元素的引用 …… 103
9.4.1 考点提炼 …… 103
9.4.2 题眼分析 …… 105
9.5 二维数组和指针 …… 107
9.5.1 考点提炼 …… 107
9.5.2 题眼分析 …… 110
9.6 二维数组名和指针数组作为实参 …… 111
9.6.1 考点提炼 …… 111
9.6.2 题眼分析 …… 111
9.7 单元训练及参考答案 …… 114
9.7.1 单元训练 …… 114
9.7.2 参考答案 …… 120
第 10 章 字符串 …… 121
10.1 用一个一维字符数组存放字符串 …… 121
10.1.1 考点提炼 …… 121
10.1.2 题眼分析 …… 122
10.2 使指针指向一个字符串 …… 124
10.2.1 考点提炼 …… 124
10.2.2 题眼分析 …… 125
10.3 字符串的输入和输出 …… 128
10.3.1 考点提炼 …… 128
10.3.2 题眼分析 …… 129
10.4 字符串数组 …… 130
10.4.1 考点提炼 …… 130
10.4.2 题眼分析 …… 131
10.5 用于字符串处理的函数 …… 132
10.5.1 考点提炼 …… 132
10.5.2 题眼分析 …… 132
10.6 单元训练及参考答案 …… 134
10.6.1 单元训练 …… 134
10.6.2 参考答案 …… 136
第 11 章 对函数的进一步讨论 …… 137
11.1 传给 main 函数的参数 …… 137
11.1.1 考点提炼 …… 137
11.1.2 题眼分析 …… 137
11.2 通过实参向函数传递函数名或指向函数的指针变量 …… 138
11.2.1 考点提炼 …… 138
11.2.2 题眼分析 …… 139
11.3 函数的递归调用 …… 140
11.3.1 考点提炼 …… 140
11.3.2 题眼分析 …… 142
11.4 单元训练及参考答案 …… 144
11.4.1 单元训练 …… 144
11.4.2 参考答案 …… 146
第 12 章 用户标识符的作用域和存储类 …… 147
12.1 局部变量、全局变量和存储分类 …… 147
12.1.1 考点提炼 …… 147
12.1.2 题眼分析 …… 147
12.2 局部变量及其作用域和生存期 …… 148
12.2.1 考点提炼 …… 148
12.2.2 题眼分析 …… 148
12.3 全局变量及其作用域和生存期 …… 150
12.3.1 考点提炼 …… 150
12.3.2 题眼分析 …… 151
12.4 函数的存储分类 …… 153
12.4.1 考点提炼 …… 153
12.4.2 题眼分析 …… 153
12.5 单元训练及参考答案 …… 154
12.5.1 单元训练 …… 154
12.5.2 参考答案 …… 155
第 13 章 编译预处理和动态存储分配 …… 156
13.1 编译预处理 …… 156
13.1.1 考点提炼 …… 156
13.1.2 题眼分析 …… 157
13.2 动态存储分配 …… 159
13.2.1 考点提炼 …… 159
13.2.2 题眼分析 …… 160
13.3 单元训练及参考答案 …… 161

13.3.1 单元训练 …… 161
13.3.2 参考答案 …… 163
第 14 章 结构体、共用体和用户定义类型 …… 164
14.1 用 typedef 说明一种新类型名 …… 164
14.1.1 考点提炼 …… 164
14.1.2 题眼分析 …… 164
14.2 结构体类型 …… 165
14.2.1 考点提炼 …… 165
14.2.2 题眼分析 …… 169
14.3 共用体 …… 175
14.3.1 考点提炼 …… 175
14.3.2 题眼分析 …… 176
14.4 单元训练及参考答案 …… 178
14.4.1 单元训练 …… 178
14.4.2 参考答案 …… 184
第 15 章 位运算 …… 185
15.1 位运算符 …… 185
15.1.1 考点提炼 …… 185
15.1.2 题眼分析 …… 186
15.2 位运算符的运算功能 …… 186
15.2.1 考点提炼 …… 186
15.2.2 题眼分析 …… 187
15.3 单元训练及参考答案 …… 189
15.3.1 单元训练 …… 189
15.3.2 参考答案 …… 190
第 16 章 文件 …… 191
16.1 C 语言文件的概念 …… 191
16.1.1 考点提炼 …… 191
16.1.2 题眼分析 …… 192
16.2 文件指针、打开文件和关闭文件 …… 192
16.2.1 考点提炼 …… 192
16.2.2 题眼分析 …… 194
16.3 文件的读写 …… 196
16.3.1 考点提炼 …… 196
16.3.2 题眼分析 …… 199
16.4 文件定位函数 …… 201
16.4.1 考点提炼 …… 201
16.4.2 题眼分析 …… 202
16.5 单元训练及参考答案 …… 203
16.5.1 单元训练 …… 203
16.5.2 参考答案 …… 205
第 17 章 上机考试专题辅导 …… 206
17.1 上机考试试题类型简介 …… 206
17.1.1 程序填空题 …… 206
17.1.2 程序改错题 …… 206
17.1.3 程序设计题 …… 207
17.2 常考题型提炼 …… 208
17.2.1 题型 1：数的转换与计算 …… 208
17.2.2 题型 2：数列与级数求和 …… 211
17.2.3 题型 3：矩阵运算 …… 213
17.2.4 题型 4：数组运算 …… 217
17.2.5 题型 5：排序 …… 220
17.2.6 题型 6：字符串运算 …… 221
17.2.7 题型 7：链表处理 …… 225
17.2.8 题型 8：其他 …… 229
第 18 章 笔试模拟试题及参考答案 …… 232
18.1 笔试模拟试题 …… 232
18.1.1 笔试模拟试题（一） …… 232
18.1.2 笔试模拟试题（二） …… 242
18.1.3 笔试模拟试题（三） …… 252
18.2 笔试模拟试题参考答案 …… 261
18.2.1 笔试模拟试题（一）答案 …… 261
18.2.2 笔试模拟试题（二）答案 …… 262
18.2.3 笔试模拟试题（三）答案 …… 262
第 19 章 上机模拟试题及参考答案 …… 264
19.1 上机考试模拟试题 …… 264
19.1.1 上机考试模拟试题（一） …… 264
19.1.2 上机考试模拟试题（二） …… 266
19.1.3 上机考试模拟试题（三） …… 268
19.2 上机考试模拟试题参考答案 …… 270
19.2.1 上机考试模拟试题（一）答案 …270
19.2.2 上机考试模拟试题（二）答案 …271
19.2.3 上机考试模拟试题（三）答案 …271

# 第 1 章　程序设计基本概念

## 1.1　程序和程序设计

### 1.1.1　考点提炼

**📖考点 1：高级语言**

计算机只能接受和处理由 0 和 1 的代码构成的二进制或数据，这种形式的指令是面向机器的，称为“机器语言”。但机器语言很难以理解和编写，人们使用接近习惯的自然语言和数学语言作为语言的表达形式，这种语言称为高级语言，C 语言就是高级语言的一种。

由高级语言编写的程序称为“源程序”，把由二进制代码表示的程序称为“目标程序”。“编译程序”就是把“源程序”翻译成“目标程序”的软件。

**📖考点 2：C 语言特点**

由 C 语言构成的指令序列称为 C 源程序，其后缀一般为.C。

用 C 语言所写的每条语句，都要经过编译最终都将转换成二进制的机器指令。其编译过程如下：首先将 C 语言源程序编译并生成一个后缀为.OBJ 的二进制文件（称为目标文件），最后还要由“连接程序”把.OBJ 文件与 C 语言的各种库函数连接起来生成一个后缀为.EXE 的可执行文件。在 DOS 状态下，只需输入这个文件的名字（不必输入后缀.EXE），就可以执行了。

### 1.1.2　题眼分析

【例 1】以下叙述中正确的是_____。（2003 年 4 月）

（A）C 语言比其他语言高级

（B）C 语言可以不用编译就能被计算机识别执行

（C）C 语言以接近英语国家的自然语言和数学语言作为语言的表达形式

（D）C 语言出现得最晚，具有其他语言的一切优点

题眼分析　计算机语言分为机器语言、汇编语言和高级语言，C 语言属于一种高级语言，但并不是说 C 语言比其他语言高级，选项（A）错误；C 语言必须编译成目标代码才能执行，选项（B）错误；C 语言出现于 1972 年至 1973 年之间，并不是出现得最晚的语言，选项（D）错误；高级语言类似于人类的自然语言和数学语言，选项（C）正确。

答案　C

【例 2】用 C 语言编写的代码_____。（2004 年 9 月）

（A）可立即执行　　（B）是一个源程序

（C）经过编译即可执行　　（D）经过编译解释才能执行

**题眼分析** C 语言是一种高级语言，由 C 语言构成的指令序列称为 C 语言源程序。C 语言源程序经过编译生成一个后缀为.OBJ 的二进制文件（称为目标文件），最后还要由“连接程序”把.OBJ 文件与 C 语言的各种库函数连接起来生成一个可执行文件。

**答案** B

# 1.2 算法

## 1.2.1 考点提炼

### 考点 1：算法的定义

算法是指为解决某个特定问题而采取的确定且有限的步骤。算法不等于程序，也不等于计算方法。算法可以用任何一种计算机高级语言转换成程序。

### 考点 2：算法的特性

一个算法具有以下 5 个特性：

（1）有穷性。一个算法应包含有限个操作步骤。

（2）确定性。算法中的每一条指令必须有确切的含义，不能有二义性，对于相同的输入必能得出相同的执行结果。

（3）可行性。算法中的操作都可以通过已经实现的基本运算执行有限次后实现。

（4）有零个或多个输入。算法是用来处理数据对象，在大数情况下，数据对象需要通过输入来得到。

（5）有一个或多个输出。算法的目的是为了求“解”，这些“解”只有通过输出才能得到。

### 考点 3：算法的描述

最常用的算法的描述方法是：伪代码和流程图，而流程图又分为传统流程图和 N-S 流程图。

1．伪代码

使用近似高级语言但又不受语法约束的一种语言描述方式。

2．流程图

流程图也叫框图，它是用各种几何图形、流程线及文字说明来描述计算过程的框图。用流程图描述算法的优点是：直观，设计者的思路表达得清楚易懂，便于检查修改。但也存在一些缺点：对流线的使用没做限制，流线可以随意地转向，这样画出的流程图使人难以理解；在描述复杂的算法时所占篇幅大。这样的流程图称为传统流程图。

3．N-S 流程图

N-S 流程图是适应结构化程序设计而出现的一种新型流程图。已经证明，由顺序、分支、循环 3 种基本结构顺序组成的算法，可以解决任何复杂的问题。1973 年，美国学者 I.Nassi 和 B.Shneiderman 提出了一种新的流程图。这种流程图，去掉了带箭头的流线，全部算法写在一

个大矩形框内，该框可包含一些从属于它的小矩形框。后来，这种流程图就以发明者名字的首字母命名为 N-S 流程图，由于 N-S 流程图的形状像一个盒子，因此，N-S 流程图又称为盒图。

### 1.2.2　题眼分析

【例 1】一个算法应该具有“确定性”等 5 个特性，下面对另外 4 个特性的描述中错误的是_____。（2004 年 4 月）

（A）有零个或多个输入　　（B）有零个或多个输出

（C）有穷性　　（D）可行性

题眼分析　一个算法具有以下 5 个特性：有穷性、确定性、可行性、有零个或多个输入、有一个或多个输出。由此可见，选项（A）、（C）、（D）所描述都是正确的。算法的目的是为了求“解”，这些“解”只有通过输出才能得到，算法必须有一个或多个输出，因此选项（B）的描述是错误的，是答案。

答案　B

【例 2】算法具有 5 个特性，以下选项中不属于算法特性的是_____。（2005 年 4 月）

（A）有穷性　　（B）简洁性　　（C）可行性　　（D）确定性

题眼分析　参见【例 1】。

答案　B

## 1.3　结构化程序设计和模块化结构

### 1.3.1　考点提炼

📖考点 1：结构化程序.

结构化程序由 3 种基本结构组成。

（1）顺序结构。顺序结构是一种简单的程序设计结构，是最基本、最常用的结构。顺序结构是顺序执行结构，程序的执行是按照程序中语句的先后顺序逐条执行。

（2）选择结构。又称为分支结构，它包括简单选择和多分支选择结构。它根据不同的条件去执行不同的分支中的语句。

（3）循环结构。又称为重复结构，根据给定的条件，判断是否需要重复执行某一相同的或类似的程序段。循环结构对应两类循环语句，先判断后执行的循环体称为当型循环结构，对先执行循环体后判断的称为直到型循环结构。

已经证明，由 3 种基本结构组成的算法结构可以解决任何复杂问题。由 3 种基本结构所构成的算法称为结构化算法；由 3 种基本结构所构成的程序称为结构化程序。

📖考点 2：模块化结构

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。模块化结构可以使程序设计变得简单，提高了程序编制的效率。

### 1.3.2 题眼分析

【例 1】结构化程序设计所规定的 3 种基本控制结构是_____。(2003 年 9 月)

(A) 输入、处理、输出　　(B) 树形、网形、环形

(C) 顺序、选择、循环　　(D) 主程序、子程序、函数

题眼分析　结构化程序设计的 3 种基本逻辑结构为顺序、选择和循环，答案选 (C)。

答案　C

【例 2】结构化程序有 3 种基本结构组成，3 种基本结构组成的算法____。(2004 年 9 月)

(A) 可以完成任何复杂的任务　　(B) 只能完成部分复杂的任务

(C) 只能完成符合结构化的任务　　(D) 只能完成一些简单的任务

题眼分析　1966 年，Boehm 和 Jacopini 证明了程序设计语言仅仅使用顺序、选择和循环 3 种基本控制结构就足以表达出各种其他形式结构的程序设计方法，也就是说可以完成任何复杂的任务，答案选 (A)。

答案　A

## 1.4 单元训练及参考答案

### 1.4.1 单元训练

1．要把高级语言编写的源程序转换为目标程序，需要使用_____。

(A) 编辑程序　　(B) 驱动程序　　(C) 诊断程序　　(D) 编译程序

2．以下叙述中正确的是_____。

(A) C 语言的源程序不必通过编译就可以直接运行

(B) C 语言中的每条可执行语句最终都将被转换成二进制的机器指令

(C) C 源程序经编译形成的二进制代码可以直接运行

(D) C 语言中的函数不可以单独进行编译

3．C 语言中用于结构化程序设计的 3 种基本结构是_____。

(A) 顺序结构、选择结构、循环结构　　(B) if、switch、break

(C) for、while、do-while　　(D) if、for、continue

4．以下叙述中正确的是_____。

(A) 用 C 语言实现的算法必须要有输入和输出操作

(B) 用 C 语言实现的算法可以没有输出但必须要有输入

(C) 用 C 程序实现的算法可以没有输入但必须要有输出

(D) 用 C 程序实现的算法可以既没有输入也没有输出

### 1.4.2 参考答案

1．D　　2．B　　3．A　　4．C

# 第 2 章　C 程序设计的初步知识

## 2.1　简单 C 语言程序的构成和格式

### 2.1.1　考点提炼

**📖考点 1：C 程序的构成**

C 程序是由一个或多个函数所组成的，每个函数完成相对独立的功能，函数是 C 程序的基本单位。一个完整的 C 程序有且仅有一个主函数（main()函数）。

程序总是从 main()函数的第 1 条语句开始执行，执行完 main()函数中的所有语句则意味着整个 C 程序执行完成。其他函数都是在执行 main()函数时，通过函数调用或嵌套调用来执行的。

**📖考点 2：main()函数在程序中的位置**

C 规定 main()函数在程序中的位置是任意的，可以放在程序的开头、中间或结尾。

**📖考点 3：函数体**

函数体是从花括号“{”开始，到花括号“}”结束。由说明部分和执行部分组成。其中说明部分的作用是定义函数中使用的变量、数组等，执行部分的作用是完成函数功能。

**📖考点 4：语句以分号“;”结束**

C 语言规定每条语句以分号“;”结束，分号是语句不可缺少的一部分。

**📖考点 5：注释信息**

“/*”与“*/”之间的信息称为注释信息。注释信息对程序的运行结果不发生任何影响，也不会被 C 语言编译程序所编译。插入注释的要求是：允许在任何能够插入空格符的位置插入注释，但 C 语言的注释不能进行嵌套。

**📖考点 6：C 程序的书写风格**

C 程序的书写风格比较自由，每条语句可从任意列开始。但是，为了使程序层次清楚，应当采用缩进格式书写程序。C 程序中每行可以写多条语句，推荐采用一行一句的书写风格。

### 2.1.2　题眼分析

【例 1】在一个 C 程序中_____。（2003 年 4 月）

（A）main 函数必须出现在所有函数之前　　（B）main 函数可以在任何地方出现

（C）main 函数必须出现在所有函数之后　　（D）main 函数必须出现在固定位置

**题眼分析** main()函数在程序中的位置是任意的，可以放在程序的开头、中间或结尾。

**答案** B

**【例2】以下叙述中正确的是_____。（2003年9月）**

**（A）C 程序中注释部分可以出现在程序中任何合适的地方**

**（B）花括号“{”和“}”只能作为函数体的定界符**

**（C）构成 C 程序的基本单位是函数，所有函数名都可以由用户命名**

**（D）分号是 C 语句之间的分隔符，不是语句的一部分**

**题眼分析** C 程序中注释部分允许在任何能够插入空格符的位置插入注释。花括号“{”和“}”不但可以作为函数体的定界符，也可以作为一个复合语句的定界符。构成 C 程序的基本单位是函数，函数分为两类：系统提供的标准函数和用户自定义函数，只有用户自定义函数名可以由用户命名。C 语言规定每条语句以分号“;”结束，它是语句不可缺少的一部分。

**答案** A

**【例 3】以下说法不正确的是_____。**

**（A）C 程序中的一行可以写多条语句** **（B）C 程序中的语句可以采用缩进格式书写**

**（C）C 程序中的每行只能写一条语句** **（D）C 程序中可以通过注释提高程序的可读性**

**题眼分析** C 程序的书写风格比较自由：每条语句可从任意列开始；每行可以写多条语句，一条语句也可以写在多行上；采用缩进格式书写和适当地添加注释信息可以提高程序的可读性。

**答案** C

# 2.2 常量、变量和标识符

## 2.2.1 考点提炼

### 考点 1：标识符

在 C 语言中，合法的标识符由字母、数字和下划线组成，并且第 1 个字符必须为字母或下划线，如：acd、d_ad、_adf123、_123 都是合法的标识符。而 2asdf、ww.da、_sd\=都是不合法的标识符。在 C 语言中大写字母和小写字母被认为两个不同的字符，如：ACD 和 acd 是两个不同的标识符。

C 语言的标识符有 3 种类型：关键字、预定义标识符和用户标识符。

在 C 语言中用户标识符要符合标识符的命名规则，同时不能和保留字同名。

### 考点 2：常量与变量

1．常量

常量又叫常数，是指在程序的运行过程中其值保持不变的量。C 语言规定常量的类型有 4 种：整型常量、实型常量、字符常量、字符串常量。常量是不需要事先定义的，在程序中可以直接使用，常量的类型也不需要说明，它是由书写方法自动默认的。

2．变量

变量是在程序运行期间可以改变的量，本质上变量就是计算机内部的一个存储单元。每个变量都有一个变量名，该变量名对应于内存中固定长度的存储单元，该存储单元的首地址称为该变量的地址，该存储单元中存放的数据称为该变量的值。

变量实质上代表某个存储单元，这个存储单元的大小以及每位的含义由变量的类型来决定，因此在使用变量之前必须对变量的类型进行说明。变量说明的形式如下所示：

类型标识符　变量名 1,变量名 2,……；

当用以上方式定义变量时，编译程序给变量分配存储单元，但是并没有在存储单元中放置任何初值，因此在这些存储单元中，原有的信息并没有被清除，这时变量中的值是无意义的。C 语言规定：可以在定义变量的同时给变量赋初值，或称变量初始化。如：char *a*='A'; int *b*=1320,*c*=3000；。

### 2.2.2　题眼分析

【例 1】下列关于 C 语言用户标识符的叙述中正确的是_____。（2003 年 4 月）

（A）用户标识符中可以出现下划线和中划线（减号）

（B）用户标识符中不可以出现中划线，但可以出现下划线

（C）用户标识符中可以出现下划线，但不可以放在用户标识符的开头

（D）用户标识符中可以出现下划线和数字，它们都可以放在用户标识符的开头

题眼分析　在 C 语言中，合法的标识符由字母、数字和下划线组成，并且必须以字母或下划线开头。

答案　B

【例 2】以下不能定义为用户标识符是_____。（2005 年 4 月）

（A）Main　　（B）_0　　（C）_int　　（D）sizeof

题眼分析　用户标识符不能和关键字同名。sizeof 是 C 语言关键字，不能用做用户定义标识符。main 是 C 语言主函数名，但不是关键字，但最好不要使用它作为用户标识符。另外，C 语言中是区分大小写的，但 Main 和 main 可以认为是两个不同的用户标识符。

答案　D

【例 3】以下 4 组用户定义标识符中，全部合法的一组是_____。（2004 年 4 月）

| （A） | （B） | （C） | （D） |
|---|---|---|---|
| _main | if | txt | int |
| enclude | -max | REAL | k_2 |
| sin | turbo | 3COM | _001 |

题眼分析　if、int 都是 C 语言关键字，不能用做用户定义标识符，因此选项（B）、（D）都是错误的。3COM 以数字开头，不是以字母或下划线开头，因此选项（C）也是错误的。

答案　A

【例 4】下列叙述中正确的是_____。（2003 年 4 月）

（A）C 语言中既有逻辑类型也有集合类型

（B）C 语言中没有逻辑类型但有集合类型

（C）C 语言中有逻辑类型但没有集合类型

（D）C 语言中既没有逻辑类型也没有集合类型

**题眼分析** C 语言的数据类型分为基本类型、构造类型、指针类型、空类型 4 大类。其中，基本类型分为整型、字符型、实型 3 类。实型又称浮点型，包括单精度型和双精度型两种类型。C 语言中没有逻辑类型，逻辑类型用整形来表示；C 语言中也没有集合类型。

**答案** D

**【例 5】下列变量定义中合法的是_____。** （2000 年 9 月）

（A）short _a=1-.le–1;　　（B）double b=1+5e2.5;

（C）long do=0xfdaL;　　（D）float 2_and=1–e–3;

**题眼分析** 选项（A）中把实型表达式作为初值赋给整型变量，显然是非法的；在选项（C）中使用了保留字作变量名是错误的；选项（D）中使用了不正确的标识符。

**答案** B

# 2.3 整型数据

## 2.3.1 考点提炼

### 考点 1：整型常量

在 C 语言中整型及常数可以是十进制、八进制或十六进制数字表示的整数值。

（1）十进制数：通常整数的写法。如：0、–1、32、+87。

（2）八进制数：书写方法是在通常八进制整数的前面加一个数字 0。如：00、011、–011、+023 等，它们分别表示十进制数：0、9、–9、+19。

（3）十六制整数：书写方法是在通常的十六进制整数的前面加 0x。如：0x0、–0x11、+023、0x98 等，它们分别表示十进制数 0、–17、+35、152。

注意：正整数前面的“+”可以省略。整型在计算机中一般占两个字节。它的取值范围是：–32768～+32767。

### 考点 2：整型变量

整型变量可分为基本型、短整型、长整型和无符号型。以基本型整型变量为例，使用关键字 int 进行定义。例如：

```
int k;
```

在一个定义语句可以同时定义多个变量，变量之间使用逗号隔开。例如：

```
int i,j,k
```

可以在定义变量的同时给变量赋初值。例如：

```
int i=0,j=1,k=20
```

### 考点 3：整型数据的分类

整型数据包括基本整型、短整型、长整型和无符号整型，如表 2-1 所示。

表 2-1　　整型数据的分类

| 数据类型 | 数据类型符 | 占用的字节数 | 数值的范围 |
|---|---|---|---|
| 基本整型 | int | 2（或 4） | 同短整型（或长整型） |
| 短整型 | short | 2 | -32768～+32767 |
| 长整型 | long | 4 | -2147483648～+2147483647 |
| 无符号整型 | unsigned int | 2（或 4） | 同无符号短整型（或长整型） |
| 无符号短整型 | unsigned short | 2 | 0～65535 |
| 无符号长整型 | unsigned long | 4 | 0～4294967295 |

长整型，在整数的末尾加上后缀字母“l”或“L”。例如 0L、–011L、+0X32L。

无符号整型在整数的末尾加上后缀字母“u”或“U”。若是长整型无符号整数，则应加后缀“lu”或“LU”。

**考点 4：整型数据在内存中的存储方式**

在 C 语言中，对于有符号整数，用最高位（最左边一位）用来存储整数的符号，若是正整数，最高位为 0，若是负数，最高位放置 1。对于正整数用“原码”形式存放，对于负整数用“补码”形式存放。

对于无符号整数，最高位（最左边一位）不再用来存储整数的符号，全部用来存放整数。

## 2.3.2　题眼分析

【例 1】以下选项中可以作为 C 语言中合法整数的是_____。（2003 年 9 月）

（A）10110B　　（B）0386　　（C）0Xffa　　（D）x2a2

题眼分析　整型数据可用十进制、八进制、十六进制表示。选项（A）是一个二进制数；选项（B）中，数字 8 不能出现在一个八进制数中；选项（D）中，十六进制数表达不正确，十六进制数应以 0x 开头。

答案　C

【例 2】在 16 位 C 编译系统上，若定义 long *a*;，则能给 *a* 赋 40000 的正确语句是_____。（2002 年 4 月）

（A）*a*=20000+20000;　（B）*a*=4000*10;　　（C）*a*=30000+10000;　（D）*a*=4000L*10L;

题眼分析　由于 20000+20000、4000*10 及 30000+10000 都是整型表达式运算的结果仍然是整型，表达式运算的结果超出了整型数据的范围，不正确。而（D）是长整型运算不会超出长整型的范围。

答案　D

【例 3】在 C 语言中，合法的长整型常数是_____。　（2001 年 9 月）

（A）0L　　（B）4962710　　（C）324562&　　（D）216D

题眼分析　长整型数据可用十进制、八进制、十六进制表示，要求在数后边要加上“l”或“L”作为后缀。

答案　A

# 2.4 实型数据

## 2.4.1 考点提炼

### 考点 1：实型常量

C 语言中，实型常量可以用小数形式和指数形式两种方法表示实型：

（1）小数形式：小数形式的实数由整数部分、小数点和小数部分组成（必须要有小数点），如：3.5、5.0、–.5、7. 等是合法的实型。

（2）指数形式：指数形式的实数由尾数部分、e（E）和指数部分组成。字母 e 或 E 的前后必须要有数字，且其后面的指数必须为整数。如：2.306 可以表示成 0.2306e1、2.306e0、23.06e-1 等形式，如：e3、0.5e3.6、2.0e 等都是不合法的实型。

### 考点 2：实型变量

实型变量分为单精度实型和双精度实型两类，如表 2-2 所示。

表 2-2　　实型数据的分类

| 数据类型 | 数据类型符 | 占用的字节数 | 数值的范围 |
|---|---|---|---|
| 单精度实型 | float | 4 | $-10^{38}$～$10^{38}$ |
| 双精度实型 | double | 8 | $-10^{308}$～$10^{308}$ |

单精度实型变量和双精度实型变量分别使用类型名 float 和 double 进行定义，在变量定义时可以给变量赋初值。

## 2.4.2 题眼分析

### 一、选择题分析

【例 1】以下选项中，不能作为合法常量的是_____。（2005 年 4 月）

（A）1.234e04　　（B）1.234e0.4　　（C）1.234e+4　　（D）1.234e0

题眼分析　指数形式的实数是由尾数、小写字母 e 或大写字母 E、指数 3 部分组成。字母 e 或 E 的前后必须要有数字，且其后面的指数必须为整数。因此选项（B）不是合法的实型常量。

答案　B

【例 2】以下选项中可作为 C 语言合法常量的是_____。（2005 年 4 月）

（A）–80.　　（B）–080　　（C）–8e1.0　　（D）–80.0e

题眼分析　根据【例 1】的分析，选项（C）、（D）都不是合法常量。选项（B）中，数字是 0 开头应当是一个八进制数，但八进制数只能出现数字 0～7，不能出现数字 8，因此也不是合法常量。选项（A）是使用小数形式表示实型常量，是合法的。

答案　A

**二、填空题分析**

**【例】把 *x*、*y* 定义成单精度实型变量，并赋初值 10.5 的定义语句是_____。**

**题眼分析　单精度实型变量用类型名 float 进行定义，并可以在定义时给变量赋初值。**

**答案**　float *x*=10.5,*y*=10.5

# 2.5　算术表达式

## 2.5.1　考点提炼

### 考点 1：基本算术运算符

**基本的算术运算符的运算对象、运算规则及结果如表 2-3 所示。**

**表 2-3　　基本的算术运算符的运算对象、运算规则及结果**

| 对象数 | 名称 | 运算符 | 运算规则 | 运算对象 | 运算结果 |
|---|---|---|---|---|---|
| 单目 | 正 | + | 取原值 | 整型或实型 | 整型或实型 |
| | 负 | − | 取负值 | | |
| 双目 | 加 | + | 加法 | | |
| | 减 | − | 减法 | | |
| | 乘 | * | 乘法 | | |
| | 除 | / | 除法 | | |
| | 模 | % | 整除取余数 | 整型 | 整型 |

### 考点 2：算术运算符的优先级、结合性和算术表达式

**算术运算符和圆括号的优先级如图 2-1 所示。**

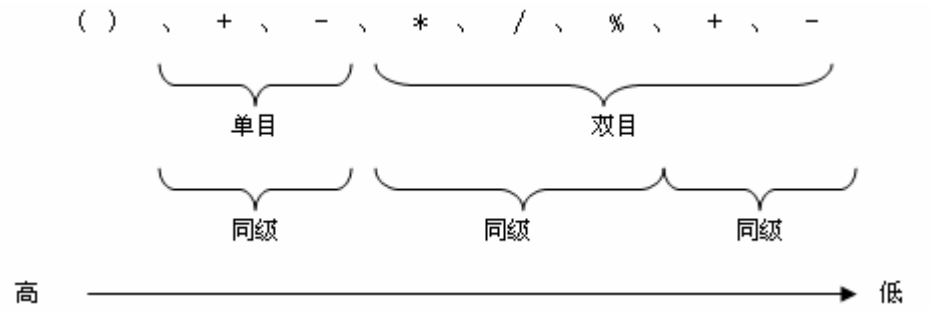



**图 2-1　算术运算符和圆括号的优先级**

**结合性：同级单目基本算术运算符的结合性自右向左，同级双目基本算术运算符的结合性是自左向右。**

**算术表达式是由算术运算符连接数值类型运算对象构成的。运算对象包括常量、变量、函**

数等。使用括号可以改变运算处理顺序，在括号里的运算总是先进行。

📖考点 3：强制性数据类型转换

转换格式：(数据类型符)(表达式)

说明：C 语言先按自动转换的原则计算表达式的值，然后在将其转换成指定的数据类型。

## 2.5.2 题眼分析

### 一、选择题分析

【例 1】C 语言中运算对象必须是整型的运算符是_____。（2000 年 9 月）

（A）%=　　（B）/　　（C）=　　（D）<=

题眼分析　取模运算要求两边的对象全为整型。

答案　A

【例 2】以下程序的输出结果是_____。（2001 年 9 月）

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

（A）–6　　（B）12　　（C）0　　（D）–12

题眼分析　根据运算符的优先级和结合性，该表达式等价于：*a*+=(*a*–=(*a***a*))。

答案　D

### 二、填空题分析

【例 1】若有语句

```
int  i=–19,j=i%4;
printf("%d\n",j);
```

则输出结果是_____。（2003 年 4 月）

题眼分析　运算符“%”是求两个数相除的余数，其结果与被除数的符号一致。

答案　–3

【例 2】设有以下变量定义，并已赋确定的值（2000 年 4 月）

```
char w; int x; float y; double z;
```

则表达式：*w***x*+*z*–*y* 所求得的数据类型为_____。

题眼分析　参加运算的各个数据类型都转换成数据类型最长的数据类型，然后计算，表达式值的类型也就是数据长度最长的数据类型，在 char、int、float、double 中，double 型是最长的。

答案　double

# 2.6　赋值表达式

## 2.6.1　考点提炼

### 考点 1：赋值运算符和赋值表达式

赋值运算是双目运算符，赋值运算符前面必须要有变量，后面是常量、表达式或变量。赋值运算符“ = ”及算术自反赋值运算符的运算对象、运算规则如表 2-4 所示。

优先级：算术运算符 优先于 关系运算符 优先于 双目逻辑运算符 优先于 赋值运算符；算术运算符 优先于 关系运算符 优先于 双目逻辑运算符 优先于 算术自反赋值运算符。

结合性：赋值运算的结合性是自右向左；算术自反赋值运算符同赋值运算符是同级别的，结合性是自右向左。

由赋值运算符连接运算对象组成的式子：“ 变量=表达式 ”称赋值表达式。其值等于赋值运算符右边的表达式的值。

### 考点 2：复合赋值运算符

复合赋值运算符的运算规则如表 2-4 所示。

表 2-4　　赋值运算符“ = ”及复合赋值运算符的运算对象、运算规则

| 对象数 | 名称 | 运算符 | 运算规则 | 运算对象 | 运算结果 |
|---|---|---|---|---|---|
| 双目 | 赋值 | = | 将表达式赋值给变量 | 任意类型 | 表达式类型 |
| | 加赋值 | += | A+=B（相当于 A=A+(B)） | 数值型 | 数值型 |
| | 减赋值 | -= | A-=B（相当于 A=A–(B)） | | |
| | 除赋值 | /= | A/=B（相当于 A=A/(B)） | | |
| | 乘赋值 | *= | A*=B（相当于 A=A*(B)） | | |
| | 模赋值 | %= | A%=B（相当于 A=A%(B)） | 整型 | 整型 |

### 考点 3：赋值运算中的类型转换

在 C 语言中提供两条自动转换规则：

（1）表达式在计算中数据的类型自动转换原则。参加运算的各个数据类型都转换成数据长度最长的数据类型，然后再计算，计算的结果是数据长度最长的数据类型。

（2）运算结果在存入变量时数据的类型的自动转换原则。先将运算结果的数据类型自动转换成变量的数据类型，然后在赋值给该变量。

## 2.6.2　题眼分析

### 一、选择题分析

【例 1】若有以下程序段

```
int m=0xabc,n=0xabc;
```

```
    m–=n;
    printf("%X\n",m);
```

执行后输出结果是_____。（2003 年 4 月）

（A）0X0　　（B）0x0　　（C）0　　（D）0XABC

题眼分析　“*m*– =*n*”相当于“*m*=*m*–*n*”，两个相等的数相减，其值为 0。

答案　C

【例 2】有以下程序，

```
main()
{   int a;      char c=10;
    float f=100.0; double x;
    a=f/=c*=(x=6.5);
    printf(“%d %d %3.1f %3.1f\n”,a,c,f,x);
}
```

程序运行后的输出结果是_____。（2003 年 9 月）

（A）1　65　1　6.5　　（B）1　65　1.5　6.5

（C）1　65　1.0　6.5　　（D）2　65　1.5　6.5

题眼分析　程序的执行顺序如下：①*x*=6.5，*x* 的值为 6.5；②*c**=*x*，即 *c*=*c***x*，*c* 的值为 65；③*f*/=*c*，即 *f*=*f*/*c*，f 的值为 1.538；④*a*=*f*，但 *a* 为整型变量，因此只取 *f* 的整数部分 1。在输出时，变量 *f* 的域宽为 3，小数位为 1，因此输出 1.5。

答案　B

【例 3】设 *a* 和 *b* 均为 double 型变量，且 *a*=5.5、*b*=2.5，则表达式(int)*a*+*b*/*b* 的值是_____。（2002 年 9 月）

（A）6.500000　　（B）6　　（C）5.500000　　（D）6.000000

题眼分析　在表达式中，优先级最高的是(int)a，结果是 5，其次是 b/b，结果是 1.0，最后相加结果为 6.000000（表达式的最终结果为实型）。

答案　D

【例 4】若有定义 int *a*=8,*b*=5,*c*;，执行语句“*c*=*a*/*b*+0.4;”后，*c* 的值为_____。（2002 年 4 月）

（A）1.4　　（B）1　　（C）2.0　　（D）2

题眼分析　在表达式中，根据运算的结合性和运算的优先级，首先计算的是 *a*/*b* 其值为 1，再将 1+0.4 赋值给 *c*，由于 *c* 为整型变量所以要将 1.4 转换为整型数，即舍弃小数位为 1。

答案　B

## 二、填空题分析

【例 1】以下程序的输出结果是_____。（2001 年 4 月）

```
main()
{   int a=1, b=2;
    a=a+b; b=a–b; a=a–b;
    printf("%d,%d\n", a, b );
}
```

题眼分析　首先将“*a*+*b*”赋值给变量 *a*，*a* 的值为 3，再将“*a*–*b*”赋值给变量 *b*，此时

变量 *b* 的值为 1，最后再将“*a*–*b*”赋值给变量 *a*，此时变量 *a* 的值为 2。

**答案**　2,1

**【例 2】以下程序的输出结果是_____。（2002 年 4 月）**

```
main()
{   int a=0;
    a+=(a=8);
    printf("%d\n",a);
}
```

**题眼分析**　在程序中首先将 8 赋值给变量 *a*，然后在进行自反运算，其等价形式为

```
{   a=8;a=a+8;  }。
```

**答案**　16

# 2.7　自加、自减运算符和逗号运算符

## 2.7.1　考点提炼

### 考点 1：自增运算符、自减运算符

自增运算符、自减运算符的运算对象、运算规则与结果如表 2-5 所示。

表 2-5　　自增运算符、自减运算符的运算对象、运算规则与结果

| 对象数 | 名称 | 运算符号 | 运算规则 | 运算对象 | 运算结果 |
|---|---|---|---|---|---|
| 单目 | 增 1（前缀） | ++ | 先加 1 后再使用 | 整型变量<br>字符型变量<br>指针型变量<br>数组元素 | 同运算符对象的类型 |
| | 减 1（前缀） | – – | 先减 1 后再使用 | | |
| | 增 1（后缀） | ++ | 先使用后再加 1 | | |
| | 减 1（后缀） | – – | 先使用后在减 1 | | |

优先级：增 1 减 1 运算符优先于双目基本算术运算符号。

结合性：增 1 减 1 运算符和单目的+、–是同级别，结合性自右向左。

### 考点 2：逗号表达式

逗号表达式是双目运算符，其运算对象是表达式，其一般形式为：

表达式 1，表达式 2，……，表达式 *n*

逗号表达式的结合性为从左向右，即先计算表达式 1，再计算表达式 2，最后运算表达式 *n*。最后一个表达式的值就是逗号表达式的值。

## 2.7.2　题眼分析

**【例 1】有以下程序**

```
main()
{   int m=12,n=34;
```

```
        printf("%d%d",m++,++n); printf("%d%d\n",n++,++m);
    }
```

程序运行后的输出结果是_____。（2005 年 4 月）

（A）12353514　　（B）12353513　　（C）12343514　　（D）12343513

题眼分析　++、--运算符在变量之前是先使变量的值加 1 或减 1，然后再使用变量的值，如果在变量之后则先使用变量的值，再把变量的值加 1 或减 1。

答案　A

【例 2】设有如下程序段

```
int x=2002, y=2003;
printf("%d\n",(x,y));
```

则以下叙述中正确的是_____。（2003 年 9 月）

（A）输出语句中格式说明符的个数少于输出项的个数，不能正确输出

（B）运行时产生出错信息

（C）输出值为 2002　　（D）输出值为 2003

题眼分析　逗号表达式的值是其最后一个表达式的值，逗号表达式(*x*,*y*)的值为 *y* 的值，即 2003。

答案　D

# 2.8　单元训练及参考答案

## 2.8.1　单元训练

### 一、选择题

1．请选出可以作为 C 语言用户标识符的一组标识符号_____。

（A）void　define　WORD　　（B）a3_b3　_123　IF

（C）for　-abc　case　　（D）2a　D0　sizeof

2．以下选项中属于 C 语言的数据类型是_____。

（A）复数型　　（B）双精度型　　（C）逻辑型　　（D）集合型

3．在 C 语言中，不正确的 int 类型的常数是_____。

（A）32768　　（B）0　　（C）037　　（D）0xAF

4．请选出合法的 C 语言赋值语句_____。

（A）*a*=*b*=58;　　（B）i++;　　（C）*a*=58,*b*=58;　　（D）*k*=int(*a*+*b*);

5．设已定义整型变量 *k* 和 *g*，则下面的程序输出为_____。

```
k=017;  g=111;
printf("%d\n",++k);
printf("%x\n",g++);
```

（A）15　6f　　（B）16　70　　（C）15　71　　（D）16　6f

6．下面语句中_____是 C 语言的正确赋值语句。

（A）*a*=1.b=2　　（B）i++　　（C）*a*=*b*=5　　（D）*y*=int(*x*);

7．下列选项中，不能用做标识符的是_____

（A）_1234_　　（B）_1_2　　（C）int_2_　　（D）2_int_

8．下面程序的输出是_____。

```
main( )
{   int x = 10, y = 3;
    printf("%d\n",y=x/y);
}
```

（A）0　　（B）1　　（C）3　　（D）不确定的值

9．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

10．C 语言中，int 类型数据占两个字节，则 long 类型数据占_____个字节。

（A）1　　（B）2　　（C）4　　（D）8

11．若 int 类型数据占两个字节，则下列语句

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

的输出结果为_____。

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

12．若有定义 int *a*=8,*b*=5,*c*;，执行语句“*c*=*a*/*b*+0.4;”后，*c* 的值为_____。

（A）1.4　　（B）1　　（C）2.0　　（D）2

13．设有定义：float *a*=2，*b*=4，*h*=3;，以下 C 语言表达式中与代数式$\frac{1}{2}(a+b)h$计算结果不相符的是_____。

（A）(*a*+*b*)*h*/2　　（B）(1/2)*(*a*+*b*)*h*　　（C）(*a*+*b*)*h*l/2　　（D）*h*/2*(*a*+*b*)

14．若有以下程序：

```
main()
{   int k=2,i=2,m;
    m=(k+=i*=k);
    printf("%d,%d\n",m,i)；
}
```

执行后的输出结果是_____。

（A）8,6　　（B）8,3　　（C）6,4　　（D）7,4

15．有如下程序

```
main()
{   int    y=3,x=3,z=1;
    printf(“%d  %d\n”,(++x,y++),z+2);
}
```

运行该程序的输出结果是_____。

（A）3　4　　（B）4　2　　（C）4　3　　（D）3　3

## 二、填空题

1．C 语言提供的基本数据类型包括：__(1)__、__(2)__、__(3)__、__(4)__、

（5）。

2．C 语言的标识符只能有 3 种字符组成，它们是：（6）、（7）、（8）。

3．若 *x*，*y*，*z* 均是整型变量，则执行表达式 *x*=(*y*=4)+(*z*=2)后，*x* 的值为（9），*y* 的值为（10）。

4．假设所有的变量都为整型，则表达式（*a*=2,*b*=*a*++,*b*++,*a*+*b*）值为（11）。

5．设 *x* 为 int 型变量，请写出描述“*x* 是奇数”的表达式（12）。

6．已知整型数据 *a*=3，*b*=–4，*c*=5，则表达式 *a*++–*b*+(++*c*)的值是（13）。

7．在 C 语言中（以 16 位 PC 机为例），一个 float 型数据在内存中所占的字节数为 4，一个 double 型数据在内存中所占的字节数为（14）。

8．若有定义：int *a*=10，*b*=9，*c*=8；接着顺序执行下列语句，变量 *b* 中的值是（15）。

```
c=(a-=(b-5));
c=(a%11)+(b=3);
```

## 2.8.2 参考答案

### 一、选择题

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

### 二、填空题

（1）单精度型
（2）双精度型
（3）整型
（4）字符型
（5）枚举型
（6）字母
（7）数字
（8）下划线
（9）6
（10）4
（11）6
（12）(*y*%2)==1
（13）13
（14）8
（15）3

# 第 3 章　顺序结构

## 3.1　赋值语句

### 3.1.1　考点提炼

在赋值表达式后加一个“ ; ”就构成赋值语句。逗号表达式和自加、自减运算表达式后加一个“ ; ”也构成赋值语句。赋值时只能变量进行赋值，不能对表达式和常量进行赋值。赋值语句是一种可执行语句，只能出现在函数的可执行部分。

### 3.1.2　题眼分析

【例 1】以下非法的赋值语句是_____。（2002 年 9 月）

（A）*n*=(*i*2,*i*++);　　（B）*j*++;　　（C）++(*i*+1);　　（D）*x*=*j*>0;

题眼分析　在表达式的运算中，双目赋值运算符的格式为“ 变量=表达式 ”，单目运算符一般形式为“ 运算符 表达式 ”或“ 表达式 运算符 ”。常量和表达式是不能被赋值的。选项（C）试图给表达式 i+1 赋值，因此是错误的。

答案　C

【例 2】以下合法的赋值语句是_____。（2001 年 9 月）

（A）*x=y=100;*　　（B）*d–;*　　（C）*x+y;*　　（D）*c*=int*(a+b);*

题眼分析　（B）和（C）没有赋值运算符，所以不是赋值语句，（D）有语法错误，int 应加括号。

答案　A

## 3.2　数据输出

### 3.2.1　考点提炼

📖考点 1：格式化输出函数 printf()的一般调用形式

格式化输出函数 printf()的一般调用形式为：

printf(输出格式字符串,输出表达式表);

功能：按照自右向左的顺序，依次计算“ 输出表达式 ”中各个表达式的值，再按“ 输出格式字符串 ”中规定的格式输出到显示器上。

**说明：输出格式字符串由控制输出格式的字符和非格式字符组成的字符串，通常是一个字符串常量。非格式字符是作为输出时数据的间隔，输出时原样输出。而格式字符对应着数据，输出时按照规定的格式输出数据。输出表达式表是由若干个需要计算和输出的表达式组成的，表达式之间用逗号分隔开。要注意的它不是逗号表达式，它的计算顺序是自右向左的。**

### 考点 2：函数 printf()的常用格式

**常用输出格式字符号如表 3-1 所示。**

**表 3-1　　输出格式字符号列表**

| 数据格式 | 数据对象 | 输出格式 | 数据输出方法 |
|---|---|---|---|
| %–md | int<br>short<br>unsigned int<br>unsigned short<br>char | 十进制整数 | 无 m 按实际位数输出<br>有 m 输出 m 位<br>超过 m 位，按实际位输出<br>不足 m 位，补空格<br>无-右对齐（不够左不空格），<br>有-左对齐（不够右不空格） |
| %–mo | | 八进制整数 | |
| %–mx | | 十六进制整数 | |
| %–mu | | 无符号整数 | |
| %mld | long<br>unsigned long | 十进制整数 | 无 m 位按实际位数输出<br>有 m 位输出 m 位<br>超过 m 位按实际位输出<br>不足 m 位补空格<br>无-右对齐(左补空格)，有-左对齐(右补空格) |
| %–mlo | | 八进制整数 | |
| %–mlx | | 十六进制整数 | |
| %–mlu | | 无符号整数 | |
| %–m.nf | float<br>double | 十进制小数 | 无 m.n 按实际位数输出<br>有 m.n 输出 n 位小数，m 为总宽度<br>超过 m 位时按实际位输出，不足 m 位补空格<br>无-右对齐(左补空格)，有-左对齐(右补空格) |
| %–m.ne | | 八进制小数 | |
| %g | | 自动选取 f 或 e 中宽度的格式 | |
| %–mc | char<br>int<br>short | 单个字符 | 无 m 输出单个字符<br>有 m 输出 m 位，补空格<br>无-右对齐(左补空格)，有-左对齐(右补空格) |
| %–m.ns | 字符串 | 一个字符串 | 无 m.n 按实际字符串输出全部字符<br>有 m.n 仅输出 n 个字符，补空格<br>超过 m 位时按实际位输出，不足 m 位,补空格<br>无-右对齐(左补空格)，有-左对齐(右补空格) |

### 考点 3：调用 printf 函数时的注意事项

**在调用 printf 函数时需注意以下几点：**

**（1）在格式控制串中，格式说明与输出项从左到右在类型上必须一一对应匹配。**

**（2）在格式控制串中，格式说明与输出项个数相同。如果格式说明的个数少于输出项的个数，多余的项将不予输出。反之，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）。

（4）如果想输出百分号“%”，则在格式控制串中用两个连续的百分号“%%”来表示。

（5）在输出语句中改变输出变量的值，如：i=5;printf(“%d%d\n”,i,++i);不能保证先输出 *i* 的值，然后再求 *i*++，并输出。

（6）printf 函数的返回值为本次调用输出字符的个数。

### 3.2.2　题眼分析

**一、选择题分析**

【例 1】有以下程序

```
main( )
{   int  m=0256,n=256 ;
    printf("%o  %o\n", m , n ) ;
}
```

程序运行后的输出结果是_____。（2004 年 9 月）

（A）0256　0400　　（B）0256　256　　（C）256　400　　（D）400　400

题眼分析　用格式字符 *o* 和 *x* 按八进制和十六进制数的形式输出整数时，在数据前面并不出现 0 和 0x。如果需要输出，可在%和 *o* 或 *x* 之间插入一个#号。

答案　C

【例 2】有以下程序

```
main( )
{   int  a=666 , b=888 ;
    printf("%d\n", a , b ) ;
}
```

程序运行后的输出结果是_____。（2004 年 9 月）

（A）错误信息　　（B）666　　（C）888　　（D）666，888

题眼分析　在格式控制串中，格式说明应与输出项个数相同。如果格式说明的个数少于输出项的个数，多余的项将不予输出。

答案　B

【例 3】有以下程序

```
main ()
{   int  x=l02,y=012;
    printf("%2d,%2d\n",x,y);
}
```

执行后输出结果是_____。（2004 年 4 月）

（A）10,01　　（B）02,12　　（C）102,10　　（D）02,10

题眼分析　变量 y 被赋值 012，012 是用八进制表示的整数，其值对应于十进制为 10。在“%”和格式字符之间插入一个整数用来指定输出的宽度，如果指定的宽度不够，并不影响数据的完整输出，系统会代之以隐含的输出宽度。因此，选项 C 是正确答案。

**答案** C

**【例 4】设有定义：long *x*=–123456L;，则以下能够正确输出变量 *x* 的语句是_____。（2002 年 9 月）**

（A）printf("x=%d\n",x);　　（B）printf("x=%ld\n",x);

（C）printf("x=%8dL\n",x);　　（D）printf("x=%LD\n",x) ;

**题眼分析** *x* 为一个长整型的变量，而且是一个十进制的数，它的输出控制符是“ %ld ”或“ %Ld ”，由于 C 语言中是区分大小写的所以“ ld ”中的“ d ”不能为大写。

**答案** B

## 二、填空题分析

**【例 1】以下程序运行后的输出结果是_____。（2004 年 9 月）**

```
main( )
{   int   a,b,c;
    a=25;
    b=025;
    c=0x25;
    printf("%d  %d  %d\n" , a , b , c );
}
```

**题眼分析** 变量 *a*、*b*、*c* 赋值时分别以十进制、八进制和十六进制，输出时都用十进制输出。八进制 025 对应的十进制为 21，十六进制 0x25 对应的十进制为 37。

**答案** 25　21　37

**【例 2】有以下语句段**

```
int  nl=10，n2=20;
printf("_____",n1,n2);
```

**要求按以下格式输出 *n*1 和 *n*2 的值，每个输出行从第 1 列开始，请填空。（2004 年 4 月）**

```
n1=10
n2=20
```

**题眼分析** 变量 n1 和 n2 用十进制进行赋初值，输出也是十进制，因此需要使用格式字符 *d*。输出时要进行换行输出，因此在格式控制中要加入转义字符\n。因此该处需填写“ *n1=%d\n n2=%d\n* ”。

**答案** n1=%d\n n2=%d\n

# 3.3 数据输入

## 3.3.1 考点提炼

### 考点 1：scanf 函数的一般调用形式

scanf 函数的一般调用形式为：

scanf(输入格式字符串，输入变量地址列表)；

格式控制部分是字符串，主要由“%”号和格式字符组成。地址列表是由多个地址组成，可以是变量的地址，也可以是字符串的首地址。

### 考点 2：scanf 函数中常用的格式说明

scanf 函数的格式字符如表 3-2 所示。

表 3-2　　scanf()的格式字符

| 格式字符 | 数据对象 | 输入形式 | 数据输入方法 |
|---|---|---|---|
| %md | int<br>short<br>unsigned int<br>unsigned short | 十进制 | 无 m 时按实际位输入，有 m 时输入 m 位，不足 m 位则跟回车键 |
| %mo | | 八进制 | |
| %mx | | 十六进制 | |
| %mld | long<br>unsigned long | 十进制 | 无 m 时按实际位输入，有 m 时输入 m 位，不足 m 位则跟回车键 |
| %mlo | | 八进制 | |
| %mlx | | 十六进制 | |
| %mf | float | 十进制小数<br>十进制指数 | 无 m 时按实际位输入，有 m 时输入 m 位，不足 m 位则跟回车键 |
| %me | | | |
| %mlf | double | 十进制小数<br>十进制指数 | 无 m 时按实际位输入，有 m 时输入 m 位，不足 m 位则跟回车键 |
| %mle | | | |
| %mc | char | 单个字符 | 无 m 仅取单个字符<br>有 m 位输入 m 位，仅取第 1 个字符 |
| %ms | 字符串 | 一串字符 | 无 m 位取若干个字符直到回车或空格为止，有 m 仅取前 m 个字符 |

### 考点 3：通过 scanf 函数从键盘输入数据

通过 scanf 函数从键盘输入数据时，需要注意以下几点：

（1）其中 *m* 是一个整数，主要是用来控制输入的数据的位数，*m* 可以省略。省略时可以用非格式字符作为两个数据的间隔；也可以在输入时用空格、Tab、回车作为两个数据的间隔。

（2）在 scanf()语句中非格式字符是作为输入数据时的间隔，输入时必须原样地输入。而格式字符对应的数据，输入时必须按照规定的格式输入。

（3）当所有数据输入后，可用一个回车符结束输入。

（4）用%*c* 作为输入字符时仅接受单个字符。从键盘输入单个字符后应该跟回车键作为输入数据的结束，此时回车被作为一个字符存放在缓冲区中，如再有%*c* 作为输入字符时，将不在从键盘读入数据，而是从缓冲区读入还未读完的数据。

（5）输入变量的地址是由接受输入数据的变量地址组成的，变量地址之间用“逗号”分隔开。变量的地址格式为：&变量名。

## 3.3.2 题眼分析

### 一、选择题分析

**【例 1】有以下语句：**int *b*;char *c*[10]; **，则正确的输入语句是_____。（2005 年 4 月）**

（A）scanf("%d%s",&b,&c);　　（B）scanf("%d%s",&b,c);

（C）scanf("%d%s",b,c);　　（D）scanf("%d%s",b,&c);

**题眼分析　函数** scanf()**的第** 2 **参数（即输入列表）需要使用变量的地址。对于一个简单变量** *b*，**其地址是**&*b*，**对于字符数组** *c*[10]，**数组名** *c* **就代表其首地址。**

**答案**　B

**【例** 2**】有以下程序**

```
main()
{   int m,n,p;
    scanf("m=%dn=%dp=%d",&m,&n,&p);
    printf("%d%d%d\n",m,n,p);
}
```

**若想从键盘上输入数据，使变量** *m* **中的值为** 123，*n* **中的值为** 456，*p* **中的值为** 789，**则正确的输入是_____。（2005 年 4 月）**

（A）*m*=123*n*=456*p*=789　　（B）*m*=123　*n*=456　*p*=789

（C）*m*=123,*n*=456,*p*=789　　（D）123　456　789

**题眼分析　函数** scanf()**要求除格式控制字符以外的字符都要按原样输入。该题中“** *m*= **”、“** *n*= **”和“** *p*= **”为非格式符，要按原样输入。由于这些非格式符是紧跟在格式字符之后，因此它们之间不需要用空格隔开。**

**答案**　A

### 二、填空题分析

**【例** 1**】若有程序**

```
main()
{   int i,j;
    scanf("i=%d,j=%d",&i,&j);
    printf("i=%d,j=%d\n",i,j);
}
```

**要求给** *i* **赋** 10，**给** *j* **赋** 20，**则应该从键盘输入_____。（2003 年 4 月）**

**题眼分析　该函数的第** 1 **个参数是格式字符串，主要有两类字符组成，一类是非格式字符，另一类是格式字符对应要输入的变量。对于非格式字符，要按原样输入。**

**答案**　i=10,j=20

**【例** 2**】以下程序运行时若从键盘输入：**10　20　30<**回车**>。**输出结果是_____。（2005 年** 4 **月）**

```
#include <stdio.h>
main()
{   int i=0,j=0,k=0;
```

```
scanf("%d%*d%d",&i,&j,&k);printf("%d%d%d\n",i,j,k);    }
```

**题眼分析**　在格式字符和%之间加一个“*”号，其作用是跳过对应的输入数据。在本题中，把 10 赋给变量 *i*，跳过 20，把 30 赋给变量 *j*，变量 *k* 没有读入数据，仍为 0。

**答案**　10300

# 3.4　复合语句和空语句

## 3.4.1　考点提炼

### 考点 1：复合语句

复合语句是用一对花括号“{}”把若干条语句括起来构成一个语句组。复合语句在语法上视为一条语句，在一对花括号内的语句数量不限。

### 考点 2：空语句

如果一条语句只一个分号“;”称为空语句。程序执行空语句不产生任何动作。

## 3.4.2　题眼分析

**【例】以下 4 个选项中，不能看做一条语句的是_____。（2004 年 4 月）**

（A）{;}　　（B）*a*=0,*b*=0,*c*=0;

（C）if(*a*>0)　　（D）if(*b*==0)*m*=l;*n*=2;

**题眼分析**　选项（A）是一条空语句；选项（B）是由一个逗号表达式构成的赋值句；选项（C）是一条判断语句，但不完整，不能看做一条语句；而选项（D）是一条判断语句和一条赋值语句，共有两条语句。

**答案**　C

# 3.5　单元训练及参考答案

## 3.5.1　单元训练

### 一、选择题

1．以下程序输出的结果是_____。

```
main()
{   printf("\n*a=%15s, ""chinazhongguo");
    printf("\n*b=%-5s*","chi");}
```

（A）*a=chinazhongguo　*　　（B）*a=chinazhongguo　*

b= * *chi*　　*b=chi　*

（C）*a=*　chinabeijing*　　（D）*a=　chinazhongguo*

*b= chi*　　*b=chi　*

2．在 C 语言中，不同的类型数据混合运算时，要先转换成同一类型后进行运算。设一表达式中包含有 int、long、float 和 char 类型的变量和数据，则表达式最后的运算结果是_____。

（A）int　（B）char　（C）float　（D）long

3．若以下选项中的变量已正确定义，则正确的赋值语句是_____。

（A）x1=26.8%3;　（B）1+2=x2;　（C）x3=0x12;　（D）x4=1+2=3;

4．设正 *x*、*y* 均为整型变量，且 *x*=10,*y*=3，则以下语句的输出结果是_____。

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

（A）10,3　（B）9,3　（C）9,2　（D）10,2

5．*a* 和 *b* 均定义为 int，*c* 定义为 double，以下不正确的 scanf( )是_____。

（A）scanf("%d%lx,%le",&a,&b,&c);　（B）scanf("%2d*%d%lf",&a,&b,&c);

（C）scanf("%x%*d%o",&a,&b);　（D）scanf("%x%lo,%6.2f",&a,&b,&c);

6．有如下程序段：

```
int a1,a2;  char b1,b2;
scanf("%d%c%d%c",&a1,&b1,&a2,&b2);
```

则正确的输入是_____。

（A）10A　20B<CR>　（B）10　A　20　B<CR>

（C）10　A20B<CR>　（D）10A20　B<CR>

7．执行语句“scanf("x=%d,y=%d,z=%d",&x,&y, &z);”后，若要求 *x*、*y*、*z* 的值分别为 1、3、2，正确的输入是_____。

（A）132<CR>　（B）1,3,2<CR>

（C）*x*=1,*y*=3,*z*=2<CR>　（D）*x*=1　*y*=3　*z*=2<CR>

8．若变量已正确说明为 float 类型，要通过语句“scanf("%f %f %f ",&a,&b,&c);”给 *a* 赋值 10.0，*b* 赋值 22.0，*c* 赋值 33.0，不正确的输入形式是_____。

（A）10<回车>
22<回车>
33<回车>

（B）10.0,22.0,33.0<回车>

（C）10.0<回车>
22.0　33.0<回车>

（D）10 22<回车>
33<回车>

## 二、填空题

1．以下程序的输出结果是___（1）___。

```
main()
{   int a=177;
    printf("%o\n",a);  }
```

2．执行下列程序时输入：123<空格>456<空格>789<回车>，输出结果是___（2）___。

```
main()
{   char  s[100];   int  c, i;
```

```
    scanf("%c",&c);     scanf("%d",&i);     scanf("%s",s);
    printf("%c,%d,%s \n",c,i,s);
}
```

3．以下程序的执行结果是___（3）___。

```
main( )
{   int a1,a2,b1 ,b2;
    a1=5;   a2=7;   b1=++a1;b2=a2++;
    printf("%d,%d,%d,%d\n",a1,a2,b1,b2);}
```

4．以下程序的执行结果是___（4）___。

```
main()
{   float a=13.8;   int b =5;   b=( (int)a) %3;
    printf("b=%d\n",b);}
```

5．以下程序的执行结果是___（5）___。

```
main()
{   int a=2;    a+=a－=a*a;
    printf("a=%d\n",a);}
```

6．以下程序的执行结果是___（6）___。

```
main()
{   int a,b,c;  c=(a=3,b=a--);
    printf("c=%d,a=%d,b=%d\n",c,a,b);}
```

7．若想通过以下输入语句使 $a$=5.0，$b$=4，$c$=3，则输入数据的形式应该是___（7）___。

```
int b，c；float a；
scanf("%f,%d,c=%d",&a,&b,&c)
```

8．以下程序的执行结果是___（8）___。

```
main()
{   short a =-1;
    printf("dec:%d,oct:%o,hex:%x,unsigned:%u\n", a,a,a,a);}
```

## 3.5.2　参考答案

### 一、选择题

1．D　　2．C　　3．C　　4．D　　5．D

6．A　　7．C　　8．B

### 二、填空题

（1）261　　（2）1,23,456

（3）6,8,6,7　　（4）*b=1*

（5）*a=–4*　　（6）*c=3,a=2,b=3*

（7）*5.0,4,c=3*　　（8）dec: –1,oct:177777,hex:ffff,unsigned:65535

# 第 4 章　选择结构

## 4.1　关系运算和逻辑运算

### 4.1.1　考点提炼

**📖考点 1：C 语言中的逻辑值**

C 语言中没有逻辑型数据，用非零整数表示“真”，用零值表示“假”。

**📖考点 2：关系运算符和关系表达式**

1．关系运算符

关系运算符只用于比较两个数据的大小，运算的结果为成立或不成立。如果成立，则结果是逻辑值“真”，用整数 1 表示；如果不成立，则结果为逻辑值“假”，用整数 0 表示。

关系运算符的运算对象、运算规则与结果如表 4-1 所示。

表 4-1　　关系运算符的运算对象、运算规则与结果

| 对象数 | 名称 | 运算符 | 运算规则 | 运算对象 | 运算结果 |
|---|---|---|---|---|---|
| 双目 | 大于 | > | 满足则为真<br>结果为 1<br>不满足为假<br>结果为 0 | 整型<br>实型<br>字符型 | 逻辑值<br>（整型） |
| | 小于 | < | | | |
| | 小于等于 | <= | | | |
| | 大于等于 | >= | | | |
| | 等于 | == | | | |
| | 不等于 | != | | | |

优先级：算术运算优先于关系运算；<、<=、>= 优先于 ==、!=。

结合性：<、<=、>=是同级自左向右；==、!=同级结合性自左向右。

2．关系表达式

关系表达式是由关系运算符连接表达式构成的，规则如下：

格式：表达式 关系运算符 表达式

说明：其中表达式包括算术表达式、字符型数据、关系表达式、逻辑表达式、条件表达式、逗号表达式、赋值表达式等。

**📖考点 3：逻辑运算符和逻辑表达式**

1．逻辑运算符

逻辑运算符的运算对象、运算规则与结果如表 4-2 所示。

表 4-2　　逻辑运算符的运算对象、运算规则与结果

| 对象数 | 名称 | 运算符 | 运算对象 | 运算结果 |
|---|---|---|---|---|
| 单目 | 逻辑非 | ! | 数值型字符型 | 逻辑值 |
| 双目 | 逻辑与 | && | | |
| | 逻辑或 | \|\| | | |

逻辑运算符运算规则如表 4-3 所示。

表 4-3　　逻辑运算符运算规则

| 对象 1(A) | 对象 2(B) | 逻辑与运算(A&&B) | 逻辑或运算(A\|\|B) | 逻辑非（!A） |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 |
| 0 | 非 0 | 0 | 1 | 1 |
| 非 0 | 0 | 0 | 1 | 0 |
| 非 0 | 非 0 | 1 | 1 | 0 |

注意：逻辑值的表示方法：对于参加运算的逻辑真值，是用非 0 表示的，而运算的结果用数值 1 来表示的。而对逻辑假不管是参加运算，还是运算结果，都是用数值 0 表示的。

优先级：! 优先于 双目算术运算符 优先于 关系运算符 优先于 && 优先于 ||。

结合性：! 和单目算术运算符是同级的，结合性自右向左。双目运算符的结合性是自左向右。! 的结合性是自右向左，依次计算。

2．逻辑表达式

逻辑表达式是有逻辑运算符连接表达式构成的，格式如下：

格式 1：单目逻辑运算符　　表达式

格式 2：表达式　　双目逻辑运算符　　表达式

说明：其中表达式包括算术表达式、字符型数据、关系表达式、逻辑表达式、条件表达式、逗号表达式和赋值表达式等。但是主要的是关系表达式。

注意：用&&对两个表达式计算时，若第 1 个表达式的值为“假”则运算结果与第 2 个表达式无关，此时第 2 个表达式将不再计算。当用||对两个表达式进行运算时，如果第 1 个表达式的值为“真”，则运算结果与第 2 个表达式的结果无关，此时第 2 个表达式不再计算。

### 4.1.2　题眼分析

#### 一、选择题分析

【例 1】能正确表示逻辑关系：“$a \geq 10$ 或 $a \leq 0$”的 C 语言表达式是_____。（2000 年 9 月）

（A）a>=10 or a=0　　（B）a>=0|a<=10　　（C）a>=10&&a<=0　　（D）a>=10||a<=0

题眼分析　在 C 语言中逻辑运算符号有&&、||、!这 3 个，关系运算符有>、<、>=、<=、==5 个。根据逻辑关系很容易看出符合条件的选项（D）是正确选项。

答案　D

【例 2】有以下程序

```
main()
{   int a,b,d=25;
    a=d/10%9;   b=a&&(-1);
    printf("%d,%d\n",a,b);      }
```

程序运行后的输出结果是_____。（2005 年 4 月）

（A）6,1　　（B）2,1　　（C）6,0　　（D）2,0

题眼分析　经过执行 *a=d*/10%9;,变量 *a* 的值为 2,2 和–1 都是非零值,两者相与将得‘真’。在 C 语言中，逻辑值“真”用整数 1 表示，也就是说变量 *b* 的值为 1。

答案　B

【例 3】有以下程序

```
main()
{   int i=1,j=2,k=3;
    if(i++==1&&(++j==3||k++==3))   printf("%d  %d  %d\n",i,j,k);
}
```

程序运行后的输出结果是_____。（2005 年 4 月）

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

题眼分析　在条件表达式中，当“&&”运算的第 1 个表达式的值为“假”时，第 2 个表达式将不再运算；当“||”运算中第 1 个表达式的值为“真”时，第 2 个表达式将不再运算。在表达式 i++==1&&(++j==3||k++==3)中，先运算第 1 个表达式 i++==1，其值为“真”，*i* 变为 2，再运算第 2 个表达式++j==3||k++==3。在表达式++j==3||k++==3 中，先运算第 1 个表达式 ++j==3，其值为“真”，*j* 变为 3，这时第 2 个表达式 k++==3 将不再运算，因此 *k* 的值不变。这样变量 *i* 为 2，*j* 为 3，*k* 不变，仍为 3。又由于整个表达式 i++==1&&(++j==3||k++==3)值为“真”，所以输出 2　3　3，即选项（D）是正确答案。

答案　D

## 二、填空题分析

【例】以下程序运行后的输出结果是_____。（2005 年 4 月）

```
main()
{   int a,b,c;
    a=10;b=20;c=(a%b<1)||(a/b>1);
    printf("%d %d %d\n",a,b,c);     }
```

题眼分析　算术运算符的优先级比关系运算符要高，所以表达式 *a*%*b*<1 相当于(*a*%*b*)<1，其值为逻辑“假”；同样，表达式 *a*/*b*>1 的值也为逻辑“假”；两者进行“逻辑或”运算仍为逻辑“假”，用整数 0 来表示，即变量 c 的值为 0。

答案　10　20　0

# 4.2　if 语句和用 if 语句构成的选择结构

## 4.2.1　考点提炼

### 考点 1：单分支选择语句（if 语句）

**单分支 if 语句一般形式为：**

**格式：**if(**条件**)　**语句**;

**功能：先判断条件是否为真（非 0），若为真则执行语句，否则不执行语句。**

**说明：条件可以是任意的类型的表达式，但常用的是关系表达式或逻辑表达式。**

### 考点 2：双分支选择语句（if-else 语句）

**格式：**if (**表达式**)　**语句** 1;

　　else　**语句** 2;

**功能：先计算表达式的值。如果为真（非 0）执行“ 语句 1 ”，否则执行“ 语句 2 ”。**

### 考点 3：选择结构的嵌套

C **语言规定，在分支语句** if **的各个语句组中也可以出现** if **语句或** switch **语句，在** switch **语句的各个分支中也可以出现** if **语句和** switch **语句，把这种情况称为选择结构的嵌套。**

**说明：**

**（1）在使用** if-else **嵌套时注意保证** if-else **语句的完整性，书写或输入源程序时应采用缩进原则，这样也便于检查是否出错。**

**（2）由于** C **程序对书写规则并无特定的要求，所以在程序中使用** if **语句的嵌套可能给理解上带来“ 二义性 ”。因此** C **语言规定，在出现这种情况时，将按照下列原则处理：**else **总是与最近的** if **配对。**

## 4.2.2　题眼分析

### 一、选择题分析

**【例** 1**】下列条件语句中，功能与其他语句不同的是_____。（**2004 **年** 9 **月）**

（A）if (a) printf("%d\n", x ) ;　else printf("%d\n", y ) ;

（B）if (a= =0)　printf("%d\n", y ) ;　else printf("%d\n", x ) ;

（C）if (a!=0)　printf("%d\n", x ) ;　else printf("%d\n", y ) ;

（D）if (a= =0)　printf("%d\n", x ) ;　else printf("%d\n", y ) ;

**题眼分析　选项（**A**）的含义是：若** *a* **非零，则输出变量** *x*，**否则输出变量** *y*。**选项（**B**）的含义是：若** *a* **为零，则输出变量** *y*，**否则输出变量** *x*。**选项（**C**）的含义是：若** *a* **不等于零，则输出变量** *x*，**否则输出变量** *y*。**这** 3 **个含义是完全一样。选项（**D**）的含义是：若** *a* **等于零，则输出变量** *x*，**否则输出变量** *y*，**明显不同于前** 3 **个选项。**

**答案** D

**【例 2】有以下程序**

```
main()
{   int i=l,j=1,k=2;
    if((j++||k++)&&i++)  printf("%d,%d,%d\n",i,j,k);
}
```

**执行后输出结果是_____。（2003 年 4 月）**

（A）1,1,2　　（B）2,2,1　　（C）2,2,2　　（D）2,2,3

**题眼分析** 首先计算 if 语句后面的表达式值，先计算&&前面括号里的“||”运算。在||运算中先计算 j++，值为 1（*j* 的值变为 2），为“真”，后面的 k++将不再计算，*k* 的值依旧为 2。计算&&后面的表示式 i++，值为 1（*i* 的值变为 2）。整个表达式的值为“真”，所以执行后面的输出语句。

**答案** C

**【例 3】有以下程序**

```
main( )
{   int  a=3 , b=4 , c=5 , d=2 ;
    if (a>b)
        if(b>c)
            printf("%d", d++ + 1 ) ;
        else
            printf("%d", ++d + 1 ) ;
    printf("%d\n", d ) ;
}
```

**程序运行后的输出结果是_____。（2004 年 9 月）**

（A）2　　（B）3　　（C）43　　（D）44

**题眼分析** C 语言规定 else 总是和离它最近的 if 语句配对。程序中的 else 语句与第 2 个 if 配对。由于第 1 个 if 语句的判断条件(a>b)为假，第 2 个 if 语句并没运行，因此变量 *d* 的值没有发生改变。

**答案** A

**【例 4】有以下程序**

```
main()
{   int a=5,b=4,c=3,d=2;
    if(a>b>c)
        printf("%d\n",d);
    else     if ((c-1>=d)==1)
            printf("%d\n",d+1)
        else
            printf("%d\n”,d+2);
}
```

**执行后输出结果是_____。（2003 年 4 月）**

（A）2　　（B）3

（C）4　　（D）编译时有错，无结果

**题眼分析**　根据 else 总是和离它最近的 if 语句配对这一规则，第 1 个 else 和第 1 个 if 配对，第 2 个 else 和第 2 个 if 配对。首先计算第 1 个 if 后面的表达式“ a>b>c ”，“ a>b ”是“ 真 ”，为 1，“ 1>c ”为“ 假 ”，值为 0，所以执行 else 后面的语句。先执行 if 后面的表达式，“ c-1>=d ”为“ 真 ”，值为 1，“ 1==1 ”为“ 真 ”，所以执行紧接后面的 printf 语句。

**答案**　B

**【例 5】有一函数：** $y=\begin{cases}1 & x>0\\ 0 & x=0\\ -1 & x<0\end{cases}$ **。则以下程序段中不能根据 $x$ 值正确计算出 $y$ 值的是_____。（2002 年 9 月）**

（A）
```
if(x>0) y=1;
else if(x==0) y=0;
else y=–1;
```

（B）
```
y=0;
if (x>0) y=1;
else  if(x<0) y=–1;
```

（C）
```
y=0;
if(x>=0)
if(x>0) y=1;
else  y= –1;
```

（D）
```
if(x>=0)
if(x>0)  y=1;
else y=0;
else y= –1;
```

**题眼分析**　首先检查 if 与 else 的配对，然后再分析各分支实现的功能。答案（C）描述的意思是：当 $x\geq 0$ 的情况下，如果 $x>0$ 时 $y$ 为 1，否则即 $x==0$ 时，$y$ 为–1；剩下的 $x<0$ 时，$y$ 为 0。可见答案（C）实现的结果不是给定的表达式。

**答案**　C

## 二、填空题分析

**【例 1】若有以下程序**

```
main()
{   int a=4,b=3,c=5,t=0;
    if(a<b)t=a; a=b; b=t;
    if(a<c)t=a; a=c; c=t;
    printf("%d%d%d\n",a,b,c);
}
```

**执行后输出结果是_____。（2003 年 4 月）**

**题眼分析**　如果 if 后面的条件为“ 真 ”，只执行其后的一条语句或一条复合语句，如果 if 后面的条件为“ 假 ”，只执行 else 后面的一条语句或一条复合语句。先判断表达式“ $a<b$ ”为“ 假 ”，不执行“ $t=a;$ ”，但执行“ $a=b;b=t;$ ”，$a$ 的值为 3，$b$ 的值为 0。再判断表达式“ $a<c$ ”，值为“ 真 ”，所以执行后面的 3 条语句“ $t=a;\ a=c;\ c=t;$ ”，结果 $a$ 的值为 5，$c$ 的值为 3。

**答案**　503

**【例 2】以下程序运行后的输出结果是_____。（2003 年 9 月）**

```
main()
{   int a=1,b=3,c=5;
    if (c=a+b)      printf("yes\n");
```

```
        else            printf("no\n");     }
```

**题眼分析** if 语句判断条件是一个赋值表达式。赋值表达式的值是赋值结果，在本题中即为 *c* 的值，由于 *c=a+b=4*，非零条件为真，因此将输出“ yes ”。

**答案** yes

**【例 3】若有以下程序**

```
main()
{   int  p,a=5;
    if(p=a!=0)
        printf("%d\n",p);
    else
        printf(”%d\n",p+2); }
```

**执行后输出结果是_____。（2003 年 4 月）**

**题眼分析** 首先计算 if 语句后面的表达式，根据运算符的优先级可知，先算“ *a!=0* ”，值为 1，再把 1 赋值给 *p*，结果为 1（真），执行其后的 printf 语句，输出的值为 1。

**答案** 1

# 4.3 条件表达式构成的选择结构

## 4.3.1 考点提炼

### 考点 1：条件运算符

条件运算符由两个运算符组成，它们是：？ ： 。这是 C 语言中惟一的一个三目运算符，即要求有 3 个运算对象。

### 考点 2：条件表达式

条件表达式是由条件运算符连接表达式构成的，格式如下：

表达式 1？表达式 2：表达式 3

功能：若表达式 1 的值为“ 真 ”，则返回表达式 2 的值，否则返回表达式 2 的值。

说明：其中表达式 1 主要是由逻辑表达式或关系表达式，也可以是其他的表达式（当值为非 0 看成逻辑真，值为 0 时看成逻辑假）。表达式 2 和表达式 3 一般是同类型的表达式。

### 考点 3：条件运算符的优先级

条件运算符优先于赋值运算，但低于逻辑运算、关系运算和算术运算。

## 4.3.2 题眼分析

### 一、选择题分析

**【例 1】以下程序段中与语句 *:k=a>b?(b>c?l:0):0;* 功能等价的是_____。（2004 年 4 月）**

（A）if((a>b&&(b>c))k=1
　　else　k=0

（B）if((a>b)||(b>c))k=1;
　　else　k=0;

（C）if(a<=b)　k=0;
　　else if (b<=c)　k=1;

（D）if(a>b)　k=1;
　　else if (b>c)　k=1;
　　else　k=0；

**题眼分析**　根据条件表达式的功能，该表达式对应的分支结构是：

```
if(a>b)
    if(b>c)  k=1;
    else k=0;
else k=0
```

显然，它的含义是当 *a>b* 且 *b>c* 时，给 *k* 赋值 1，否则赋值 0，即选项（A）所完成的功能。

**答案**　A

**【例 2】以下程序的输出结果是_____。（2002 年 4 月）**

```
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

**题眼分析**　首先计算括号内的条件表达式，它的值为 6，然后再计算外面条件表达式的值，赋值给 d。

**答案**　C

**【例 3】若整型变量 *a*、*b*、*c*、*d* 中的值依次为：1、4、3、2。则条件表达式 *a<b?a:c<d?c:d* 的值是_____。（2005 年 4 月）**

（A）1　　（B）2　　（C）3　　（D）4

**题眼分析**　条件表达式的结合方向是“自右向左”。所以上述表达式等价于 a<b?a:(c<d?c:d)。

**答案**　A

## 二、填空题分析

**【例】以下程序运行后的输出结果是_____。（2003 年 9 月）**

```
main()
{   int  p=30;
    printf("%d\n",(p/3>0?p/10:p%3));
}
```

**题眼分析**　首先计算括号内的条件表达式，由于 *p/3=10>0*，因此该表达式的值为 *p/10=3*。

**答案**　3

# 4.4 switch 语句

## 4.4.1 考点提炼

**多分支选择语句的格式为：**

```
switch(表达式)
    {   case 常量表达式1: 语句组1;
                          [break;]
        case 常量表达式2: 语句组2;
                          [break;]
          ……
        case 常量表达式n: 语句组n;
                          [break;]
        [default:        语句组n+1;]
      }
```

**功能：先计算表达式的值，再依次和 case 中的“常量表达式”的值比较，若与某个常量表达式的值相等，则执行其后的语句组，如果该语句组中没有 break 语句，则接着向下执行，直到遇到“**break;**”语句或语句执行完为止。如果表达式的值和任一常量表达式的值都不相等，则执行 default 后面的语句，若无“**default:**语句组;**”，则不执行任何语句。**

**注意：此处 break 语句的作用是退出分支语句，若无则一直向下执行。**

## 4.4.2 题眼分析

**【例 1】若 *a*、*b*、*c1*、*c2*、*x*、*y* 均是整型变量，正确的 switch 语句是_____。（2001 年 4 月）**

**（A）** swich(a+b);
{ case 1:y=a+b; break;
case 0:y=a–b; break;
}

**（B）** switch(a*a+b*b)
{ case 3:
case 1:y=a+b;break;
case 3:y=b–a,break;
}

**（C）** switch a
{ case c1 :y=a–b; break;
case c2: x=a*d; break ;
default:x=a+b;
}

**（D）** switch(a–b)
{ default:y=a*b;break;
case 3:case 4:x=a+b;break;
case 10:case 11:y=a–b;break;
}

**题眼分析　选项（A）中关键词“**switch**”拼写错，且 switch 语句后不能有分号；对于选项（C），switch 后的表达式必须要加括号，case 后的表达式必须是常量，所以选项（C）是错误的；C 语言规定 switch 语句中的常量不能在分支中重复出现，所以选项（B）也是错误的。因此，只有选项（D）是正确的。**

**答案**　D

【例 2】有以下程序

```
main()
{       int i;
    for(i=0;i<3;i++)
    switch(i)
        {  case 1: printf("%d",i);
           case 2: printf("%d",i);
           default: printf("%d",i);
       }
}
```

执行后输出结果是_____。（2003 年 4 月）

（A）011122　　（B）012　　（C）012020　　（D）120

题眼分析　for 循环执行 3 次，第 1 次时 *i* 的值为 0，执行其后的 switch 语句，没有匹配的 case，执行 default 语句后的 printf 语句，输出为 0；第 2 次循环时 *i* 的值为 1，执行其后的 switch 语句，与第 1 个 case 语句匹配，执行其后的 printf 语句，输出 1，由于没有遇到 break 语句，所以一直向下执行，又输出了两个 1；第 3 次循环时 *i* 的值为 2，执行其后的 switch 语句，同理可知输出两个 2。

答案　A

【例 3】以下程序运行后的输出结果是_____。（2004 年 9 月）

```
main( )
{   int   x=1,y=0,a=0,b=0;
    switch (x)
    {   case  1:switch (y)
                {   case  0:  a++; break;
                    case  1:  b++; break;
                }
        case  2:a++; b++; break;
    }
    printf("%d  %d\n" , a , b );
}
```

题眼分析　这是一个 switch 语句的嵌套，由于 *x*=1，则从 case 1 开始执行，在内层 switch 语句中，由于 *y*=0，则从 case 0 开始执行，*a* 自加变为 1，遇到 break 语句跳出内层 switch 语句。由于外层 switch 语句的 case 1 没有跳出语句，接下来执行外层 switch 语句的 case 2，*a* 再次自加变为 2，*b* 自加变为 1，最后跳出外层 switch 语句，并输出 *a* 和 *b*。

答案　2　1

# 4.5 语句标号和 goto 语句

## 4.5.1 考点提炼

### 考点 1：语句标号

在 C 语言中，可以在任何语句前加上语句标号。标号可以是任意合法的标识符，并在标识符后面加一个冒号“ :”。

### 考点 2：goto 语句

goto 语句的作用是把程序执行转向语句标号所在的位置。goto 语句的一般格式为：

goto 语句标号;

goto 语句的跳转仅限在一个函数内。在结构化程序设计中，主张限制使用 goto 语句，因为滥用 goto 语句将使程序流程无规律、可读性差。

## 4.5.2 题眼分析

【例】下面说法正确的是_____。

（A）goto 语句可以提高程序效率，因此在程序设计中提倡使用 goto 语句

（B）goto 语句可以在一个程序内随意跳转

（C）“ 123 :” 是一个合法语句标号

（D）以上说法都不正确

题眼分析　虽然 goto 语句在有的时候可以提高程序效率，但将使程序流程无规律、可读性差，不符合结构化程序设计思想；goto 语句仅限在一个函数内的跳转，不能跨越函数；语句标号必须是一个合法标识符，并在后面加一个冒号“ :”,“ 123 ” 不是一个合法标识符。

答案　D

# 4.6 单元训练及参考答案

## 4.6.1 单元训练

### 一、选择题

1．语句“ printf("%d",(a=2)&&(b=–2)); ” 的输出结果是_____。

（A）无输出　　（B）结果不确定　　（C）–1　　（D）1

2．设有定义：int x=1,y=-1;。则语句“ printf("%d\n",(x--&&++y); ”的输出结果是_____。

（A）1　　（B）0　　（C）–1　　（D）2

3．当 *c* 的值不为 0 时，_____能正确将 *c* 的值赋给变量 *a*、*b*。

（A）c=b=a;　　（B）(a=c)||(b=c)　　（C）(a=c)&&(b=c)　　（D）a=c=b

4．在 C 语言的 if 语句中，用做判断的表达式为_____。

（A）关系表达式　（B）逻辑表达式　（C）算术表达式　（D）任意表达式

5．在以下运算符中，优先级最高的运算符是_____。

（A）<=　（B）/　（C）!=　（D）&&

6．以下不正确的 if 语句形式是_____。

（A）if(x>y&&!=y);　（B）if(x==y) x+=y;

（C）if(x!=y) scanf("%d",&x) else scanf("%d",&y);　（D）if(x<y) {x++;y++;}

7．已知 int x=10,y=20,z=30;以下语句执行后的值是_____。

```
if (x>y) z=x;x=y;y=z;
```

（A）x=10,y=20,z=30　（B）x=20,y=30,z=30

（C）x=20,y=30,z=10　（D）x=20,y=30,z=30

8．请阅读以下程序：

```
main()
{   int a=5,b=0,c=0;
    if(a=b+c) printf("***\n");
    else printf("$$$"\n);
}
```

以上程序_____。

（A）有语法错误不能被编译　（B）可以通过编译但不能通过连接

（C）输出***　（D）输出$$$

## 二、填空题

1．在 C 语言中逻辑“真”值用__（1）__。

2．设 *a*，*b*，*c* 均为整型变量，请描述出“*a* 或 *b* 中有一个小于 *c*”的表达式__（2）__。

3．已知 *x*=7.5，*y*=2，*z*=3.6，则表达式 *x>y&&z>x<y&&!z>y* 的值是__（3）__。

4．已知 *x*=6，*y*=4，*c*=2，则表达式 *!(x–y)+z–1&&y+z/2* 的值是__（4）__。

5．已知 *x*=2，*y*=4，则表达式 *!(x=a)||(y=b)&&0* 的值是__（5）__。

6．已知 *x*=1，*y*=4，*z*=3，则表达式 *!(x<y)||!z&&1* 的值是__（6）__。

7．以下程序运行的结果是__（7）__。

```
main()
{   int a,b,c;
    a=1;b=2;c=3; a=b--<=a||a+b!=c;
    printf("%d,%d",a,b);   }
```

8．以下程序的执行结果是__（8）__。

```
main( )
{   float f1,f2,f3,f4; int m1,m2; f1=f2=f3=f4=2; m1=m2=1;
    printf("%d\n",(m1=f1>=f2)&&(m2=f3<f4));   }
```

9．以下程序的执行结果是__（9）__。

```
main()
{   int a,b,c; a=2;b=3;c=1;
    if (a>b)  if (a>c)printf("%d\n",a);
```

```
    else printf("%d\n",b);
    printf("end\n");
}
```

10．以下程序的执行结果是__（10）__。

```
main()
{   int x=1,y=0;
    switch(x)
    {   case 1: switch (y)
                {   case 0: printf("first  ");break;
                    case 1: printf("second  ");break;
                }
        case 2:printf("third  ");
    }
}
```

11．以下程序运行后的输出结果是__（11）__。

```
main()
{   int a=3,b=4,c=5,t=99;
    if(b<a&&a<c)t=a;a=c;c=t;
    if(a<c&&b<c)t=b;b=a;a=t
    printf("%d%d%d\n",a,b,c);
}
```

### 4.6.2 参考答案

**一、选择题答案**

1．D　　2．B　　3．C　　4．D　　5．B

6．C　　7．B　　8．D

**二、填空题**

（1）整数　　（2）$a<c||b<c$

（3）0　　（4）1

（5）0　　（6）0

（7）1,1　　（8）0

（9）end　　（10）first　third

（11）4599

# 第 5 章　循环结构

## 5.1　while 语句和用 while 语句构成的循环结构

### 5.1.1　考点提炼

**📖考点 1：while 循环结构**

while 语句用来实现“当型”循环结构。一般形式如下：

```
while （表达式）
    语句;                    /* 即循环体部分 */
```

**📖考点 2：while 循环结构的执行过程**

其执行过程为：先计算 while 后面圆括号内的表达式，如果其值为“真”(非 0)，则执行循环语句部分，然后再次计算 while 后面圆括号内的表达式，重复上述过程，直到表达式的值为“假”，退出循环，转入下一条语句去执行。

**📖考点 3：while 循环结构使用注意事项**

使用 while 语句时，需要注意的几个问题：

(1) 循环体如果包含一个以上的语句，应该用花括号括起来，以复合语句形式出现。

(2) 其特点是：先判断表达式，后执行语句。

(3) 在循环体中应有使循环趋于结束的语句。可以在表达式本身中实现，也可以在循环体中实现。

### 5.1.2　题眼分析

**一、选择题分析**

【例 1】有以下程序段：

```
int k=0
while (k=1)  k++;
```

while 循环执行的次数是_____。(2001 年 4 月)

(A) 无限次　　(B) 有语法错，不能执行

(C) 一次也不执行　　(D) 执行 1 次

题眼分析　$k$ 的初值为 0。执行 while 后面小括号中的赋值表达式“$k$=1”，则 $k$ 的值为 1，即整个表达式的值为真，所以执行循环体，执行语句“$k$++;”。因为循环控制表达式为赋值表

达式“*k*=1”，其值永远是 1（即逻辑真），没有值为 0（即逻辑假）的可能，因此会出现死循环。

答案　A

【例 2】以下程序的输出结果是_____。（2001 年 9 月）

```
main()
{   int  num= 0;
    while(num<=2)
    {   num++;  printf("%d\n",num); }
}
```

（A）1　2　3　4　　（B）1　2　3　　（C）1　2　　（D）l

题眼分析　程序中，变量 num 初值为 0，判断循环表达式“num<=2”是否成立，表达式“0<=2”成立，执行循环体，输出 num 的值为 1，因输出语句中有“\n”，所以输出第 1 个值后，回车换行；继续判断循环条件，表达式“1<= 2”成立，执行循环体，输出 num 的值为 2；继续判断循环条件，表达式“2<= 2”成立，执行循环体，输出 num 的值为 3；此时再继续判断，表达式“3<=2”不成立，结束循环。

答案　B

【例 3】有以下程序

```
main()
{   int  x=0,y=5,z=3;
    while(z-->0&&++x<5)y=y-l;
    printf("%d, %d ,%d\n",x,y,z);
}
```

程序执行后的输出结果是_____。（2004 年 4 月）

（A）3,2,0　　（B）3,2, –1　　（C）4,3, –1　　（D）5, –2, –5

题眼分析　循环表达式中用两个变量来控制循环，但两个变量有很大区别。对于变量 *z* 是先判断后自减，对于变量 *x* 是先自增再判断。第 1 次循环时，循环表达式等价于(3>0&&1<5)，条件成立，执行循环体，这样在第 1 次循环结束时 *z*=2、*x*=1、*y*=4。第 2 次循环时，循环表达式等价于（2>0&&2<5），条件成立，执行循环体，这样在第 2 次循环结束时 *z*=1、*x*=2、*y*=3。第 3 次循环时，循环表达式等价于（1>0&&3<5），条件成立，执行循环体，这样在第 3 次循环结束时 *z*=0、*x*=3、*y*=2。第 4 次循环时，运算循环表达式时，发现 *z*=0，即条件 *z*>0 不成立，这样就不计算++*x*<5，也不执行循环体，但执行了 *z* 的自减。这样最后的结果是 *z*=–1、*x*=3、*y*=2，即选项（B）为正确答案。这一题所考的考点比较多，要考虑自增自减运算的时机、&& 的运算规则和 while 循环。

答案　B

【例 4】有以下程序

```
main()
{   int p[8]={11,12,13,14,15,16,17,18},i=0,j=0;
```

```
        while(i++<7) if(p[i]%2) j+=p[i];
        printf("%d\n",j);
    }
```

程序运行后的输出结果是_____。（2005 年 4 月）

（A）42　　（B）45　　（C）56　　（D）60

**题眼分析**　语句 while(i++<7) if(p[i]%2) j+=p[i];**等价于** while(i<7){i++; if(p[i]%2) j+=p[i];}。**这就是说程序的功能是求** p[1]～p[7]**中所有奇数元素的和，结果为** 45。**这里需要注意的是从** p[1]**开始，而不是从** p[0]**开始。**

**答案**　B

**二、填空题分析**

**【例】设有以下程序：**

```
main()
 {   int n1,n2;
     scanf("%d",&n2);
     while(n2!=0) {  n1=n2%10; n2=n2/10; printf("%d",n1); }
 }
```

程序运行后，如果从键盘上输入 1298；则输出结果为_____。（2001 年 9 月）

**题眼分析**　此程序的功能是从键盘上输入一个数，然后反序输出这个数。第 1 次执行循环时，判断循环条件‘ n2!=0 ’是否成立，即表达式‘ 1298!=0 ’成立，执行语句‘ n1=n2%10;n2=n2/10; ”后，n1 和 n2 的值分别为 8 和 129，输出第 1 个数 8。第 2 次循环执行过后，n1 和 n2 的值分别为 9 和 12，输出第 2 个数 9。第 3 次循环执行过后，n1 和 n2 的值分别为 2 和 1，输出第 3 个数 2；第 3 次循环执行过后，n1 和 n2 的值分别为 1 和 0，输出第 4 个数 1。当第 5 次执行循环时，循环条件为假，结束循环。

**答案**　8921

# 5.2　do-while 语句和用 do-while 语句构成的循环结构

## 5.2.1　考点提炼

**考点 1：do-while 循环结构**

do-while 语句用来实现“ 直到型 ”循环结构。其一般形式为：

```
do 语句;
while (表达式);
```

**考点 2：do-while 循环结构的执行过程**

do-while 循环结构的执行过程如下：

（1）执行 do 后面循环体中的语句。

（2）计算 while 后面括号中表达式的值，当值为非 0 时，转去执行步骤（1）；值为 0 时，执行步骤（3）。

（3）退出 do-while 循环

📖考点 3：do-while 循环结构使用注意事项

使用 do-while 语句时，需要注意以下问题：

（1）do-while 语句的特点是：先执行语句，后判断表达式。所以，无论一开始表达式的值为“真”还是“假”，循环体中的语句至少执行一次，这一点与 while 语句不同。

（2）如果 do-while 语句的循环体部分是由多个语句组成的，则必须用花括号括起来，使其形成复合语句。

（3）与其他语言中类似语句不同的是，C 语言中的 do-while 语句是在表达式为“真”时重复执行循环体。

## 5.2.2 题眼分析

### 一、选择题分析

【例 1】以下叙述正确的是_____。（2000 年 4 月）

（A）do-while 语句构成的循环不能用其他语句构成的循环来代替。

（B）do-while 语句构成的循环只能用 break 语句退出。

（C）用 do-while 语句构成的循环，在 while 后的表达式为非零时结束循环。

（D）用 do-while 语句构成的循环，在 while 后的表达式为零时结束循环。

题眼分析　do-while 语句构成的循环能用 for 语句、while 语句构成的循环来代替。若处理同一问题，一般情况下 3 种循环语句可以互相代替。do-while 语句可以用 break 语句退出，也可以利用循环条件控制循环的执行。do-while 语句是在表达式为真（非 0）时重复执行循环体。

答案　D

【例 2】有如下程序

```
main()
{   int s=0,a=1,n;
    scanf("%d",&n);
    do
    {   s+=1;a=a-2; }
    while(a!=n)
    printf("%d\n",s);
}
```

若要使程序输出 2，则应该从键盘输入的值是_____。（2003 年 9 月）

（A）–1　　（B）–3　　（C）–5　　（D）0

题眼分析　do-while 语句的特点：先执行循环体，后判断条件。从程序中可以看出，其实变量 s 记录的是循环体的执行次数，循环通过变量 a 来控制循环，其初值为 1，每执行一次 *a*

的值减 2，变量 *n* 用来控制循环结束。当循环体执行完一次时，*a*=–1，*s*=1；当循环体执行完第 2 次时，*a*=–3，*s*=2，这时就必需要退出循环，即要使循环条件 *a*!=*n* 为假，因此 *n*=–3。

**答案**　B

**【例 3】有以下程序段**

```
int x=3
do {  printf ("%d", x –=2); }
while (!(—x) );
```

其输出结果是_____。（2001 年 4 月）

（A）1　　（B）3 0　　（C）1 –2　　（D）死循环

**题眼分析**　变量 *x* 的初值为 3，执行输出语句后，输出 *x* 的值为 1，再执行表达式“!(—*x*)”后，*x* 的值为 0；继续执行循环体，执行输出语句后 *x* 的值为–2，再判断条件表达式“!(—*x*)”的值为假，结束循环。

**答案**　C

## 二、填空题分析

**【例】以下程序运行后的输出结果是_____。（2001 年 9 月）**

```
main()
{   int i=10, j=0;
    do{  j=j+i; i—; } while(i>2);
    printf("%d\n",j);
}
```

**题眼分析**　在程序中，输出语句在循环体外，因此，只输出最后一个 *j* 的值。do-while 语句的特点：先执行循环体，后判断条件。第 1 次循环执行后，*j* 和 *i* 的值分别为 10 和 9，判断循环条件为真，继续执行循环体；*j* 和 *i* 的值分别为 19 和 8，继续上述执行过程，直到 *i* 的值为 2 时，表达式“2>2”不在成立，退出循环，此时，*j* 的值为 47。

**答案**　47

# 5.3　for 语句和用 for 语句构成的循环结构

## 5.3.1　考点提炼

**考点 1：for 的一般形式**

for 语句的一般形式为：

```
for（表达式 1;表达式 2;表达式 3 ）
    语句;
```

**考点 2：for 循环结构的执行过程**

for 循环结构的执行过程如下：

（1）求解表达式 1。

（2）求解表达式 2，若其值为“真”，则执行 for 语句中的循环体，然后执行下面第（3）步。若为假，则结束循环转到第（5）步。

（3）求解表达式 3。

（4）转回上面第（2）步继续执行。

（5）执行 for 语句下面的一个语句。

📖考点 3：使用 for 语句须注意事项

使用 for 语句时，需要注意以下几个问题：

（1）for 语句一般形式中的“表达式 1”可以省略，此时应在 for 语句之前给循环变量赋初值。注意：省略表达式 1 时，其后的分号不能省略，如

```
for( ; i<=100; i++ )  sum+=i;
```

执行时，跳过“求解表达式 1”这一步，其他不变。

（2）如果表达式 2 省略，即不判断循环条件，相当于表达式 2 始终为“真”。

（3）表达式 3 也可以省略，但此时程序设计者应保证循环能正常结束。如：

```
for(sum=0, i=1; i<=100; )
{   sum+=i; i++ ;   }
```

本例把 i++的操作不放在 for 语句的表达式 3 的位置处，而作为循环体的一部分，效果是一样的，都能使循环正常结束。

（4）可以省略表达式 1 和表达式 3，只有表达式 2，即只给循环条件。如：

```
for( ; i<=100; ) { sum+=i; i++;}
```

（5）3 个表达式都可以省略，如：

```
for(  ;  ;  )  语句
```

退出循环可用 break 语句。

## 5.3.2 题眼分析

### 一、选择题分析

【例 1】有以下程序

```
main()
{   int  i,s=0;
    for(i=1;i<10;i+=2)   s +=i+1
    printf("%d\n",s);
}
```

程序执行后的输出结果是_____。（2004 年 4 月）

（A）自然数 1~9 的累加和　　（B）自然数 1~ 10 的累加和

（C）自然数 1~9 中奇数之和　　（D）自然数 1~ 10 中偶数之和

题眼分析　先看循环变量 *i* 的变化情况，*i* 的初值为 1，终止条件是 *i*<10，每次循环时增加 2，因此第 1 次执行循环体的 *i*=1，最后一次执行循环体的 *i*=9，当 *i*=11 时退出循环。这样变量 *s* 依次累加 2，4，…，10。

**答案**　D

**【例 2】有如下程序段，其中 *s*、*a*、*b*、*c* 均已经定义为整型变量，且 *a*、*c* 均已经赋值（*c* 大于 0）**

```
s=a;
for(b=1;b<=c;b++) s=s+1;
```

**则与上述程序段功能等价的赋值语句是_____。（2003 年 9 月）**

（A）s=a+b;　　（B）s=a+c;　　（C）s=s+c　　D）s=b+c;

**题眼分析**　循环变量 *b* 的初值为 1，终止条件是 *b*<=*c*，每次循环时增加 1，因此循环体共执行 *c* 次。每执行一次循环，*s* 值加 1，因此 *s* 最后的值为 *a*+*c*。

**答案**　B

**【例 3】以下程序执行后 sum 的值是_____。（2001 年 4 月）**

（A）15　　（B）14　　（C）不确定　　（D）0

```
main()
{   int i , sum;
    for(i=1;i<6;i++) sum+=i;
    printf("%d\n",sum);
}
```

**题眼分析**　C 语言中，如果对定义的变量没有赋初值，那么在后面使用时，变量的值是不确定的。所以，在本程序中，没有对 sum 赋值，它的值不确定。在 for 循环结束后，sum 的值也是不确定的。

**答案**　C

**【例 4】有如下程序**

```
main()
{   int i,sum;
    for(i=1;i<=3;sum++)  printf("%d\n",sum);
}
```

**该程序的执行结果是_____。（2000 年 9 月）**

（A）6　　（B）3　　（C）死循环　　（D）0

**题眼分析**　程序中，for 后面小括号中的第 3 个表达式是 sum 自加，并不是 *i* 自加，所以在执行循环时，*i* 的值不会得到改变，它的值永远是初值 1。表达式“*i*<=3”永远成立，此循环是死循环。

**答案**　C

**【例 5】要求以下程序的功能是计算：** $s=1+\frac{1}{2}+\frac{1}{3}+\Lambda+\frac{1}{10}$

```
main()
{   int n; float s;
    s=1.0;
    for(n=10;n>1;n—)
        s=s+1/n;
    printf("%6.4f\n",s);
}
```

**程序运行后输出结果错误，导致错误结果的程序行是_____。（2003 年 9 月）**

（A）s=1.0;　　（B）for(n=10;n>1;n—)

（C）s=s+1/n;　　（D）printf("%6.4f\n",s);

**题眼分析　选项（A）、（B）、（D）在语法上和逻辑上都没有错误。选项（C）中，变量 *n* 为整型变量，1 是整型常数，整数和整数相除得到的也是整数，因此每次执行后都累加一个 0。可以将该语句改为“ *s*=*s*+1.0/*n*; ”。**

**答案**　C

## 二、填空题分析

**【例 1】有以下程序**

```
main()
{   int  t=1,i=5;
    for(;i>=0;i--)t*=i;
    printf("%d\n",t);
}
```

**执行后输出的结果是_____。（2004 年 4 月）**

**题眼分析　循环变量 *i* 从 5 循环到 0，每次累乘变量 t。由于最后一次执行 *t**=*i* 时，*i*=0，所以 *t* 结果为 0。**

**答案**　0

**【例 2】以下主程序运行后的输出结果是_____。（2003 年 9 月）**

```
main()
{   int i,m=0,n=0,k=0;
    for(i=9;i<=11;i++)
        switch(i%10)
        {   case 0:m++;n++;break;
            case 10:n++;break;
            default:k++;n++;
         }
    printf("%d  %d  %d \n",m,n,k);
}
```

**题眼分析　这个 for 循环共执行 3 次。第 1 次（*i*=9）循环时，执行 default 后的语句，使 *k*=1，*n*=1；第 2 次（*i*=10）循环时，执行 case 0 后的语句，使 *m*=1，*n*=2，*k* 不变；第 3 次（*i*=11）循环时，执行 default 后的语句，使 *k*=2，*n*=3，*m* 不变。这样最后结果是 *m*=1 ，*n*=3，*k*=2。**

**答案**　1　3　2

**【例 3】以下程序的输出结果是_____。（2002 年 4 月）**

```
main()
{   int s,i;
    for(s=0,i=1;i<3;i++,s+=i); printf("%d\n",s);
}
```

**题眼分析　此程序中，for 循环语句后面直接跟了一个“ ; ”，循环体为空。输出语句不在 for 循环体内，所以在循环执行完毕后才输出 *s* 的值。*s*,*i* 的初值分别为 0 和 1，判断循环条件，**

表达式“ 1<3 ”成立。执行“ *i*++,*s*+=*i* ”后，*i* 和 *s* 的值分别为 2 和 2，继续判断循环条件，表达式“ 2<3 ”成立；执行“ *i*++,*s*+=*i* ”后，*i* 和 *s* 的值分别为 3 和 6，再次判断循环条件，表达式“ 3<3 ”不成立，结束循环。

**答案**　6

**【例 4】以下 sum 函数的功能是计算下列级数之和。**

$$s=1+x+\frac{x^2}{2!}+\frac{x^3}{3!}+\Lambda+\frac{x^n}{n!}$$

**请给函数中的各变量正确赋初值。（2004 年 4 月）**

```
double  sum(double  x，int  n )
{   int  i;   double  a,b,s;

    ________
    for(i=1;i<=n ;i++)
    {   a=a*x;  b=b*i;  s=s+a/b; }
        return  s;
}
```

**题眼分析**　在循环开始前，需要对循环中使用的几个变量赋初值。从程序中可以看出，变量 *a* 用来存放 $x^i$ 的值，变量 b 用来存放 *i*!，变量 *s* 用来存放级数的和。因此它们的初值都是 1.0。

**答案**　*a*=1.0;*b*=1.0;*s*=1.0;

# 5.4　循环结构的嵌套

## 5.4.1　考点提炼

**在一个循环体内又包含了另一个循环，称为循环嵌套。前面介绍的 3 种循环可以相互嵌套，循环嵌套可以多层，但每一层在逻辑上必须是完整的。**

## 5.4.2　题眼分析

### 一、选择题分析

**【例 1】有以下程序**

```
main()
{   int num[4][4]={{1,2,3,4},{5,6,7,8},{9,10,11,12},{13,14,15,16}},i,j;
    for(i=0;i<4;i++)
    {   for(j=0;j<=i;j++)   printf("%4c",’ ’);
        for(j=_____;j<4;j++)    printf("%4d",num[i][j]);
        printf("\n");
    }
```

```
}
```

若要按以下形式输出数组右上半三角

```
1   2   3   4
    6   7   8
        11  12
            16
```

则在程序下划线处应填入的是_____。（2005 年 4 月）

（A）*i*–1　　（B）*i*　　（C）*i*+1　　（D）4–*i*

题眼分析　程序中使用了两层 for 循环，外层循环的作用是控制输出行数（从 0 开始），循环一次输出一行。在外层循环体中有两个循环，第 1 个循环用于控制在输出数组元素前应输出的空格数，第 2 个循环用于控制在输出数组元素。第 0 行从第 0 列开始，第 1 行从第 1 列开始，第 2 行从第 2 列开始，第 3 行从第 3 列开始，这可以用外循环变量 *i* 表示，即 $i_0$。

答案　B

【例 2】下面程序的功能是输出以下形式的金字塔图案:

```
    *
   ***
  *****
 *******
```

```
main()
{   int  i,j;
    for(i=1;i<=4;i++;)
    {   for(j=l;j<=4–i;j++)printf("  ");
        for(j=1; ;j<=_____;j++)printf("*");
        printf("\n");
    }
}
```

在下划线处应填入的是_____。（2004 年 4 月）

（A）i　　（B）2*i–l　　（C）2*i+l　　（D）i+2

题眼分析　程序中使用了两层 for 循环，外层循环的作用是控制输出行数，循环一次是输出一行。在外层循环体中有两个循环，第 1 个循环用于控制在输出“*”前应输出的空格，第 2 个循环用于控制在输出“*”个数。第 1 行 1 个，……，第 4 行 7 个，这可以用外循环变量 *i* 表示，即 2**i*–1。

答案　B

【例 3】以下程序的功能是：按顺序读入 10 名学生 4 门课程的成绩，计算出每位学生的平均分并输出，程序如下：

```
main()
{   int n,k;
    float score ,sum,ave;
    sum=0.0;
    for(n=1;n<=10;n++)
    {   for(k=1;k<=4;k++) { scanf("%f",&score); sum+=score;}
```

```
            ave=sum/4.0;
            printf("NO%d:%f\n",n,ave);
        }
    }
```

**上述程序运行后结果不正确，调试中发现有一条语句出现在程序中的位置不正确。这条语句是_____。（2002 年 9 月）**

（A）sum=0.0;　　（B）sum+=score;

（C）ave=sum/4.0;　　（D）printf("NO%d:%f\n",n,ave);

**题眼分析**　程序中使用了两层 for 循环，外层循环的作用是控制人数，循环一次是求一个人的成绩和，然后除以 10 得到平均成绩。每个人的成绩总和一开始时都必须清 0，否则就会出现后面人的总成绩越来越大。“ sum=0.0; ”应在外层循环中。

**答案**　A

## 二、填空题分析

**【例】执行以下程序后，输出'#'号的个数是_____。（2003 年 9 月）**

```
#include<stdio.h>
main()
{   int i,j;
    for(i=1;i<5;i++)
        for(j=2;j<=i;j++)  putchar('#');
}
```

**题眼分析**　程序中使用了两层 for 循环，外层循环的循环体共运行 4 次。内循环的循环体运行次数由外层循环变量决定。当外层循环第 1 次运行时（$i$=1），内层循环等价于“ for(j=2;j<=1;j++)putchar('#'); ”，循环条件不满足，循环体运行次数 0，即没有输出“ # ”；当外层循环第 2 次运行时（$i$=2），内层循环等价于“ for(j=2;j<=2;j++)putchar('#'); ” 循环体运行次数 1，即输出 1 个。依此类推，当 $i$=3 时，输出两个“ # ”；当 $i$=4 时，输出 3 个“ # ”。因此一共输出 6 个“ # ”。

**答案**　6

# 5.5　break 和 continue 语句在循环体中的作用

## 5.5.1　考点提炼

### 考点 1：break 语句

break 语句可用于 switch 语句和 3 种循环语句中，这 3 种循环语句都是在执行循环体之前或之后通过对一个表达式的测试来决定是否终止对循环体的执行。另外，在循环体中，也可以通过使用 break 语句来立即终止循环的执行，而转入下一语句的执行。

break 的一般形式如下：

break；

**其执行过程是：终止对** switch **语句或循环语句的执行，即跳出这两种语句，而转入下一语句执行。**

**注意：**break **只能中止一层循环。**

**考点** 2：continue **语句**

continue **语句的一般形式为：**

continue;

**其作用是结束本次循环，即跳过本次循环体中余下尚未执行的语句，接着再一次进行循环的条件判断。**

**注意：执行** continue **语句并没有使整个循环终止。**

**在** while **和** do-while **循环中，**continue **语句使得流程直接跳到循环控制条件的测试部分，然后决定循环是否继续进行。在** for **循环中，遇到** continue **后，跳过循环体中余下的语句，而去对** for **语句中的“表达式** 3**”求值，然后进行“表达式** 2**”的条件测试。**

## 5.5.2 题眼分析

### 一、选择题分析

**【例** 1**】有以下程序**

```
main( )
{   int  i=0 , s=0 ;
    for ( ; ; )
    {   if ( i= =3 || i= =5 ) continue ;
        if ( i= =6 )  break ;
        i++;
        s+= i ;
    } ;
    printf("%d\n", s ) ;
}
```

**程序运行后的输出结果是_____。（**2004 **年** 9 **月）**

A）10　　B）13　　C）21　　D）**程序进入死循环**

**题眼分析**　**当** *i*=0 **时，不满足两个** if **语句的判断条件，则执行** if **语句之后的两条语句，执行自加语句后** *i*=1**。同样当** *i*=1 **和** *i*=2 **时，仍不满足两个** if **语句的判断条件。当** *i* **自加到** 3 **时，满足第** 1 **个** if **语句的条件，执行“**continue;**”，跳过本次循环余下的语句，进入下一次循环，***i* **仍然等于** 3**，则再一次进入下一次循环。由于没有循环的终止条件，所以程序进入死循环。**

**答案**　D

**【例** 2**】有以下程序**

```
main()
{   int  i,n=0;
```

```
        for(i=2;i<5;i++)
        {   do
            {   if(i%3)  continue;
                n++;
            }while(!i);
            n++;
        }
        printf("n=%d\n",n);
    }
```

**程序执行后输出结果是_____。（2004 年 4 月）**

（A）n=5　　（B）n=2　　（C）n=3　　（D）n=4

**题眼分析　外循环共运行 3 次，每次 *n* 自加 1。在 do-while 内循环中，由于*(!i)*条件始终不成立，因此该循环始终只运行一次，当 *i*%3=0 时(即 *i*=3 时)，*n* 自加 1，其他时候什么也没做。这样 *n* 自加了 4（1+3）次。**

**答案**　D

**【例 3】有如下程序**

```
main()
{   int k=4, n=0;
    for(;  n<k;  )
    {   n++;
        if(n%3!=0) continue;
        k--;}
    printf("%d, %d\n",k,n);
    }
```

**程序运行后的输出结果是_____。（2003 年 9 月）**

（A）1,1　　（B）2,2　　（C）3,3　　（D）4,4

**题眼分析　第 1 次循环时，*n* 自加 1 后，即 *n*=1，由于 *n*%3=1，跳过“ *k*-- ;”; 第 2 次循环时，*n* 自加 1 后，即 *n*=2，由于 *n*%3=2，跳过“ *k*-- ;”; 第 3 次循环时，*n* 自加 1 后，即 *n*=3，由于 *n*%3=0，将执行“ *k*-- ;”，*k*=3；第 4 次循环时，不满足 *n*<*k*，退出循环。最终，*k*=3，*n*=3。**

**答案**　C

**【例 4】有以下程序**

```
main()
{   int i=0,s=0;
    do{
        if(i%2) {   i++;    continue;}
        i++;
        s+=i;
    } while(i<7);
    printf("%d\n",s);
}
```

**执行后输出结果是_____。（2003 年 4 月）**

（A）16　　（B）12　　（C）28　　（D）21

**题眼分析　在循环体中有一条 if 语句，其后的表达式为“ i%2 ”，当 *i* 的值为奇数时，值为**

“真”，执行其后面的语句，*i* 的值加 1，重新开始循环；当 *i* 的值为偶数时，“i%2”为“假”，执行“*i*++;*s*+=*i*;”。在循环中 *i* 为偶数时的值分别为 0、2、4、6，加 1 过后的值分别为 1、3、5、7，s 中存放的是它们的和，值为 16。

**答案** A

**【例 5】有以下程序**

```
main()
{   int a=1,b;
    for(b=1;b<=10;b++)
    {   if(a>=8)break;
        if(a%2==1){a+=5;continue;}
        a-=3;
    }
    printf("%d\n",b);
}
```

程序运行后的输出结果是_____。（2005 年 4 月）

（A）3　　（B） 4　　（C）5　　（D）6

**题眼分析** 这一题和【例 1】是同一类型的题目，程序的执行顺序是根据变量 a 值变化，以决定是否结束一次循环还是中止循环。程序执行过程这里就不再赘述了。

**答案** B

## 二、填空题分析

**【例】以下程序运行后的输出结果是_____。（2002 年 9 月）**

```
main()
{   int x=15;
    while(x>10&&x<50)
    {   x++;
        if(x/3) {   x++;    break;  }
        else continue;
    }
    printf("%d\n",x);
}
```

**题眼分析** 首先定义一个变量 *x* 并赋初值 15，然后判断循环条件，为真，执行循环体。语句“*x*++;”执行后，*x* 的值变为 16，“x/3”的值为 5(真)，执行其后的语句“*x*++;”；*x* 的值变为 17，执行语句“break;”，循环退出，输出 *x* 的值为 17。

**答案** 17

# 5.6　单元训练及参考答案

## 5.6.1　单元训练

### 一、选择题

1．C 语言中 while 和 do-while 循环的主要区别是_____。

（A）do-while 的循环体至少无条件执行一次

（B）do-while 允许从外部转到循环体内

（C）while 的循环控制条件比 do-while 的循环控制条件严格

（D）do-while 的循环体不能是复合语句

2．语句 while (!x);中的条件*!x* 等价于_____。

（A）*x==0*　　（B）*x!=1*　　（C）*x!=0*　　（D）*~x*

3．以下 for 循环是_____。

```
for (a=0,b=0;(b!=123) && (a<=4);a++);
```

（A）无限循环　　（B）循环次数不定　　（C）执行 4 次　　（D）执行 5 次

4．以下关于 for 循环的说法不正确的是_____。

（A）for 循环只能用于循环次数已经确定的情况

（B）for 循环是先判定表达式，后执行循环体语句

（C）for 循环中，可以用 break 语句跳出循环体

（D）for 循环体语句中，可以包含多条语句，但要用花括号括起来

5．以下程序的输出结果是_____。

```
#include <stdio.h>
main()
{   int i=0,a=0;
    while(i<20)
    {   for(; ;) {  if((i%10)= =0) break; else  i--; }
        i+=11; a+=i;
    }
    printf("%d\n",a);
}
```

（A）21　　（B）32　　（C）33　　（D）11

6．下列程序的输出结果是_____。

```
#include<stdio.h>
void  main()
{   int y=10;
    while(y--);
    printf("y=%d",y);
}
```

（A）*y*=0　　（B）*y*=1　　（C）*y*=随机值　　（D）*y*= –1

7．以下程序的输出结果是_____。

```
main()
{   int i;
    for (i=1;i<=5;i++)
    {   if (i%2)        printf("#");
        else        continue;
        printf("*");
    }
    printf("$\n");
}
```

（A）*#*#*#$　　（B）#*#*#*$　　（C）*#*#$　　（D）#*#*$

8．以下程序的输出结果是_____。

```
main()
{   int  m=1;
    while (m<=3) {  m++; printf("%d",m);    }
}
```

（A）2　　（B）23　　（C）234　　（D）2345

9．以下程序段的输出结果是_____。

```
p=5;
do {    p=p*p;  }   while(!p);
```

（A）是死循环　　（B）循环执行两次　　（C）循环执行 1 次　　（D）循环执行 5 次

10．若有如下语句

```
int f=1;
while (!(—f)){     printf("%d\n",f–=2);    }
```

则上面程序段_____。

（A）输出的是–2　　（B）输出的是 1 和–2　（C）输出的是 3 和 0　（D）是死循环

11．设有以下程序段

```
int x=0,s=0;
while(!x!=0) {  s+=++x; ++x;    }
printf("%d, %d",s,x)
```

则_____。

（A）运行程序段后输出为 0　　（B）运行程序段后输出为 1,2

（C）程序段中的控制表达式是非法的　　（D）循环体语句执行一次

12．设 $x$ 和 $y$ 均为整型变量，则执行下面的循环后，$y$ 的值为_____。

（A）2　　（B）4　　（C）6　　（D）8

```
for(y=1,x=1;y<=50;y++)
{   if(x>=10)  break;
    if(x%2==1) {    x+=5;   continue;   }
    x–=3;
}
```

13．以下程序片段执行后的输出结果是_____。

```
int  b;
for (b=1;  ; b++);printf ("%d", b++);
```

（A）2　　（B）1　　（C）4　　（D）死循环

14．*t* 为 int 类型，进入下面的循环之前，*t* 的值为 0。

```
while ( t=l ) { … …}
```

则以下叙述中正确的是_____。

（A）循环控制表达式的值为 0　　（B）循环控制表达式的值为 1

（C）循环控制表达式不合法　　（D）以上说法都不对

15．有以下程序

```
main()
{   char   k;   int  i;
    for(i=l;i<3;i++)
    {   scanf("%c",&k);
        switch(k)
        {   case ’0’:  printf("another\n");
            case ’l’:  printf("number\n");
        }
    }
}
```

程序运行时，从键盘输入：01< 回车>，程序执行后的输出结果是_____。

（A）another　　（B）another　　（C）another　　（D）number

number　　number　　number　　number

another　　number

## 二、填空题

1．以下程序的输出结果是___（1）___。

```
main()
{   int s=0,k;
    for (k=5;k>=0;k—)
    {   switch(k)
        {   case 1:
            case 5:  s++; break;
            case 3:
            case 4:  break;
            case 0:
            case 2:  s+=2; break;
        }
    }
    printf("s=%d\n",s);
}
```

2．以下程序是利用公式 $\pi = 4*(\frac{1}{1}-\frac{1}{3}+\frac{1}{5}-\frac{1}{7}+\Lambda)$ 来计算 $\pi$ 的值。其中，变量 *k* 表示当前符号项，*t* 表示当前项，*n* 表示当前项的序号，要求精度控制在 0.00001 内。试分析程序，完成空标号处应填的内容。

```
#include<math.h>
```

```
main()
{   float pi,t,n,k;
    pi=0.0; n=k=t=1.0;
    while(  (2)  )
    {   pi+=t;  k=-k; t=  (3)  ; n++;  }
    pi=  (4)  ;
    printf("pi=%f\n",pi);
}
```

3．以下程序的输出结果是 (5) 。

```
main()
{   int i,j,sum=0;
    for (j=1;j<=5;j++)
    {   m=1;
        for(i=1;i<=j;i++) m=m*i;
        sum=sum+m;      }
    printf("sum=%d\n",sum);
}
```

4．下面程序的输出结果是 (6) 。

```
#include<stdio.h>
main()
{   int  x,y;
    for(x=1,y=1; x<100;x++)
    {   if (y>=20)  break;
        if(y%3==1) {    y=y+3;  continue;   }
        y=y-5;
    }
    printf("x=%d, y=%d",x,y);
}
```

## 5.6.2 参考答案

### 一、选择题

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

### 二、填空题

（1）$s$=6
（2）fabs(t)>0.00001
（3）$s$/(2*$n$+1)
（4）4*pi
（5）153
（6）$x$=8,$y$=22

# 第 6 章　字符型数据

## 6.1　字符型常量

### 6.1.1　考点提炼

**📖考点 1：字符型常量**

**用一对单引号括起来的一个字符称为字符，一对单引号为字符的定界符。**

**字符型常量在内存中占用一个字节，存放字符的 ASCII 代码值。所有字符型常量都作为整形数字处理。**

**📖考点 2：转义字符**

**C 语言中还有一类以反斜杠开头的字符序列，称为转义字符。可分为以下几种情况：**

**（1）用反斜杠加一个字母代表一个控制字符。如：'\n'表示换行控制符。**

**（2）用反斜杠加 1～3 位八进制数字代表 ASCII 码为该八进制数的字符。如：'\101'是字符'A'的另外一种表示。**

**（3）用反斜杠加 x 和 1～2 位十六进制数字来代表 ASCII 码为该十六进制数的字符，如：'\x41' 是字符'A'的另外一种表示。**

**📖考点 3：字符串常量**

**字符串常量是由双引号括起来的一串字符。在 C 语言中，系统在每个字符串的最后自动加入一个字符'\0'作为字符串结束标志。要注意字符常量和字符串常量的区别，例如：'\101'、'Z' 是字符常量，在内存中占用一个字节，而"ABC\n"、"Z"是字符串常量，前者占 5 个字节，后者占两个字节。**

**📖考点 4：字符串常量**

**字符量可参与任何整数的运算。例如：**

'B'–'A'=66–65=1　　'a'+1=97+1=98='b'　　'9'–'0'=57–48=9

'A'+32=65+32=97='a'　　'b'–32=98–32=66='B'　　'0'+8=8+48=56='8'

**字符量也可进行关系运算。例如：**

'a'<'b'**为真**　　'0'>'7'**为假**　　'A'<'a'**为真**

'a'&&'b'**为真**　　'0'||0 **为真**　　'0'&&0 **为假**

## 6.1.2 题眼分析

### 一、选择题分析

【例 1】以下选项中合法的字符常量是_____。（2001 年 9 月）

（A）"B"　　（B）'\10'　　（C）68　　（D）D

题眼分析　选项（A）是字符串的表示方法，选项（B）是一个转义字符，选项（C）是整型常量，选项（D）不是常量的表示。

答案　B

【例 2】已知大写字母 A 的 ASCII 码是：65，小写的 *a* 的 ASCII 码是 97，则用八进制表示的字符常量'\101'是_____。（2002 年 9 月）

（A）字符 A　　（B）字符 a　　（C）字符 e　　（D）非法常量

题眼分析　在题目中的关键是八进制 101 的十进制是多少。八进制 101 转换为十进制为 65，也就是 A 的 ASCII 码。

答案　A

【例 3】若变量 *a* 是 int 类型，并执行了语句：*a*='A'+1.6；，则正确的叙述是_____。（2002 年 4 月）

（A）a 的值是字符 C　　（B）a 的值是浮点型

（C）不允许字符型和浮点型相加　　（D）a 的值是字符'A'的 ASCII 值加上 1

题眼分析　在 C 语言中规定允许不同类型的量进行运算，但在运算时需转换成高级的类型进行运算。在表达式中有字符型和实型参加运算，统一转换为实型在运算，当运算结果存入变量时再转换为该变量的类型。

答案　D

### 二、填空题分析

【例 1】若有以下程序

```
main()
{   char  a;
    a='H'-'A'+'0';
    printf("%c\n",a);
}
```

执行后输出结果是_____ 。（2003 年 4 月）

题眼分析　字符型数据可作为整型参加算术运算，其值为其对应的 ASCII 码。'H'–'A'的结果是 7，加'0'后是'7'的 ASCII 码，所以输出的字符是'7'。

答案　7

【例 2】以下程序运行后的输出结果是_____。（2003 年 9 月）

```
main()
{   char  m;
    m='B'+32;   printf("%c\n",m);
}
```

**题眼分析**　在 C 语言中，字符常量存储是字符 ASCII 码值，'B'的 ASCII 的值为 66，则变量 *m* 存放的值为 98，这个值是'b'的 ASCII 码值。该题要求考生记住大写字母和其对应的小写字母的 ASCII 码的关系，它们之间差为 32。例如：'a'-'A'=32。

**答案**　b

# 6.2　字符变量

## 6.2.1　考点提炼

字符型变量用关键字 char 进行定义，在定义时可以赋初值。字符变量在内存中占 1 个字节，存放字符的 ASCII 码值。字符变量可以作为整型变量来处理，可以参与对整型变量所允许的任何运算。

## 6.2.2　题眼分析

### 一、选择题分析

**【例 1】已经定义 ch 为字符型变量，以下赋值语句中错误的是_____。（2003 年 9 月）**

（A）ch='\';　　（B）ch=62+3;　　（C）ch=NULL;　　（D）ch='\xaa';

**题眼分析**　对一个字符串变量 ch 赋字符"\"，需要使用转义字符'\\'，因此选项（A）是错误的；选项（B）是给字符变量 ch 赋一个 ASCII 码为 65 的字符；选项（C）是给字符变量 *ch* 赋空字符（不是空格字符）；选项（D）是给字符变量 ch 赋一个 ASCII 码十六进制为 aa 的字符。

**答案**　A

**【例 2】已定义 *c* 为字符型变量，则下列语句中正确的是_____。（2003 年 9 月）**

（A）c='97';　　（B）c="97";　　（C）c=97;　　（D）c="a";

**题眼分析**　字符常量是用单引号括起来的一个字符或一个转义字符，因此选项（A）不正确。选项（B）、（D）是用双引号括起来，都是字符串。字符变量可以参加任何整数的运算，因此选项（C）是正确的。

**答案**　C

**【例 3】数字字符 0 的 ASCII 值为 48，若有以下程序**

```
main()
{   char a='1',b='2';
    printf("%c,",b++);
    printf("%d\n",b-a);
}
```

**程序运行后的输出结果是_____。（2005 年 4 月）**

（A）3,2　　（B）50,2　　（C）2,2　　（D）2,50

**题眼分析**　在 C 语言中，字符变量中存储的是字符的 ASCII 码值，字符变量可以作为整

型变量来处理，因此，变量 *a* 和 *b* 实际上是存储字符'1'和'2'的 ASCII 码值 49、50。第 1 个 printf() 按字符方式输出 *b* 后（即'2'），*b* 自动加 1，这样 *b* 的值为 51。第 2 个 printf()按整数方式输出 *b*–*a* 值，其值为 2。因此最后输出的是 2,2，但前一个是字符 2，后一个是整数 2。

**答案** C

**【例 4】有以下程序**

```
main( )
{   char   a='a' , b ;
    printf("%c,",++a) ;
    printf("%c\n",b=a++) ;
}
```

**程序运行后的输出结果是_____。（2004 年 9 月）**

A）b,b　　B）a,b　　C）b,c　　D）a,c

**题眼分析** 参见【例 3】分析。

**答案** A

**【例 5】有以下程序**

```
main()
{   char a,b,c,*d;
    a='\';   b='\xbc';
    c='\0xab';    d="\0127";
    printf("%c%c%c%c\n",a,b,c,*d);
}
```

**编译时出现错误，以下叙述中正确的是_____。（2003 年 4 月）**

**（A）程序中只有 a='\';语句不正确**　　**（B）b='\xbc';语句不正确**

**（C）d="\0127";语句不正确**　　**（D）a='\';和 c='\0xab';语句都不正确**

**题眼分析** 给字符变量赋值只能赋一个字符，包括转义字符，语句“a='\';”是错误的，“\”是转义字符，应用“\\”两表示。语句“B='\xbc';”是正确的，它给一个字符型变量赋一个用十六进制表示的转义字符；语句“c='\0xab';”是错误的，这是因为反斜线后的十六进制数只可由小写字母 x 开头，不允许用大写字母 X，也不能用 0x 开头，\0xab 表示的是 4 个字符，而不是一个字符；语句“d="\0127";”是正确的，可以给字符型指针变量赋一个字符串，其作用是让该指针变量指向该字符串。

**答案** D

## 二、填空题分析

**【例 1】已知字符 A 的 ASCII 码值为 65 以下语句的输出结果是_____。（2004 年 4 月）**

```
char   ch='B';
printf("%c%d\n"ch,ch);
```

**题眼分析** 字符 A 的 ASCII 码为 65，则字符 B 的 ASCII 码为 66。

**答案** B 66

**【例 2】以下程序运行后的输出结果是_____。（2005 年 4 月）**

```
main()
{   char c1,c2;
```

```
        for(c1='0',c2='9';c1<c2;c1++,c2--) printf("%c%c",c1,c2);
        printf("\n");
    }
```

**题眼分析**　本题中通过两个字符变量来控制 for 循环，循环共执行了 5 次，第 1 次输出 09，第 2 次输出 18，第 3 次输出 27，第 4 次输出 36，第 5 次输出 45。

**答案**　0918273645

# 6.3　字符的输入和输出

## 6.3.1　考点提炼

**考点 1：调用 printf 和 scanf 函数输出和输入字符**

用 printf 函数输出字符时只须使用格式说明%c。

用 scanf 函数输入字符时也要使用格式说明%c。在用 scanf 函数输入字符时需要注意以下几点：

（1）当使用格式说明%c 一个紧接一个时，在输入字符时，字符之间没有间隔符，这时空格、回车和横向跳格符都将按字符读入。如：

```
char a,b,c;
scanf("%c%c%c", &a,&b,&c);
```

若从第 1 列开始输入：TH<回车>，则变量 *a* 将存放字符 T，变量 *b* 将存放字符 H，而变量 *c* 将存放回车符。

（2）如果在格式控制串加入空格，如 scanf("%c　%c　%c", &a,&b,&c); ，输入形式可以和不加空格的 scanf("%c%c%c", &a,&b,&c);一样，但这时空格、回车和横向跳格符都作为间隔符而不被读入。

（3）如果在格式字符前加一个整数，用来指定输入数据所占宽度。这时在输入字符数据时，应严格按指定的宽度输入数据，且取指定宽度的第 1 个字符作为输入数据。

（4）当交叉输入数值数据和字符数据时，即在输入表中交替出现字符变量和数值变量，输入时要多加注意，否则将产生输入错误。例如，

```
int a1,a2;      char c1,c2;
scanf("%d%c%d%c",&a1,&c1,&a2,&c2);
```

必须用以下形式输入数据：

```
10A<空格>20B<回车>
```

如果用以下形式输入数据：

```
10<空格>A<空格>20<空格>B<回车>
```

则将 10 赋给 *a1*，将空格赋给 *c1*，接着 *a2* 将遇到字母 A，因为类型不匹配，尽管程序照常运行而并不报错，但 scanf 函数将结束运行，使得 *a2* 和 *c2* 没有从终端接受数据。

**考点 2：调用 putchar 和 getchar 函数输出和输入字符**

**使用 putchar 和 getchar 函数时，必须在程序的开头出现包含头文件 stdio.h 的命令行：**

```
#include stdio.h
```

putchar **函数的作用是向终端输出一个字符，其调用形式如下：**

```
putchar(ch);
```

**其中 *ch* 可以是字符变量或是字符常量。**

getchar **函数的作用是向终端输入一个字符，其调用形式如下：**

```
ch=putchar();
```

getchar **函数没有参数，但这对圆括号不能少。它把输入的字符赋值给变量 *ch*。**

## 6.3.2 题眼分析

### 一、选择题分析

**【例 1】执行下面语句后，输出的结果是_____。**

```
char a=97;b=98; printf("%d%c",a,b);
```

（A）97 98　　（B）97 b　　（C）a 98　　（D）a b

**题眼分析　整型数据和字符型数据可以转换，97 和 98 分别是'a'和'b'的 ASCII 码值。在输出语句中，%d 和%c 是输出格式，要求在相应的位置上将变量 *a* 以整数的形式输出，变量 *b* 以字符型输出。**

**答案**　B

**【例 2】有以下程序**

```
main( )
{   char  a , b , c , d ;
    scanf("%c,%c,%d,%d", &a , &b , &c , &d ) ;
    printf("%c,%c,%c,%c\n", a , b , c , d ) ;
}
```

**若运行时从键盘上输入：**6,5,65,66<**回车**>**。则输出结果是_____。（2004 年 9 月）**

（A）6,5,A,B　　（B）6,5,65,66　　（C）6,5,6,5　　（D）6,5,6,6

**题眼分析**　scanf()**的格式控制串为**"%c,%c,%d,%d"**，先读入两个字符，再等两个整数。这样 *a*='6'，*b*='5'，*c*=65，*d*=66。又知** 65、66 **分别是字母** A **和** B **的** ASCII **码，实际上 *c*='A'，*d*='B'。输出这** 4 **个字符变量都是按字符形式输出的，因此选项（**A**）是正确答案。**

**答案**　A

### 二、填空题分析

**【例 1】下列程序运行的结果是_____。**

```
main ()
{   char a='a',b='b',c='c';
    printf("a%cb%c\tc%c\n",a,b,c);
}
```

**题眼分析　输出语句依次输出为：非格式字符'a'，变量 *a*，非格式字符'b'，变量 *b*、**Tab（\t

为 Tab 的转义字符)、变量 *c*、回车。

**答案**　aabb　　c

**【例 2】输入“ 12345,xyz ”，下列程序输出的结果是_____。**

```
main()
{   int x;   char y ;
    scanf("%3d%3c",&x,&y);
    printf("%d,%c",x,y); }
```

**题眼分析**　在输入数据时，如果指定了数据所占的列数后，系统会自动按要求截取数据，不需要分隔符。程序中变量 *a* 占 3 列，在输入“ 12345，xyz ”后系统自动将数据 123 赋值给变量 *x*，剩余的数据仍放在缓冲区中，对于变量 y 它是字符型的数据要求是占 3 列，但是字符型变量只能有一个字符。剩余数据中的第 1 个，也就是 4 被赋值给了 *y*。

**答案**　123,4

**【例 3】有以下程序**

```
#include  <stdio.h>
main()
{   char  ch1,ch2;    int  n1,n2
    ch1=getchar();    ch2= getchar();
    n1=ch1-'0';      n2=n1*10+(ch2-'0');
    printf("%d\n ",n2);
}
```

**程序运行时输入:** 12**<回车>，执行后输出结果是_____。（2004 年 4 月）**

**题眼分析**　输入 12 后，字符变量 ch1='1'，ch1='2'。这样 n1='1'-'0'=1，n2=10+'2'-'0'=12。

**答案**　12

**【例 4】已知字符 A 的 ASCII 代码值为 65，以下程序运行时若从键盘输入：**B33<**回车**>，**则输出结果是_____。（2005 年 4 月）**

```
#include
main()
{   char a,b;
    a=getchar();scanf("%d",&b);
    a=a-'A'+'0';    b=b*2;
    printf("%c %c\n",a,b);
}
```

**题眼分析**　在 C 语言中，字符变量中存储的是字符的 ASCII 码值，字符变量可以作为整型变量来处理，因此可以使用整数格式%d 进行读入。输入 B33 后，字符变量 *a* 中存放字符'B'的 ASCII 码，即 66，字符变量 *b* 中存放着 33。当执行 *a*=*a*-'A'+'0';后，字符变量 *a* 中存放字符'1'的 ASCII 码值，当执行 *b*=*b**2；后，字符变量 *b* 的值为 66，这是字符'B'的 ASCII 码。最后以字符方式输出这两个变量，因此得到 1　B。

**答案**　1　B

# 6.4 单元训练及参考答案

## 6.4.1 单元训练

### 一、选择题

1．_____是非法的 C 语言转义字符

（A）'\t'　　（B）'\018'　　（C）'\n'　　（D）'\xaa'

2．在执行完下面的 C 语句段之后，则 B 的值是_____。

```
int x=241,y=15;
char Z='A';int B;
B=((241&15)&&(Z<'a'));
```

（A）0　　（B）1　　（C）TRUE　　（D）FALSE

3．putchar()函数可以向终端输出一个_____。

（A）整型变量表达式　　（B）实型变量值

（C）字符串　　（D）字符或字符型变量值

4．getchar()函数可以接受一个_____。

（A）整型变量表达式　（B）实型变量值　　（C）字符串　　（D）字符

5．有如下程序段：

```
int a1,a2;  char b1,b2;
scanf("%d%c%d%c",&a1,&b1,&a2,&b2);
```

则正确的输入是_____。

（A）10A　20B<CR>　　（B）10　A　20　B<CR>

（C）10　A20B<CR>　　（D）10A20　B<CR>

6．运行以下程序后,如果从键盘上输入 china#<回车>,则输出结果为_____。

（A）2,0　　（B）5,0　　（C）5,5　　（D）2,5

```
#include
main( )
{   int v1=0,v2=0;
    char ch ;
    while ((ch=getchar())!='#')
        switch (ch )
        {   case 'a':
            case 'h':
            default: v1++;
            case '0':v2++;
        }
    printf("%d,%d\n",v1,v2);
}
```

7．以下程序的输出结果是_____。

```
main()
```

```
{   char c='z';
    printf("%c",c-25);
}
```

（A）a　　（B）Z　　（C）z–25　　（D）y

**二、填空题**

1．**以下的程序执行时，先输入** a↙**；后输入** b↙**。最后显示的结果是**（1）**。**

```
#include <stdio.h>
main()
{   int x,y;
    printf("Enter a character:");
    x=getchar();    y=getchar();
    printf("Enter a character,again :");
    x=getchar();    y=getchar();
    printf("%c,%c\n",x,y);
}
```

2．**有以下程序：**

```
#include <stdio.h>
main()
{   char c;
    while((c=getchar())!='?')  putchar(--c);
}
```

**程序运行时，如果从键盘输入：**Y? N?<**回车**>**，则输出结果为**（2）**。**

3．**若输入字符串：**abcde<**回车**>**，则以下** while **循环体将执行**（3）**次。**

```
while((ch=getchar())=='e')printf("*");
```

## 6.4.2　参考答案

**一、选择题**

1．B　　2．B　　3．D　　4．D　　5．A

6．C　　7．A

**二、填空题**

（1）b,　　（2）X

（3）0

# 第 7 章　函数

## 7.1　库函数

### 7.1.1　考点提炼

**📖考点 1：库函数的含义**

**从使用的角度来分，C 语言中的函数可以分为用户函数和库函数（系统函数）。为方便用户编制程序，C 语言把一些常用的功能（如一些数学函数，如求绝对值、求平方根、求正余弦等）预先编制成函数，放在函数库中。用户要使用该功能时，只需直接调用相应的库函数就可以了。需注意的是调用库函数时，需在程序的开头用“ #include ”命令把包含库函数说明的相应的头文件（后缀名为“ .h ”）包含进来。例如，在程序中需要调用数学库函数，则必须在调用数学库函数之前包含以下 include 命令：**

```
#include  "math.h"
```

**📖考点 2：库函数的调用形式**

**库函数的调用形式如下：**

**格式：函数名(实际参数列表)**

**库函数的调用需要注意的是：函数的功能，函数的参数个数、类型，函数的返回值，对参数的一些特殊要求。**

**如调用平方根函数，其调用格式为 sqrt(x)。其功能是求 *x* 的平方根；其参数个数为 1，类型为实型；其返回值就是 *x* 的算术平方根；要求参数 *x* 的值必须大于等于 0。**

### 7.1.2　题眼分析

**【例】下列说明不正确的是_____。**

**（A）使用 putchar 和 getchar 函数时，必须在程序的开头出现包含头文件 stdio.h 的命令行**

**（B）使用数学库函数时，必须在程序的开头出现包含头文件 math.h 的命令行**

**（C）使用 printf 和 scanf 函数时，必须在程序的开头出现包含头文件 stdio.h 的命令行**

**（D）以上说法都正确**

**题眼分析**　在 C 语言中，尽管 printf 和 scanf 函数是库函数，但不需要指定头文件，而 putchar、getchar 函数和数学库函数在使用时，必须在程序的开头出现包含头文件的命令行。

**答案**　C

# 7.2　函数的定义和返回值

## 7.2.1　考点提炼

### 考点 1：有参函数的定义方法

有参函数的定义格式如下：

```
存储类型说明符　数据类型说明符　函数名([形式参数说明列表])　/*函数头*/
{　说明部分;
　　执行部分;
}
```

说明：

（1）存储类型说明符可以是 static 或 extern，说明该函数是内部函数还是外部函数。若为 static 则是内部函数，内部函数只能被本编译单位中的其他函数调用；若为 extern 则是外部函数，可被其他编译单位中的函数调用。缺省时为“ extern ”。

（2）数据类型说明符用来说明该函数返回值的类型。缺省时为“ int ”。

（3）函数头也可写成下列两行：

```
存储类型说明符　数据类型说明符　函数名(形式参数列表)
形式参数说明;
{　说明部分;
　　执行部分;
}
```

（4）要想让函数返回一个确定的值，必须通过语句“ return(表达式) ”来实现，其中表达式就是函数的返回值。如果没有 return 语句或 return 语句不带表达式并不表示没有返回值，而是表示返回一个不确定的值。如果不希望有返回值，必须在定义函数时把“ 数据类型说明符 ”说明为“ void ”。

### 考点 2：无参函数的定义方法

无参函数的定义形式如下：

```
存储类型说明符　数据类型说明符　函数名()
{　说明部分;
　　执行部分;
}
```

说明：

无参函数与有参函数基本一样，不同的只是它没有形式参数，调用时不需实参。

**考点 3：函数的类型和返回值**

**所谓函数的类型指的是函数的返回值类型。函数的类型由定义时的类型说明符确定，而不是由“ return(表达式) ”中的“ 表达式 ”类型确定。如果“ 表达式 ”类型与函数定义时的类型不一致，应把“ 表达式 ”转换为相应的类型，再作为函数值返回。**

## 7.2.2 题眼分析

### 一、选择题分析

**【例 1】下列函数定义中，会出现编译错误的是_____。（2003 年 9 月）**

（A）max(int x,int y,int *z)
{ *z=x>y?x:y; }

（B）int max(int x,y)
{ int z;
z=x>y?x:y;
return z;
}

（C）max(int x, int y)
{ int z;
z=x>y?x:y; return(z);
}

（D）int max(int x, int y)
{ return(x>y?x:y); }

**题眼分析** 函数定义时，必须为每个形式参数指定数据类型，因此选项（B）是错误的。选项（A）、（C）和（D）的定义都是正确的。

**答案** B

**【例 2】以下所列的各函数首部中，正确的是_____。（2001 年 4 月）。**

（A）void play(var :Integer,var b:Integer)　（B）void play(int a,b)

（C）void play(int a,int b)　（D）Sub play(a as integer,b as integer)

**题眼分析** C 语言规定如果在定义时进行形式参数的说明，必须对每一种形参都要进行类型说明。选项（A）的参数定义中用到了 var 和 integer，它们均不是变量的类型，也不是 C 语言的保留字，是错误的；选项（B）的参数定义并没有说明每一个参数的类型；选项（C）符合 C 语言的函数定义形式，“ void ”说明函数无返回值，play 是函数名，有两个形参并都进行了类型说明；选项（D）中的 sub 不是 C 语言的关键字更不是数据类型，参数定义也不对。

**答案** C

**【例 3】C 语言中，函数值类型的定义可以缺省，此时函数值的隐含类型是_____。（2002 年 9 月）**

（A）void　（B）int　（C）float　（D）double

**题眼分析** C 语言规定在定义函数时，若其返回值类型为 int，则可以缺省。

**答案** B

**【例 4】有以下程序**

```
#define P 3
void F(int x){  return(P*x*x);  }
```

```
main()
{   printf("%d\n",F(3+5)); }
```

**程序运行后的输出结果是_____。（2005 年 4 月）**

**（A）** 192　　**（B）** 29　　**（C）** 25　　**（D）编译出错**

**题眼分析**　**由于函数** F(int x)**被定义为无返回值函数，因此在调用时不允许使用该函数的返回值（会使编译出错）。**

**答案**　D

## 二、填空题分析

**【例】以下程序是计算** $s=\frac{1}{1!}+\frac{1}{2!}+\frac{1}{3!}+\Lambda+\frac{1}{n!}$**，请填空_____。（2002 年 9 月）**

```
double  fun(int n)
{   double s=0.0,fac=1.0;int i;
    for(i=1;i<=n;i++)
    {   fac=fac_____;     s=s+fac;        }
    return(s);
}
```

**题眼分析**　**函数中** *s* **和** fac **的作用是存放和要加到的那一项的值。通过分析可知，第** *i* **项的值可以由第** *i*–1 **项的值得到，即第** *i* **项是第** *i*–1 **项的值除** *i*。**可见横线处应填“** /*i* **”或与它等价的表达式。**

**答案**　/*i*　**或** *1.0/*i*　**或**　*1/*i*　**或** *(1.0/*i*)　**或** /(double)*i*

# 7.3　函数的调用

## 7.3.1　考点提炼

### 考点 1：形式参数与实际参数

**定义函数时所使用的参数称“形式参数”。调用函数时所使用的参数称“实际参数”。**

### 考点 2：函数的调用

**1．有返回值的函数调用形式**

**有返回值的函数调用，可以作为表达式或表达式的一部分，也可以作为一条语句。其调用形式如下：**

**函数名(实际参数列表);**

**调用的结果是获得一个返回值，该返回值可以参加相应类型的计算。**

**2．无返回值的函数调用形式**

**无返回值的函数调用只能作为一条函数调用语句来使用，其调用形式如下：**

**函数名(实际参数列表);**

## 7.3.2 题眼分析

### 一、选择题分析

**【例 1】若已经定义的函数有返回值，则以下关于该函数调用的叙述中错误的是_____。（2003 年 9 月）**

（A）函数调用可以作为独立的语句存在　　（B）函数调用可以作为一个函数的实参

（C）函数调用可以出现在表达式中　　（D）函数调用可以作为一个函数的形参

**题眼分析　对于有返回值的函数，可以作为表达式或表达式的一部分，也可以作为一条语句，因此选项（A）、（B）和（C）都是正确的。函数的形参是指函数定义时的参数，必须指定参数名称和类型，而函数调用的结果是一个值，因此选项（D）是错误的。**

**答案**　D

**【例 2】有以下函数定义：**

```
void fun(int n, double x) {……}
```

**若以下选项中的变量都已经正确定义并赋值，则对函数 fun 的正确调用语句是_____。（2003 年 9 月）**

（A）fun(int y, double m);　　（B）k=fun(10,12.5);

（C）fun(x,n);　　（D）void fun(n,x);

**题眼分析　从函数 fun 定义可以看出，这是一个无返回值的函数。在函数调用的时候不需指定实际参数的类型，因此选项（A）是错误的，它实际上是一个函数定义的形式；选项（B）中，实际参数列表是正确的，但 fun 无返回值，因此不能赋值给变量 *k*，因此也是错误的；在函数调用时，不需要指定函数的返回类型，因此选项（D）也是错误的。**

**答案**　C

**【例 3】有以下程序**

```
int f1(int x,int y)
{   return x>y?x:y; }
int f2(int x,int y)
{   return x>y?y:x; }
main()
{   int a=4,b=3,c=5,d,e,f;
    d=f1(a,b);      d=f1(d,c);
    e=f2(a,b);      e=f2(e,c);
    f=a+b+c-d-e;    printf("%d,%d,%d\n",d,f,e);
}
```

**执行后输出结果是_____。（2003 年 4 月）**

（A）3,4,5　　（B）5,3,4　　（C）5,4,3　　（D）3,5,4

**题眼分析　函数 *f*1 的作用是返回形参 *x* 和 *y* 的较大值，函数 *f*2 的作用是返回形参 *x* 和 *y* 的较小值，在 main 函数中通过调用两次 *f*1 函数求得 *a*、*b*、*c* 的最大值并存放到变量 *d* 中，通过调用两次 *f*2 函数求得 *a*、*b*、*c* 的最小值并存放到变量 *e* 中。不难算出 *d* 的值为 5，*f* 的值为 4，*e* 的值为 3。**

**答案**　C

**【例 4】有以下程序**

```
fun(int a, int b)
{   if(a>b)  return(a);
    else  return(b);
}
main()
{   int x=3,y=8,z=6,r;
    r=fun(fun(x,y),2*z);
    printf("%\d\n",r); }
```

**程序运行后的输出的结果是_____。（2003 年 9 月）**

（A）3　　（B）6　　（C）8　　（D）12

**题眼分析**　**函数** fun **有两个整型形式参数，其功能是返回两者中较大的值。主函数** main **中调用函数** fun**，其实调用了两次，第** 1 **次是** fun(*x*,*y*)**，显然返回** *x* **和** *y* **的较大者，结果是** 8**。这个结果作为第** 2 **次调用的第** 1 **个实参，于是第** 2 **次调用是求** 8 **和** 2*z **的较大者，结果是** 12**。**

**答案**　D

**【例 5】有以下程序**

```
char fun(char  x，char  y)
{   if(x<y)  return  x;
    return y;
}
main()
{   int  a='9'，b='8'，c='7';
    phintf("%c\n",fun(fun(a,b),fun(b,c)));
}
```

**程序的执行结果是_____。（2004 年 4 月）**

（A）函数调用出错　　（B）8　　（C）9　　（D）7

**题眼分析**　**程序中的函数** fun **被调用了** 3 **次，第** 1 **次是求** fun(*a*,*b*)**，其返回值是**'8'**；第** 2 **次是求** fun(*b*,*c*)**，其返回值是**'7'**；第** 3 **次是将前面的两个返回值作为函数** fun **的实参，即求** fun('8','7')**，得到的返回值是**'7'**。**

**答案**　D

**【例 6】有以下程序**

```
int f1(int x,int y){   return x>y?x:y;    }
int f2(int x,int y){   return x>y?y:x;    }
main()
{   int a=4,b=3,c=5,d=2,e,f,g;
    e=f2(f1(a,b),f1(c,d)); f=f1(f2(a,b),f2(c,d));
    g=a+b+c+d-e-f;
    printf("%d,%d,%d\n",e,f,g);
}
```

**程序运行后的输出结果是_____。（2005 年 4 月）**

（A）4,3,7　　（B）3,4,7　　（C）5,2,7　　（D）2,5,7

**题眼分析　参见【例 3】、【例 4】和【例 5】的分析。**

**答案**　A

## 二、填空题分析

**【例】以下程序的功能是调用函数** fun **计算：**m=1–2+3–4+…+9–10**，并输出结果。请填空。（**2003 **年** 9 **月）**

```
int  fun(int  n)
{   int  m=0,f=1,i;
    for(i=1;i<=n;i++)
    {   m+=i*f;
        f=   (1)   ;
    }
    return  m;
}
main()
{   printf("%d\n",   (2)   );  }
```

**题眼分析　函数** fun **通过一个** for **循环求各项的和，其中变量** *f* **是用来控制该项是正数还负数，其初值为** 1 **表示第** 1 **项是正数，第** 2 **项是负数，第** 3 **项又是正数，这样就是通过赋值语句** *f*=*f**(–1);**来控制。因此空（**1**）处应填入“** *f**(–1) **”。从函数** fun **的定义可以看出，形式参数** n **是用来控制求和的项数，现要求求** *m*=1–2+3–4+…+9–10 **的值，其项数是** 10**，因此其调用的形式为** fun(10)**。这就是空（**2**）处需要填写的内容。**

**答案**　**（**1**）** *f**(–1)**或**–*f*　　**（**2**）** fun(10)

# 7.4　函数的说明

## 7.4.1　考点提炼

### 考点 1：函数说明

**在** C **语言中，除主函数外，对于用户定义的函数要遵循“先定义，后使用”的规则。函数说明可遵循以下规则：**

**（**1**）调用库函数时，需要在程序的开头包含相应的头文件。但** scanf()**和** printf()**等少数的几个函数不需要。**

**（**2**）当被调函数定义在主调函数之前时，对被调函数的说明可以省去，也可以不省。**

**（**3**）当被调函数的返回值类型是整形或字符型时，不管其定义在主调函数之前还是之后，对被调函数的说明都可以省去，也可以不省。**

**（**4**）其他情况一律需要对被调函数进行说明。**

**（**5**）当被调函数和主调函数在同一个程序文件中，可在主调函数的函数体说明部分对被调函数进行说明。**

函数说明格式有 3 种形式，分别是：

格式 1：类型名　函数名()

格式 2：类型名　函数名(参数类型 1，参数类型 2，……)

格式 3：类型名　函数名(参数类型 1　参数名 1，参数类型 2　参数名 1，……)

### 7.4.2　题眼分析

#### 一、选择题分析

【例 1】若程序中定义了以下函数

```
double myadd(double a,double b)
{return (a+b);}
```

并将其放在调用语句之后，则在调用之前应该对函数进行说明，以下选项中错误的说明是_____。（2004 年 4 月）

（A）double myadd(double a, b);　　（B）double myadd(double，double);

（C）double myadd(double b，double a)　　（D）double myadd(double x,double y);

题眼分析　若一个函数定义在被调函数之后，其返回值不是 int 或 char 类型，则必须对之进行函数说明。函数说明时，参数名完全是虚设的，它可以是任意的用户标识符，既不必与函数首部的形参名一致，也可以与程序中任意用户标识符同名，甚至可以省略。因此选项（B）、（C）和（D）都是正确的函数说明。选项（A）中，没有指明函数 myadd 第 2 个形参的类型，因此是错误的。

答案　A

【例 2】若有以下程序

```
#include <stdio.h>
void f(int n);
main()
{   void f(int n); f(5);    }
void f(int n)
{   printf("%d\n",n);    }
```

则以下叙述中不正确的是_____。（2002 年 4 月）

（A）若只在主函数中对函数 f()进行说明，则只能在主函数中正确调用函数 f()

（B）若在主函数前对函数 f()进行说明，在主函数和其后其他函数中都可以调用函数 f()

（C）对于以上程序，编译时系统会提示出错信息：提示对 f()函数重复说明

（D）函数 f()无返回值，所以可用 void 将其类型定义为无值型

题眼分析　函数说明既可以在函数外说明，也可以在函数内说明。在函数外说明，在其后的所有函数均可调用该函数，若在函数内进行说明，则只能在本函数内调用该函数。函数只能定义一次，但函数说明可以出现多次。因此不难看出答案（C）是错误的描述。

答案　C

**二、填空题分析**

**【例】请在以下程序第1行的下划线处填写适当内容，使程序能正确运行。（2003年9月）**

```
_____(double,double);
main()
{   double  x, y;
    scanf("%lf%lf",&x,&y);
    printf("%lf\n",max(x,y));
}
double  max(double  a, double  b)
{   return(a>b?a:b);    }
```

**题眼分析** 函数 main 调用了用户定义函数 max，但函数 max 放在函数 main 之后，而且返回值的类型是 double，不是 int 或 char，因此需要进行函数说明。根据函数说明形式，空格处应填写函数 max 的返回值类型名和函数名，即 double max

**答案** double max

# 7.5 调用函数和被调用函数之间的数据传递

## 7.5.1 考点提炼

**考点 1：调用函数和被调用函数之间的数据传递**

C 语言中，调用函数和被调用函数之间的数据要以通过 3 种方式进行传递：

（1）实在参数和形式之间进行数据传递；

（2）通过 return 语句把函数值返回调用函数；

（3）通过全局变量。

在数据传递过程中，只是“按值”传递，即数据只能从实参单向传递给形参，函数中形参的值发生变化不会改变相对的实参。

## 7.5.2 题眼分析

**一、选择题分析**

**【例 1】有以下程序**

```
void f(int v, int w)
{   int  t;
    t=v; v=w; w=t;      }
main ()
{   int  x=1,y=3,z=2
    if(x>y)                 f(x,y);
    else    if (y>z)        f(y,z);
```

```
            else            f(x,z);
        printf("%d,%d,%d\n",x,y,z);
    }
```

**执行后输出结果是_____。（2004 年 4 月）**

（A）1,2,3　　（B）3,1,2　　（C）　1,3,2　　（D）　2,3,1

**题眼分析**　fun()函数调用时，数据只能从实参单向传送形参，形参的值修改不会影响到实参的值。因此主函数 main 中变量 *x*、*y*、*z* 不会因函数调用而修改，输出的值依旧是原来的值。

**答案**　A

**【例 2】有以下程序：**

```
float  fun(int x,int y)
{   return(x+y);  }
main()
{   int a=2,b=5,c=8;
    printf("%3.0f\n",fun((int)fun(a+c,b),a-c));
}
```

**程序运行后的输出结果是_____。（2002 年 9 月）**

（A）编译出错　　（B）9　　（C）21　　（D）9.0

**题眼分析**　fun()函数的是对传进来的两个整型参数相加，把和作为函数值返回，注意返回值为 float 型。在主函数调用了两次 fun()函数，第 1 次调用时把表达式 *a*+*c* 和变量 *b* 的值求和，得到 15 转换为 15.0 作为函数值返回。第 2 次调用把第 1 次调用的返回值通过强制类型转换成 int 型 15 再和表达式 a–c 相加，得到结果 9，转换成 float 型作为函数值返回。由于输出格式符为“ %3.0f ”，输出时没有小数位，故输出为 9。

**答案**　B

## 二、填空题分析

**【例 1】函数 fun 的功能是计算 $x^n$ 。**

```
double fun(double x,int n)
{   int i;  double y=l;
    for(i;=1; i<=n;i++)y=y*x;
    return y;
}
```

**主函数中已正确定义 *m*、*a*、*b* 变量并赋值，并调用 fun 函数计算：$m=a^4+b^4-(a+b)^3$。实现这一计算的函数调用语句为_____。（2004 年 4 月）**

**题眼分析**　从函数定义可以看出，函数 fun 有两个形式参数，第 1 个参数是基数的值，第 2 个参数是指数的值。因此，求 $a^4$ 的表达式是 fun(*a*,4)，求 $b^4$ 的表达式是 fun(*b*,4)，求$(a+b)^3$ 的表达式是 fun(*a*+*b*,3)。综上所述，实现计算 $m=a^4+b^4-(a+b)^3$ 的语句是 *m*= fun(*a*,4)+ fun(*b*,4) – fun(*a*+*b*,3); 。

**答案**　*m*= fun(*a*,4)+ fun(*b*,4) – fun(*a*+*b*,3);

**【例 2】以下程序运行后的输出结果是_____。（2005 年 4 月）**

```
void swap(int x,int y)
{   int t;
```

```
    t=x;x=y;y=t;printf("%d  %d  ",x,y);
}
main()
{   int a=3,b=4;
    swap(a,b);  printf("%d  %d\n",a,b);
}
```

**题眼分析**　变量作为形参是传值的，对形参的修改并不影响对应的实参。主函数中调用 fun()函数把实参 *a* 和 *b* 的值传给形式参数 *x* 和 *y*，实参和形参不再有联系。在 fun()函数中通过运算使 *x* 和 *y* 的值交换过来，所以 fun()函数中输出的结果是“4　3　”。fun()函数调用返回后，输出 *a* 和 *b* 依旧是原来的 *x* 和 *y*，为“3　4”。

**答案**　4,3,3,4

# 7.6　单元训练及参考答案

## 7.6.1　单元训练

### 一、选择题

1．若调用一个函数，且此函数中没有 return 语句，则正确的说法是_____。

（A）该函数没有返回值　　（B）返回若干个系统默认值

（C）能返回一个用户所希望的函数值　　（D）返回一个不确定的值

2．以下正确的说法是_____。

（A）定义函数时，形参的说明可以放在括号里，函数头后应加“ ;”

（B）没有 return 语句的函数将不返回值

（C）如果函数类型是 void，则函数中不能有 return 语句

（D）对定义在主调函数后面的非整型和字符型的函数应进行函数说明

3．若有以下函数调用语句：fun(a+b,(x,y),fun(n+k,d,(a,b)));在此函数调用语句中实参的个数是_____。

（A）3　　（B）4　　（C）5　　（D）6

4．以下函数值的类型是_____。

```
fun ( float x )
{   float y;
    y= 3*x-4;
    return y;   }
```

（A）int　　（B）不确定　　（C）void　　（D）float

5．以下程序的输出结果是_____。

```
fun(int x, int y, int z)
{   z=x*x+y*y; }
main()
{   int a=31;
```

```
    fun(5,2,a);
    printf("%d",a);
}
```

（A）0　　（B）29　　（C）31　　（D）无定值

6．有如下程序

```
int func(int a,int b)
{    return(a+b);}
main()
{   int x=2,y=5,z=8,r;
    r=func(func(x,y),z);
    printf("%\d\n",r);
}
```

该程序的输出的结果是_____。

（A）12　　（B）13　　（C）14　　（D）15

## 二、填空题

1．在 C 语言中，一个函数一般由两个部分组成，分别是__（1）__和__（2）__。

2．若没有定义函数的类型，该函数的类型是__（3）__，如果在函数中没有使用 return 语句，则返回的是__（4）__值。

3．程序段如下：

```
#include "stdio.h"
int abc(int u,int v);
main ()
{   int a=24,b=16,c;
    c=abc(a,b);
    printf('%d\n",c);
}
int abc(int u,int v)
{   int w;
    while(v) { w=u%v; u=v; v=w ;}
    return u;
}
```

输出结果是__（5）__。

4．下面 pi()函数的功能是根据以下的公式，返回满足精度 $f$ 要求的 $\pi$ 值。请填空。

$\pi=(1+1/3+2/(3*5)+(3*2/(3*5*7))+(4*3*2)/(3*5*7*9)+\cdots\cdots)$

```
double pi(double eps)
{   double s=0.0, t=1.0;
    int n;
    for(__(6)__; t>eps; n++) { s+=t; t=n*t/(2*n+1);}
    return(2.0 * __(7)__);
}
```

5．下面程序的输出是__（8）__。

```
unsigned fun6( unsigned num)
```

```
{   unsigned k=1;
    do {  k*=num%10;  num/=10; } while(num);
    return(k);
}
main()
{   unsigned n=26;
    printf("%d\n", fun6(n));
}
```

6．**以下程序运行后的输出结果是___（9）___。**

```
void fun(int x,int y)
{   x=x+y;y=x-y;x=x-y;
    printf("%d,%d,",x,y);  }
main()
{   int x=2,y=3;
    fun(x,y);
    printf("%d,%d\n",x,y);
}
```

### 7.6.2 参考答案

**一、选择题**

1． D　　2． D　　3． A　　4． A　　5． C

6． D

**二、填空题**

**（1）函数头**　　**（2）函数体**

（3）int　　**（4）不确定**

（5）8　　（6）n=1

（7）s　　（8）12

（9）3,2,2,3

# 第 8 章　指针

## 8.1　变量的地址和指针

### 8.1.1　考点提炼

**📖考点 1：变量地址和指针**

**计算机的内存是以字节为单位的一片连续的存储空间，每个字节都有一个编号，这个编号称为内存地址。在程序中，每定义一个变量，系统就会根据其类型为其分配一定数量的地址空间，那么该变量所占内存的第 1 个字节的地址称为该变量的地址。**

**在程序执行中变量的地址起到了寻找变量的作用，好像是一个指针指向了变量，所以常把变量的地址称为“ 指针 ”。指针（地址）在 C 语言中也是一种数据类型，它可以存放在一种特殊的变量中，存放变量地址（指针）的变量称“ 指针变量 ”。**

**📖考点 2：直接存取和间接存取**

**存取变量的值有两种方式，一种是直接使用变量的地址来存取变量的值的方式称为“ 直接存取 ” 方式。另一种是使用指针变量来存取变量的值的方式称为 “ 间接存取 ” 方式。**

### 8.1.2　题眼分析

**【例 1】设有定义：**int　a,*pa=&a; **以下** scanf **语句中能正确为变量** *a* **读入数据的是_____。（2004 年 4 月）**

**（A）**scanf("%d",pa);　　**（B）**scanf("%d",a);

**（C）**scanf("%d",&pa);　　**（D）**scanf("%d",*pa);

**题眼分析　函数** scanf()**的第** 2 **参数（即输入列表）需要使用变量的地址。变量** *a* **的地址是**&a**，经过***pa=&a;**语句后，指针变量** *pa* **中存在着变量** *a* **的地址。**&pa **是指针变量** *pa* **的地址，***pa **是通过间接存取方式得到变量** *a* **的值。**

**答案**　A

**【例** 2**】有以下程序**

```
main( )
{   int  a=7 , b=8 , *p , *q , *r ;
    p=&a ;  q=&b ;
    r=p ;       p=q;    q=r ;
    printf("%d,%d,%d,%d\n", *p , *q , a , b ) ;
}
```

程序运行后的输出结果是_____。（2004 年 9 月）

A）8,7,8,7　　B）7,8,7,8　　C）8,7,7,8　　D）7,8,8,7

题眼分析　函数运行完 *p*=&*a* ; *q*=&*b* ;两条语句后，指针变量 *p* 存放着变量 *a* 的地址，指针变量 *q* 存放着变量 *b* 的地址。接下来 3 条赋值语句是交换变量 *p* 和变量 *q* 的值，这样指针变量 *p* 存放着变量 *b* 的地址，指针变量 *q* 存放着变量 *a* 的地址。因此*p 值就是 *b* 的值，**q* 值就是 *a* 的值，而变量 *a* 和 *b* 的值没有改变。

答案　C

# 8.2　指针变量的定义和引用

## 8.2.1　考点提炼

### 考点 1：指针变量的定义

定义指针变量的一般形式为：

类型名　*指针变量名 1，*指针变量名 2，……；

例如：

```
int *p,*q;
```

以上语句中，*p* 和 *q* 都是用户标识符，在每个变量前的星号（*）是一个说明符，用来说明该变量是一个指针变量。

### 考点 2：指针变量的基类型

以上语句中，类型名 int 说明这两个指针变量只能用来存放整型变量的地址，我们称 int 是变量 *p* 和 *q* 基类型。

为什么指针变量要定义指针变量的基类型呢？一个指针变量中存放一个存储单元的地址值。一个存储单元中的“一”，对于不同的数据类型所代表的字节数是不同的，例如，对于整型来说，它代表两个字节，而对于实型来说，它则代表 4 个字节。对于指针的移动和地址的加减运算，指针移动的最小单元是一个存储单元而不是一个字节，因此指针变量必须区分基类型。

### 考点 3：给指针变量赋值

1．通过求地址运算（&）获取地址值

定义一个指针变量，然后把一个变量的地址赋给它，就可以使该指针变量指向该变量。通过求地址运算（&）获取地址值的方法有两种：一种是赋初值，另一种是赋值。

以下是赋初值方式的例子：

```
int a,*p=&a;  /*把a的地址作为初值赋给指针变量p，p指向了变量a*/
```

以下是赋值方式的例子：

```
int a,*p;p=&a;  /* 把a的地址赋值给指针变量p，p指向了变量a*/
```

此时，*p 代表的就是变量 *a*，给 *a* 赋值 3 可以写成“*p=3;”。

2．通过指针变量获得地址值

通过赋值运算，把一个指针变量中的地址值赋给另一个指针变量，从而使两个指针变量指向同一地址。

3．通过标准函数获取地址值

通过调用库函数 malloc 和 calloc 在内存中开辟动态存储单元，并把所开辟的动态存储单元的地址赋值给指针变量。

### 考点 4：给指针变量赋“空”值

指针变量可以赋“空”值，其方法是：

```
p=NULL; 或p='\0' ;      或p=0 ;
```

注意：由于 NULL 是 stdio.h 头文件的预定义符，在使用 NULL 时，在程序的前面应包含定义行：include "stdio.h" 。

### 考点 5：取地址运算符（&）和指针运算符（*）

与指针有关的运算符主要有两个，分别是“取地址运算符(&)”和“指针运算符(*)”。

（1）&运算符。为取地址运算符，其作用是返回操作对象（变量或数组元素）的地址。例如，“&x;”返回变量 *x* 的地址，“&*a*[5];”返回的是数组元素 *a*[5]的地址。

（2）*运算符。为指针运算符，其作用是返回以操作对象的值作为地址的变量（或内存单元）的内容。例如，有变量定义语句“int *a*,**p*1=&*a*;”，则**p*1 代表的就是变量 *a*。

它们都是单目运算符，优先级高于所有的双目运算符，它们的结合性均是自右向左。在使用这两个运算符需要注意以下几点：

（1）如果已经执行了“*p*=&*a* ;”语句，若有

```
&*p
```

由于“&”和“*”两个运算符的优先级别相同，但按自右向左方向结合，因此先运算**p*，它就是变量 a，再执行&运算。因此它等同于&a。

（2）*&*a* 的含义是 *a*。因为先进行&*a* 运算，得到 *a* 的地址，再进行*运算，即&*a* 所向的变量，因此*&*a* 等价于 *a*。

（3）(**p*)++相当于 *a*++。注意括号是必要的，如果没有括号，就成为*(*p*++)，这时使指针变量本身增 1，并不是使 *p* 所指的存储单元的值增 1。

### 考点 6：移动指针

移动指针就是对指针变量进行加上或减去一个整数或通过赋值运算，使指针变量指向相邻的存储单元。因此，只有当指针指向一串连续的存储单元时，指针的移动才有意义。

在对指针进行加、减运算中，数字“1”不再代表十进制整数“1”，而是指 1 个存储单元长度；至于 1 个长度占多少存储空间，则视指针变量的基类型而定。如果指针变量的基类型是 int，则移动 1 个存储单元长度就是移动两个字节；如果指针变量的基类型是 double，则移动 1 个存储单元长度就是移动 8 个字节，等等。

**考点 7：指针比较**

**若两个指针变量 *p* 和 *q* 的基类型相同，则：**

**（1）*p* 等于 *q*（即 *p*==*q* 为真）表示两个指针指向同一目标；**

**（2）*p* 大于 *q*（即 *p*>*q* 为真）表示 *p* 指向的内存地址比 *q* 指向的内存地址高；**

**（3）*p* 等于空（即 *p*=='\0'为真）表示 *p* 是一个空指针。**

**考点 8：指向指针的指针**

**指针变量本身是变量，因此它在内存中也占有存储单元，如果指针变量的地址又放在另一类特殊的变量中，这类特殊的变量就指向了指针变量。我们把这类特殊的变量称为“指针的指针”，也称“多级指针”。通常使用的多级指针是二级指针，而前面介绍的指针变量也称为“一级指针变量”。**

**二级指针变量的定义及赋初值的形式如下：**

**格式：存储类型符　数据类型符　**指针变量名[=初值]；**

**功能：定义一个二级指针变量，它的存储类型由“存储类型符指定”，它指向的指针变量指向的变量的数据类型由“数据类型符”决定，还可以同时给它赋初值。**

**说明：**

**（1）二级指针变量定义时前面必须有两个“*”；**

**（2）初值必须是一级指针变量的地址，通常形式是“&一级指针变量名”或“指针数组名（数组元素为一级指针）”。**

**例如：**

```
int a,*pa=&a,**ppa=&pa;
```

**该语句定义一个整型变量 *a*，一个一级指针 *pa* 和一个二级指针 *ppa*，通过赋初值使 *pa* 指向了 *a*，*ppa* 指向了 *pa*。**

**可以通过赋值使某个二级指针变量指向一级指针变量。赋值的形式是：**

**二级指针变量=&一级指针变量；**

## 8.2.2　题眼分析

### 一、选择题分析

**【例 1】设有定义：**int　n=0，*p=&n，**q=&p;，则以下选项中，正确的赋值语句是_____。（2004 年 4 月）**

（A）p=1;　　（B）*q=2　　（C）q=p　　（D）*p=5

**题眼分析　本题先定义一个变量 *n*，再定义一个指针变量 *p* 通过赋初值让它指向了变量 *n*，最后定义一个指向指针的指针变量 *q*，并通过赋初值让它指向了指针变量 *p*。指针变量 *p* 中存放的是变量 *n* 的地址，不能直接赋值，因此选项（A）不正确；指针变量 *q* 存放的是指针变量 *p* 地址，也不能直接赋值，因此选项（B）不正确。指针变量 *q* 的基类型是指针变量，而指针变量 *p* 的基类型是整型变量，因此选项（C）不正确。选项（D）等价于 *n*=5。**

**答案**　D

【例 2】若程序中已包含头文件 stdio.h，以下选项中，正确运用指针变量的程序段是_____。（2003 年 9 月）

（A）int *i=NULL;
　　scanf("%d",i);

（B）float *f=NULL
　　*f=10.5;

（C）char t='m',*c=&t;
　　*c=&t;

（D）long *L;
　　L='\0';

题眼分析　选项（A）中，指针变量 i 被赋“空”值，不能直接访问，因此是错误的。同样的道理，选项（B）也是错误的。选项（C）中，指针变量 *c* 通过赋初值让它指向了变量 *t*，对**c* 赋值就等同于给变量 *t* 赋值，必须赋字符型数据，而&*t* 是地址，因此选项（C）也是错误的。选项（D）是给指针变量 L 赋“空”值，是正确的。

答案　D

【例 3】若有以下定义和语句

```
#include <stdio.h>
int a=4,b=3,*p,*q,*w;
p=&a; q=&b; w=q; q=NULL;
```

则以下选项中错误的语句是_____。（2003 年 4 月）

（A）*q=0;　　（B）w=p　　（C）*p=a;　　（D）*p=*w;

题眼分析　本题中定义了 3 个指向整型的指针变量 *p*、*q* 和 *w*，让 *p* 指向了 *a*，让 q 指向了 *b*，把 *q* 的值赋给了 *w*，*w* 指向了 *b*，把 NULL 赋值给 *q*，*q* 不指向任何变量。可以给整型变量指向的变量赋一个整型值（表达式），故答案（C）和（D）是正确的；可以给指针变量赋一个同种类型的指针变量（或地址值），故答案（B）也是正确的；答案（A）是不正确的，因为 *q* 不指向任何变量，给一个空指针指向的变量赋一个值是错误的。

答案　A

【例 4】有以下程序段

```
main()
{   int a=5,*b,**c;
    c=&b;b=&a;
}
```

程序在执行了 *c=&b;b=&a;* 语句后，表达式：**c* 的值是_____。（2003 年 9 月）

（A）变量 *a* 的地址　　（B）变量 *b* 中的值

（C）变量 *a* 中的值　　（D）变量 *b* 的地址

题眼分析　本题先定义了一个整型变量 *a*、一个基类型为整型变量的指针变量 *b* 和一个基类型为指针的指针变量 *c*。通过 *c=&b;* 语句，使指针变量 *c* 存放指针变量 *b* 的地址，再通过 *b=&a;* 语句，使指针变量 *b* 存放变量 *a* 的地址。于是，**c* 代表变量 *b*，**b* 代表变量 *a*，因此**c 就代表变量 *a*。

答案　C

【例 5】有以下程序

```
main()
{   char s[]="159",*p;
    p=s;
```

```
        printf("%c",*p++);printf("%c",*p++);
    }
```

程序运行后的输出结果是_____。（2005 年 4 月）

（A）15　　（B）16　　（C）12　　（D）59

题眼分析　由于指针运算符*的优先级比自加运算符++的优先级低，*p++相当于*(p++)，其含义是先引用指针 p 所指的变量的值，再将指针 p 本身加 1，使其指向下一个变量。在本题中，指针 p 指向字符串的第 1 个字符，第 1 个 prinft()函数输出字符'1'后，指针 p 加 1 指向字符'5'；第 2 个 prinft()函数将输出字符'5'，并使指针 p 加 1 指向字符'9'。这里一定要注意，*p++相当于*(p++)，而不等价于(*p)++，后者的含义是将 p 所指变量加 1，而不是移动指针。

答案　A

## 二、填空题分析

【例 1】有如下程序段

```
int *p,a=10,b=1;  p=&a; a=*p+b;
```

执行该程序段后，a 的值为_____。（2000 年 9 月）

题眼分析　本题首先定义了一个整型指针变量 p 和两个整型变量 a 和 b，并给 a 赋初值 10，给 b 赋初值 1。接着通过赋值语句把 a 的地址赋给指针变量 p，p 指向了 a。语句“a=*p+b;”中的*p 其实就是 a，所以该语句是把 a 和 b 相加赋给 a，结果 a 的值为 11。

答案　11

【例 2】设有以下程序：

```
main()
{   int a, b, k=4, m=6, *p1=&k, *p2=&m;
    a=pl= =&m;  b=(*p1)/(*p2)+7;
    printf("a=%d\n",a);  printf("b=%d\n",b);
}
```

执行该程序后，a 的值为___（1）___，b 的值为___（2）___。（2001 年 9 月）

题眼分析　程序中定义了两个指针变量 p1 和 p2，并通过赋初值让它们分别指向了变量 k 和 m。语句“a=pl==&m;”中先执行关系运算“p1==&m”，显然是 0（假），然后把 0 赋值给 a。接着执行语句“b=(*p1)／(*p2)+7;”，此处*p1 的值就是 k 的值 4，*p2 的值就是 m 的值 6，即把“4/6+7”的结果赋给 b，b 的值为 7。

答案　（1）0　　（2）7

# 8.3　函数之间地址值的传递

## 8.3.1　考点提炼

### 考点 1：用指针作函数参数

由于变量是传值的，变量作为函数的参数不能从被调函数中带出结果。但通过指针可以作

为参数，可以实现在主调函数和被调函数之间传递数据。

例　利用指针做参数，交换两个变量的值。

程序清单如下：

```
void rev(int *a,int *B)
{   int t;  t=*a;*a=*b;*b=t;}
main()
{   int x=3,y=5; rev(&x,&y);  printf("\n%d,%d",x,y);}
```

程序运行结果为：5,3。

分析

（1）变量 x 和 y 的值已经交换了，交换的原因是函数的形式参数使用了指针变量，而相应的实参是变量的地址，是一种“传址”的关系。

（2）本例的执行过程可通过图 8-1 来说明。主函数调用 rev 时，把 *x* 和 *y* 的地址分别传给了形参 *a* 和 *b*，形参指针变量 *a* 和 *b* 就指向了实参变量 *x* 和 *y*。因此在 rev 中交换了**a* 和**b* 的值，也就是交换了实参变量 *x* 和 *y* 的值。

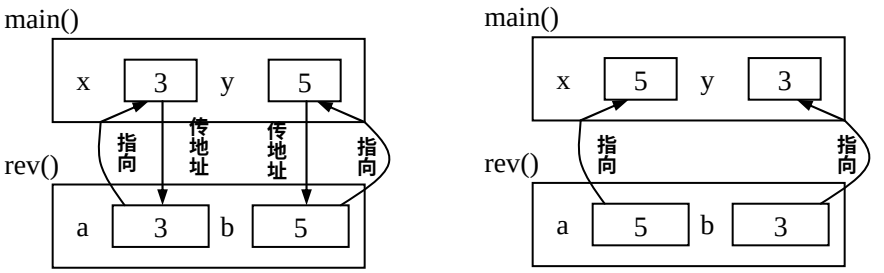



图 8-1　函数间指针作为参数的数据传递

（3）指针变量作为形参时，其对应的实参应是指针变量、数组名或地址表达式。

### 考点 2：返回地址值的指针函数

函数的返回值的类型不但可以是普通的数据类型，而且可以是地址即指针类型。指针函数的定义格式如下。

格式：存储类型　数据类型　　*函数名(形式参数说明列表)

作用：定义一个函数，返回值为“数据类型”的变量地址。

说明：

（1）存储类型有两种，static 和 extern，默认为 extern。

（2）“*函数名”不能写成“(*函数名)”，否则就成了指向函数的指针。

（3）此类函数的调用形式通常是：*p*=函数名(实际参数列表)。其中 *p* 通常是调用函数中定义的一个指针变量。

## 8.3.2 题眼分析

### 一、选择题分析

**【例 1】有以下程序**

```
void fun(char *a, char *b)
{   a=b; (*a)++;    }
main()
{   char c1='A', c2='a', *p1, *p2;
    p1=&c1; p2=&c2; fun(p1,p2);
    printf("%c%c\n", c1, c2);
}
```

**程序运行后的输出结果是_____。（2003 年 9 月）**

（A）Ab　　（B）aa　　（C）Aa　　（D）Bb

**题眼分析　在主函数中定义了两个字符变量 *c1* 和 *c2* 并赋初值'A'和'a'，并通过&运算将指针变量 *p1* 和 *p2* 分别指向这两个变量，然后调用函数 fun()把 *c1* 的地址和 *c2* 的地址传给对应的形参指针变量 *a* 和 *b*，所以在 fun()函数中指针变量 *a* 指向了 main()函数中的 *c1*，指针变量 *b* 指向了 main()函数中的 *c2*。fun()函数中的语句"** *a=b* **"，使指针 *a* 也指向变量 *c2*。然后执行语句"** (**a*)++; **"，即给指针变量 *a* 指向的变量（即 main()函数中的 *c2*）增加 1，即 *c2* 值为 *b*。因此，fun()函数返回后在 main 后中执行输出语句"** printf("%c%c\n", *c1*, *c2*); **"，得到的结果是 Ab。**

**答案**　A

**【例 2】有以下程序**

```
void f(int *x,int *y)
{   int t;
    t=*x;*x=*y;*y=t;
}
main()
{   int a[8]={1,2,3,4,5,6,7,8},i,*p,*q;
    p=a;q=&a[7];
    while(p<q)
    {   f(p,q);     p++;    q--;    }
    for(i=0;i<8;i++)printf("%d,",a[i]);
}
```

**程序运行后的输出结果是_____。（2005 年 4 月）**

（A）8,2,3,4,5,6,7,1,　　（B）5,6,7,8,1,2,3,4,

（C）1,2,3,4,5,6,7,8,　　（D）8,7,6,5,4,3,2,1,

**题眼分析　本题定义了一个函数 *f*，它有两个指针型的形式参数 *x* 和 *y*，该函数的功能是交换这两个指针所指单元的值。在主函数中定义了两个指针变量 *p* 和 *q* 并让它们指向了数组 a 的第 0 个元素和第 7 个元素。在 while 循环中，通过函数 *f* 依次交换 a[0]和 a[7]，a[1]和 a[6]，a[2]和 a[5]，a[3]和 a[4]的值。**

**答案**　D

**【例 3】程序中对 fun 函数有如下说明：**

```
void  *fun( ) ;
```

此说明的含义是_____。（2004 年 9 月）

（A）fun 函数无返回值

（B）fun 函数的返回值可以是任意的数据类型

（C）fun 函数的返回值是无值型的指针类型

（D）指针 fun 指向一个函数，该函数无返回值

题眼分析　在 C 语言中，运算符（）的优先级比*高，所以 void　*fun( )说明了一个函数，该函数的返回值是无值型的指针类型。而 void (*fun)( )说明指针 fun 指向一个函数，而该函数无返回值。

答案　C

【例 4】有以下程序

```
int *f(int  *x, int *y)
{   if(*x<*y)
        return  x;
    else
        return  y;
}
main()
{   int a=7,b=8,*p,*q,*r;
    p=&a;   q=&b;
    r=f(p,q);
    print("%d,%d,%d\n",*p,*q,*r);
}
```

执行后输出结果是_____。（2003 年 4 月）

（A）7,8,8　　（B）7,8,7　　（C）8,7,7　　（D）8,7,8

题眼分析　本题定义了一个返回值为指针的函数 *f*，它有两个指针型的形式参数 *x* 和 *y*，该函数的功能是返回 *x* 和 *y* 指向的变量的值较小的那个指针变量。在主函数中定义了两个指针变量 *p* 和 *q* 并让它们指向了变量 *a* 和 *b*，调用函数 *f*，返回指向的变量值较小的指针变量，*r* 的值为指针变量 *p* 的值（变量 *a* 的地址），故最后输出的*p、*q 和*r 的值是 7、8 和 7。

答案　B

## 二、填空题分析

【例 1】有以下程序

```
void f(int  y,int  *x)
{   y=y+ *x;   *x;=*x+y; }
main()
{   int   x=2,y=4;
    f(y,&x);
    printf("%d  %d\n",x,y);
}
```

执行后输出结果是_____。（2004 年 4 月）

题眼分析　在 main()函数中定义了两个整型变量 *x*、*y*，并分别赋初值 2 和 4，然后调用函

数 f()，调用时把实参 *y* 的值传送形参 *y*，把实参 *x* 的地址传给形参指针变量 *x*，指针变量 *x* 就指向了 main()函数中的变量 *x*，在 f()函数中*x 其实就是 main()函数中的变量 *x*。先执行“ *y=y+*x;* ”语句，使得形参变量 *y* 变为 6，但这一值不能返回到 main()函数中的变量 *y*，再执行“ *x=*x+y;* ”语句，使*x 的值变为 8，也就是 main()函数中的变量 *x* 的值变为 8。

**答案** 8 4

**【例 2】函数** void fun(float *sn, int n)**的功能是：根据以下公式计算 S，计算结果通过形参指针 sn 传回。n 通过形参传入，n 的值大于等于 0。请填空。（2000 年 4 月）**

$$s=1-\frac{1}{3}+\frac{1}{5}-\frac{1}{7}+\Lambda\pm\frac{1}{2n-1}$$

```
void fun( float *sn, int n)
{   float s=0.0, w, f=-1.0;
    int i=0;
    for(i=0; i<n;i++)
    {   f= __(1)__* f; w=f/(2*i+1); s+=w; }
        __(2)__=s;
}
```

**题眼分析** 从程序中可以看出，变量 *f* 的作用就是求当前项的项号。分析公式可知第 1 项的符号为 1，然后每一项的符号是前一项的符号乘–1。因此空（1）处应填写“ –1 ”。当所有项的值都加到了 *s* 中后，应把 *s* 的值送到形参 sn 指向的主调函数中的变量中，所以空（2）处应填“ *sn ”。

**答案** （1）–1 （2）*sn

# 8.4 单元训练及参考答案

## 8.4.1 单元训练

### 一、选择题

1．**下面程序的输出是_____。**

```
void prtv(int *x)
{   printf("%d\n", ++*x);}
main()
{   int a=25;       prtv(&a);   }
```

（A）23 （B）24 （C）25 （D）26

2．**若有说明：**double *p,a;**则能通过** scanf **语句正确给输入项读入数据的程序段是_____。**

（A）*p=&a; scanf("%lf",p); （B）p=(double *)malloc(8);scanf("%f",p);

（C）p=&a;scanf("%lf",a); （D）p=&a; scanf("%le",p);

3．**要求函数的功能是交换 *x* 和 *y* 中的值，且通过正确地调用返回交换结果。能正确执行此功能的函数是_____。**

（A）funa(int *x,int *y)
{ int *p;

（B）funb(int x,int y)
{ int t;

*p=*x;*x=*y;*y=*p;}　　t=x;x=y;y=t;}

（C）func(int *x,int *y)　　（D）fund(int x,int y)

{　*x=*y;*y=*x;}　　{　*x=*x+*y; *y=*x-*y; *x=*x-*y; }

4．若有定义和语句：

```
int **qq,*q,c=10,d=20;
qq=&q;q=&c;q=&d;printf("%d\n",*q,**qq);
```

则输出结果是_____。

（A）10,20　（B）10,10　（C）20,10　（D）20,20

5．设 *p*1 和 *p*2 是指向同一个 int 型一维数组的指针变量，*k* 为 int 型变量，则不能正确执行的语句是_____。

（A）k=*p1+*p2;　（B）p2=k;　（C）p1=p2;　（D）k=*p1 * (*p2);

6．若有说明语句：int　a,b,c,*d=&c ;，则能正确从键盘读入 3 个整数分别赋给变量 *a*、*b*、*c* 的语句是_____。

（A）scanf("%d%d%d",&a,&b,d);　（B）scanf("%d%d%d",&a,&b,&d);

（C）scanf("%d%d%d",a,b,d);　（D）scanf("%d%d%d",a,b,*d);

7．若定义：int a=511,*b=&a;，则 printf("%d\n",(*b)++);的输出结果为_____。

（A）无确定值　（B）*a* 的地址　（C）512　（D）511

8．有以下程序

```
void fun(char *c,int d)
{   *c=*c+1;d=d+1;
    printf("%c,%c,",*c,d);     }
main()
{   char a='A',b='a';
    fun(&b,a);  printf("%c,%c\n",a,b);     }
```

程序运行后的输出结果是_____。

（A）B,a,B,a　（B）a,B,a,B　（C）A,b,A,b　（D）b,B,A,b

9．在说明语句“ int *f(); ”中，标识符 *f* 代表_____。

（A）一个用于指向整型数据的指针变量　（B）一个用于指向一维数组的行指针

（C）一个用于指向函数的指针变量　（D）一个返回值为指针型的函数名

## 二、填空题

1．下面程序的运行结果是___（1）___。

```
void swap(int *a,int *b)
{   int *t; t=a;a=b;b=t; }
main()
{   int x=3,y=5,*p=&x,*q=&y;
    swap(p,q);
    printf("%d %d\n",*p,*q);
}
```

2．fun1 函数的调用语句为:fun1(&a,&b,&c); 它将 3 个整数按由大到小的顺序调整后依次放入 *a*，*b*，*c*，3 个变量中，*a* 中放最大数。请填空。

```
void fun2 (int *x,int *y)
{   int t;
    t=*x; *x=*y;*y=t;
    }
void fun1 (int *pa,int *pb,int *pc)
{   if(*pc>*pb) fun2(   (2)   )
    if(*pa<*pc) fun2(   (3)   )
    if(*pa<*pb) fun2(   (4)   )
}
```

**3．下列程序的输出结果是__(5)__。**

```
int ast(int x,int y,int * cp,int * dp)
{   *cp=x+y; *dp=x-y; }
main()
{   int a,b,c,d;
    a=4;b=3;
    ast(a,b,&c,&d);
    printf("%d  %d\n",c,d);
}
```

**4．下面函数用来求出两个整数之和，并通过形参传回两数相加的和，请填空。**

int add(int x, int y,__(6)__z)

{ __(7)__ =x+y ; }

## 8.4.2 参考答案

### 一、选择题

| | | | | |
|---|---|---|---|---|
| 1． D | 2． D | 3． A | 4． D | 5． B |
| 6． A | 7． D | 8． D | 9． D | |

### 二、填空题

(1) 3 5

(2) pc ,pb **或** pb , pc

(3) pc , pa **或** pa , pc

(4) pb , pa **或** pa , pb

(5) 7　1

(6) int *

(7) *z

# 第 9 章　数组

## 9.1　一维数组的定义和元素的引用

### 9.1.1　考点提炼

**📖考点 1：一维数组的定义**

**一维数组在使用之前必须要定义，具体的定义语句的格式与功能如下。**

**格式：存储类型　数据类型　数组名[长度];**

**功能：定义指定“ 存储类型 ”与“ 数据类型 ”的一维数组，其名称由“ 数组名 ”指定。**

**说明：**

**（1）存储类型可以是自动型（auto）、静态型（static）、外部参照型（extern），但不能是寄存器型（register）。存储类型省略时，默认为自动型。**

**（2）数据类型可以是任何基本类型，也可以是其他组合类型，如结构型、指针型等。**

**（3）数组名是标识符，应符合标识符的取名规则。**

**（4）长度是一个“ 整型常量表达式 ”，通常是一个整型常量。**

**（5）可以通过一条定义语句，定义一个或多个数组及变量。**

**（6）数组定义后，将占用连续的存储空间，其大小为“ 长度*数据类型所占用的字节数 ”。**

**定义数组时要注意以下几点：**

**（1）定义的时候数组的长度使用了变量，如** int *a*[n] **（**n **为变量或没有定义）；**

**（2）将**[]**写成了**()**，如** int a(n) **；**

**（3）数组的定义语句写在了执行语句之后，如“** scanf("%d",&n);int a[n]; **”，当然此语句也违反了第** 1 **个错误。**

**📖考点 2：一维数组的初始化**

**数组元素和变量一样，可以在定义时赋初值，给数组元素赋初值称数组的初始化。**

**格式：存储类型　数据类型　数组名[长度]={初值表};**

**功能：定义一个“ 数组名 ”的数组，并把“ 初值表 ”中的值依次赋给相应数组元素。**

**说明：**

**（1）初值表中的初值用“ 逗号 ”隔开。**

**（2）可对数组中的所有元素赋初值，也可对数组的前若干个元素赋初值（部分赋初值）。**

**（3）若对所有元素赋初值，数组的长度可以省略。**

**（4）若只对数组的前若干元素赋初值，没有赋初值的元素也有初值。若是数值类型为** 0**，是字符型则为**'\0'**。**

（5）若数组的存储类型为 auto，如果不进行初始化，则其元素初值不确定。若数组的存储类型为 static，没有进行初始化，其元素均有初值：对于数值类型是 0，对于字符型则是'\0'。

注意：

（1）可把字符型数据作为初值赋给整型数组元素，赋的是对应的 ASCII 码值，如：

```
static int d[10]={ 'c', 'd', 'e'};
```

（2）可把整型数据作为初值赋给字符型数组元素，赋的是把该整型数作为 ASCII 码值找到的对应字符，如：

```
static char d[10]={97,98,99};
```

**考点 3：一维数组的引用**

在程序中使用的是数组元素，当定义了一个数组后就可以引用它的数组元素了。引用方法有两种，分别是“指针法”和“下标法”。本节只讨论“下标法”，其引用形式为：

数组名[下标]

C 语言规定，数组下标从 0 开始，因此具有 $N$ 个元素的数组，其下标范围为 0～$N-1$。例如有定义：int a[5];，那么其元素为 a[0]、a[1]、a[2]、a[3]和 a[4]。值得注意的是在 C 语言中允许下标越界，也就是说 C 对下标越界不进行检查，a[5] 、a[6]均是可用的，但它们不是数组元素。虽然 C 允许下标越界，但应尽量避免，否则可能产生意想不到的结果。

## 9.1.2 题眼分析

### 一、选择题分析

【例 1】以下叙述中错误的是_____。（2005 年 4 月）

（A）对于 double 类型数组，不可以直接用数组名对数组进行整体输入或输出

（B）数组名代表的是数组所占存储区的首地址，其值不可改变

（C）当程序执行中，数组元素的下标超出所定义的下标范围时，系统将给出“下标越界”的出错信息

（D）可以通过赋初值的方式确定数组元素的个数

题眼分析　在 C 语言中，程序运行时系统并不自动检验数组的下标是否越界，数组元素的下标超出所定义的下标范围时，系统不会有出错信息，因此选项（C）错误，是答案。在各种类型数组中，只有字符数组可以用来存储字符串，可直接用数组名对数组进行整体输入或输出，而其他类型数组都不行，因此选项（A）描述是正确的。选项（B）和（D）的叙述也都是正确的。

答案　C

【例 2】以下能正确定义一维数组的选项是_____。（2005 年 4 月）

（A）int a[5]={0,1,2,3,4,5};　　（B）char a[]={0,1,2,3,4,5};

（C）char a={'A','B','C'};　　（D）int a[5]="0123";

题眼分析　选项（A）数组定义的长度为 5，但赋了 6 个初值，所以是错误的；选项（C）中，少了[]，也是错误的；选项（D）定义的是一个整数数组，但给它赋初值时却赋了一个字

符串，因此也是错误的；只有选项（B）定义了一个字符数组 *a* 同时给它赋初值（字符的 ASCII 码），由于省去了长度，所以长度为由初值个数确定，为 6。

**答案**　D

**【例 3】以下定义语句中，错误的是_____。（2001 年 9 月）**

（A）int a[]={1,2};　　（B）char *a[3];　　（C）char s[10]="test";　（D）int n=5,a[n];

**题眼分析**　选项（A）定义了一个整型数组 *a* 同时给它赋初值，由于省去了长度，所以长度由初值个数确定为 2；选项（B）定义的是一个具有 3 个元素的字符指针数组；选项（C）定义了一个具有 10 个元素的字符数组 *s* 并用一个字符串给它赋初值，初值个数没有超过数组元素个数，因此是正确的；选项（D）是错误的，因为在定义数组时，要求长度应为整型常量或整型常量表达式，不能为变量，而此处用了变量 n。

**答案**　D

**【例 4】有以下程序**

```
int  f (int  a)
{   return  a%2 ;       }
main( )
{   int  s[8]={1,3,5,2,4,6}, i , d=0 ;
    for (i=0; f(s[i]) ; i++) d+=s[i] ;
    printf("%d\n", d ) ;
}
```

**程序运行后的输出结果是_____。（2004 年 9 月）**

（A）9　　（B）11　　（C）19　　（D）21

**题眼分析**　函数 f 的功能是判断一个整数是否为奇数，如果是奇数则返回 1，否则返回 0。在函数 main 中，for 循环的结束条件是 f(s[i])，也就是说当 *s*[*i*]是偶数时就退出循环，否则累加 *s*[*i*]。换句话说，程序的功能是从数组 *s* 中的第 1 个元素开始，如果元素是奇数则累加到变量 d 中，直到遇到偶数，结束循环，最后输出 *d*。在本题中，*d*=1+3+5=9。

**答案**　A

**【例 5】以下程序的输出结果是_____。（2001 年 4 月）**

（A）20　　（B）21　　（C）22　　（D）23

```
main()
{   int i, k, a[10], p[3];
    k=5;
    for (i=0;i<10;i++) a[i]=i;
    for (i=0;i<3;i++) p[i]=a[i *(i+1)];
    for (i=0;i<3;i++) k+=p[i] *2;
    printf("%d\n",k);
}
```

**题眼分析**　本题定义了两个数组 a[10]和 p[3]，首先给变量 *k* 赋值 5，然后通过一个循环给 a[0]～a[9]依次赋值 0～9。第 2 个循环执行了 3 次，第 1 次相当于执行“ p[0]=a[0*1]; ”，结果给 p[0]赋一个值 0；第 2 次相当于执行“ p[1]=a[1*2]; ”，结果给 p[1]赋一个值 2；第 3 次相当于执行“ p[2]=a[2*3]; ”，结果给 p[2]赋一个值 6。第 3 个循环也执行了 3 次，第 1 次相当于执行了“ k=k+p[0]*2; ”，得到的 *k* 值是 5；第 2 次相当于执行了“ k=k+p[1]*2; ”，得到的 *k* 值是 9；

第 3 次相当于执行了“ k=k+p[2]*2; ”，得到的 *k* 值是 21。

**答案** B

## 二、填空题分析

**【例 1】以下程序运行后的输出结果是_____。（2003 年 9 月）**

```
main()
{   int i,n[]={0,0,0,0,0};
    for(i=1;i<=4;i++)
    {  n[i]=n[i-1]*2+1;
        printf("%d",n[i]);
    }
}
```

**题眼分析** 本题定义了一个数组 a[]，并给每个元素赋初值 0。循环从第 2 个元素 a[1]开始，使这个元素的值为前一个元素的 2 倍再加 1 并输出，循环体共运行了 4 次。因此，a[1]=a[0]×2+1=1、a[2]=a[2]×2+1=3、a[3]=a[2]×2+1=7、a[4]=a[3]×2+1=15。

**答案** 1 3 7 15

**【例 2】以下程序从终端读入数据到数组中，统计其中正数的个数，并计算它们之和。请填空。（2004 年 9 月）**

```
main( )
{   int  i , a[20] , sum , count ;
    sum=count=0;
    for(i=0; i<20; i++ ) scanf("%d ", __(1)__ );
    for(i=0; i<20; i++ )
    {   if (a[i]>0 )
        {   count++;
            sum+= __(2)__ ;
        }
    }
printf("sum=%d, count=%d \n" , sum , count );
}
```

**题眼分析** 空（1）是函数 scanf 的第 2 参数，即地址列表，应该为该元素的地址，所以用取地址运算符号&，或者用地址首地址加元素的序号，即 *a*+*i*。空（2）所在语句实现对正数的求和，所以应为 a[i]。

**答案** （1）&a[i]或 a+i （2）a[i]

**【例 3】以下程序执行的结果是_____。**

```
main( )
{   int a[ ]={1,2,3,4},i,j,s=0;j=1;
    for (i=3;i>=0;i--){s=s+a[i]*j; j=j*10;}
    printf("s=%d\n",s);
}
```

**题眼分析** 首先定义了一个一维数组 *a*，具有 4 个元素，并依次赋初值 1,2,3,4，然后执行一个循环。该循环执行 4 次，第 1 次时 *i*=0，*j*=1 相当于执行了"s+=a[3]*1;j=j*10; "，过后 *s* 的

值为 4，*j* 的值为 10；第 2 次时 i=2，j=10 相当于执行了"s+=a[2]*10;j=j*10;"，过后 *s* 的值为 34，*j* 的值为 100；第 3 次时 *i*=1，*j*=100 相当于执行了"s+=a[1]*100;j=j*10;"，过后 *s* 的值为 234，*j* 的值为 1000；第 4 次时 *i*=0，*j*=1000 相当于执行了"s+=a[0]*1000;j=j*10;"，过后 *s* 的值为 1234，*j* 的值为 10000。故本题最后的输出为：*s*=1234。

**答案**　*s*=1234

# 9.2　一维数组和指针

## 9.2.1　考点提炼

**考点 1：定义指向一维数组首元素的指针变量的方法**

一维数组在内存中占连续的存储空间，数组名代表的是数组的首地址。可定义一个指针变量，通过赋值或赋初值的形式，把数组名或数组的第 1 个元素的地址赋值该指针变量，该指针变量就指向了该一维数组的首元素。

以下是两种赋值的方式让指针变量 *p* 指向 *a*：

```
int a[5],*p;p=a;  /*数组名赋值给指针变量名，指针变量指向了数组*/
int a[5],*p;p=&a[0];  /*a[0]的地址赋值给指针变量名，指针变量指向了数组*/
```

**考点 2：利用指向一维数组首元素的指针变量表示数组元素**

在 C 语言中，有一个等式永远成立，即 a[i]无条件等价于*(a+i)，此处 *a* 和 *p* 可以是指针变量名和数组名。

因此，当指针变量 p 指向了一维数组 a 的首元素后，元素 a[i]可表示为下列几种形式：

```
*(a+i)  *(p+i)  a[i]  p[i]
```

数组 a[i]的地址可表示为下列几种形式：

```
a+i   p+i   &a[i]  &p[i]
```

在引用数组元素时，通常把*(a+i)和*(p+i)称为“指针法”；a[i]和 p[i]称为“下标法”。

**考点 3：指向一维数组某元素的指针变量的定义及其使用**

当一个指针变量指向了一维数组的某元素后，可以使用该指针变量来引用该数组的其他数组元素。

例　写出下列程序的运算结果。

程序清单如下：

```
main()
{   int a[10]={1,2,3,4,5,6,7,8,9,10},k,*p;
    p=&a[2]; k=*(p+4)+*p++;
    printf("%d",k);
    }
```

分析

（1）此题的变量 *p* 并不是指向一维数组首元素的指针变量，所以*(p+4)并不是 a[4]。

（2）程序中指针变量 *p* 首先指向了 *a* 数组元素 a[2]，故*(p+4)引用的是数组元素 a[6]，注意此时 *p* 的值并没有变化，它仍然指向数组元素 a[2]。*p++相当于先计算*p（即为 a[2]）的值，然后 *p* 再指向下一个元素，所以 *k*=a[6]＋a[2]=7＋3=10。结果是：6

## 9.2.2 题眼分析

### 一、选择题分析

【例 1】已有定义：int i,a[10],*p; ，则合法的赋值语句是_____。（2004 年 9 月）

（A）p=100; （B）p=a[5]; （C）p=a[2]+2; （D）p=a+2;

题眼分析 *p* 是一个指向整型数据的指针变量，*p* 中应存放的是一个整型数据的地址，因此选项（A）、（B）和（C）都试图给 *p* 赋一个整数，都是错误的。选项（D）中，指针变量 *p* 中存放的是 a[2]的地址，是一条合法的赋值语句。

答案 D

【例 2】设有定义语句

```
int  x[6]={2,4,6,8,5,7},*p=x,i;
```

要求依次输出 x 数组 6 个元素中的值，不能完成此操作的语句是_____。（2004 年 9 月）

（A）for(i=0;i<6;i++) printf("%2d", *(p++) ) ;

（B）for(i=0;i<6;i++) printf("%2d", *(p+i) ) ;

（C）for(i=0;i<6;i++) printf("%2d", *p++ ) ;

（D）for(i=0;i<6;i++) printf("%2d", (*p)++ ) ;

题眼分析 很明显，选项（A）和（B）都是一维数组元素的引用方法，能正确输出数组 *x* 的 6 个元素。由于运算符“ ++ ”的优先级要高于运算符“ * ”，因此选项（C）和选项（A）的本质是一样的。选项（D）中，指向数组的指针 *p* 并没有移动，始终指数组的第 1 个元素 x[0]，是对 x[0]进行依次加 1 后输出，即输出 2 3 4 5 6 7。

答案 D

【例 3】以下函数的功能是：通过键盘输入数据，为数组中的所有元素赋值。

```
#define  N  10
void arrin(int x[N])
{   int i=0;
    while(i<N)  scanf("%d",_______);
}
```

在下划线处应填入的是_____。（2003 年 4 月）

（A）x+i （B）&x[i+1] （C）x+(i++) （D）&x[++i]

题眼分析 不能发现，while 循环中的 *i* 代表的是数组元素的下标，*i* 的初值是 0，每输入一个元素值后，*i* 值应加 1。因此循环中的 scanf 语句应实现两个功能：一是输入元素 *x*[*i*]的值，另一个是使 *i* 的值加 1，注意是先输入 x[i]的值，再使 *i* 的值加 1。输入数组元素值的时候，应该使用数组元素的地址，*x* 代表的是数组的首地址，*x*+*i* 是 *x*[*i*]的地址。

答案 C

【例 4】有以下程序段

```
int  a[10]={1,2,3,4,5,6,7,8,9,10},*p=&a[3],b;
b=p[5];
```

*b* 中的值是_____。（2004 年 4 月）

（A）5　　（B）6　　（C）8　　（D）9

题眼分析　通过赋值语句“*p=&a[3];”使指针变量 *p* 指向了数组元素 a[3]，p[5]等价于 *(p+5)，因此 p[5]就是数组 a[8]，所以 *b*=9。

答案　D

【例 5】有以下程序

```
#include <stdio.h>
main( )
{   int  a[ ]={1,2,3,4,5,6,7,8,9,10,11,12} , *p = a+5, *q=NULL ;
    *q=*(p+5) ;
    printf("%d  %d\n", *p , *q ) ;
}
```

程序运行后的输出结果是_____。（2004 年 9 月题）

（A）运行后报错　　（B）6　6　　（C）6　11　　（D）5　10

题眼分析　在定义时，指针 *p* 被赋了一个空指针，并没有指向任何一个单元，给它赋值将会出现错误。

答案　A

【例 6】有以下程序

```
main()
{   int a[]={1,2,3,4,5,6,7,8,9,0},*p;
    for(p=a;p<a+10;p++) printf("%d,",*p);
}
```

程序运行后的输出结果是_____。（2005 年 4 月）

（A）1,2,3,4,5,6,7,8,9,0,　　（B）2,3,4,5,6,7,8,9,10,1,

（C）0,1,2,3,4,5,6,7,8,9,　　（D）1,1,1,1,1,1,1,1,1,1,

题眼分析　数组名 *a* 代表数组 *a* 的首地址，*a*+*i* 代表数组第 *i* 个元素的地址，因此 *a*+9 是数组 *a* 的最后元素地址，而地址 *a*+10 所存储的数据不是数组元素。在 for 循环中，通过 *p*=*a* 使指针指向数组 *a* 的第 1 个元素，并输出该元素；输出该元素后，通过 *p*++使指针指向第 2 个元素……当指针没超过数组地址范围（即 *p*<*a*+*10*）时继续循环，否则退出循环。综上所述，程序的功能是输入数组 *a* 的所有元素。

答案　A

## 二、填空题分析

【例 1】若有以下定义，则不移动指针 *p*，且通过指针 *p* 引用值为 98 的数组元素的表达式是_____。（2000 年 9 月）

```
int w[10]={23,54,10,33,47,98,72,80,61}, *p=w;
```

题眼分析　本题首先定义了一个具有 10 个元素的整型数组 *w* 并给它赋初值，然后定义一

个指针变量 *p* 并通过赋初值使它指向了该数组的第 1 个元素。数值为 98 的数组元素是 w[5]，用 *p* 表示 w[5]的表达式为“ *(p+5) ”。

**答案** *(p+5)

**【例 2】以下程序的输出结果是_____。（2001 年 4 月）**

```
main()
{   int arr[ ]={30,25,20,15,10,5}, *p=arr;
    p++;
    printf("%d\n",*(p+3));
}
```

**题眼分析** 本题首先定义了一个一维数组 arr 并给它赋初值，再定义了一个指针变量 *p*，通过赋初值让它指向了一维数组 arr 的首元素。执行“ *p*++;”后，*p* 指向元素 arr[1]，此时“ *p*+3 ”应是 arr[4]的地址，“ *(*p*+3) ”就是 arr[4]，故最后输出的是 arr[4]的值为 10。

**答案** 10

# 9.3 函数之间对一维数组和元素的引用

## 9.3.1 考点提炼

### 考点 1：数组元素作实参

在调用函数时，数组元素可以作为实参传送给形参，每个数组元素实际上代表内存中的一个存储单元，故和普通变量一样，对应的形参必须是类型相同的变量。数组元素的值可以传送给形参，在函数中只能对形参进行操作，其结果不影响对应的数组元素。

### 考点 2：数组名做实参

在调用函数时，数组名也可以作为实参传送给形参，但数组名本身是一个地址值，因此对应的形参应当是一个指针变量或数组。数组名做实参时需要注意以下几点：

（1）当使用指针变量作为形参时，其基类型必须与数组类型一致。

（2）当使用数组作为形参时，形参数组可以指定数组长度，也可以不指定长度。不指定长度时，其长度和实参数组的长度一致。

（3）数组名做实参时，实参数组和形参数组共占用一段内存空间。函数调用时，形参数组并没有另外开辟新的存储单元，而是以实参数组的首地址作为形参数组的首地址。这样实参数组的第 1 个元素和形参数组的第 1 个元素占有的是同一个存储单元，实参数组的第 2 个元素和形参数组的第 2 个元素占有的是同一个存储单元，依此类推。因此，形参数组中元素值发生改变也就是实参数组中相对应的元素也发生了改变。

（4）实参可以不用数组名，而使用指向该数组的指针变量来代替。

**考点 3：数组元素地址做实参**

**当使用数组元素地址做实参时，传递的也是一个地址，因此，形参数组中元素值发生改变也会影响到实参数组中元素的值。但与数组名做实参有所不同，形参数组的首地址是传递给它的实参数组元素的地址，而不是实参数组的首地址。实际上，数组名做实参是数组元素地址做实参的一个特例，它是传递第 1 个元素的地址。**

## 9.3.2　题眼分析

### 一、选择题分析

**【例 1】有以下程序**

```
void  swap 1(int  c[])
{   int   t;
    t=c[0]; c[0]= c[1]; c[1]=t;
}
void  swap2(int   c0，int  cl)
{   int   t;
    t=c0;     c0=cl;  cl=t;
}
main()
{   int  a[2]={3,5},b[2]={3,5};
    swapl(a);        swap2(b[0]，b[1]);
    printf("%d%d%d%d\n"，a[0]，a[1]，b[0]，b[1]);
}
```

**其输出结果是_____。（2004 年 4 月）**

（A）5353　　（B）5335　　（C）3535　　（D）3553

**题眼分析　从程序中可以看出，函数** swap1 **是使用数组名作为实参，传送的是数组首地址，实参数和形参数组共用一块内存空间，形参数组中元素值改变就是改变实参数组的元素值，因此数组** *a* **中的第** 1 **元素和第** 2 **元素进行了交换。函数** swap2 **是使用数组元素作为实参，实参向形参传送的是元素的值，形参变量的改变不会影响到实参数组元素值，因此数组** *b* **中的第** 1 **元素和第** 2 **元素没有进行交换。**

**答案**　B

**【例 2】有以下程序**

```
#define N 20
fun(int a[],int n,int m)
{   int i,j;
    for(i=m;i>=n;i--)a[i+1]=a[i];
}
main()
{   int i,a[N]={1,2,3,4,5,6,7,8,9,10};
    fun(a,2,9);
    for(i=0;i<5;i++)printf("%d",a[i]);
```

```
}
```

程序运行后的输出结果是_____。（2005 年 4 月）

（A）10234　　（B）12344　　（C）12334　　（D）12234

**题眼分析**　在 main()函数中定义了一个有 20 个元素的数组 a，并给前 10 个元素赋初值 1～10。执行函数调用 f(a,2,9)，把 *a* 的首地址赋给形参数组 *a*，把 2 和 9 分别赋给形参变量 *n* 和 *m*。在函数 f()中，for 循环执行了 8 次：第 1 次，*i* 值为 9，相当于执行了语句“a[10]=a[9];”，a[10]的值就是 10；第 2 次，*i* 值为 8，相当于执行了语句“a[9]=a[8];”，a[9]的值就是 9，……，最后一次，*i* 的值为 2，相当于执行了语句“a[3]=a[2];” a[3]的值就是 3。a[0]、a[1]、a[2]的值没有改变，仍为 1、2、3，因此前 5 个元素的值为 1、2、3、3、4。

**答案**　C

【例 3】有以下程序

```
void sum(int  *a)
{   a[0]= a[1];  }
main()
{   int  aa[10]={1,2,3,4,5,6,7,8,9,10},i;
    for(i=2; i>=0;i--)  sum(&aa[i]);
    printf("%d\n", aa[0]);
}
```

执行后的输出结果是_____。（2004 年 4 月）

（A）4　　（B）3　　（C）2　　（D）1

**题眼分析**　主函数 main 调用函数 sum 时，使用数组元素地址做实参，因此函数 sum 对形参数组的修改会影响到实参数组的值。循环共运行了 3 次，第 1 次循环实际上是将 a[3]赋值给 a[2]，使 a[2]=4；第 2 次循环实际上是将 a[2]赋值给 a[1]，使 a[1]=4；第 3 次循环实际上是将 a[1]赋值给 a[0]，使 a[0]=4。

**答案**　A

【例 4】有以下程序

```
prt(int *m,int n)
{   int i;
    for(i=0;i<n;i++)m[i]++;
}
main()
{   int a[]={1,2,3,4,5},i;
    prt(a,5);
    for(i=0;i<5;i++)    printf("%d,",a[i]);
}
```

程序运行后的输出结果是_____。（2005 年 4 月）

（A）1,2,3,4,5,　　（B）2,3,4,5,6,　　（C）3,4,5,6,7,　　（D）2,3,4,5,1,

**题眼分析**　该题比较简单，主要是考查数组名做实参时，对应的形参可以是一个数组，也可以是一个指针变量。在本题中，函数 fun 的功能是将数组每个元素加 1。

**答案**　B

**二、填空题分析**

**【例】函数** swap(arr, n)**可完成对** arr **数组从第** 1 **个元素到第** *n* **个元素两两交换。在运行调用函数中的如下语句后，**a[0]**和** a[1]**的值分别为_____。**

```
a[0]=1;a[1]=2;swap(a,2);
```

**题眼分析　数组名作为函数参数传递的是数组的首地址，即实参数组名把实参数组的首地址传给了形参数组名，形参数组名就指向了相应的实参数组，也就是说形参数组和实参数组其实就是同一个数组，对形参数组元素的修改也同样影响到对应的实参数组元素。**

**答案**　2，1

# 9.4　二维数组的定义和元素的引用

## 9.4.1　考点提炼

**考点 1：二维数组的定义**

**当数组元素的下标为两个时，该数组称为二维数组。**

**1．二维数组的定义**

**格式：存储类型　数据类型　数组名[长度** 1**][长度** 2**]；**

**功能：定义一个二维数组，有“长度** 1**×长度** 2**”个元素。其元素的存储类型和数据类型分别由定义中的“存储类型”和“数据类型”指定。**

**说明：**

**（**1**）存储类型、数据类型、数组名和长度的含义和选取方法同一维数组。**

**（**2**）数组元素的各维下标从** 0 **开始，最大下标为“长度–**1**”。**

**2．二维数组的逻辑结构和存储结构**

**二维数组的逻辑结构，可以看成是由若干行，每行由若干列组成。例如有如下数组定义语句：**int a[3][4]**；则其逻辑结构如下：**

```
a[0][0]  a[0][1]  a[0][2]  a[0][3]
a[1][0]  a[1][1]  a[1][2]  a[1][3]
a[2][0]  a[2][1]  a[2][2]  a[2][3]
```

**二维数组存储结构是“按行存放，先行后列”，其在内存中的存放顺序为：**a[0][0]、a[0][1]、a[0][2]、a[0][3]、a[1][0]、a[1][1]、a[1][2]、a[1][3]、a[2][0]、a[2][1]、a[2][2]、a[2][3]。

**假设需要求出数组** *a* **中第** 10 **个元素是哪个元素，只需写出它的逻辑结构，根据存放规律，很容易就知道是** a[2][1]。

**考点 2：二维数组元素的引用**

**二维数组元素的引用方法也有两种，分别是“指针法”和“下标法”。这里只讨论“下标法”。假设定义了一个二维数组：**int a[$N_1$][$N_2$]**，其引用形式为：**

数组名[下标表达式 1][下标表达式 2]

**二维数组元素的引用时注意事项：**

**（1）二维数组各维的下标也是从 0 开始，因此二维数组某维长度为 $N_i$，其下标范围为 0～$N_i$–1。例如有定义：**int a[3][4] ;**那么其元素有** a[0][0]、a[0][1]、a[0][2]、a[0][3]、a[1][0]、a[1][1]、a[1][2]、a[1][3]、a[2][0]、a[2][1]、a[2][2]、a[2][3]。

**（2）下标表达式的值必须是整数，且不得超越数组定义的上、下界。值得注意的是在 C 语言中允许下标越界，也就是说 C 对下标越界不进行检查，**a[3][4] 、a[1][4]**均是可用的，但它们都不是数组元素。**

**（3）引用二维数组元素时，一定要把两个下标分别放在两个括号内。例如，引用上面定义的数组 a 数组元素时，不能写成** a[0,1]、a[i,j]、a[i+k,j+k]。

### 考点 3：二维数组的初始化

**二维数组初始化的方法有以下几种。**

**（1）分行给二维数组所有元素赋初值。**

**例如：**int a[3][4]={{0,1,2,3},{4,5,6,7},{8,9,10,11}};

**初始化后，**a[1][2]**的值是多少？答案为** 6。

**（2）不分行给二维数组所有元素赋初值。**

**例如：**int a[3][4]={0,1,2,3,4,5,6,7,8,9,10,11};

**初始化后，**a[2][1]**的值是多少？答案为** 9。

**注意：如果对所有元素赋初值，其第 1 维的长度可以省去，所以（1）和（2）中的** int a[3][4] **可以写成** a[][4]。

**（3）只对每行或前若干行的前若干个元素赋初值。**

**例如：**int a[3][4]={{0,1},{4},{8,9,10}};

**初始化后，问** a[0][1]**和** a[2][3]**的值是多少，显然答案为 1 和** 0。

**注意：如果给数组中的某些元素赋初值，没有赋初值的元素也有初值，对于数值型数组，其值为** 0，**对于字符型数组，其值为**'\0'。

**（4）若数组的存储类为** auto，**如果不进行初始化，则其元素初值不确定。若数组的存储类型为** static，**若没有进行初始化，其元素均有初值。对于数值类型是** 0，**对于字符型则是**'\0'。

**注意：给二维数组赋初值时，行数不能超过定义的行数，每行的初值个数不能超过定义时的列数。下列的定义是错误的：**

```
int a[2][3]={{1,2,3},{3,4,5},{4,5,6}};        /*行数超过定义的行数*/
int a[3][4]={{1,2,3,4},{1,2,3,4,5}};          /*第 2 行超过了定义时的列数*/
```

### 考点 4：通过赋初值确定二维数组的大小

**在定义一个二维数组时，可以通过赋初值的个数来确定数组的大小，但只可以省略第** 1 **个方括号的常量表达式，而不能省略第** 2 **个括号的常量表达式。**

**当使用行花括号赋初值时，第** 1 **维的大小由赋初值的行数来决定。当省略行花括号时，第** 1 **维的大小按以下规则来决定：**

（1）当初值的个数能被第 2 维的常量表达的值除尽时，所得商数就是第 1 维的大小；

（2）当初值的个数不能被第 2 维的常量表达的值除尽时，则：

第 1 维的大小 = 所得商数+1

### 9.4.2　题眼分析

**一、选择题分析**

【例 1】有以下程序：

```
main()
{   int aa[4][4]={{1,2,3,4},{5,6,7,8},{3,9,10,2},{4,2,9,6}}; int i,s=0;
    for(i=0;i<4;i++)s+=aa[i][1];
    printf("%d\n",s);
}
```

程序运行后的输出结果是_____。（2002 年 9 月）

（A）11　　（B）19　　（C）13　　（D）20

题眼分析　本题首先定义了一个二维数组 aa 并按行赋初值，定义了一个变量 *s* 用于求和。for 循环执行了 4 次，分别把数组元素 aa[0][1]、aa[1][1]、aa[2][1]和 aa[3][1]的值（2、6、9、2）加到变量 *s* 中，*s* 的值为 19。

答案　B

【例 2】有以下程序

```
main()
{   int m[][3]={1,4,7,2,5,8,3,6,9};
    int i,j,k=2;
    for(i=0;i<3;i++)
        {   printf("%d",m[k][i]);  }
}
```

执行后输出结果是_____。（2003 年 4 月）

（A）456　　（B）258　　（C）369　　（D）789

题眼分析　变量 *k* 的初值为 2，循环执行了 3 次，分别输出 m[2][0]、m[2][1]和 m[2][2]的值，为 3、6 和 9。

答案　C

【例 3】以下能正确定义二维数组的是_____。（2004 年 9 月）

（A）int　a[ ][3] ;　　（B）int　a[ ][3]={2*3};

（C）int　a[ ][3]={ } ;　　（D）int　a[2][3]={ {1},{2},{3,4} } ;

题眼分析　在定义一个二维数组时，可以通过赋初值的个数来确定数组的大小，但只可以省略第 1 个方括号的常量表达式，而不能省略第 2 个括号的常量表达式。也就说，第 2 维的大小必须是确定的，而第 1 维大小由赋初值的行数或元素的个数来确定。在本题中，选项（A）没有赋初值，所以必须要指定第 1 维和第 2 维的大小；选项（C）所赋初值为一个空，没办法确定第 1 维的大小；选项（D）中，所赋初值的行数大于定义的行数，也是错误的。选项（B）能够正确定义二维数组，其等价于“int　a[1][3]={6};”。

**答案** B

## 二、填空题分析

**【例 1】以下程序运行后的输出结果是_____。（2004 年 9 月）**

```
main( )
{   int  a[4][4]={{1,2,3,4},{5,6,7,8},{11,12,13,14},{15,16,17,18}};
    int  i=0,j=0,s=0;
    while (i++<4)
    {   if (i= =2 || i= = 4 )  continue ;
        j=0;
        do {    s+=a[i][j];  j++; }while (j<4) ;
    }
    printf("%d\n" , s );
}
```

**题眼分析** 当 *i*=0 时，满足 while 循环的循环条件，并在条件判断完成后，*i* 自增变为 1，由于 *i*=1 不满足 if 语句的条件，执行 do…while 循环，将第 1 行的 4 个元素累加到变量 *s* 之中。再一次进入 while 语句时，还满足 while 循环的循环条件，并在条件判断完成后，*i* 自增变为 2，由于 *i*=2 满足 if 语句的条件，结束了这一趟循环；第 3 趟循时，又将三行的元素累加到变量 *s* 之中；第 4 趟时，满足 if 语句的条件，跳过 do…while 循环；第 5 次执行 while 语句时，退出循环。变量 *s* 的最后结果是第 1 行和第 3 行的 8 个元素的和。

**答案** 92

**【例 2】下面 rotate 函数的功能是:将 n 行 n 列的矩阵 A 转置为 A'，例如**

$$\text{当 A}=\begin{vmatrix}1&2&3&4\\5&6&7&8\\9&10&11&12\\13&14&15&16\end{vmatrix}\qquad \text{则 A'}=\begin{vmatrix}1&5&9&13\\2&6&10&14\\3&7&11&15\\4&8&12&16\end{vmatrix}$$

**请填空。（2004 年 4 月）**

```
#define  N  4
void rotate(int a[][N])
{   int  i,j,t;
    for(i=0;i<N;j++)
        for{j=0; ___(1)___;j++}
        {   t=a[i][j];
            ___(2)___;
            a[j][i]=t;
        }
}
```

**题眼分析** 在矩阵转置时，只需将对角线对称的元素互换就可以了。函数中使用一个两重 for 循环来完成，变量 *i* 用来控制行，变量 *j* 用来控制列。当 *i*=0 时，不需要互换；当 *i*=1 时，a[1][0]与 a[0][1]需要互换；当 *i*=2 时，需要互换 a[2][0]与 a[0][2]、a[2][1]与 a[1][2]需要互换，依此类推，因此内循环的结束条件时 j<i，这就是空（1）所填写的内容。内循环的循环体是完成 a[i][j]与 a[j][i]的交换，因此，空（2）处应填写“a[i][j]= a[j][i]”。

**答案**　（1）j<i　　（2）a[i][j]= a[j][i]

**【例 3】以下程序中，fun 函数的功能是求 3 行 4 列二维数组每行元素中的最大值，请填空。（2005 年 4 月）**

```
void fun(int,int,int(*)[4],int *);
main()
{   int a[3][4]={{12,41,36,28},{19,33,15,27},{3,27,19,1}},b[3],i;
    fun(3,4,a,b);
    for(i=0;i<3;i++)printf("%4d",b[i]);
    printf("\n");
}
void fun(int m,int n,int ar[][4],int *bar)
{   int i,j,x;
    for(i=0;i<m;i++)
    {   x=ar[i][0];
        for(j=0;j<n;j++) if(x<ar[i][j])x=ar[i][j];
        _____=x;
    }
}
```

**题眼分析**　fun 函数的功能是求 3 行 4 列二维数组每行元素中的最大值，并将这些最大值存放在数组 *b*[]中。该函数共有 4 个参数，其中最后一个参数*bar 是一个整数指针，函数调用时，该指针将指向数组 *b* 的首地址，它可以当做数组来使用。该函数中使用一个循环嵌套，外循环用来控制每一行，内循环用来找第 *i* 行的最大值。当找到第 *i* 行最大值后，需要将该值保存在指针 bar 所指的数组中，因此空格处应填写 bar[i]。当然也可以直接使用指针形式，但要将指针后移，每完成一行，指针 bar 加 1，即*(bar++)。

**答案**　bar[i]或*(bar++)

# 9.5　二维数组和指针

## 9.5.1　考点提炼

### 考点 1：用指针变量指向二维数组首元素的方法

二维数组在内存中是按行连续存放的，当用指针变量指向二维数组的首地址后，利用该指针变量就可以处理二维数组的任一个元素。

可采用赋初值和赋值两种方式使指针变量指向二维数组的首元素。

赋初值的方法有两种，格式如下：

格式 1：数据类型符　*指针变量=&二维数组名[0][0];

格式 2：数据类型符　*指针变量=*二维数组名;

赋值的方式也有两种，格式如下：

格式 1：指针变量=&二维数组名[0][0];

**格式 2：指针变量=*二维数组名;**

**考点 2：用指向二维数组首元素的指针变量引用二维数组元素**

**当 *p* 指向二维数组的首元素后，*p*+1 将指向元素第 2 个元素，*p*+2 将指向第 3 个元素……依此类推。**

**例如，有定义：**

```
int a[3][4],*p=&a[0][0];/*p指向a数组的首元素*/
```

**此时有以下结论：**

*a*[*i*][*j*]**的地址为** *p*+*i**4+*j*，a[*i*][*j*]**可表示为** *(*p*+*i**4+*j*)

**更为一般的结论如下。假设指针变量 *p* 已经指向 *M* 行 *N* 列的数组 A 的首元素（**0≤*i*<*M*,0≤*j*<*N*）**，则**

A[i][j]**的地址为：***p*+*i***N*+*j*

A[i][j]**可表示为：** *(*p*+*i***N*+*j*)

**考点 3：指向二维数组某元素的指针变量的定义及其使用**

**指针变量指向二维数组某元素的方法有两种，赋初值与赋值。**

**赋初值：**

**数据类型符 *指针变量名=&数组名[下标 1][下标 2]**

**赋值方式：**

**指针变量=&数组名[下标 1][下标 2]**

**指针变量指向二维数组某元素后，引用该数组元素的方法是：*指针变量。**

**例 写出下列程序的执行结果**

```
main()
{  int a[3][4]={1,2,3,4,5,6,7,8,9,10,11,12},*p=&a[1][3],k;
   k=*p+*(p+2);  printf("%d",k);
}
```

**分析**

**（1）此题首先定义了一个 3 行 4 列的数组 *a*，然后定义了一个指针变量 *p* 让它指向了数组元素 *a*[1][2]。**

**（2）执行语句“** *k*=**p*+*(*p*+2); **”，此处**p* 代表的就是数组元素 *a*[1][2]，其值是 8，*(*p*+2) 是其后的第 2 个元素即 *a*[2][0]，其值是 10。所以最后输出的 *k* 值为 18。**

**考点 4：指针数组**

**指针数组也是一种数组，所有有关数组的概念都适用于它。但它与普通的数组又有区别，它的数组元素是指针类型的，只能用来存放地址值。其定义形式如下：**

**格式：存储类型 数据类型 *指针数组名[长度]={初值列表};**

**作用：定义一个指针数组，数组名为“指针数组名”，它的元素个数由“长度”指定，它的每个元素是一个指向“数据类型”的指针，“存储类型”决定了数组的存储类型。**

**说明：**

（1）赋初值和普通数组赋初值基本上是一样的，不同的是初值必须是地址值，通常的形式有“ &变量名 ”、“ &数组元素 ”、“ 数组名 ”，对应的变量和数组必须在前面已经定义。

（2）注意指针数组定义与指向数组的指针变量定义的区别，不能写成“ (*数组名)[长度] ”，否则“ 数组名 ”就成了一个指向数组的指针变量。

### 考点 5：指向二维数组首行的指针变量的定义及其使用

1．二维数组的模型

可以把二维数组看成一个一维数组，二维数组的每一行是其一个元素，每一行又可以看成一个一维数组。假设有一个 3 行 4 列的二维数组，可以把它看成是一个由 3 个元素组成的一维数组 *a*。它的每一个元素都是一维数组，其数组名分别是 *a*[0]、*a*[1]、*a*[2],由于一维数组名代表的是数组的首地址，所以 *a*、*a*+1、*a*+2 分别是 *a*[0]、*a*[1]和 *a*[2]的地址；而 *a*[0]、*a*[1]和 *a*[2]代表的分别是 *a*[0][0]、*a*[1][0]和 *a*[2][0]的地址，如图 9-1 所示。那么用 *a* 表示的各元素地址如图 9-2 所示。

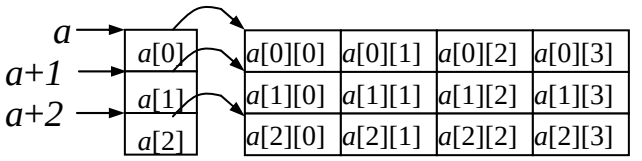



图 9-1　二维数组的模型图

| | | | |
|---|---|---|---|
| *a*[0]+0 | *a*[0]+1 | *a*[0]+2 | *a*[0]+3 |
| *a*[1]+0 | *a*[1]+1 | *a*[1]+2 | *a*[1]+3 |
| *a*[2]+0 | *a*[2]+1 | *a*[2]+2 | *a*[2]+3 |

图 9-2　用数组名表示的各元素地址

可见元素 *a*[*i*][*j*]的地址可表示为：*a*[*i*]+*j*。由于 *a*[*i*]无条件地等价于*(*a*+*i*)，*a*[*i*][*j*]的地址也可以表示为：*(*a*+*i*)+*j*。同样数组元素 *a*[*i*][*j*]也可以表示为：*(*a*[*i*]+*j*)和*(*(*a*+*i*)+*j*)。

2．指向一维数组的指针变量的定义。

格式：数据类型符　(*指针变量)[*m*]

功能：定义一个指针变量，指向由 *m* 个“ 数据类型符 ”指定类型的元素组成的一维数组。

注意：定义时，圆括号不能丢。因为如果写成“ *指针变量 ”，由于“ [ ] ”运算符的优先级高于“ * ”运算符，指针变量先和“ [ ] ”结合，定义的是一个数组，然后再和“ * ”结合，结果定义成指针型的数组，即数组的每个元素是指针。

3．用指向一维数组的指针指向二维数组的首行。

可以先定义一个指向一维数组的指针变量，它指向的一维数组的元素个数与二维数组的列数一致，然后可以通过赋初值或赋值的方式来使指针变量指向二维数组的首行，形式如下：

赋初值：数据类型符　(*指针变量)[*m*]=二维数组名

赋值：指针变量=二维数组名

例如：

```
int  a[3][4],(*p)[4]=a;  /*赋初值使指针变量p指向a数组的首行*/
int  a[3][4],(*p)[4];p=a;  /*赋值使指针变量p指向a数组的首行*/
```

4．用指向二维数组首行的指针变量表示数组元素及其地址。

让指向数组的指针变量指向二维数组 *a* 的首行后，数组元素 *a*[*i*][*j*]的地址可表示为：

```
&a[i][j]     a[i]+j    *(a+i)+j   *(p+i)+j  p[i]+j
```

数组元素 *a*[*i*][*j*]可表示为：

```
a[i][j]   *(a[i]+j)  *(*(a+i)+j)  *(*(p+i)+j)  *(p[i]+j)
```

**考点 6：指向二维数组某行的指针变量的定义及其使用**

**可以用指向一维数组的指针变量指向二维数组的某一行，然后就可以使用该指针变量来引用二维数组的数组元素。用指向一维数组的指针变量指向二维数组的某一行的方法有两种，一种是赋初值，另一种是赋值。假使数组定义语句为“** int *a*[3][4]; **”，则让指针变量 *p* 指向它的第** 2 **行的方法如下：**

**赋初值方式：**int (**p*)[4]=*a*+1;　　**赋值方式：**int (**p*)[4];*p*=*a*+1;

## 9.5.2　题眼分析

**【例** 1**】有以下程序**

```
main()
{   int a[3][3],*p,i;
    p=&a[0][0];
    for(i=0;i<9;i++)p[i]=i;
    for(i=0;i<3;i++)printf("%d",a[1][i]);
}
```

**程序运行后的输出结果是_____。（**2005 **年** 4 **月）**

**（**A**）**0　1　2　　**（**B**）**1　2　3　　**（**C**）**2　3　4　　**（**D**）**3　4　5

**题眼分析　通过赋值语句“** p=&a[0][0]; **”使指针 *p* 指向二维数组的首地址，因此** p[0]**就代表** a[0][0]**；**p[1]**就代表** a[0][1]**；**p[2]**就代表** a[0][2]**；**p[3]**就代表** a[1][0]**……**p[8]**就代表** a[2][2]**。在** for **循环中，分别把** 0**，**1**，**2**…**8 **赋值给** a[0][0]**，**a[0][1]**，**a[0][2]**……**a[2][2]**。**

**答案**　D

**【例** 2**】设有以下定义和语句**

```
int  a[3][2]={1,2,3,4,5},*p[3] ;
p[0]=a[1] ;
```

**则***(p[0]+1)**所代表的数组元素是_____。（**2004 **年** 9 **月）**

**（**A**）**a[0][1]　　**（**B**）**a[1][0]　　**（**C**）**a[1][1]　　**（**D**）**a[1][2]

**题眼分析　在说明符***p[3]**中，方括号**[]**的优先级高于星号*****，因此 *p* 首先与**[]**结合，构成** p[3]**，说明 *p* 是一个数组名，系统将为它开辟** 3 **个连续的存储单元，在它前面的*****号则说明数组 *p* 是指针类型，它的每一个元素都是基类型** int **的指针。通过赋值语句“** p[0]=a[1] ; **”，使** p[0] **指向数组 *a* 的第** 2 **行第** 1 **列，即** a[1][0]**，因此，**p[0]+1 **指向数组 *a* 的第** 2 **行第** 2 **列，即** a[1][1]**。**

**答案**　C

**【例** 3**】若有以下说明和语句**

```
int  c[4][5],(*p)[5] ;
p=c ;
```

**能够正确引用** c **数组元素的是_____。（**2004 **年** 9 **月）**

A）p+1　　B）*(p+3)　　C）*(p+1)+3　　D）*(p[0]+2)

**题眼分析　在说明符（***p**）**[5]**中，由于一对圆括号（）的存在，所以*****号首先与** p **结合，说明 *p* 是一个指针变量，然后再与说明符**[5]**结合，说明指针变量 *p* 的基类型是一个包含** 5 **个**

int 元素的一维数组。通过赋值语句“ p=c; ”，使 *p* 指向二维数组 *c* 的首地址。选项（A）是二维数组的第 1 行的地址，选项（B）是二维数组的第 3 行的地址，选项（C）是二维数组元素 c[1][3]的地址，选项（D）是数组元素 c[0][2]的值。

**答案**　D

**【例 4】有以下程序**

```
main()
{  int a[][3]={{1,2,3},{4,5,0}},(*pa)[3],i;
    pa=a;
    for(i=0;i<3;i++)
      if(i<2)  pa[1][i]=pa[1][i]-1;
      else   pa[1][i]=1;
    printf("%d\n",a[0][1]+a[1][1]+a[1][2]);
}
```

执行后输出结果是_____。（2003 年 4 月）

（A）7　　（B）6　　（C）8　　（D）无确定值

**题眼分析**　本题定义了一个指向由 3 个元素组成的一维数组的指针变量 pa，通过赋值让它指向具有两行三列的数组 a 的首行，此时用指针变量 pa 表示数组元素 a[i][j]的形式是 pa[i][j]。for 循环执行了 3 次：第 1 次 *i* 值为 0，执行 pa[1][0]=pa[1][0]–1，执行后 a[1][0]的值变为 3；第 2 次 *i* 值为 1，执行 pa[1][1]=pa[1][1]–1，执行后 a[1][1]的值变为 4；第 3 次 i 值为 2，执行 pa[1][2]=1，执行后 a[1][2]的值变为 1。输出“ a[0][1]+a[1][1]+a[1][2] ”的值为：2+4+1=7。

**答案**　A

# 9.6　二维数组名和指针数组作为实参

## 9.6.1　考点提炼

**考点 1：二维数组名作为实参**

当二维数组名作为实参时，对应的形参必须是一个行指针变量。

**考点 2：指针数组作为实参**

当指针数组作为实参时，对应的形参应当是一个指向指针的指针。

## 9.6.2　题眼分析

### 一、选择题分析

**【例 1】以下程序的输出结果是_____。（1999 年 9 月）**

```
#include "stdio.h"
int a[3][3]={1,2,3,4,5,6,7,8,9},*p;
```

```
main( )
{   p=(int*)malloc(sizeof(int));
    f(p,a);
    printf("%d \n",*p);
}
f(int *s, int p[][3])
{   *s=p[1][1]; }
```

（A）1　　（B）4　　（C）7　　（D）5

**题眼分析**　本题首先定义了一个全局数组 *a* 并赋初值，定义了一个全局的指针变量 *p*。在主函数中首先给通过 malloc 函数申请分配了能够存放一个整型数的存储区并把首地址赋给了指针变量 *p*，然后调用函数 *f* 把 *p* 的值传给了形参指针变量 *s*，*s* 就和实参 *p* 指向同一个存储区；把数组 *a* 的首地址传给形参数组名 *p*，则形参数组 *p* 与实参数组 *a* 就是同一个数组。在函数 *f* 中执行语句“*s=*p*[1][1];”，由于 *s* 数组和 *a* 数组就是同一个数组，所以也就是把 *a*[1][1]的值 5 赋值给了 s 指向的存储区。故函数调用返回后，输出的*p 的值为 5。

**答案**　D

**【例 2】有以下程序**

```
int f(int  b[][4])
{   int  i,j,s=0;
    for(j=0;j<4;j++)
    {   i=j;
        if(i>2)i=3-j;
        s+=b[i][j];
    }
    return s;
}
main()
{   int   a[4][4]={{1,2,3,4 },{0,2,4,6 }, {3,6,9,12 }, {3,2,1,0}};
    printf("%d\n", f(a));
}
```

**执行后的输出结果是_____。（2004 年 4 月）**

（A）l2　　（B）11　　（C）18　　（D）16

**题眼分析**　主函数 main 向函数 *f* 传送的是数组 *a* 的首地址，因此形参数组 *b* 和实参数 *a* 占用同一段存储单元。在函数 *f* 中，for 循环执行过程如下：

| | *j* | *i* | *i=3–j* | *s* |
|---|---|---|---|---|
| ① | 0 | 0 | **不执行** | **累加** b[0][0]，**结果为** 1 |
| ② | 1 | 1 | **不执行** | **累加** b[1][1]，**结果为** 3 |
| ③ | 2 | 2 | **不执行** | **累加** b[2][2]，**结果为** 12 |
| ④ | 3 | 3 | 0 | **累加** b[0][3]，**结果为** 16 |
| ⑤ | 4 | **结束循环** | | |

**答案**　D

## 二、填空题分析

**【例 1】设在主函数中有以下定义和函数调用语句，且** fun()**函数为** void **类型；请写出** fun() **函数的首部_____，要求形参名为** *b*。**（**2000 **年** 9 **月）**

```
main()
{   double s[10][22];
    int n;
    ……
    fun(s);
    ……
}
```

**题眼分析　函数头的定义形式是“存储类型说明符　数据类型说明符　函数名([形式参数说明列表]) ”，此题没有涉及到函数的存储类别。如果函数的参数是数组，其第** 1 **维的长度可以缺省，因此本题的函数的首部可以写成：**void fun(b[][22])。

**答案**　void fun(b[][22])

**【例** 2**】以下程序中，函数** SumColumMin **的功能是：求出** M **行** N **列二维数组每列元素中的最小值，并计算它们的和值。和值通过形参传回主函数输出。请填空。（**2004 **年** 9 **月）**

```
#define M   2
#define N   4
void  SumColumMin( int  a[M][N] , int  *sum )
{   int  i , j , k , s=0 ;
    for (i=0 ; i<N ; i++ )
    {   k=0 ;
        for (j=1 ; j<M ; j++ )
            if (a[k][i]>a[j][i]) k=j ;
        s+=  (1)  ;
    }
     (2)  = s ;
}
main( )
{   int  x[M][N]={3,2,5,1,4,1,8,3}, s ;
    SumColumMin(  (3)  );
    printf("%d \n" , s );
}
```

**题眼分析　通过对题意的分析不难看出，在函数** SumColumMin **中，变量** *i* **的作用是控制列下标，变量** *j* **的作用是控制行下标，变量** *k* **的作用是记录每一列最小数的行数。当内循环结束时，**a[k][i]**是第** *i* **列的最小数，要将它累加到变量** *s* **中，因此，空（**1**）处应填写“**a[k][i]**”。当整个两重循环都结束时，变量** *s* **就存放着所有列的最小元素和，要通过形参** sum **返回，形参** sum **是一个指向整型变量的指针变量，因此，空（**2**）处应填写“***sum**”。在函数** main **中，需要调用函数** SumColumMin**，空（**3**）需要写出调用时实参表。根据函数** SumColumMin **的形参表，第** 1 **个参数应当是一个数组首地址，第** 2 **参数应当是返回值存放的地址，因此，空（**3**）处应填写“** *x,&s* **”或其等价形式。**

**答案** （1）a[k][i] （2）*sum （3）x,&s

**【例 3】函数 YangHui 的功能是把杨辉三角形的数据赋给二维数组的下半三角，形式如下**

```
1
1   1
1   2   1
1   3   3   1
1   4   6   4   1
```

**其构成规律是：**

**（1）第 0 列元素和主对角线元素均为 1；**

**（2）其余元素为其左上方和正上方元素之和；**

**（3）数据的个数每行递增 1。**

**请将程序补充完整。（2003 年 4 月）**

```
#define  N  6
void YangHui(int x[N][N])
{   int i,j; x[0][0]=1;
    for(i=1;i<N;i++)
    {   x[i][0]=  (1)  =1;
        for(j=1;j<i;j++)
            x[i][j]=  (2)  ;
    }
}
```

**题眼分析** 根据杨辉三角的规律可知，每行的首列和对角线的值为 1，故第 1 空应填对角线元素 x[i][i]。其他元素为其左上方和正上方（即上一行的前一列和当前列）元素之和，即：x[i][j]=x[i−1][j−1]+x[i−1][j]。

**答案** （1）x[i][i] （2）x[i−1][j−1]+x[i−1][j]

# 9.7 单元训练及参考答案

## 9.7.1 单元训练

### 一、选择题

1．以下关于数组的描述正确的是_____。

（A）数组的大小是固定的，但可以有不同的类型的数组元素

（B）数组的大小是可变的，但所有数组元素的类型必须相同

（C）数组的大小是固定的，所有数组元素的类型必须相同

（D）数组的大小是可变的，可以有不同的类型的数组元素

2．以下一维数组 *a* 的正确定义是_____。

（A）int a{18} （B）int n=20,a[n];

（C）int n=10;n=20; int a[n];　　（D）int a[]={0};

3．**以下对一维数组 *a* 的正确说明是_____。**

（A）int a(10);　　（B）int n=10,a[n];

（C）int n; scanf("%d",&n);int a[n]　　（D）#define SIZE 10 /int a[SIZE];

4．**以下不正确的定义语句是_____。**

（A）double　a[5]={2.0,4.0,6.0,8.0,10.0}　　（B）int　b[5]={0,1,3,5,7,9}

（C）char　c[]={'1', '2', '3', '4', '5'}　　（D）char　d[]={'\x10', '\xa', '\x8'}

5．**假定** int **类型变量占用两个字节，其有定义：**int x[10]={0,2,4};**，则数组 *x* 在内存中所占字节数是_____。**

（A）3　　（B）6　　（C）10　　（D）20

6．**在定义** int a[10];**之后，对 *a* 元素的引用正确的是_____。**

（A）a[10]　　（B）a[6,3]　　（C）a(6)　　（D）a[10–10]

7．**以下程序的输出结果是_____。**

```
main()
{   int i, a[10];
    for(i=9;i>=0;i--) a[i]=10-i;
    printf("%d%d%d",a[2],a[5],a[8]);
}
```

（A）258　　（B）741　　（C）852　　（D）369

8．**以下程序运行后，输出结果是_____。**

（A）1000　　（B）10010　　（C）00110　　（D）10100

```
main()
{   int y=18,i=0,j,a[8];
    do {
        a[i]=y%2;  i++;  y=y/2;
    } while(y>=1);
    for(j=i-1;j>=0;j--)
        printf(%d",a[j]);
    printf("\n");
}
```

9．**下列程序的执行结果是_____。**

```
main()
{   int a[]={1,2,3,4,5},i;
    for(i=1;i<5;i++) printf("%1d",a[i]-a[i-1]);
}
```

（A）11111　　（B）1111　　（C）111　　（D）222

10．**有以下程序**

```
main( )
{   int  x[ ]={1,3,5,7,2,4,6,0},i,j,k ;
    for (i=0; i< 3 ; i++ )
        for (j=2; j>= i ; j-- )
            if (x[j+1]>x[j]) { k=x[j];  x[j]=x[j+1];  x[j+1]=k;  }
```

```
    for (i=0; i< 3 ; i++ )
        for (j=4; j<7-i ; j++ )
            if (x[j]>x[j+1]) { k=x[j];  x[j]=x[j+1];  x[j+1]=k;  }
    for (i=0; i< 8 ; i++ )  printf("%d", x[i] ) ;
    printf("\n") ;
}
```

**程序运行后的输出结果是_____。**

（A）75310246　　（B）01234567　　（C）76310462　　（D）13570246

11．**有如下说明**

```
int a[10]={1,2,3,4,5,6,7,8,9,10},*p=a;
```

**则数值为 9 的表达式是_____。**

（A）*p+9　　（B）*(p+8)　　（C）*p+=9　　（D）p+8

12．**若已定义：**

```
int a[ ]={0,1,2,3,4,5,6,7,8,9}, *p=a,i;
```

**其中 0≤*i*≤9, 则对 *a* 数组元素不正确的引用是_____。**

（A）a[p–a]　　（B）*(&a[i])　　（C）p[i]　　（D）a[10]

13．**下列程序的输出结果是_____。**

（A）**非法**　　（B）a[4]**的地址**　　（C）5　　（D）3

```
main()
{   char a[10]={9,8,7,6,5,4,3,2,1,0},*p=a+5;
    printf("%d",*--p);
}
```

14．**下面程序的输出是_____。**

（A）3　　（B）4　　（C）1　　（D）2

```
main()
{   int a[10]={1,2,3,4,5,6,7,8,9,10},*p=a;
    printf("%d\n",*(p+2));
}
```

15．**下列程序的输出结果是_____。**

```
main()
{   int a[5]={2,4,6,8,10},*p,**k;
    p=a; k=&p;
    printf("%d",*(p++)); printf("%d\n",**k);
}
```

（A）24　　（B）46　　（C）26　　（D）36

16．**执行以下程序后，*y* 的值是_____。**

```
main()
{   int a[]={2,4,6,8,10};
    int y=1,x,*p;  p=&a[1];
    for(x=0;x<3;x++) y += * (p + x);
    printf("%d\n",y); }
```

（A）17　　（B）18　　（C）19　　（D）20

17．**下面程序输出数组中的最大值，由 *s* 指针指向该元素。**

```
main()
{   int a[10]={6,7,2,9,1,10,5,8,4,3,},*p,*s;
    for(p=a, s=a; p-a<10; p++)
        if(_______)s=p;
    printf("The max:%d",*s):
}
```

**则在 if 语句中的判断表达式应该是_____。**

（A）p>s　　（B）*p>*s　　（C）a[p]>a[s]　　（D）p–a>p–s

18．**设有如下定义：**

```
int arr[]={6,7,8,9,10};
int *ptr;
```

**则下列程序段的输出结果为_____。**

```
ptr=arr;  *(ptr+2)+=2;
printf ("%d,%d\n",*ptr,*(ptr+2));
```

（A）8,10　　（B）6,8　　（C）7,9　　（D）6,10

19．**要求函数的功能是在一维数组中查找值，若找到则返回所在的下标值，否则返回 0；数组放在 *a* 中。不能正确执行的函数是_____。**

（A）
```
funa(int *a,int n,int x )
{   *a=x;
    while(a[n]!=x)n––;
    return n;
}
```

（B）
```
funb(int *a,int n,int x)
{   int k;
    for(k=1;k<=n;k++)
        if(a[k]==x)return k;
    return 0; }
```

（C）
```
func(int a[],int n,int x )
{   int *k;
    a[0]=x;k=a+n;
    while(*k!=x) k––;
    return k–n;
}
```

（D）
```
fund(int a[],int n,int x )
{   int k=0;
    do
        k++;
    while((k<n+1)&&(a[k]!=x));
    if(a[k]==x)return k; else return 0; }
```

20．**以下能正确定义数组并正确赋初值的语句是_____。**

（A）int N=5,b[N][N];　　（B）int a[1][2]={{1},{3}};

（C）int c[2][]={{1,2},{3,4}};　　（D）int d[3][2]={{1,2},{34}};

21．**以下不能正确定义二维数组的选项是_____。**

（A）int a[2][2]={{1},{2}};　　（B）int a[][2]={1,2,3,4};

（C）int a[2][2]={{1},2,3};　　（D）int a[2][]={{1,2},{3,4}};

22．**有定义语句“** int a[ ][3]={1,2,3,4,5,6}; **”，则** a[1][0]**的值是_____。**

（A）4　　（B）1　　（C）2　　（D）5

23．**在定义“** int a[5][4]; **”之后，对 *a* 数组的数组元素的正确引用是_____。**

（A）a(1,2)　　（B）a[5][0]　　（C）a[0][0]　　（D）a[0,0]

24．**在定义“** int a[5][6]; **”后，第** 10 **个元素是_____。**

（A）a[2][5]　　（B）a[2][4];　　（C）a[1][3];　　（D）a[1][5];

25．若二维数组 b 有 m 列，则在 b[i][j]前的元素个数为_____。

（A）j*m+i　　（B）i*m+j　　（C）i*m+j–i　　（D）i*m+j+1

26．已有说明：int a[3][4];。则对 a 数据元素的非法引用是_____。

（A）a[0][2*1]　　（B）a[1][3]　　（C）a[4–2][0]　　（D）a[0][4]

27．若有定义：int *p[3];，则以下叙述中正确的是_____。

（A）定义了一个基类型为 int 的指针变量 *p*，该变量具有 3 个指针

（B）定义了一个指针数组 *p*，该数组含有 3 个元素，每个元素都是基类型为 int 的指针

（C）定义了一个名为*p 的整型数组，该数组含有 3 个 int 类型元素

（D）定义了一个可指向一维数组的指针变量 *p*，所指一维数组应具有 3 个 int 类型元素

28．若有以下的说明和语句：

```
main()
{   int t[3][2], *pt[3],k;
    for(k=0; k<3;k++)pt[k]=t[k];   }
```

则以下选项中能正确表示 t 数组元素地址的表达式是_____。

（A）&t[3][2]　　（B）*pt[0]　　（C）*(pt+1)　　（D）&pt[2]

29．若有以下定义和语句：

```
    int s[4][5],(*ps)[5];ps=s;
```

则对 s 数组元素的正确引用形式是_____。

（A）ps+1　　（B）*(ps+3)　　（C）ps[0][2]　　（D）*(ps+1)+3

30．下面程序的输出是_____。

（A）60　　（B）68　　（C）99　　（D）108

```
main()
{   int a[3][4]={ 1,3,5,7,9,11,13,15,17,19,21,23};
    int ( *p)[4]=a,i,j,k=0;
    for(i=0; i<3; i++)
        for(j=0; j<2; j++)k=k+*( *(p+i)+j);
    printf("%d\n",k);
}
```

31．若有以下说明：

```
    int w[3][4]={{0,1},{2,4},{5,8}};
    int(*p)[4]=w;
```

则数值为 4 的表达式是_____。

（A）*w[1]+1　　（B）p++,*(p+1)　　（C）w[2][2]　　（D）p[1][1]

32．若有以下定义和语句：

```
    int w[2][3],(*pw)[3]; pw=w;
```

则对 w 数组元素非法引用是_____。

（A）*(w[0]+2)　　（B）*(pw+1)[2]　　（C）pw[0][0]　　（D）*(pw[1]+2)

33．执行以下程序段后，*m* 的值为_____。

```
int a[2][3]={ {1,2,3},{4,5,6} };
int m,*p;
p=&a[0][0];  m=(*p)*(*(p+2))*(*(p+4));
```

（A）15　　（B）14　　（C）13　　（D）12

## 二、填空题

1．若有定义 int a[3][4]={{1,2,3,4},{'a', 'b', 'c', 'd'},{4,6,8,10}};，则初始化后，a[1][2]得到的初值是（1），a[2][3]得到的初值是（2）。

2．在 C 语言中，二维数组在内存中的存放顺序是（3）。

3．若有定义：double x[3][5]，其最后一个元素是（4）。

4．若有定义：int a[3][4]={{1,2},{0},{4,6,8,10}}。则初始化后，a[1][2]得到的初值是（5），a[2][1]得到的初值是（6）。

5．以下程序运行后的输出结果是（7）。

```
main( )
{   int  p[7]={11,13,14,15,16,17,18};
    int  i=0, j=0;
    while (i<7 && p[i]%2= = 1 )  j+= p[i++];
    printf("%d \n" , j );
}
```

6．下面程序通过函数 average()计算数组中各元素的平均值，请填空。

```
float average(int *pa,int n)
{   int i;
    float avg=0.0;
    for(i=0;i<n;i++) avg=avg+ (8) ;
    avg= (9) ;
    return avg;
}
main()
{   int i,a[5]={2,4,6,8,10};
    float mean;
    mean=average(a,5); printf("mean=%f\n",mean);
}
```

7．下述程序的功能是：从键盘中输入 *n* 个整数，使用冒泡法对这 *n* 个整数从小到大进行排序并输出。请在横线上填上相应的内容。

```
main()
{   int i,j,temp,a[100],n;
    scanf("%d",&n);
    for (i=0;i<n;i++)  scanf("%d",&a[i]);
    for (i=1;i<n;i++)
        for(j=0;j< (10) ;j++)
            if ( (11) ) { temp=a[j];a[j]=a[j+1];a[j+1]=temp;}
    for (i=0;i<n;i++) printf("%d",a[i]);
    printf("\n");
}
```

8．下面程序的输出是（12）。

```
main()
```

```
{   int a[]={ 2,4,6}, *prt=&a[0], x=8,y,z;
    for(y=0; y<3; y++)
        z=(*(prt+y)<x)? *(prt+y):x;
    printf("%d\n", z);
}
```

**9．以下函数用来在 *w* 数组中插入 *x*，*w* 数组中的数已按由小到大顺序存放，*n* 指存储单元中存放数组中数据的个数。插入后数组中的数仍有序。请填空。**

```
void fun (char *w,char x,int *n)
{   int i,p;
    p=0;  w[*n]=x;
    while (x>w[p]) __(13)__;
    for(i=*n;i>p;i--)w[i]= __(14)__;
    w[p]=x; ++ *n;
}
```

## 9.7.2 参考答案

### 一、选择题

| | | | | |
|---|---|---|---|---|
| 1．C | 2．D | 3．D | 4．B | 5．D |
| 6．D | 7．C | 8．B | 9．B | 10．A |
| 11．B | 12．D | 13．C | 14．A | 15．A |
| 16．C | 17．B | 18．D | 19．C | 20．D |
| 21．D | 22．A | 23．C | 24．C | 25．B |
| 26．D | 27．B | 28．C | 29．C | 30．A |
| 31．D | 32．B | 33．A | | |

### 二、填空题

（1）99　　（2）10

（3）**按行存放**　　（4）x[2][4]

（5）0　　（6）6

（7）24　　（8）*(pa+i)**或** pa[i]

（9）avg/n　　（10）n–i

（11）a[j]>a[j+1]　　（12）6

（13）p++**或**++p **或** p=p+1　　（14）w[i–1]

# 第 10 章　字符串

## 10.1　用一个一维字符数组存放字符串

### 10.1.1　考点提炼

**📖考点 1：字符串基本概念**

**1．字符串定义**

**C 语言本身并没有设置一种类型来定义字符串变量，字符串的存储完全依赖于字符数组。字符串是借助字符型一维数组来存放的，以'\0'作为字符串结束标志。'\0'是一个转义字符，称为“空值”，其 ASCII 码值为 0。'\0'作为标志占用存储空间，但不计入字符串的实际长度。**

**2．字符串常量**

**C 语言中，无字符串数据类型，但允许使用字符串常量。C 语言中，字符串常量给出的是地址值。即在 C 语言中，字符串常量被隐含处理成一个以'\0'结尾的无名字符型一维数组。**

```
char *sp,s[10];
s="hello!"; /*该赋值不合法*/
sp="hello!"; /*该赋值合法*/
```

**3．字符数组与字符串的区别**

**（1）字符数组的每个元素中可存放一个字符，但它并不限定最后一个字符应该是什么。**

**（2）在字符数组中的有效字符后面加上'\0' 把这种一维字符型数组“看做”字符串变量。**

**（3）字符串是字符数组的一种具体应用。**

**📖考点 2：通过赋初值的方式给一维字符数组赋字符串**

**1．用给一般数组赋初值的相同方法给一维字符数组赋初值**

**给一维字符数组赋初值的方式是把所赋初值依次放在一对花括号中，如：**

```
char c[10]={'s', 't','r','i','n','g','\0 '};
```

**所赋初值的个数少于数组的元素个数时，系统将自动在其后面的元素中加'\0'。**

**当用赋初值方式来定义字符数组大小时，这时定义应写成：**

```
char c1[ ]={'c', ' ','p','r','o','g','r','a','m', '\0 '};
```

**在此，定义了一个包括 9 个字符的字符串 c1。而以下定义：**

```
char c2[ ]={'c', ' ','p','r','o','g','r','a','m'};
```

**只是定义了一个包括 9 个字符的字符数组 *c2*。这时不能认为 *c2* 存放了字符串，不能把它当做字符来使用。**

**2．在赋初值时直接赋字符串常量**

**可以用字符串常量给一维字符数组赋初值。例如**

```
char c[12 ]={"C program"};
```

**或去掉{}写为：**

```
char c[12]="C program";
```

**或**

```
char c[]="C program";
```

**考点 3：在程序执行过程中给一维字符数组赋字符串**

**1．不可以用赋值语句给字符数组整体赋一串字符**

**当做字符串变量使用的字符数组，不能由赋值语句直接赋字符串常量。例如：**

```
char mark[10];
mark="C program";   /*赋值不合法*/
```

**这是因为数组名 mark 是一个地址常量，不能被重新赋值。同理以下赋值方式也是错误的：**

```
char str1[10]="computer",str2[10];
str2=str1;          /*赋值不合法*/
```

**2．给数组元素逐个赋字符值，最后要人为加入串结束标志**

**在程序执行过程中，可以通过逐个给数组元素赋值的方式，达到给一维数组赋字符串的目的。例如：**

```
char mark[10];
int i;
for(i=0;i<9;i++)
    scanf("%c",&mark[i]);
mark[i]='\0'; /*人为加入串结束标志*/
```

## 10.1.2 题眼分析

### 一、选择题分析

**【例 1】有以下程序**

```
main()
{   char   a[]="abcdefg"，b[l0]="abcdefg";
    printf("%d   %d\n"，sizeof(a),sizeof(b));
}
```

**执行后输出结果是_____。（2004 年 4 月）**

**（A）** 7　7　　**（B）** 8　8　　**（C）** 8　10　　**（D）** 10　10

**题眼分析**　sizeof()**函数的作用是返回运算对象占用的字符数。对于字符数组** *a*，**由于它的存储空间长度省略，所以其存储空间长度由初值个数确定，并自动增加一个字符串结束标志** '\0'，**其长度为** 8。**对于字符数组** *b*，**已经固定分配存储空间长度为** 10。

**答案**　C

**【例 2】有以下程序：**

```
main()
{   char a[]={'a','b','c','d','e','f','g','h','\0'}; int i,j;
    i=sizeof(a); j=strlen(a);
```

```
    printf("%d,%d\n",i,j) ;
}
```

**程序运行后的输出结果是_____。（2002 年 9 月）**

（A）9,9　　（B）8,9　　（C）1,8　　（D）9,8

**题眼分析　本题定义的字符数组 a，由于它的存储空间长度省略，所以其存储空间长度由初值个数确定，为 9。sizeof()函数的作用是返回运算对象占用的字符数，所以为 9。strlen()函数的作用是返回字符串的长度，不包含字符串结束标记'\0'，所以是 8。**

**答案**　D

**【例 3】已有定义：char a[]="xyz",b[]={'x','y','z'};，以下叙述中正确的是_____。（2005 年 4 月）**

（A）数组 *a* 和 *b* 的长度相同　　（B）*a* 数组长度小于 *b* 数组长度

（C）*a* 数组长度大于 *b* 数组长度　　（D）上述说法都不对

**题眼分析　数组 *a* 是一个字符串，在赋初值时自动加一个字符串结束标志'\0'，因此它的长度是 4；而数组 *b* 是一个字符数组，根据赋初值，该数组只有 3 个元素，其长度是 3。所以选项（C）是答案。**

**答案**　C

## 二、填空题分析

**【例 1】函数 fun 的功能是：使一个字符串按逆序存放，请填空。（1998 年 9 月）**

```
void fun (char str[])
{   char m; int i,j;
    for(i=0,j=strlen(str);i<j;i++,j--)
        {   m=str[i]; str[i]= _____; str[j-1]=m; }
    printf("%s\n",str);
}
```

**题眼分析　把字符串反序的算法如下：用一个变量 *i* 记下第 1 个字符的下标，开始为 0，用另一个变量 *j* 记下最后一个元素的下标，然后下标为 *i* 的数组元素与下标为 *j* 的数组元素进行交换；然后 *i* 加 1，*j* 减 1，下标为 *i* 的数组元素与下标为 *j* 的数组元素进行交换；……如此反复直到 i 的值大于或等于 *j*。此题用变量 *i* 作为交换的前一个元素的下标，开始赋值为 0，为第 1 个元素的下标，*j* 赋初值为“ strlen(str) ”，所以最后一个元素的下标应为 *j*–1，*j*–1 也作为交换的后一个元素的下标。在循环中应该是 str[*i*]和 str[*j*–1]交换，故横线处应填“ str[*j*–1] ”。**

**答案**　str[*j*–1]

**【例 2】下面程序的功能是：将字符数组 *a* 中下标值为偶数的元素从小到大排列，其他元素不变。请填空_____。（2000 年 4 月）**

```
#include <stdio.h>
#include <string.h>
main()
{   char a[]="clanguage",t,m; int i, j, k;
    k=strlen(a);
    for(i=0; i<=k-2; i+=2)
    {   m=i;
```

```
            for(j=i+2; j<=k;  (1)  )if(a[m]>a[j])m=j;
            if(  (2)  ) { t=a[i]; a[i]=a[m]; a[m]=t; }
        }
        puts(a);
        printf("\n");
    }
```

**题眼分析** 本题将字符数组 *a* 中下标值为偶数的元素从小到大排列，采用选择法排序。和一维数组所有元素的选择法不同的是，它只对偶数下标元素进行排序，因此循环中牵涉到的下标不是加 1，而是加 2。根据选择法的排序方法是每轮选择 1 个最小的数和本轮的第 1 个数交换可知，本题用 *m* 表示每轮最小值的下标，一轮开始认为该轮的最小值下标为第 1 个元素的下标即 *i*，然后最小值 a[m]和所有后面的元素比较，如果后面的元素值小，则记下它的下标。第（1）空处应填下标 *j* 的变化，由于只对偶数下标排序，*j* 的值应增加 2，故第 1 空处应填：*j*=*j*+2(或 *j*+=2)。当一轮比较后，该轮的最小值下标存放在变量 *m* 中，如果 *m* 的值就是该轮比较的第 1 个元素的下标值 *i*，则不需要交换。只有在 *m* 不等于 *i* 的时候，才交换，所以第（2）空处应填：*m!=i*。

**答案** （1）*j*=*j*+2（或 *j*+=2） （2）*m!=i*

**【例 3】以下程序的功能是将字符串 *s* 中的数字字符放入 *d* 数组中，最后输出 *d* 中的字符串。例如，输入字符串：**abc123edf456gh**，执行程序后输出：**123456**。请填空。（2003 年 9 月）**

```
#include <stdio.h>
#include <ctype.h>
main()
{   char  s[80], d[80];  int i, j;
    gets(s);
    for(i=j=0;s[i]!= '\0';i++)
        if(_______)  { d[j]=s[i]; j++;}
    d[j]= '\0';
    puts(d);
}
```

**题眼分析** 程序的思路是：遍历字符数组 *s* 中每一个字符，如果该字符是一个数字字符就将其加入到字符数组 *d* 中。字符的存储是以字符的 ASCII 码存储的，而数字字符在 ASCII 码表中是连续的，根据这一特点可以写出 if 语句的条件为“s[i]>='0'&& s[i]<='9'”。

**答案** s[i]>='0'&& s[i]<='9'

# 10.2 使指针指向一个字符串

## 10.2.1 考点提炼

### 考点 1：通过赋初值的方式使指针指向一个字符串

可以在定义字符指针变量的同时，将存放字符串的存储单元起始地址赋给指针变量。例如：

```
char *ps1="form one";
```

也可以先定义一个存放字符串的字符数组，并将数组的首字地址赋给指针变量。例如：

```
char str[ ]="form two", *ps2;
ps2=str;
```

**考点 2：通过赋值运算使指针指向一个字符串**

如果已经定义了一个字符型指针变量，可以通过赋值运算将某个字符串的起始地址赋给它，从而使其指向一个具体的字符串。例如：

```
char *ps1;
ps1="form one";
```

**考点 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**

若有以下定义：

```
char mark[]="A PROGRAM";
char *pmark="A PROGRAM";
```

虽然字符串的内容相同，但它们占有不同的存储空间。

（1）mark 是一个字符数组，通过赋初值，系统为它开辟了刚好存放以上字符序列再加'\0'的存储空间（10 个字符）。而 pmark 是一个字符指针，通过赋初值，使其指向一个字符串常量。

（2）字符数组 mark 引用固定的存储空间，最多只能存放有 9 个字符的字符串。但指针变量中地址可以改变而指向可另外的字符串，另外字符串的长度不受限制。这里需要注意的是，一旦字符指针指向其他字符串，如果没有另外的指针指向原来的字符串，则此字符串将“丢失”，所占存储空间就再也无法引用。

## 10.2.2　题眼分析

### 一、选择题分析

【例 1】有以下程序

```
main( )
{   char  s [ ]= "ABCD",*p ;
    for (p=s+1; p<s+4 ; p++ )  printf("%s\n", p ) ;
}
```

程序运行后的输出结果是_____。（2004 年 9 月）

（A）ABCD　　（B）A　　（C）B　　（D）BCD

BCD　　B　　C　　CD

CD　　C　　D　　D

D　　D

题眼分析　循环共执行 3 次，第 1 次循环时，*p* 指向 s[1]，输出从字符 B 开始的字符串，直到遇到字符串结束标志'\0'，所以输出 BCD；第 2 次循环时，*p* 指向 s[2]，则输出 CD；第 3 次循环时，*p* 指向 s[3]，则输出 D。

答案　B

**【例 2】有以下函数**

```
fun(char *a,char *b)
{   while((*a!='\0')&&(*b!='\0')&&(*a==*b))
    {   a++;    b++;    }
    return(*a-*b);
}
```

**该函数的功能是_____。（2005 年 4 月）**

**（A）计算 a 和 b 所指字符串的长度之差**

**（B）将 b 所指字符串连接到 a 所指字符串中**

**（C）将 b 所指字符串连接到 a 所指字符串后面**

**（D）比较 a 和 b 所指字符串的大小**

**题眼分析　函数 fun 中有两个字符型指针变量 *a* 和 *b*，在函数中执行 while 循环，该循环的退出条件有两个：一是两个字符串中字符两两相等，但遇到字符串结束标志'\0'（即其中一个字符串已结束而另一个字符串没有结束，或者两个字符串都结束）；二是出现不相等字符。循环退出后，执行“ return(*a–*b); ”语句，即*a 和*b 两个字符 ASCII 出现码差，如果是大于 0，表示字符串 *a* 大于字符串 *b*；如果是小于 0，表示字符串 *a* 小于字符串 *b*；如果是等于 0，则表示两个字符串相等。**

**答案**　D

**【例 3】以下语句或语句组中，能正确进行字符串赋值的是_____。（2005 年 4 月）**

**（A）** char *sp;*sp="right!";　　**（B）** char s[10];s="right!";

**（C）** char s[10];*s="right!";　　**（D）** char *sp="right!";

**题眼分析　选项（A）中，sp 代表一个字符指针，*sp 是一个字符，不能通过**sp="right!"**使指针 sp 指向一个字符串常量，正确的赋值方法是 sp="right!"；选项（B）中，数组名 *s* 代表数组的首地址，是一个地址常量，不能重新赋值。选项（C）的错误和选项（A）的错误是一样的；选项（D）是正确的赋值方式，它的含义是在定义字符指针变量的同时，将存放字符串的存储单元起始地址赋给指针变量。**

**答案**　D

**【例 4】下列选项中正确的语句组是_____。（2003 年 9 月）**

**（A）** char s[8]; s={"Beijing"};　　**（B）** char *s; s={"Beijing"};

**（C）** char s[8]; s="Beijing";　　**（D）** char *s; s="Beijing";

**题眼分析　花括号{}只能在定义变量的同时赋初值时使用，在赋值时不能使用，因此选项（B）不正确。选项（A）、（C）和（D）的分析参见【例 3】。**

**答案**　D

## 二、填空题分析

**【例 1】以下程序运行后的输出结果是_____。（2004 年 9 月）**

```
main( )
{   char  a[ ]= "123456789",*p;
    int   i=0;
    p=a;
```

```
    while (*p)
    {   if ( i%2= =0 )  *p= '*';
        p++;        i++;
    }
    puts(a);
}
```

**题眼分析**　本程序的功能是将数组 *a* 中下标为偶数的元素变为字符“ * ”，因此程序最后输出结果是“ *2*4*6*8* ”。

**答案**　*2*4*6*8*

**【例 2】以下程序运行后的输出结果是_____。（2004 年 9 月）**

```
main( )
{   char  a[ ]= "Language",b[ ]="Programe" ;
    char  *p1,*p2;
    int   k;
    p1=a;   p2=b;
    for(k=0; k<=7; k++)
      if (*(p1+k)= =*(p2+k) )   printf("%c" , *(p1+k) );
}
```

**题眼分析**　本程序定义了两个字符指针 *p*1 和 *p*2，分别指向字符数组 a[]和 b[]。于是，*(p1+k)所指的字符就是 a[k]，*(p2+k)所指的字符就是 b[k]。因此程序的功能是：如果两个字符串中对应位置上的字符相同，则输出这个字符，否则不输出。

**答案**　gae

**【例 3】以下程序中函数 huiwen 的功能是检查一个字符串是否是回文，当字符串是回文时，函数返回字符串：yes!；否则函数返回字符串：no!，并在主函数中输出。所谓回文即正向与反向的拼写都一样，例如：adgda。请填空。（2005 年 4 月）**

```
#include   <string.h>
char *huiwen(char *str)
{   char *p1,*p2;int i,t=0;
    p1=str;    p2=__(1)__;
    for(i=0;i<=strlen(str)/2;i++)
        if(*p1++!=*p2--){t=1;break;}
    if(__(2)__)return("yes!");
    else return("no!");
}
main()
{   char str[50];
    printf("Input:");scanf("%s",str);
    printf("%s\n",__(3)__);
}
```

**题眼分析**　函数*huiwen()实现思路是：使用两个字符指针 *p*1 和 *p*2 分别指向字符串的首字符和最后一个字符，通过一个循环，使这两个指针都向字符串中间移动，依次比较所指的字符，当遇到所指字符不相等时或移动到中间位置时退出循环。程序中使用一个标志变量 *t* 来识别是通过什么方式结束循环的，若 *t*=0，表示指针已经移动到中间位置，两两都相同，说明它

是一个回文字符串；若 *t*=1，表示遇到所指字符不相等，说明它不是一个回文字符串。综上所述，第（1）个空是使字符指针 *p2* 指向字符串的最后一个字符，字符串的首地址是 str，长度是 strlen(str)，因此最后第 1 个字符的地址是 str+strlen(str)−1，或使用 & 来取得，即 &str[strlen(str)−1]。第（2）个空是判断是以何种方式结束循环的，以决定返回字符串，因此，该处应填写 t= =0，以及其等价形式 *t*!=1、*!t*。第（3）个空需要填写调用函数*huiwen()方式，函数*huiwen()有一个形参，其返回值是字符指针，因此应填写 huiwen(str)及其等价形式。

**答案** （1）str+strlen(str)−1 **或** &str[strlen(str)−1]

（2）*t*= =0 **或** *t*!=1 **或** !*t*

（3）huiwen(str) **或** huiwen(**&**str[0])

# 10.3 字符串的输入和输出

## 10.3.1 考点提炼

### 考点 1：用格式说明符%s 进行整串输入和输出

字符数组的输入输出可以像整型数组一样利用格式符“ %c ”进行单个字符的输入输出，也可以利用“ %s ”进行整体的输入输出。利用“ %c ”进行输入输出时，每按下的一个键均作为一个字符，包括回车键。利用“ %s ”进行输入时，输入的结束标记是空格或回车，输出时输出到字符串结束标记为止。

### 考点 2：puts，gets 函数

用于输入输出字符串的 puts,gets 函数， 在使用前应包含头文件"stdio.h"。

1．字符串输入函数 gets

gets 的调用形式如下：

gets (存储输入字符串的起始地址)；

功能：从标准输入设备键盘上输入一个字符串。 本函数得到一个函数值，即为该字符数组的首地址。

gets 函数并不以空格作为字符串输入结束的标志，而只以回车作为输入结束，系统将自动用'\0'代码换行符。

2．字符串输出函数 puts

puts 的调用形式如下：

puts (存放待输出字符串的起始地址)；

功能： 从待输出字符串的起始地址开始，依次输出存储单元中的字符，遇到第 1 个'\0'即输出结束，并自动输出一个换行符。

## 10.3.2　题眼分析

### 一、选择题分析

**【例 1】有以下程序：**

```
#include<stdio.h>
main()
{   char  *p,*q;  p=(char *)malloc(sizeof(char)*20);q=p;
    scanf("%s %s",p,q);printf("%s %s\n",p,q);
}
```

**若从键盘输入：**abc　def<**回车**>，**则输出结果是**_____。（2002 **年** 9 **月**）

（A）def def　　（B）abc def　　（C）abc d　　（D）d d

**题眼分析**　**本题首先定义两个字符型指针变量** *p* **和** *q*，**通过** malloc()**函数申请** 20 **个字符的存储空间，并把它的首地址赋给** *p*，**再把** *p* **的值赋给** *q*，*p* **和** *q* **指向同一个存储区。在** scanf() **语句中读取字符串到** *p* **和** *q* **指向的字符串，先把**"abc"**读到** *p* **指向的存储区中，第** 1 **个空格是结束标记，第** 2 **个空格是分隔符，再把**"def"**存放到** *q* **指向的存储区中，把原先的内容覆盖。所以** *p* **和** *q* **指向的存储区中的内容都是“** def **”。**

**答案**　A

**【例 2】有以下程序**

```
main( )
{   char  s [ ]= "Yes\n/No",*ps=s ;
    puts(ps+4);
    *(ps+4)=0;
    puts(s);
}
```

**程序运行后的输出结果是**_____。（**选项** D **中的第** 1 **行是空行**）（2004 **年** 9 **月**）

A）n/No　　B）/No　　C）n/No　　D）

Yes　　Yes　　Yes　　/No

/No　　/No　　Yes

**题眼分析**　**在字符串** *s* **中，包含了一个转义字符**'\n'，**它在字符串数组作为一个元素存储的。函数** puts()**的形参是要输出的字符串的起始地址，***ps*+4 **指向字符“** / **”，因此第** 1 **个** puts **语句将是“** /No **”。由“** *(*ps*+4)=0; **”语句把字符数组中** a[4]，**即“** \ **”改为字符串结束标志**'\0'（**其** ASCII **码为** 0）。**利用函数** puts()**输出字符串时，遇到第** 1 **个**'\0'**就输出结束，并自动输出一个换行符。因此第** 2 **个** puts **语句将是“** Yes<**换行符**> **”。**

**答案**　B

### 二、填空题分析

**【例 1】以下程序的输出结果是**_____。（2002 **年** 4 **月**）

```
main()
{   char s[]="abcdef";s[3]= '\0';
    printf("%s\n",s);
```

```
}
```

**题眼分析**　字符串的结束标记是'\0'，当输出一个存放在字符数组中的字符串时，只需输出到'\0'为止，而不管其后还有什么数据。本题给字符数组 *s* 的元素 s[3]的赋值为'\0'，故只能输出 3 个字符“abc”。

**答案**　abc

**【例 2】以下程序运行后的输出结果是_____。（2005 年 4 月）**

```
#include  <string.h>
void fun(char *s,int p,int k)
{   int i;
    for(i=p;i<k-1;i++) s[i]=s[i+2];
}
main()
{   char s[]="abcdefg";
    fun(s,3,strlen(s));    puts(s);
}
```

**题眼分析**　在 main()函数中通过赋初值方法定义了一个长度为 7 的字符串 *s*，并在赋值时自动加了一个字符串结束标志'\0'，因此字符数组 *s* 中实际存放的内容是" abcdefg\0"。执行函数调用 fun(a,3, strlen(s))，把 *s* 的首地址赋给形参字符指针 s，把 3 和 7 分别赋给形参 *p* 和 *k*。在函数 fun()中，for 循环执行了 3 次：第 1 次，*p* 值为 3，相当于执行了语句“s[3]=a[5];”，s[3]的值就是'f'；第 2 次，*p* 值为 4，相当于执行了语句“s[4]=s[6];”，s[4]的值就是'g'；最后一次，*p* 的值为 5，相当于执行了语句“s[5]=s[7];” s[3]的值就是字符串结束标志'\0'。这样字符数组 *s* 的值为"abcfg\0g\0"。字符串输出函数 puts 的功能从待输出字符串的起始地址开始，依次输出存储单元中的字符，遇到第 1 个'\0'即输出结束，因此输出"abcfg"。

**答案**　abcfg

# 10.4　字符串数组

## 10.4.1　考点提炼

字符串数组就是数组中的每个元素都是一个存放字符串的数组。

（1）可以将一个二维字符数组视为一个字符串数组。例如：

```
char name[10][80];
```

其中：二维数组的第 1 个下标决定了字符串的个数，第 2 个下标决定字符串的最大长度。

（2）字符串数组可以在定义的同时赋初值。例如：

```
char str[5][10]={"File","Edit","Run","Compile","Debug"};
char str[][10]={"File","Edit","Run","Compile","Debug"};
```

（3）可以定义字符型指针数组并通过赋初值来构成一个类似的字符串数组。例如：

```
char *str[5] ={"File","Edit",”Run","Compile","Debug"};
```

## 10.4.2　题眼分析

### 一、选择题分析

**【例 1】有以下程序**

```
main( )
{   char  str[ ][10]={ "China","Beijing"},*p=str ;
    printf("%s\n", p+10 ) ;
}
```

**程序运行后的输出结果是_____。（2004 年 9 月）**

（A）China　　（B）Beijing　　（C）ng　　（D）ing

**题眼分析**　二维数组 str 给每个字符串分配的存储空间是 10 个字节，而指针 *p* 又指向字符串数组的首地址，因此 *p*+10 指向 str[1][0]，输出的字符串是字符串数组的第 2 个字符串。

**答案**　B

**【例 2】有以下程序**

```
main()
{   char *s[]={"one","two","three"},*p;
    p=s[1];
    printf("%c,%s\n",*(p+1),s[0]);
}
```

**执行后输出结果是_____。（2003 年 4 月）**

（A）n,two　　（B）t,one　　（C）w,one　　（D）o,two

**题眼分析**　本题首先定义了一个有 3 个元素的指针数组 *s*，并通过赋初值使它的元素 s[0]指向"one"，s[1]指向"two"，s[2]指向"three"，通过赋值语句‘ p=s[1]; ’使 *p* 指向了"two"，故‘ *(p+1) ”就是字符“ w ”。

**答案**　C

### 二、填空题分析

**【例】以下程序运行后的输出结果是_____。（2005 年 4 月）**

```
#include  <string.h>
main()
{   char ch[]="abc",x[3][4];   int i;
    for(i=0;i<3;i++)   strcpy(x[i],ch);
    for(i=0;i<3;i++)   printf("%s",&x[i][i]);
    printf("\n");
}
```

**题眼分析**　本题中，通过第 1 个 for 循环，对字符串数组 x[3][4]进行赋值，使 x[0]、x[1]和 x[2]的值都是"abc"。关键是第 2 个 for 循环，它依次是从第 0 个位置，第 1 个位置、第 2 个位置输出字符串，因此输出内容分别是"abc"、"bc"和"c"。这里需要注意通过 printf()函数输出字符串和通过 puts()函数输出字符串的区别，后者输出一个字符串后将自动输出一个换行符。

**答案**　abcbcc

# 10.5 用于字符串处理的函数

## 10.5.1 考点提炼

**字符串处理函数应包含头文件**"string.h"**。**

1．**字符串拷贝函数** strcpy

**格式：** strcpy (**字符数组名** 1，**字符数组名** 2)

**功能：把字符数组** 2 **中的字符串拷贝到字符数组** 1 **中。字符串结束标志**'\0'**也一同拷贝。字符数名** 2，**也可以是一个字符串常量。**

2．**字符串连接函数** strcat

**格式：**strcat (**字符数组名** 1，**字符数组名** 2)

**功能：把字符数组** 2 **中的字符串连接到字符数组** 1 **中字符串的后面，并删去字符串** 1 **后的串标志**'\0'**。本函数返回值是字符数组** 1 **的首地址。**

3．**测字符串长度函数** strlen

**格式：** strlen(**字符数组名**)

**功能：测字符串的实际长度(不含字符串结束标志**'\0') **并作为函数返回值。**

4．**字符串比较函数** strcmp

**格式：**strcmp(**字符数组名** 1，**字符数组名** 2)

**功能：按照** ASCII **码顺序比较两个数组中的字符串，并由函数返回值返回比较结果。**

**若“字符串** 1**”>“字符串** 2**”，返回一个大于** 0 **的整数值；**

**若“字符串** 1**”=“字符串** 2**”，返回一个等于** 0 **的整数值；**

**若“字符串** 1**”<“字符串** 2**”，返回一个小于** 0 **的整数值。**

## 10.5.2 题眼分析

### 一、选择题分析

**【例** 1**】以下程序的输出结果是_____。(**2002 **年** 4 **月)**

```
#include <stdio.h>
#include <string.h>
main()
{   char b1[8]="abcdefg",b2[8],*pb=b1+3;
    while (--pb>=b1) strcpy(b2,pb);
    printf("%d\n",strlen(b2));     }
```

(A) 8　　(B) 3　　(C) 1　　(D) 7

**题眼分析　本题使用的是带有两个参数的** strcpy**，其作用是把第** 2 **个参数代表的字符串复制到第** 1 **个参数指向的数组中。本题首先定义了两个字符数组** *b1* **和** *b2*，**并用一个字符串给** *b1* **赋初值，然后定义了一个字符型指针变量** *pb*，**通过赋初值使它指向了** b1[3]。**接着执行** while **循环，不难得出该循环执行了** 3 **次：第** 1 **次判断条件“** *—pb>=b1* **”，使** *pb* **的值为“** *b1+2* **”，执**

行“ strcpy(b2,pb); ”后，*b2* 中的内容为“ cdefg ”；第 2 次判断条件“ *--pb>=b1* ”，使 *pb* 的值为“ b1+1 ”，执行“ strcpy(b2,pb); ”后，*b2* 中的内容为“ bcdefg ”；第 3 次判断条件“ *--pb>=b1* ”，使 *pb* 的值为“ b1 ”，执行“ strcpy(b2,pb); ”后，b2 中的内容为“ abcdefg ”。最后输出 b2 数组中存放的字符串长度，显然是 7。

**答案**　D

**【例 2】有以下程序**

```
main()
{   char a[7]="a0\0a0\0"; int i,j;
    i=sizeof(a); j=strlen(a);
    printf("%d %d\n",i,j);
}
```

程序运行后的输出结果是_____。（2005 年 4 月）

（A）2　2　　（B）7　6　　（C）7　2　　（D）6　2

**题眼分析**　函数 sizeof 是求字符数组所占用的内存空间的长度，因为在定义字符数组时已经指定的数组的长度为 7。函数 strlen 是求字符串的长度，即字符串中有效字符的个数，其值为第 1 个字符串结束标志\0前的字符数，为 2。

**答案**　C

**【例 3】** *s1* 和 *s2* 已正确定义并分别指向两个字符串。若要求：当 *s1* 所指串大于 *s2* 所指串时，执行语句 S；则以下选项中正确的是_____。（2004 年 9 月）

A）if (s1>s2) S;　　B）if (strcmp(s1,s2)) S;

C）if (strcmp(s2,s1)>0) S;　　D）if (strcmp(s1,s2)>0) S;

**题眼分析**　比较两个字符串的大小不能直接将两个字符名进行比较，可借助字符串比较函数 strcmp(s1,s2)，因此选项（A）正确。在字符串比较函数 strcmp(s1,s2)中，当 *s1* 所指的字符串大于 *s2* 所指的字符串时，strcmp(s1,s2)>0；当 *s1* 所指的字符串小于 *s2* 所指的字符串时，strcmp(s1,s2)<0；当 *s1* 所指的字符串等于 *s2* 所指的字符串时，strcmp(s1,s2)=0。

**答案**　A

**【例 4】以下程序中函数 scmp 的功能是返回形参指针 *s1* 和 *s2* 所指字符串中较小字符串的首地址。**

```
#include <stdio.h>
#include <string.h>
char *scmp(char *s1,char *s2)
{   if(strcmp(s1,s2)<0)
        return(s1);
    else  return(s2);
}
main()
{   int i;  char string[20],str[3][20];
    for(i=0;i<3;i++)     gets(str[i]);
    strcpy(string,scmp(str[0],str[1]));
    strcpy(string,scmp(string,str[2]));
    printf("%s\n",string);
```

```
}
```

**若运行时依次输入：**abcd、abba **和** abc3 **个字符串，则输出结果为_____。（**2003 **年** 9 **月）**

（A）abcd　　（B）abba　　（C）abc　　（D）abca

**题眼分析　程序的功能很明显，是找出输入的** 3 **个字符串的最小的一个字符串。这里需要注意字符串的比较方法。字符串的比较方法是：依次对** *s1* **和** *s2* **所指字符对应位置上的字符两两进行比较，当出现第** 1 **对不相等字符时，即由这两个字符决定所在字符串的大小。因此，**abcd、abba **和** abc3 **个字符串的最小一个字符串是** abba。

**答案**　B

## 二、填空题分析

**【例】以下程序运行后输入：**3，abcde<**回车**>，**则输出结果是_____。（**2003 **年** 9 **月）**

```
#include<string.h>
move(char *str, int n)
{   char  temp;  int  i;
    temp=str[n-1];
    for(i=n-1;i>0;i--)  str[i]=str[i-1];
    str[0]=temp;
}
main()
{   char s[50];  int  n, i, z;
    scanf("%d,%s",&n,s);
    z=strlen(s);
    for(i=1;i<=n;i++)  move(s,z);
    printf("%s\n", s);
}
```

**题眼分析　函数** move（）**的功能是：将字符数组的第** *n*–1 **的字符** str[*n*–1]**移动到字符串的最前端** str[0]，**原来的第** 1 **个字符** a[0]**至倒数第** 2 **个字符** a[*n*–2]**依次后移。主函数** main()**调用函数** move（）3 **次，因此输出的值是“** cdeab **”。**

**答案**　cdeab

# 10.6　单元训练及参考答案

## 10.6.1　单元训练

### 一、选择题

1．**下述对** C **语言字符数组的描述中错误的是_____。**

（A）**字符数组可以存放字符串**

（B）**字符数组的字符串可以整体输入、输出**

（C）可以在赋值语句中通过赋值运算符“ = ”对字符数组整体赋值

（D）不可以用关系运算符对字符数组中的字符串进行比较

2．以下对字符数组 word 进行不正确初始化的是_____。

（A）static char word[]= ’Turbo\0’

（B）static char word[]={ ’T’, ’u’, ’r’, ’b’, ’o’, ’\0’};

（C）static char word[]={ "Turbo\0"};

（D）static char word[]= "Turbo\0";

3．下面是对 *s* 的初始化，数组长度是 4 的是_____。

（A）char s[]={"abc"};　　（B）char s[]={’a’, ’b’, ’c’}

（C）char s[]=""　　（D）char s[]="abcdef"

4．有下面程序段

```
char  a[3],b[]="China";
a=b; printf("%s",a);
```

则_____。

（A）运行时输出 China　　（B）运行时输出 Ch

（C）运行时输出 Chi　　（D）编译出错

5．设有数组定义：char array[]="China";。则数组 array 所占的空间为_____。

（A）4 个字节　　（B）5 个字节　　（C）6 个字节　　（D）7 个字节

6．下列程序段的运行结果是_____。

```
char  c[ ]= "\t\v\\\0will\n";
printf("%d",strlen(c));
```

（A）14　　（B）3

（C）9　　（D）字符串中有非法字符，输出值不确定

7．以下程序的输出结果是_____。

```
main()
{   char st[20]= "hello\0\t\\\";
    printf(%d %d \n",strlen(st),sizeof(st));   }
```

（A）9 9　　（B）5 20　　（C）13 20　　（D）20 20

8．有以下程序：

```
void ss(char *s,char  t)
{   while(*s) {  if(*s==t) *s=t-’a’+’A’; s++; } }
main()
{   char str1[100]="abcddfefdbd",c=’d’
    ss(str1,c);printf("%s\n",str1);    }
```

程序运行后的输出结果是_____。

（A）ABCDDEFEDBD　　（B）abcDDfefDbD

（C）abcAAfefAbA　　（D）Abcddfefdbd

9．判断字符串 *a* 是否等于 *b*，应当使用_____。

（A）if (a==b)　　（B）if (a=b)　　（C）if (strcpy(a,b))　　（D）if (strcmp(a,b))

10．当接受用户输入的含空格的字符串时，应使用的函数是_____。

（A）scanf()　　（B）gets()　　（C）getchar()　　（D）getc()

11．当输出含空格的字符串时，应使用的函数是_____。

（A）printf()　　（B）put()　　（C）putchar()　　（D）putc()

## 二、填空题

1．字符串"ab\n\\012\\\""的长度是___（1）___。

2．执行程序段：char x[]="the teacher";i=0; while(x[++i]!='\0') if(x[i–1]== 't') printf("%c",x[i])后，程序的输出结果是___（2）___。

3．下面程序段的运行结果是___（3）___。

```
char  ch[]="600";int a,s=0;
for(a=0;ch[a]>= '0'&&ch[a]<= '9';a++)  s=10*s+ch[a]- '0';
printf("%d",s)
```

4．以下函数的功能是删除字符串 *s* 中的所有数字字符。请填空。

```
void dele(char *s)
{   int  n=0;i;
    for(i=0;s[i];i++)
        if(___(4)___)
            s[n++]=s[i];
    s[n]=___(5)___                }
```

5．以下 sstrcpy()函数实现字符串复制，即将 *t* 所指字符串复制到 *s* 所指内存空间中，形成一个新字符串 *s*。请填空。

```
void  sstrcpy (char  *s, char  *t)
{   while(*s++=__(6)__);   }
main()
{   char   strl[100], str2[]="abcdefgh";
    sstrcpy(strl, str2);
    printf("%s\n", strl);       }
```

## 10.6.2　参考答案

### 一、选择题

| | | | | |
|---|---|---|---|---|
| 1．C | 2．A | 3．A | 4．D | 5．C |
| 6．B | 7．B | 8．B | 9．D | 10．B |
| 11．A | | | | |

### 二、填空题

（1）9　　（2）he

（3）600　　（4）s[i]<'0'|| s[i]>9

（5）'\0'　　（6）*t++

# 第 11 章　对函数的进一步讨论

## 11.1　传给 main 函数的参数

### 11.1.1　考点提炼

C 语言规定，main()函数可以带参数。main()函数的形式参数有两个：第 1 个形参记下了命令行参数的个数，类型为整型，名称通常规定为 argc（可取其他名字）；第 2 个形参是一个字符指针数组，其每个元素指向执行本程序命令行的各个字符串，名称通常规定为 argv[]（可取其他名字）。带参数的 main 的函数头的定义形式如下：

```
main(int argc,char *argv[])
```

如果在 C 程序中使用带参数的 main()函数，该程序只能在命令行中执行，执行方式如下：

程序名　实参 1　实参 2 ……　实参 *m*

该命令行执行时，系统会自动在内存中开辟区域依次存放程序名和所有的实际参数，同时给 main()函数的形参赋值，结果如下：

形参 argc 赋值为 *m*+1，表示连同程序名在内有 *m*+1 个参数；

形参 argv[0]赋值为存放“程序名”字符串内存区域的首地址；

形参 argv[1]赋值为存放“实参 1”字符串内存区域的首地址；

……

形参 argv[m–1]赋值为存放“实参 *m*”字符串内存区域的首地址。

### 11.1.2　题眼分析

【例 1】不合法的 main()函数命令行参数表示形式是_____。（2002 年 4 月）

（A）main(int a,char *c[])　　（B）main(int arc,char **arv)

（C）main(int argc,char *argv)　　（D）main(int argv,char *argc[])

题眼分析　main()函数可以带有参数，并且参数只能有两个，第 1 个参数（argc）类型为整型用来记下命令行的参数个数；第 2 个参数（argv）为一个字符型指针数组，其各个元素用来记下命令行各参数字符串的首地址。选项（C）的第 2 个参数是一个字符数组，不符合要求。

答案　C

【例 2】有以下程序

```
#include  <string.h>
main(int argc,char *argv[])
{   int  i,len=0;
    for(i=l;i<argc;i+=2)  len+= strlen(argv[i]);
    printf("%d\n",len);
}
```

经编译连接后生成的可执行文件是 ex.exe，若运行时输入以下带参数的命令行

```
ex abcd efg h3 k44
```

执行后输出结果是_____。（2004 年 4 月）

（A） l4　　（B）l2　　（C）8　　（D）6

**题眼分析**　执行时，argc 的值为 5，argv[0]、argv[1]、argv[2]、argv[3]、argv[3]分别是指向字符串“ex”、“abcd”、“efg”、“h3”、“k44”。在循环中，依次求出 argv[1]、argv[3]指向的字符串（“abcd”和“h3”）的长度并把它们加到变量 len 中，得到 len 的值为 6。

**答案**　D

# 11.2　通过实参向函数传递函数名或指向函数的指针变量

## 11.2.1　考点提炼

### 考点 1：指向函数的指针变量

C 语言中，函数名代表该函数的入口地址，因此，可以定义一种指向函数的指针存取这个地址。指向函数的指针变量的一般定义形式为：

数据类型标识符 (*指针变量名)()；

例如：

```
int (*p)();
```

表示定义一个指向函数的指针变量，但指向的函数值必须是 int。

说明：

（1）函数的调用可以通过函数名调用，也可以通过函数指针调用（即用指向函数的指针变量调用）。

（2）(*p)()表示定义一个指向函数的指针变量，这不是固定指向哪一个函数，而只是表示定义了这样一个类型的变量，它是专门用来存放函数的入口地址的。在程序中把哪一个函数的地址赋给它，它就指向哪一个函数。在一个程序中，一个指针变量可以先后指向不同的函数。

（3）在给函数指针变量赋值时，只需给出函数名而不必给出参数。例如定义了一个函数“int max(int x,int y);”，希望指针 *p* 指向它的入口地址，其形式是：

```
p=max;
```

（4）用函数指针变量调用函数时，只需将(*p)代替函数名即可，在(*p)之后的括号中根据需要写上实参。例如上例中，要调用 max(a,b)，可以写成：

```
c=(*p)(a,b);
```

（5）对指向函数的指针变量，像 p+n、p++、p—等运算是无意义的。

### 考点 2：函数名或指向函数的指针变量做实参

函数名或指向函数的指针变量做实参可以作为实参传送给函数，这时对应的形参应当是类型相同的指针变量。

## 11.2.2　题眼分析

### 一、选择题分析

**【例 1】程序中若有如下说明和定义语句**

```
char fun(char *);
main()
{
    char *s="one",a[5]={0},(*f1)()=fun,ch;
        ...
}
```

**以下选项中对函数 fun 的正确调用语句是_____。（2005 年 4 月）**

（A）(*f1)(a);　　（B）*f1(*s);　　（C）fun(&a);　　（D）ch=*f1(s);

**题眼分析**　**程序中定义的(*f1)()是一个指向函数的指针变量，并通过赋初值使其指向函数 fun()。这样，在调用函数 fun 时，既可以通过函数名来调用，也可以通过函数指针来调用。通过后一种方式调用时，这对圆括号()不能省略。根据对函数 fun()的说明，该函数为一个形参，其类型是一个字符指针，因此其对应的实参应是一个字符数组或字符串的首地址，该函数的返回值为字符，在调用时可以给一个字符赋值。综上所述，选项（A）是正确调用；选项（B）中有两个错误，一是指向函数的指针变量引用错误，应为(*f1)()，二是*s 代表的是一个字符，而不是字符串的首地址；选项（C）中，字符数组的数组名就代表数组的首地址，不需要取地址运算符&；选项（D）中，指向函数的指针变量引用错误，应为(*f1)(s)。**

**答案**　A

**【例 2】有以下程序：**

```
int fa(int  x)
{   return  x*x;    }
int fb(int x)
{   return  x*x*x;  }
int f(int (*f1)(),int (*f2)(),int x)
{   return  f2(x)-f1(x); }
main()
{   int i; i=f(fa,fb,2);printf("%d\n",i);}
```

**程序运行后的输出结果是_____。（2002 年 9 月）**

（A）–4　　（B）1　　（C）4　　（D）8

**题眼分析**　**函数 f()有 3 个形式参数 f1、f2 和 x，其中 f1、f2 是指向函数的指针变量。在 main()函数中执行了函数调用“ f(fa,fb,2) ”，从而使 f()的形式参数 f1 指向了 fa，形式参数 f2 指向了 fb，把实参 2 传给了形参变量 x。函数 f()中的 return 语句中的相当于“ fb(2)–fa(2) ”，值为 4。函数 f()执行后把返回值 4 赋给了 i，输出 i 的值是 4。**

**答案**　C

### 二、填空题分析

**【例 1】设函数 findbig 已定义为求 3 个数中的最大值，以下程序将利用函数指针调用**

findbig 函数。请填空_____。（2003 年 4 月）

```
main()
{  int findbig(int,int,int);
   int (*f)(),x,y,z,big;
   f=___________;
   scanf("%d%d%d",&x,&y,&z);
   big=(*f)(x,y,z);
   printf("big=%d\n", big);
}
```

**题眼分析**　本题首先定义了一个指向函数的指针变量 *f*，如果希望让它指向某个函数，只需把函数名赋给该指针变量即可。

**答案**　findbig

【例 2】以下程序的输出结果是_____。

```
funa(int a,int b)
{   return a+b;     }
funb(int a,int b)
{   return a-b;     }
sub(int,int x,int y))
{   return  (*t)(x,y);      }
main()
{   int x,(*p)(int,int);
    p=funa;
    x=sub(p,9,3);
    x+=sub(funb,8,3);
    printf("%d",x);
}
```

**题眼分析**　在主函数 main 中，定义了一个指向函数的指针变量 *p*，通过赋值语句，使指针 p 指向函数 funa，即函数 funa 的入口地址。通过调用函数 sub，把函数 funa 的入口地址、9 和 3 分别传递给函数 sub 中形参(*t)()、*x* 和 *y*，执行“ (*t)(x,y) ；”语句就相当于执行了“ funa(9,3) ”，其返回值是 6。funb 是一个函数名，它也代表该函数的入口地址，因此执行“ sub(funb,8,3) ”的返回值是 5。所以变量 *x* 的最终值是 11。

**答案**　11

# 11.3　函数的递归调用

## 11.3.1　考点提炼

C 语言中允许函数的递归调用，所谓函数的递归是指在调用一个函数的过程中，又出现了直接或间接地调用该函数本身。在此仅讨论直接递归，即函数自身调用自身。

1．递归问题的特征

为求解规模为 *n* 的问题，设法将它分解成规模较小的问题，然后从这些小问题的解方便地

构造出大问题的解，并且这些规模较小的问题也能采用同样的分解和综合方法，分解成规模更小的问题，并从这些更小问题的解构造出规模较大问题的解。特别是当规模 $n$=1 时，能直接得解。

2．递归函数的执行过程

为了理解递归的含义，可通过一个简单的例子来加以说明。例如，求斐波那契数列的第 $n$ 项 fib(n)的公式为：

$$Fib(n)=\begin{cases} n, & n=0,1 \\ Fib(n-1)+Fib(n-2), & n>1 \end{cases}$$

它对应的递归过程为：

```
long Fib(long n)
{
 if (n<=1) return n;          //终止递归条件
 else Fib(n-1)+Fib(n-2);      //递归步骤
}
```

递归算法的执行过程分递推和回归两个阶段。在递推阶段，把较复杂的问题（规模为 $n$）的求解推到比原问题简单一些的问题（规模小于 n）的求解。例如上例中，求解 fib(n)，把它推到求解 fib(n–1)和 fib(n–2)。也就是说，为计算 fib(n)，必须先计算 fib(n–1)和 fib(n–2)，而计算 fib(n–1)和 fib(n–2)，又必须先计算 fib(n–3)和 fib(n–4)。依此类推，直至计算 fib(1)和 fib(0)，分别能立即得到结果 1 和 0。在递推阶段，必须要有终止递归的情况。例如在函数 fib 中，当 n 为 1 和 0 的情况。其执行过程如图 11-1 所示。

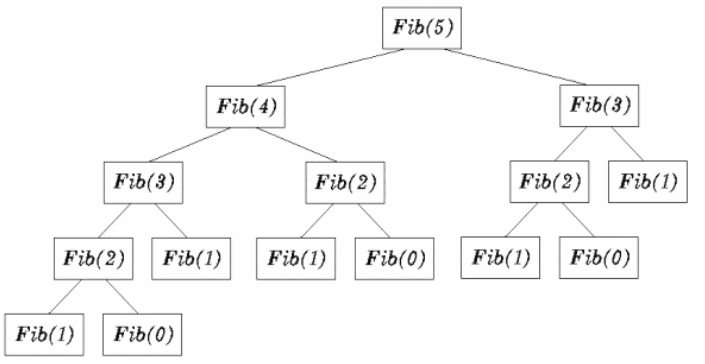



图 11-1　递归算法的执行过程

在回归阶段，当获得最简单情况的解后，逐级返回，依次得到稍复杂问题的解，例如得到 fib(1)和 fib(0)后，返回得到 fib(2)的结果……在得到了 fib(n–1)和 fib(n–2)的结果后，返回得到 fib(n)的结果。

3．递归函数的一般实现方法

从程序设计的角度来说，递归过程必须解决两个问题：一是递归计算的公式，二是递归结束的条件。如上例中的问题，可以写成：

**递归计算公式：**Fib(n)=Fib(n–1)+Fib(n–2)　　n>1

**递归结束条件：**Fib(n)=n　　n<=1

**凡是能够表示成上述式子的数学问题均可以用递归来实现，在递归函数中一般可采用双分支语句来实现：**

```
if （递归结束条件） return(递归终止值)
else return(递归计算公式)
```

## 11.3.2 题眼分析

### 一、选择题分析

**【例 1】在函数调用过程中，如果函数 funA 调用了函数 funB，函数 funB 又调用了函数 funA，则_____。（2004 年 9 月）**

**（A）称为函数的直接递归调用**　　**（B）称为函数的间接递归调用**

**（C）称为函数的循环调用**　　**（D）C 语言中不允许这样的递归调用**

**题眼分析**　C 语言中，函数有两种递归方式，一种是函数直接地调用自己，称为简单递归，另一种函数间接地调用自己，称为间接递归。很明显题中所描述的属于后者。

**答案**　B

**【例 2】有以下程序**

```
void  fun(int  *a , int  i , int  j )
{   int  t;
    if (i<j)
    {   t=a[i] ;  a[i]= a[j] ;   a[j]=t ;
        i++ ;       j-- ;
        fun (a , i , j ) ;
    }
}
main( )
{   int  x[ ]= {2,6,1,8},i ;
    fun(x,0,3);
    for (i=0; i< 4 ; i++ )  printf("%2d", x[i] ) ;
}
```

**程序运行后的输出结果是_____。（2004 年 9 月）**

A）1 2 6 8　　B）8 6 2 1　　C）8 1 6 2　　D）8 6 1 2

**题眼分析**　函数 fun 中有 3 个参数，参数 *a* 为一个指针变量，传进来的实参可以是数组名，参数 *i* 和 *j* 分别表示数组的下标。当下标 *i* 小于下标 *j* 时，*a*[*i*]和 *a*[*j*]交换，*i* 自增 *j* 自减后调用函数自身“ fun(*a*,*i*,*j*) ”。可见这是一个递归函数，其实现的功能是把 *a* 数组从下标为 *i* 的元素到下标为 *j* 的元素之间的所有元素反序存放。在主函数中调用 fun 函数，把 *a* 数组从 *a*[0]到 *a*[3]之间的所有元素反序存放，最后输出 *a*[0]到 *a*[3]。

**答案**　C

**【例 3】有以下程序：**

```
int f(int n)
```

```
{   if(n==1) return 1;
    else return(f(n-1)+1);
}
main()
{   int i,j=0;
    for(i=1;i<3;i++) j+=f(i);
    printf("%d\n",j);
}
```

**程序运行后的输出是_____。（2002 年 9 月）**

（A）4　　（B）3　　（C）2　　（D）1

**题眼分析　通过分析不难写出 f 函数的数学公式为：**

$$f(n)=\begin{cases}1 & n=1;\\ f(n)=f(n-1)+1 & n\neq 1;\end{cases}$$

**在主函数中 for 循环执行了两次函数调用 f(*i*)。第 1 次：*i* 为 1，调用 f(1)得到返回值 1 并把它加到 *j* 中，*j* 的值为 1。第 2 次：i 为 2，调用 f(2)，根据递归公式可知“ f(2)=f(1)+1 ”，得到返回值 2 并把它加到 *j* 中，*j* 的值为 3。最后输出的 *j* 的值为 3。**

**答案**　B

## 二、填空题分析

**【例 1】以下程序运行的输出结果是_____。（2003 年 9 月）**

```
fun (int  x)
{   if(x/2>0)  fun(x/2);
    printf("%d  ",x);
}
main()
{   fun(6); }
```

**题眼分析　函数 fun 是一个递归函数。第 1 次调用函数 fun 时，*x* 的值为 6，*x*/2>0 的条件成立，再一次调用函数 fun，但输出 *x*（*x*=6）的语句并没有执行，保存了断点（称为断点 1）；第 2 次调用函数 fun 时，*x* 的值为 3，*x*/2>0 的条件成立，再一次调用函数 fun，但输出 *x*（*x*=3）的语句并没有执行，保存了断点（称为断点 2）；第 3 次调用函数 fun 时，*x* 的值为 1，*x*/2>0 的条件不成立，执行输出语句输出 *x* 的值为 1；递归返回断点 2，输出 *x* 的值为 3；再次递归返回断点 1，输出 *x* 的值为 6。程序执行过程如图 11-2 所示。**

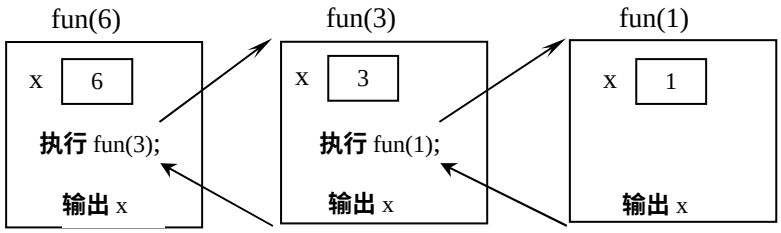



**图 11-2　sub(&x,8,1)函数的递归执行过程**

**答案**　1 3 6

【例 2】下面程序的运行结果：_____。（2004 年 4 月）

```
int f(int  a[],int  n)
{   if(n>l)  return  a[0]+f(&a[1],n-1);
    else  return  a[0];
}
main()
{   int  aa[3]={1,2,3},s;
    s=f(&aa[0],3);  printf("%d\n", s);
}
```

题眼分析　通过程序不难写出 $f$ 函数的数学公式为：

$$s(a[n],n)=\begin{cases} a[0] & \text{n}=1 \\ a[0]+s(a[n-1],n-1) & \text{n}>1 \end{cases}$$

从数学公式不难看出，当 $n>1$ 时就有：

$$\begin{aligned} s(a[n],n) &= a[0]+s(a[1],n-1) \\ &= a[0]+a[1]+s(a[2],n-2) \\ &\Lambda\ \Lambda \\ &= a[0]+a[1]+a[2]+\Lambda\ a[n-2]+s(a[n-1],1)。 \\ &= a[0]+a[1]+a[2]+\Lambda\ a[n-2]+a[n-1] \\ &= \sum_{i=0}^{n-1} a_i \end{aligned}$$

从上式中可以看出，程序所完成的是求数组前 $n$ 个元素的和，因此结果为 6。

答案　6

# 11.4　单元训练及参考答案

## 11.4.1　单元训练

### 一、选择题

1．有以下程序：

```
#include<string.h>
main(int argc,char *argv[])
{   int i,len=0;
    for(i=1;i<argc;i++)len+=strlen(argv[i]);
    printf("%d\n",len);
}
```

程序编译连接后生成的可执行文件是 ex1.exe，若运行时输入带参数的命令行是：

```
ex1  abcd  efg  10<回车>
```

则运行的结果是_____。

（A）22　　（B）17　　（C）12　　（D）9

2．若有以下说明和定义，则对 fun 函数的正确调用的语句是_____。

（A）*a*=fun;a(w);　　（B）*a*=fun;(*a)(&c);

（C）*b*=fun;*b(w);　　（D）fun(b);

```
main()
{   int (*a)(int*),*b(),w[10],c;
    ……
}
fun(int *c)  {  ……  }
```

3．有以下程序

```
void f(int a[], int i, int j)
{   int  t;
    if(i<j)
    {   t=a[i]; a[i]= a[j]; a[j]=t;
        f(a,i+1,j-1);      }
}
main()
{   int  i, aa[5]={1,2,3,4,5};
    f(aa,0,4);
    for(i=0;i<5;i++) printf("%d,",aa[i]);printf("\n");
}
```

执行后输出结果是_____。

（A）5,4,3,2,1,　　（B）5,2,3,4,1,　　（C）1,2,3,4,5,　　（D）1,5,4,3,2，

## 二、填空题

1．假定以下程序经编译和连接后生成可执行文件 PROG.EXE，如果在此可执行文件所在目录的 DOS 提示符下键入：

PROG　ABCDEFGH　IJKL<回车>，则输出结果为__（1）__。

```
main( int argc, char *argv[]))
{   while(--argc>0) printf("%s",argv[argc]);
    printf("\n");
}
```

2．以下程序通过函数指针 *p* 调用函数 fun，请在填空栏内写出定义变量 *p* 的语句。

```
void fun(int *x,int *y)
{ …… }
main()
{   int  a=10,b=20;
    __(2)__ ;      /定义变p */
    p=fun;  p(&a,&b);
    ……
}
```

3．下面程序的输出是__（3）__。

```
long fun(int n)
```

```
{   long s;
    if((n= =1)||(n= =2)) s=2;
    else     s=n+fun(n-1);
    return(s);
}
main()
{   long x;  x=fun(4);
    printf("%ld\n",x);
}
```

4．**递归函数** invert(int a[]，int k)**将指定数组中的前** *k* **个元素逆置。请填空。**

```
void invert(int a[] , int k);
{
    int t;
    if (   (4)   ){
        invert(   (5)   );
        t = a[0];
        a[0] = a[k-1];
        a[k-1] = t;
    }
}
```

5．**以下程序的输出结果是**___(6)___**。**

```
fun(int n,int *s)
{   int f1,f2;
    if(n==1||n==2) *s=1;
    else { fun(n-1,&f1); fun(n-2,&f2); *s=f1+f2;}
}
main()
{   int x;
    fun(6,&x); printf("%d\n",x);
}
```

## 11.4.2 参考答案

### 一、选择题

1．D　　2．B　　3．A

### 二、填空题

(1) IJKLABCDEFGH

(2) **void (*p)() 或 void(*p)(int*, int*)**

(3) 9

(4) k > 1

(5) a+1 ， k-2

(6) 8

# 第 12 章　用户标识符的作用域和存储类

## 12.1　局部变量、全局变量和存储分类

### 12.1.1　考点提炼

**📖考点 1：局部变量和全局变量**

**1．局部变量**

**在一个函数内部或复合语句内部定义的变量，称为局部变量，局部变量又叫内部变量。它只在本函数（或复合语句）范围内有效，其他的函数（或复合语句）是不能使用这些变量的。**

**2．全局变量**

**在函数外定义的变量称全局变量，全局变量又叫外部变量。它的作用域是全局的，即从定义处到程序的结束。**

**如果在程序中，外部变量与局部变量同名时，在局部变量范围内，外部变量不起作用。**

**📖考点 2：变量的存储分类**

**C 语言中，有两类存储类别，一种是自动类，一种是静态类。局部变量既可说明为自动类，也可以说明为静态类；而全局变量只能有静态类。存储类别确定了所说明对象在内存中的存储位置，从而确定了所说明对象的作用域和生存期。**

### 12.1.2　题眼分析

**选择题分析**

**【例】以下叙述中正确的是_____。（2003 年 4 月）**

**（A）全局变量的作用域一定比局部变量的作用域范围大**

**（B）静态(static)类别变量的生存期贯穿于整个程序的运行期间**

**（C）函数的形参都属于全局变量**

**（D）未在定义语句中赋初值的 auto 变量和 static 变量的初值都是随机值**

**题眼分析　若在函数中定义与全局变量名字相同的局部变量，则全局变量在该函数中将不起作用，因此全局变量的作用域并不一定比局部变量的作用域大；静态变量一旦定义，将在整个程序的运行期间都存在；函数的形参只在函数调用的时候分配存储空间，在退出函数时收回存储空间，因此是局部变量；未赋初值的 auto 型变量的初值是随机的，未赋初值的 static 型变量的初值是 0。**

**答案**　B

# 12.2 局部变量及其作用域和生存期

## 12.2.1 考点提炼

### 考点 1：auto 变量

当在函数内部或复合语句内定义变量时，如果没有指定存储类或使用了 auto 说明符，系统就被认为所定的变量具有自动类型。auto 变量有如下特征：

（1）局部变量的定义必须放在函数体（或复合语句）中全部可执行语句之前。

（2）自动类局部变量的存储单元是在进入这些局部变量所在函数体（或复合语句）时生成，退出其所在的函数体（或复合语句）时消失。当再次进行函数体（或复合语句）时系统将为它们另行分配存储单元，因此变量的值不可能保留。

（3）对于未赋初值的自动变量其值不定。

（4）自动变量赋初值是在程序运行过程中进行的，每进入一次函数体（或复合语句）时，就赋一次指定的初值。

### 考点 2：register 变量

寄存器变量也是自动变量。用 register 说明符说明变量是建议编译程序将变量的保留在 CPU 的寄存器，而不像一般变量那样，需要存放在内存单元中。register 变量有如下特征：

（1）访问 register 变量速度比访问一般变量要快。

（2）CPU 中寄存器的数目有限，只能说明小量的寄存器变量。

（3）register 变量没有地址，不能进行求地址运算。

### 考点 3：静态存储类的局部变量

在函数体（或复合语句）内部，用 static 说明符来说明的变量称为静态局部变量。静态局部变量有如下特征：

（1）静态局部变量存放在内存的静态存储区中，当退出函数后，下次进入该函数时，静态局部变量仍使用原来的存储单元，可以继续使用存储单元中原来的值。由此可知，静态局部变量的生存期将一直延长到程序运行结束。

（2）静态局部变量的初值是在编译时赋予的，在程序执行期间不再赋初值。对于未赋初值的静态局部变量，编译程序自动给它赋初值 0 或空字符。

## 12.2.2 题眼分析

### 一、选择题分析

【例 1】以下程序运行后，输出结果是_____。（2000 年 4 月）

（A）8,15　　（B）8,16　　（C）8,17　　（D）8,8

```
func(int a,int b)
{   int m=0 ,i=2;
    i+=m+1; m=i+a+b ;
    return(m);
}
main()
{   int k=4,m=1,p;
    p=func (k,m);printf("%d,",p); p=func (k,m);printf("%d\n",p);
}
```

**题眼分析**　局部变量的作用域是定义它的函数（或复合语句），不同的函数（或复合语句）可以定义同名的局部变量，所以本题中 main()函数和 func()函数中定义的同名变量 *m* 是不同的变量，其作用的范围也仅是定义它的函数。在主函数中第 1 次执行函数调用“ func(k,m) ”，把 *k* 和 *m* 的值传给形参 *a* 和 *b*，*a* 和 *b* 的值分别是 4 和 1。进入函数 func 后首先定义了两个变量 *m* 和 *i* 并给它们赋初值 0 和 2，然后执行语句“ i+=m+1; ”，*i* 的值变成 3，执行语句“ m=i+a+b ; ”后，*m* 的值变成 8，把 *m* 的值作为函数值返回并赋值给 *p*，输出 *p* 的值为 8。第 1 次函数调用后，返回到 main()函数，main()函数中的 *k* 和 *m* 的值并没有变化。接着再执行了一次函数调用 func(k,m)，执行过程与第 1 次完全一样，返回值也为 8 并赋给 *p*，输出的 *p* 值为 8。

**答案**　D

**【例 2】执行下面的程序段后，变量 k 中的值为_____。（2000 年 4 月）**

**（A）　不定值　　（B）　33　　（C）　30　　（D）　10**

```
int k=3, s[2]; s[0]=k; k=s[1]*10;
```

**题眼分析**　若数组为 auto 型，没有进行初始化，其元素的值为不确定。本例虽然给 s[0]赋了一个值，但 s[1]的值依旧是不确定的，所以执行语句“ k=s[1]*10; ”后，*k* 的值也不确定。

**答案**　A

**【例 3】下面程序的输出是_____。**

**（A）3　　（B）4　　（C）6　　（D）9**

```
fun3(int x)
{  static int a=3; a+=x; return(a);}
main()
{  int k=2, m=1, n;
   n=fun3(k); n=fun3(m);
   printf("%d\n",n);
}
```

**题眼分析**　在主函数中执行语句“ n=fun3(k); ”时，把 *k* 的值 2 传给形参 *x*，在 fun3()函数中定义了一个静态变量 *a* 并赋初值 3，这一初值是在编译时就赋予的，在程序执行过程中不再赋初值。当执行语句“ a+=x; ”后，a 的值变为 5，通过“ return(a) ;”语句把 *a* 作为函数值返回并赋给 *n*；执行语句“ n=fun3(m); ”时，把 *m* 的值 1 传给形参 *x*，在 fun3()函数中的静态变量 a 将保留上一次退出时的值 5，执行语句“ a+=x; ”后，*a* 的值变为 6，通过“ return(a) ;”语句把 *a* 作为函数值返回并赋值给 *n*，*n* 的值是 6。

**答案**　C

**二、填空题分析**

**【例】以下程序的输出结果是_____。（2000 年 9 月）**

```
void fun()
{  static int a=0;
   a+=2; printf("%d",a); }
main()
{  int cc;
   for(cc=1;cc<4;cc++) fun() ;
printf("\n");
}
```

**题眼分析**　**在主函数中通过一个** for **循环调用了** 3 **次** fun()**函数。第** 1 **次调用** fun()**函数执行过程如下：首先定义一个静态变量** *a* **并赋初值** 0**，执行语句“** *a*+=2; **”后，静态变量** *a* **的值变为** 2**，执行“** printf("%d",*a*); **”语句输出** *a* **的结果是** 2**；第** 2 **次调用** fun()**函数执行过程如下：静态变量** *a* **保留上一次退出时的值** 2**，执行语句“** *a*+=2; **”后，静态变量** *a* **的值变为** 4**，执行“** printf("%d",*a*); **”语句输出** a **的结果是** 4**；第** 3 **次调用** fun()**函数执行过程如下：静态变量** *a* **保留上一次退出时的值** 4**，执行语句“** *a*+=2; **”后，静态变量** *a* **的值变为** 6**，执行“** printf("%d",*a*); **”语句输出** *a* **的结果是** 6**。**

**答案**　2 4 6**。**

# 12.3　全局变量及其作用域和生存期

## 12.3.1　考点提炼

**考点** 1**：全局变量的作用域和生存期**

**全局变量是在函数外部任意位置上定义的变量，它的作用域是从变量定义的位置开始，到整个源程序结束止。全局变量的生存期是整个程序的运行期间。**

**若全局变量和某一个函数中的局部变量同名，则在该函数中，此全局变量被屏蔽，在该函数内，访问的是局部变量，与同名的全局变量不发生任何关系。**

**考点** 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**

**当全局变量定义在引用它的函数之前时，则应该在该函数中用关键字** extern **对此全局变量进行说明，表示该变量是一个已在函数的外部定义了的全局变量，并已经分配了存储单元，不需要再为它另外开辟存储单元。**

**注意：全局变量的说明和全局变量的定义的含义是不同的。全局变量的定义只能出现一次，定义时将为该变量开辟存储单元，而全局变量的说明可以多次出现在需要的地方。**

**考点 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**

**当一个程序由多个编译单位组成，并且在每个文件中均需要引用同一个全局变量。这时可以在其中一个文件中定义全局变量，而在其他用到这个全局变量的文件中用 extern 说明符对这个全局变量进行说明。**

**考点 4：静态全局变量**

**当用 static 说明符说明全局变量时，此变量称为静态全局变量。静态全局变量只限于本编译单位使用，不能被其他编译单位所引用。**

## 12.3.2　题眼分析

### 一、选择题分析

**【例 1】以下叙述中正确的是_____。（2004 年 9 月）**

**（A）局部变量说明为 static 存储类，其生存期将得到延长**

**（B）全局变量说明为 static 存储类，其作用域将被扩大**

**（C）任何存储类的变量在未赋初值时，其值都是不确定的**

**（D）形参可以使用的存储类说明符与局部变量完全相同**

**题眼分析　局部变量说明为 static 存储类，在整个程序运行期间都不释放，所以其生存期比局部动态变量的生存期长，因此选项（A）是正确答案。全局变量不论是静态的还是动态的，它的作用域都是从变量定义的位置开始，到整个源程序结束止，因此选项（B）是错误的。对于动态变量如果未赋初值，其值都是不确定的，但对于静态变量如果未赋初值，程序编译时会自动为其赋初值 0 或空字符，所以选项（C）也是错误的。函数在未调用前，函数的形参不占内存空间，只有在调用时才动态分配存储空间，所以函数形参不能说明为静态存储，而局部变量可以说明为静态存储，所以选项（D）也是错误的。**

**答案**　A

**【例 2】以下程序的输出结果是_____。（2002 年 4 月）**

```
int x=3;
main()
{   int i;
    for (i=1;i<x;i++) incre();
}
incre()
{   staic int x=1; x*=x+1;
    printf(" %d",x);
}
```

（A）3　3　　（B）2　2　　（C）2　6　　（D）2　5

**题眼分析　本题首先定义一个全局变量 *x* 并赋初值 3，主函数中使用这个全局变量控制循环次数，循环执行了 2 次，调用两次 incre()函数。第 1 次调用 incre()，定义一个静态变量 *x* 并赋初值 1，然后执行“ *x**=*x*+1; ”，使 *x* 的值变为 2，输出 *x* 的值为 2；第 2 次调用 incre()函数时，**

静态变量将保留上一次退出时的值即 2，执行语句“ *x**=*x*+1; ”后，*x* 的值变成 6，输出 *x* 的值为 6。

**答案** C

**【例 3】有以下程序**

```
int  a=2;
int f(int  *a)
{   return(*a)++;}
main()
{   int  s=0;
    {   int  a=5;
        s+= f(&a)
    }
    s+= f(&a)
    printf("%d\n" ,s)
}
```

执行后输出结果是_____。（2004 年 4 月）

（A）10　　（B）9　　（C）7　　（D）8

**题眼分析** 该程序共定义了 3 个用户标识符 *a*，但它们的含义和作用域并不相同。程序开始时定义的“ int *a*=2 ;”，它是一个全局变量；函数 *f* 的形参**a* 是一个局部变量，其作用域是该函数内部；主函数 main 中定义的“ int *a*=5 ;”，它也是一个局域变量，其作用域只是在该复合语句内部。主函数 main 中调用了两次函数 *f*，第 1 次调用使用的是该复合语句的局部变量 *a*，调用结束后，*s* 的值为 5；第 2 次调用使用的是全局变量 *a*，调用结束后，*s* 的值为 7。

**答案** C

## 二、填空题分析

**【例 1】以下程序运行后的输出结果是_____。（2003 年 9 月）**

```
int  a=5;
fun(int  b)
{   static int a=10;
    a+=b++;
    printf("%d  ",a);   }
main()
{   int  c=20;
    fun(c);
    a+=c++;
    printf("%d\n",a);
}
```

**题眼分析** 该程序定义了两个用户标识符 *a*，程序开始时定义的“ int *a*=5 ;”，它是一个全局变量；函数 fun 中定义的“ static int *a*=10; ”是一个局部静态变量，其作用域是该函数内部。主函数 main 中调用函数 fun 时，把 *c* 值传送给形参 *b*，在执行“ *a*+=*b*++; ”时，起作用的是局部静态变量 *a*，执行后该变量的值变为 30。返回主函数 main 后，在执行“ *a*+=*c*++; ”时，起作用的全局变量 *a*，执行后该变量的值变为 25。

**答案**　30 25

**【例 2】下列程序的输出结果为_____**

```
int  a=0,b=0;
fun()
{   int a=5;printf("%d,%d",a,b);          }
main()
{   b=5; fun();printf("%d,%d\n",a,b);     }
```

**题眼分析**　主函数执行时，先给全局变量 *b* 赋一个值 5，然后调用函数 fun()，fun()函数中定义一个局部变量 *a*。它和全局变量 *a* 不是一个变量，给局部变量赋一个初值 5 并不影响到全部变量 *a*，故在 fun()函数中输出的 *a* 为局部变量值，是 5，输出的 *b* 是全局变量的值，是 5。fun()函数调用返回后，在 main()函数中输出变量 *a* 和变量 *b*，它们都是全局变量，故 *a* 的值为 0，*b* 的值是 5。

**答案**　5,5 0,5

# 12.4　函数的存储分类

## 12.4.1　考点提炼

一个用 C 语言编写的程序通常由许多编译单位（源程序文件）组成，根据定义的函数能否被其他编译单位调用，可将函数分成内部函数和外部函数。

1．用 extern 说明函数

外部函数是指可以被其他编译单位（文件）中的函数调用的函数，在定义函数的时候如果把函数的存储类型定义成“ extern ”，则此函数就为外部函数。其定义格式如下：

格式：extern　类型说明符　函数名(形参说明列表)

如果在定义函数时函数的存储类型缺省，则为“ extern ”，即外部函数。

2．用 static 说明函数

如果一个函数只能被本编译单位（源文件）中的其他函数调用，而不能被其他编译单位（源文件）中的函数调用，称它为内部函数，有时也称静态函数。在定义内部函数时，其函数的存储类型符应为 static。其定义格式如下：

格式：static　类型说明符　函数名(形参说明列表)

## 12.4.2　题眼分析

**【例】在 C 语言中，函数的隐含存储类别是_____。**

（A）auto　　（B）static　　（C）extern　　（D）无存储类别

**题眼分析**　此题考核的考点是函数的存储类别。C 语言中函数的存储类别有“ static ”和“ extern ”两种，缺省时默认为“ extern ”。

**答案**　C

# 12.5 单元训练及参考答案

## 12.5.1 单元训练

### 一、选择题

1．凡是函数中未指定存储类别的局部变量，其隐含的存储类别为_____。

（A） auto （B）static （C）extern （D）register

2．以下叙述中不正确的是_____。

（A）在 C 中，函数中的自动变量可以赋初值，每调用一次，赋一次初值

（B）在 C 中，在调用函数时，实参和对应形参在类型上只需赋值兼容

（C）在 C 中，外部变量的隐含类别是自动存储类别

（D）在 C 中，函数形参可以说明为 register 变量

3．以下叙述中不正确的是_____。

（A）在不同的函数中可以使用相同名字的变量

（B）函数中的形式参数是局部变量

（C）在一个函数内定义的变量只在本函数范围内有效

（D）在一个函数内的复合语句中定义的变量在本函数范围内有效

4．以下只有在使用时才为该类型变量分配内存的存储类说明是_____。

（A）auto 和 static （B）auto 和 register

（C）register 和 static （D）extern 和 register

5．在 C 语言中，形参的缺省存储类是_____。

（A）auto （B）register （C）static （D）extern

6．下列程序执行后输出的结果是_____。

```
int d=1;
fun (int p)
{    int d=5; d +=p++; printf("%d",d); }
main( )
{   int a=3;
    fun(a);   d+= a++;
    printf("%d\n",d);   }
```

（A）8 4 （B）9 6 （C）9 4 （D）8 5

### 二、填空题

1．以下程序的输出结果是___（1）___。

```
int fun(int x,int y)
{   static int m=0,i=2;
    i+=m+1;  m=i+x+y;
    return m    }
```

```
main()
{   int j=4,m=1,k;
    k=fun(j,m); printf("%d,",k);
    k=fun(j,m); printf("%d\n",k);      }
```

2．以下程序执行的结果是___(2)___。

```
int x=1;
main()
{   func(x);    printf("x=%d\n",x);}
func(int x)
{   x=3;    }
```

3．以下程序执行的结果是___(3)___。

```
/*void  func(int a);*/
main()
{   int k=4;    func(k);    func(k);    }
func(int a)
{   static int m=0;
    m+=a;
    printf("x=%d, ",m);    }
```

4．以下程序执行的结果是___(4)___。

```
main()
{   int swap();
    extern int a,b;
    a=3;b=10;
    swap();
    printf("a=%d,b=%d\n",a,b);
}
int a,b;
swap()
{   int temp;   temp=a; a=b; b=temp; }
```

## 12.5.2　参考答案

### 一、选择题

1．A　　2．C　　3．D　　4．B　　5．A

6．A

### 二、填空题

(1) 8,17　　(2) x=1

(3) x=4,x=8,　　(4) a=10,b=3

# 第 13 章　编译预处理和动态存储分配

## 13.1　编译预处理

### 13.1.1　考点提炼

**📖考点 1：无参数宏定义**

**格式：**# define　**宏名称　一串符号**

**功能：在编译预处理时，将“ 宏名称 ”替换成“ 一串符号 ”。其中宏名称为标识符，一串符号为任意的一个字符串。**

**说明：**

**（1）名称对应一串符号。**

**（2）宏名称前后应该有空格，以便辨认宏名称。**

**（3）宏的编译预处理命令不是语句，其后不要跟分号（;）。**

**（4）在一串符号中如果出现运算符，需要注意替换后的结果，为了方便可在合适的位置加上括号。**

**（5）C 语言规定，宏名称如果出现在字符串常量中，将不作为宏名称处理，不进行替换。**

**（6）在宏定义的一个字符串中可以出现已经定义过的另一个宏名称，称为嵌套宏定义。**

**📖考点 2：带参数的宏定义**

**C 语言规定，在宏定义时可以使用参数（包括形式参数和实际参数）；宏替换时先将实际参数替换成形式参数，然后再进行宏替换，这样就增强了宏的功能。**

**带有参数宏的格式为：**

**格式：**#define　**宏名称(形式参数)　一串符号**

**功能：其中“ 宏名称 ”通常是有大写字母组成的标识符，形式参数是由逗号分隔开的若干个形式参数表。在程序开始编译前，程序中所有引用“ 宏名称(实参表) ”被替换成对应的一串符号，然后开始编译程序。**

**说明：在定义带参数的宏时，除了在不带参数的宏定义中要注意的几点外，还要注意在宏的实参中使用表达式，要将相应的表达式加上圆括号，作为一个整体来处理。**

**📖考点 3：终止宏定义**

**可以用**#undef **提前终止宏定义的作用域。所定义宏的作用域从**#define **命令行开始，到**#undef **命令行结束。**

**考点 4：文件的包含处理**

**“文件包含”是 C 语言的编译预处理命令之一。在程序进行预编译处理时，用指定的“包含文件名”中的内容替代该语句。“文件包含”的命令格式如下：**

**格式 1：**#include "**包含文件名**"

**格式 2：**#include <**包含文件名**>

**功能：在编译预处理时，用指定的“包含文件名”中的文本内容替代该命令，使包含文件成为本程序的一部分。用“格式 1”系统先在本程序文件所在的磁盘和路径下寻找包含文件，若找不到，再按系统规定的路径搜索包含文件。用格式 2 系统仅按指定的路径去搜索包含文件。**

**说明：**

**（1）“文件包含”命令一般要求放在程序的头部，因为系统的函数及某些宏的定义都放在系统文件中。这些文件被称为“头文件”，“头文件”的扩展名一般为“h”**

**（2）通常使用格式 1，这样可减少了寻找包含文件出错。**

**（3）包含文件的内容必须是 C 语言程序，因为包含文件的内容要全部出现在源程序中。**

**（4）包含文件除了可以将系统函数和系统宏定义包含到用户程序中，还可以将多个程序合并到一个程序中进行编译。**

## 13.1.2　题眼分析

### 一、选择题分析

**【例 1】以下叙述中正确的是_____。（2005 年 4 月）**

**（A）预处理命令行必须位于源文件的开头**

**（B）在源文件的一行上可以有多条预处理命令**

**（C）宏名必须用大写字母表示**

**（D）宏替换不占用程序的运行时间**

**题眼分析　预处理命令行可以定义在程序的任意位置，而不必位于源文件的开头，因此选（A）是错误的；在定义预处理命令时，一行只能有一条预处理命令，因此选项（B）也是错误的；宏名可以是大写字母，也可以是小写字母，用大写字母的目的是方便与程序中的其他标识符区别，因此选项（C）也是错误的。宏替换是在程序编译时完成，因此它不占用程序的运行时间，因此选项（D）是正确答案。**

**答案**　D

**【例 2】有以下程序**

```
#define  f(x)  x*x
main()
{   int  i;
    i=f(4+4)/f(2+2);
    printf("%d\n", i);  }
```

**执行后输出结果是_____。（2004 年 4 月）**

**（A）28　　（B）22　　（C）16　　（D）4**

**题眼分析　程序编译时先进行宏替换，将“** f(4+4) **”替换为“** 4+4*4+4 **”,将** f(2+2) **替换为“** 2+2*2+2 **”。因此表达式“** *i*=f(4+4)/f(2+2); **”将被变成“** *i*=4+4*4+4/2+2*2+2 **;”，程序运行时，** *i* **的值将被赋值为** 28**。**

**答案**　A

**【例** 3**】以下叙述正确的是_____。（**2002 **年** 4 **月）**

**（**A**）可以把** define **和** if **定义为用户标识符**

**（**B**）可以把** define **定义为用户标识符，但不能把** if **定义为用户标识符**

**（**C**）可以把** if **定义为用户标识符，但不能把** define **定义为用户标识符**

**（**D**）**define **和** if **都不能定义为用户标识符**

**题眼分析　此题考核的考点是** C **语言中的保留字。**if **是保留字，而** define **不是保留字。用户标识符不允许使用保留字。**

**答案**　B

**【例** 4**】程序中头文件** type1.h **的内容是_____。（**2002 **年** 9 **月）**

```
#define  N  5
#define  M1   N*3
```

**程序如下：**

```
#include  "type1.h"
#define  M2  N*2
main( )
{   int i;  i=M1+M2;printf("%d\n",i);       }
```

**程序编译后运行的输出结果是：**

**（**A**）**10　　**（**B**）**20　　**（**C**）**25　　**（**D**）**30

**题眼分析　编译预处理时，用“** type1.h **”中的内容替代命令“** #include "type1.h" **”。表达式“** *i*=M1+M2 **”经过宏替换为“** *i*=5*3+5*2 **”。**

**答案**　C

## 二、填空题分析

**【例** 1**】以下程序的输出结果是_____（**2003 **年** 4 **月）**

```
#define  MCRA(m)    2*m
#define  MCRB(n,m)  2*MCRA(n)+m
main()
{   int i=2,j=3;
    printf("%d\n",MCRB(j,MCRA(i)));
}
```

**题眼分析　首先将程序中的宏替换掉，先把“** MCRA(*i*) **”替换成“** 2**i* **”，然后把“** MCRB(j,2**i*) **”替换成“** 2*2**j*+2**i* **”。输出为** 16**。**

**答案**　16

**【例** 2**】以下程序执行的结果是_____。（**2002 **年** 9 **月）**

```
#define N  10
#define s(x)   x*x
#define f(x)   (x*x)
```

```
main()
{   int i1,i2;
    i1=1000/s(N); i2=1000/f(N);
    printf("%d  %d\n",i1,i2);  }
```

**题眼分析**　首先将程序中的宏替换掉，两处的宏替换后分别为 *i*1=1000/10*10 和 *i*2=1000/(10*10)。

**答案**　1000　10

**【例 3】以下程序运行后的输出结果是_____。（2005 年 4 月）**

```
#define S(x) 4*x*x+1
main()
{   int i=6,j=8;
    printf("%d\n",S(i+j));
}
```

**题眼分析**　程序编译时先进行宏替换，替换的方法是：第 1 步用“ *i*+*j* ”替换宏定义 S(*x*) 的 *x*，即得到“ 4**i*+*j***i*+*j*+1 ”；第 2 步用“ 4**i*+*j***i*+*j*+1 ”来替换程序中“ S(*i*+*j*) ”。因此程序运算结果为 81。

**答案**　81

# 13.2　动态存储分配

## 13.2.1　考点提炼

### 考点 1：malloc 函数

malloc 函数的调用格式是：

malloc(size);

其功能是分配 size 个字节的存储区，返回一个指向存储区首地址的基类型为 void 的地址。如果没有足够的内存单元分配，则返回空（NULL）。

在调用 malloc 函数时需要注意以下几点：

（1）size 的类型必须是 unsigned int。

（2）malloc 函数返回的地址为 void *(无值型)，因此在调用时，必须利用强制类型转成所需要的类型。

（3）由于动态分配得到的存储单元没有名字，只能靠指针来引用它。一旦指针改变指向，原存储单元所存储的数据将无法引用。

（4）通过 malloc 函数所分配的动态存储单元中没有确定的初始值。

（5）在分配时，若不能确定数据类型所占字节数，可以使用 sizeof 运算符来求得。

### 考点 2：calloc 函数

calloc 函数的调用格式是：

calloc(n,size);

**其功能是给** *n* **个同一类型的数据项分配连续的存储空间。每一个数据项的长度为** size **个字节。若分配成功，则返回一个指向存储区首地址的基类型为** void **的地址。如果没有足够的内存单元分配，则返回空（**NULL**）。**

**在调用** calloc **函数时需要注意以下几点：**

**（**1**）** *n* **和** size **的类型必须是** unsigned int**。**

**（**2**）以** calloc **函数返回的地址为** void *(**无值型**)**，因此在调用时，必须利用强制类型转成所需要的类型。**

**（**3**）与** malloc **函数不同，**calloc **函数所分配的存储单元，系统自动置初值** 0**。**

**考点** 3**：**free **函数**

free **函数的调用格式是：**

free(p);

**其功能是将** *p* **所指的存储空间释放，使这部分空间可以由系统重新支配。该函数没有返回值。**

## 13.2.2 题眼分析

### 一、选择题分析

**【例】若指针** *p* **已正确定义，要使** *p* **指向两个连续的整型动态存储单元，正确的语句是_____。（**2002 **年** 4 **月）**

（A）p=2*(int*)malloc(sizeof(int));　　（B）p=(int*)malloc(2*sizeof(int));

（C）p=(int*)malloc(2*2);　　（D）p=(int*)malloc(2,sizeof(int));

**题眼分析　该函数调用格式是“** malloc(n) **”，作用是申请** *n* **个字节的存储单元并把该存储区的首地址作为返回值。实际调用的时候可用在前面加上“** (**类型说明符　***) **”转换成需要的类型地址。可见选项（**D**）不符合** malloc **函数的调用格式，多了一个参数；整型变量在有的机器上占** 4 **个字节，并不一定是两个字节，所以选项（**C**）也不精确；答案（**A**）是先申请一个整型的存储空间并把该存储空间的首地址***2 **赋值给** p**，显然也是不正确的。只有选项（**B**）符合题意要求。**

**答案**　B

### 二、填空题分析

**【例** 1**】以下程序中给指针** *p* **分配** 3 **个** double **型动态内存单元,请填空。（**2004 **年** 4 **月）**

```
#include  <stdlib.h>
main()
{   double  *P;
    P=(double*)malloc(_____);
    p[0]=1.5;p[1]=2.5;p[2]=3.5
    printf("%f %f%f\n",p[0], p[1],p[2]);   }
```

**题眼分析**　需要分配 3 个 double 型动态内存单元，每个 double 型所占用的空间是可用 sizeof(double)表示，因此 3 个 double 型动态内存单元数为 sizeof(double)*3。

**注意：这里不能直接使用 8 来代替 sizeof(double)，虽然在一般计算机系统中，double 型被分配 8 个字节，但有的计算机却不是。**

**答案**　sizeof(double)*3

**【例 2】若要使指针 *p* 指向一个 double 类型的动态存储单元，请填空。（2000 年 9 月）**

```
p=_____malloc(sizeof(double));
```

**题眼分析**　函数 malloc 返回的是 void*，所以若要使指针指向一个 double 类型，则必须进行类型转换，类型转换格式为：(数据类型*)。

**答案**　(double*)

# 13.3　单元训练及参考答案

## 13.3.1　单元训练

**一、选择题**

1．以下有关宏替换的叙述不正确的是_____。

（A）宏替换不占用运行时间　　（B）宏名称无类型

（C）宏替换只是字符替换　　（D）宏名称必须用大写字母表示

2．C 语言的编译系统对宏命令的处理是_____。

（A）在程序运行时进行的

（B）在程序连接时进行的

（C）和 C 程序中的其他语句同时进行编译的

（D）在对源程序中其他成分正式编译之前进行的

3．以下程序的输出结果是_____。

```
#define M(x,y,z) x*y+z
main()
{   int a=l,b=2,c=3;
    printf("%d\n",M(a+b,b+c,c+a));
}
```

（A）19　　（B）17　　（C）15　　（D）12

4．以下程序的输出结果是_____。

```
#define SQR(X) X*X
main()
{   int a=16,k=2,m=1;
    a/=SQR(k+m)/SQR(k+m);
printf("%d\n",a);
}
```

（A）16　　（B）2　　（C）9　　（D）1

5．若有宏定义如下_____。

```
#define A 5
#define b A+1
#define c b*A/2
```

则执行以下 printf 语句，输出的结果是：

```
int a; a=A; printf("%d,",c);
printf("%d\n",--a);
```

（A）7,6　　（B）12,6　　（C）12,5　　（D）7,5

6．若有以下宏定义_____。

```
#define N 2
#define Y(a) ((N+1)*a)
```

执行语句 b=2*(N+Y(5));后 b 的结果是_____。

（A）语句有错误　　（B）b=34　　（C）b=70　　（D）b 值不定

7．若有宏定义_____。

```
#define MOD(a,b) a%b
```

则执行以下语句的输出为：

```
int x,y=15,z=100;
x=MOD(z,y); printf("%d\n",x++);
```

（A）11　　（B）10　　（C）6　　（D）宏定义不合法

8．#define 能做简单的代替，用宏代替计算多项式 5*x*x+4*x+3 之值的函数 f()正确的定义是_____。

（A）#define f(x)　5*x*x+4*x+3　　（B）#define f　5*x*x+4*x+3

（C）#define f(x)　(5*(x)*(x)+4*(x)+3)　　（D）#define　(5*x*x+4*x+3)　f(x)

9．对下面的程序段：

```
#define A  3
#define B(a) ((A+1)*a)
……
x=3*(A+B(7));
```

正确的判断是_____。

（A）程序错误，不许嵌套定义　　（B）*x*=93

（C）*x*=21　　（D）程序错误，宏定义不许有参数

10．在以下的任何情况下计算平方数都不会引起二义性的宏定义是_____。

（A）#define power(x)　x*x　　（B）#define power(x)　(x)*(x)

（C）#define power(x)　(x*x)　　（D）#define power(x)　((x)*(x))

11．若要用下面的程序片段使指针变量 *p* 指向一个存储整型变量的动态存储单元

```
int *p;
p=_____malloc(sizeof(int));
```

则应填入_____。

（A）int　　（B）int*　　（C）(*int)　　（D）(int*)

**二、填空题**

1．**设有以下宏定义：**

```
#define WIDTH 80
#define LENGTH WIDTH+40
```

**则执行赋值语句：***a*=LENGTH*20；(*a* 为 int x）后，*a* 的值是___（1）___。

2．**用以下语句调用库函数** malloc()，**使字符指针** st **指向具有** 11 **个字节的动态存储空间，请填空。**

```
st=(char*) ___(2)___;
```

3．**下面程序的运行结果**___（3）___。

```
#define DOU(R)  R*R
main()
{  int a=1,b=2,c;
   c=DOU(a+b);  printf("%d\n",c);
}
```

4．**下面程序的运行结果是**___（4）___。

```
 #define M(z) (z)*(z)
 main()
 {   printf("%d\n",M(1+2+3);}
```

5．**下面程序的运行结果**___（5）___。

```
#define POW(x) ((x)*(x))
main()
{   int j=1;
    while (j<=4) printf("%d\n",POW(j++));
}
```

## 13.3.2　参考答案

**一、选择题**

| | | | | |
|---|---|---|---|---|
| 1．D | 2．D | 3．D | 4．B | 5．D |
| 6．B | 7．B | 8．C | 9．B | 10．D |
| 11．D | | | | |

**二、填空题**

（1）880
（2）malloc(11)
（3）5
（4）12
（5）2
　　12

# 第 14 章　结构体、共用体和用户定义类型

## 14.1　用 typedef 说明一种新类型名

### 14.1.1　考点提炼

**📖考点 1：用 typedef 说明一种新类型名**

C 语言允许用户用 typedef 定义一种新的类型名，一般形式为：

typedef　类型名　标识符。

其中，“类型名”必须是在此语句之前已经有定义的类型标识符；“标识符”是一个用户定义标识符，用做新的类型名。

说明：

（1）typedef 语句的作用是用“标识符”来代表已存在的“类型名”，并未产生新的数据类型。

（2）原有类型依然有效。

（3）为了便于识别，一般习惯将新的类型句用大写字母表示。

**📖考点 2：说明一种新类型名的步骤**

说明一种新类型名的步骤如下：

（1）首先按通常定义变量的方法写出定义的主体。例如：

```
char    *p;
```

（2）将变量名换成新的类型名：

```
char    *CHARP;
```

（3）在最左边加上关键字 typedef：

```
typedef    char    *CHARP;
```

（4）可以用新类型名定义变量了：

```
CHARP p;
```

### 14.1.2　题眼分析

【例 1】若要说明一个类型名 STP，使得定义语句 STP　*s;等价于 char　*s;，以下选项中正确的是_____。（2003 年 4 月）

（A）typedef STP char *s;　　（B）typedef *char　STP;

（C）typedef STP *char；　　（D）typedef char * STP;

题眼分析　指针类型的自定义形式如 typedef 类型说明符　*用户类型名。不难看出只有

答案（D）是正确的。

**答案**　D

**【例 2】若有以下说明和定义：**

```
typedef  int  *INTEGER;
INTEGER  p,*q;
```

以下叙述正确的是_____。（2002 年 9 月）

（A）*p* 是 int 型变量　　（B）*p* 是基类型为 int 的指针变量

（C）*q* 是基类型为 int 的指针变量　　（D）程序中可用 INTEGER 代替 int 类型名

**题眼分析**　INTEGER 是类型名，由它来间接定义 *p* 和*q 的类型，因此，*p* 是基类型为 int 的指针变量，*q* 是二级指针变量，它指向基类型是 int 的指针变量。

**答案**　B

# 14.2　结构体类型

## 14.2.1　考点提炼

### 考点 1：结构体类型定义

**格式：**struct 结构名

{　成员说明列表　};

**说明：**

（1）结构体类型通常由多个成员组成，各个成员类型可相同也可不同。

（2）每个成员的定义形式可写成：类型说明符 成员名;。

可用如下所示定义一个学生数据类型：

```
struct student
{   int      NO;           /*学号*/
    char     name[20];       /*姓名*/
    int      age;           /*年龄*/
    char     sex;           /*性别*/
    char     grade;         /*年级*/
    float     score[5];       /*5门课成绩*/
};
```

### 考点 2：结构体类型变量的定义与初始化

结构体类型变量的定义有 4 种形式：

（1）定义结构型的同时定义结构体变量并初始化。

如定义学生结构体类型的同时定义变量 *s*1 和 *s*2 的形式如下，并给 *s*1 初始化：

```
struct student{/* 略*/}s1={1, "TongAiHong",32, 'M',6,{65,76,89}},s2;
```

（2）先定义一个结构体类型，然后使用该类型来定义结构体变量并初始化。例如：

```
struct student{/* 略*/};
……
struct student s1={1, "TongAiHong",32, 'M',6,{65,76,89}},s2;
```

（3）定义一个无名称的结构体类型的同时定义结构体变量并初始化。例如：

```
struct{/* 略*/}s1={1, "TongAiHong",32, 'M',6,{65,76,89}},s2;
```

（4）先使用 typedef 说明一个结构体类型名，再用新类型名定义变量。例如：

```
typedef struct{/* 略*/}ST
ST s1={1, "TongAiHong",32, 'M',6,{65,76,89}},s2;
```

### 考点 3：结构体成员的引用方法

对于结构体主要是引用的它的成员，结构体成员的引用方法如下：

结构变量名.成员名

如上述定义要给变量 s2 的年龄赋值 32，可使用下面的语句：

```
s2.age=32;
```

注意：在定义结构体类型的同时，其成员也可以是一个结构体，此时要寻找内层的结构体成员变量名，可通过下述形式：

外层结构体变量名.内存结构体成员名.成员名

### 考点 4：结构体数组的定义与初始化

结构体数组的定义与初始化有以下几种方式：

（1）定义结构型的同时定义结构体数组并初始化。如定义学生结构体类型的同时定义具有 10 个元素的数组 *s* 并给它前两个元素赋初值。

```
struct student{/* 略*/}s[10]={ {1, "TongAiHong",32, 'M',6,{65,76,89}},
                               {2, "TongEn",31, 'M',4,{65,76,89}}};
```

（2）先定义一个结构体类型，然后使用该类型来定义结构体数组并初始化。例如：

```
struct Student{/* 略*/};struct student s[10]={ {1, "TongAiHong",32,
'M',6,{65,76,89}},  {2, "TongEn",31, 'M',4,{65,76,89}}};
```

（3）定义一个无类型的结构体的同时定义结构体数组并初始化。例如：

```
struct{/* 略*/}student s[10]={ {1, "TongAiHong",32, 'M',6,{65,76,89}},
                               {2, "TongEn",32, 'M',4,{65,76,89}}};
```

### 考点 5：结构体数组元素的成员的引用

结构体数组元素的成员引用的一般形式如下：

数组名[下标].成员名

如要给上面定义的 *s* 数组的元素 s[3]的成员 age 赋值 23，所用语句如下：

```
s[3].age=23;
```

### 考点 6：指向结构体变量的指针变量的使用

结构体变量也有地址，因此可以把它的地址赋值给一个指针变量，然后通过该指针变量来引用结构体的成员。其引用形式如下：

(*指针变量名).成员名　或　指针变量名->成员名

注意：要使用指针运算符，则括号不可以省略。

例如：有以下定义：

```
struct student s1={1, "TongAiHong",32, 'M',32,{65,76,89}},s2,*p=&s1;
```

则利用指针变量 *p* 把 *s*1 的年龄增加 1 岁可有两种形式，分别为：

```
(*p).age=(*p).age+1;   或  p->age=p->age+1;
```

此处运算符“ -> ”为指向运算符。

### 考点 7：指向结构体一维数组的指针变量的使用

当指针变量指向了某一维数组的首元素后，利用它引用数组元素的成员形式如下：

```
(*(指针变量名+i)).成员名    或   (指针变量名+i).成员名
```

例如，有以下定义：

```
struct student s[10]={ {1, "TongAiHong",32, 'M',6,{65,76,89}},
                       {2, "TongEn",32, 'M',4,{65,76,89}}},*p=s;
```

则利用指针变量 *p* 把 *s*[2]的年龄增加 1 岁可有两种形式，分别为：

```
(*(p+2)).age=(*(p+2)).age+1;  或   (p+2)->age=(p+2)->age+1;
```

### 考点 8：函数之间结构体变量的数据传递

函数之间结构体变量的数据可以通过以下几种方法进行传递：

（1）向函数传递结构体变量的成员。

结构体变量中的每个成员可以参与所属类型允许的任何操作。向函数传递结构体变量的成员和传递一般变量的方法一样。

（2）向函数传递结构体变量。

使用结构体变量做实参，传递给函数对应的形参是它的值，因此函数体内对形参结构变量中任何成员的操作，都不影响对应的实参中成员的值。

（3）向函数传递结构体变量的地址。

结构体变量的地址作为实参时，对应的形参应该是一个基类型相同的结构体类型的指针。在程序运行时，系统只为形参指针开辟一个存储单元存放实参结构体变量的地址值，不必另行建立一个结构体变量。函数中，修改形参指针所指结构变量成员的值就是修改实参的值。

（4）函数的返回值可以是结构体类型。

（5）函数的返回值可以是指向结构体变量的指针类型。

### 考点 9：用指针和结构体构成链表的方法

将若干个数据项按地址链接在一起就形成了链表。链表是一种常用的数据结构，它动态地进行存储分配。链表中每一环节称结点，每一结点包含两部分内容，一是数据，二是指向下一结点的指针。链表的连接原则是：前一结点指向下一结点，通过前一结点才能找到下一结点。为了确定链表中的第 1 个结点，需设置一个指向第 1 个结点的头指针，为了标识链表的结束，把最后一个结点的指针域设置为 NULL（空指针）。

采用结构体的类型时，允许定义一个指针域，该指针可以指向定义的结构型，可以使用带

用指针域的结构体来构成链表。例如有以下结构体类型的定义：

```
struct    node{ int data; struct link_node    *next; }
```

利用该结构体类型，可以构成如图 14-1 所示的链表。

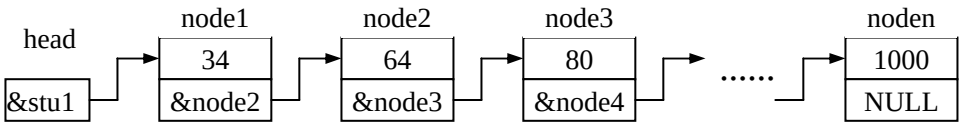



图 14-1　链表的形式

### 考点 10：单向链表的建立、输出、删除与插入

1．链表的建立

（1）在程序中定义两个指向结点类型的指针，并把 head 赋值 NULL。

```
struct  node *head,*p;  head=NULL;
```

（2）使用 malloc()函数为第 1 个结点分配存储空间，并将该空间的地址赋给指针变量 *p*，使 p 指向该结点。

```
p=(struct node *)malloc(sizeof*(struct node));
```

（3）将 head 的值赋给 *p*->next，并将 *p* 的值赋给 head，使 head 也指向该结点。

```
p->next=head;head=p;
```

（4）通过输入语句给结点中的 data 域赋值。

```
scanf("%d",&p->data);
```

（5）再次调用 malloc()函数为第 2 个结点分配存储空间，并将第 2 个结点的地址赋给 *p*，使 *p* 指向第 2 个结点。然后再给第 2 个结点成员赋值。

```
p=(struct node *)malloc(sizeof*(struct node));
scanf("%d",&p->data);
```

（6）将指向第 1 个结点的指针 head 赋给第 2 个结点的 next，再把 *p* 同给 head，使 *p* 和 head 均指向第 2 个结点。

```
p->next=head;head=p;
```

（7）反复执行上面的操作，就可以完成具有 *n* 个结点的单向链表。

注意：这样建立的链表，结点顺序是从右向左，指针 head 始终指向最后建立的结点。

2．输出链表

（1）定义一个指向结点类型的指针变量 *p*，把 head 赋给它，让 *p* 指向第 1 个节点。

```
struct  node *p;  p=head;
```

（2）输出 p 所指向结点的数据域。

```
printf("%d",p->data);
```

（3）将 p 指向下一个节点。

```
p=p->next;
```

（4）反复执行（2）和（3），直到所有的结点都输出，即遇到 *p* 的值为 NULL。

3．对链表的插入操作

链表的插入过程如图 14-2 所示。一般的操作步骤如下：

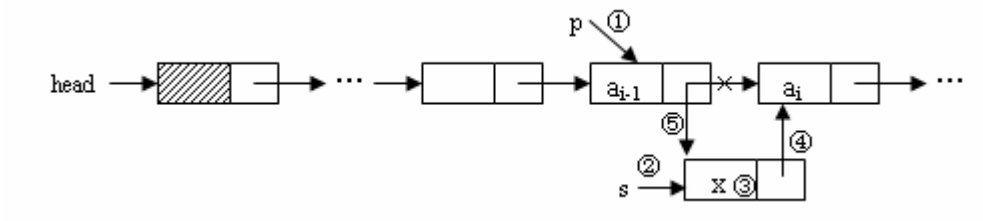



图 14-2　链表的插入操作的方法

（1）确定待插入位置，使用指针变量 *p* 指向待插入结点的前一结点；

（2）生成一个结点空间，并用指针变量 *s* 指向该结点，即：

```
s= (struct node *)malloc(sizeof*(struct node));
```

（3）给待插入结点数据域赋值，即：

```
scanf("%d",&s->data);
```

（4）修改插入结点指针域，即：

```
s->next=p->next;
```

（5）修改 *p* 所指结点的指针域，使其指向该结点，即：

```
p->next=s;
```

4．对链表的删除操作

链表的删除过程如图 14-3 所示。一般的操作步骤如下：

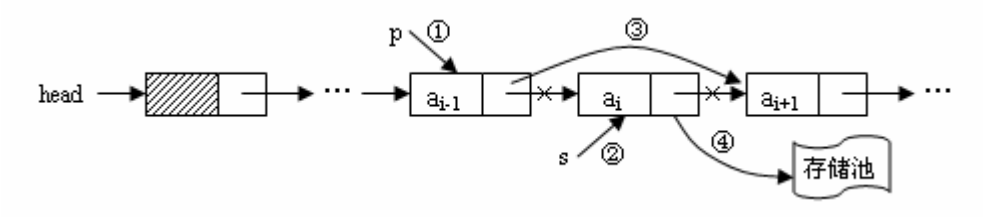



图 14-3　删除链表的操作方法

（1）确定待插入位置，使用指针变量 *p* 指向待删除结点的前一结点；

（2）使用指针变量 *s* 指向待删除的结点；

（3）修改 *p* 所指结点的指针域，使其指向删除的结点的后一结点，即：

```
p->next=s->next;
```

（4）释放 *s* 指向的结点，删除完毕，即：

```
free(s);
```

## 14.2.2　题眼分析

### 一、选择题分析

【例 1】设有如下说明

```
typedef struct ST
{long a;    int b;  char c[2];} NEW;
```

则下面叙述中正确的是_____。（2005 年 4 月）

（A）以上的说明形式非法　　（B）ST 是一个结构体类型

（C）NEW 是一个结构体类型　　（D）NEW 是一个结构体变量

**题眼分析**　此题通过 typedef 在定义一个结构体类型的同时把它自定义成类型名 NEW，而 ST 是结构体标识符，不是结构体类型。

**答案**　C

【例 2】以下对结构体类型变量 td 的定义中，错误的是_____。（2005 年 4 月）

（A）typedef struct aa
```
{    int n;
     float m;
}AA;
AA   td;
```

（B）struct aa
```
{    int n;
     float m;
 }td;
struct   aa   td;
```

（C）struct
```
{    int n;
     float m;
}aa;
struct   aa    td;
```

（D）struct
```
{    int n;
     float m;
}td;
```

**题眼分析**　对结构体类型变量的定义有 4 种方式：（1）在定义结构体类型时，同时定义结构体变量；（2）直接定义结构体变量，而不写出结构体标识符选项；（3）先定义结构体类型，然后再定义结构体变量。（4）先使用 typedef 说明一个新的结构体类型名，再用新类型名定义变量。选项（A）采用的是第 4 种定义方式；选项（B）采用的是第 1 种定义方式；选项（D）采用的是第 2 种定义方式；在选项（C）中，aa 是一个结构体变量，不是结构体标识符，因此是错误的。

**答案**　C

【例 3】有以下说明和定义语句
```
struct  student
{   int  age ;      char  num[8]; };
struct  student stu[3]={{20, "200401"},{21, "200402"},{19, "200403"}};
struct  student *p=stu;
```
以下选项中引用结构体变量成员的表达式错误的是_____。（2004 年 9 月）

（A）(p++)->num　　（B）p->num　　（C）(*p).num　　（D）stu[3].age

**题眼分析**　数组 stu 是一结构体类型的数组，它的最大长度是 3，选项（D）的下标超出允许的范围。选项（A）、（B）和（C）都能正确引用结构体变量成员，它们的值都等于 stu[0].num。

**答案**　B

【例 4】有以下程序：
```
struct s
{   int x,y;}data[2]={10,100,20,200};
main()
{   struct s *p=data;
    printf("%d\n",++(p->x));
}
```

程序运行后的输出结果是_____。（2003 年 9 月）

（A）10　　（B）11　　（C）20　　（D）21

题眼分析　数组 data 是一结构体类型的数组，指针 *p* 是一个基类型为结构体类型的指针变量。在定义 *p* 时赋初值，使它指向数组 data 首地址，*p*->x 等价于 data[0].x，因此输出的结果是 data[0].x+1。

答案　B

【例 5】设有如下定义

```
struct  ss
{   char name[10];
    int  age;
    char sex;
}std[3], *p=std;
```

下面各输入语句中错误的是_____。（2003 年 4 月）

（A）scanf("%d",&(*p).age);　　（B）scanf("%s",&std.name);

（C）scanf("%c",&std[0].sex);　　（D）scanf("%c",&(p->sex));

题眼分析　要给结构型成员输入数据，在 scanf 语句中需使用结构型成员的地址。选项（A）“ &(*p).age ”代表的是 std[0].age 的地址，是正确的；选项（C）显然也是正确的；选项（D）先用指针变量引用结构型的成员 sex，然后取它的地址，也是正确的。选项（B）中的“ std.name ”是错误的引用，因为 std 是数组名代表的是数组的首地址，地址没有成员“ name ”。

答案　B

【例 6】有以下程序

```
#include  <stdlib.h>
struct NODE{
    int    num;
    struct NODE *next;
};
main()
{   struct NODE  *p,*q,*r;
    int   sum=0;
    p=(structNODE*) malloc(sizeof(structNODE));
    q=(structNODE *) malloc(sizeof(structNODE));
    r=(structNODE *) malloc(sizeof(structNODE));
    p->num=l;q->num=2;r->num=3;
    p->next=q;q->next=r; r->next=NULL;
    sum+=q->next->num;sum+=p->num;
    printf("%d\n", sum);
}
```

执行后输出结果是_____。（2004 年 4 月）

（A）3　　（B）4　　（C）5　　（D）6

题眼分析　程序中先定义了 3 个 NODE 类型指针变量 *p*、*q*、*r*，并生成了 3 个结点。通过修改结点的 next 成员的值，使 3 个结点首尾相连形成了一个链表。通过“ *q*->next=*r*; ”这条语句可以看出，指针变量 *q*->next 指向指针变量 *r*，所以变量 *q*->next->num 就等价于 *r*->num，其

值为 3。因此，sum 值为 *r*->num+ *p*->num=4。

**答案** B

**【例 7】以下程序的功能是：建立一个带有头结点的单向链表，并将存储在数组中的字符依次转储到链表的各个结点中，请从与下划线处号码对应的一组选项中选择出正确的选项。（2004 年 9 月）**

```
#include <stdio.h>
struct node
{   char data; struct node *next ; };
   (1)   CreatList(char *s)
{   struct node *h,*p,*q;
    h=(struct node *)malloc(sizeof(struct node));
    p=q=h;
    while (*s!= '\0' )
    {   p=(struct node *)malloc(sizeof(struct node));
        p->data=   (2)    ;
        q->next=p;
        q=    (3)    ;
        s++;
    }
    p->next='\0' ;
    return h ;
}
main( )
{   char str[ ]="link list" ;
    struct node *head ;
    head=CreatList(str) ;
    ……
}
```

（1）（A）char * （B）struct node （C）struct node * （D）char

（2）（A）*s （B）s （C）*s++ （D）*(s)++

（3）（A）p->next （B）p （C）s （D）s->next

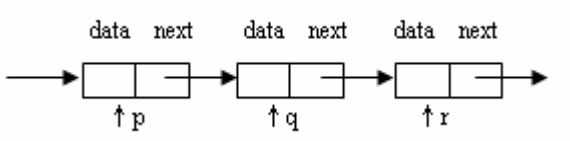
**题眼分析** 这是一个典型的建立链表算法。函数 CreatList()是链表建立函数，链表建成后，链表的头结点的地址由函数 CreatList()返回，并赋值给函数 main 中的指针变量 *h*，因此函数的类型应该是基类型为 struct node 的指针类型，因此（1）应选（C）。空（2）处是将指针 *s* 所指的字符作为链表数据域存储在链表中，考虑到下面有“ *s*++; ”语句移动指针，因此空（2）只需直接引用就可以了，所以应选（A）。从程序中可以看出，指针 *q* 是用来指向链表的最后一个结点，通过“ *q*->next=*p*; ”语句使新开辟的结点 *p* 链接在 *q* 的后面，这时指针 *p* 所指的结点是链表最后一结点，为了完成下一次插入，空（3）处应将指针 *q* 再次指向链表最后一个结点，因此空（3）应填入“ *p* ”，即选（B）。

**答案** （1）C （2）A （3）B

**【例 8】有以下结构体说明和变量定义，如图所示，指针 *p*、*q*、*r* 分别指向此链表中的 3 个连续结点。**

```
struct node
{   int data;   struct node *next;}*p,*q,*r;
```

现要将 *q* 所指结点从链表中删除，同时要保持链表的连续，




以下不能完成指定操作的语句是_____。（2005 年 4 月）

（A）*p*->next=*q*->next;　　（B）*p*->next=*p*->next->next;

（C）*p*->next=*r*;　　（D）*p*=*q*->next;

**题眼分析**　将 *q* 所指结点从链表中删除，只需要修改 *p* 所指结点的 next 成员的值，使其指向 *r* 所指的结点。在这个链表中，*r* 的值可以表示 *q*->next、*p*->next->next，因此选项（A）、（B）和（C）都是正确的方法。选项（D）只是修改指针 *p* 指向的结点，使指针 *p* 和 *r* 指向同一结点。

**答案**　D

## 二、填空题分析

【例 1】以下程序的运行结果是_____。（2004 年 4 月）

```
#include  <string.h>
typedef    struct student{
    char name[10];
    long sno;
    float score
} STU;
main()
{   STU  a={"Zhangsan " ,2001,95 },b={"Shangxian", 2002,90 }
    c={"Anhua" ,2003,95 },d, *p=&d;
    d=a;
    if(strcmp(a.name,b.name)>0) d=b;
    if(strcmp(c.name,d.name)>0) d=c;
    printf("%ld %s\n",d.sno,p->name);
}
```

**题眼分析**　经过赋值和字符串比较，结构体变量 *d* 的值为{"Shangxian", 2002,90}，而结构体指针变量 *p* 就是指向变量 *d* 的。因此 *d*.sno 值为 2002，*p*->name 的值就是 *d*.name 的值，即“Shangxian”。

**答案**　2002 Shangxian

【例 2】以下定义的结构体类型包含两个成员，其中成员变量 info 用来存入整形数据；成员变量 link 是指向自身结构体的指针。请将定义补充完整。（2002 年 4 月）

```
struct node
{   int info;
    _____link;
```

```
}
```

**题眼分析** link 是指向自身结构体的指针，因此 link 是指针，指向 struct node 类型数据，可定义成“struct node * link”。

**答案** struct node *

**【例 3】已有定义如下：**

```
struct node
{   int data;
    struct node *next;
}*p;
```

**以下语句调用 malloc 函数，使指针 *p* 指向一个具有 struct node 类型的动态存储空间。请填空。（2003 年 9 月）**

```
p=(struct node*)malloc(_____);
```

**题眼分析** 动态存储空间申请函数 malloc()中的参数是结点占用的存储空间大小，此题申请的空间用来存储 struct node 类型的数据，而 sizeof()函数的作用是求得数据类型或数据占用的存储空间的大小。

**答案** sizeof(struct node )

**【例 4】以下程序运行后的输出结果是_____。（2005 年 4 月）**

```
struct NODE
{   int k;
    struct NODE *link;
};
main()
{   struct NODE m[5],*p=m,*q=m+4;
    int i=0;
    while(p!=q)
    {   p->k=++i;   p++;
        q->k=i++;   q--;
    }
    q->k=i;
    for(i=0;i<5;i++)    printf("%d",m[i].k);
    printf("\n");
}
```

**题眼分析** 程序首先说明了一个结构体类型，并定义一个有 5 个元素的结构体类型数组 *m* 和两个基类型为结构类型的指针变量 *p* 和 *q*，通过赋初值的方式，使 *p* 指向数组 *m* 的第 1 个元素 m[0]，使 *q* 指向数组 *m* 的最后一个元素 *m*[4]。在 while 循环中，第 1 次循环时给 *m*[0]->*k* 赋值 1，*m*[4]->*k* 也赋值 1；第 2 次循环时给 *m*[1]->*k* 赋值 3，*m*[3]->*k* 也赋值 3。退出循环时，*p* 和 *q* 都指向 m[2]，*i* 的值为 4，通过赋值，使 *m*[2]->*k* 的值为 4。最后通过一个 for 循环输出数组 *m* 每个元素的 *k* 字段，因此输出结果是 13431。这里需要说明的一点是，虽然结构体中包含一个指针成员，但在程序中并没有用到，只是使用一个结构体数组，并没有构成链表，考生不要被这一假相所迷惑。

**答案** 13431

# 14.3　共用体

## 14.3.1　考点提炼

### 考点 1：共用体类型与变量定义

共用体与结构体的一个本质不同在于结构体每个成员占有不同的存储空间，而共用体的各个成员占有相同的存储空间，也就是说其各个成员的首地址是相同的。结构型变量占用的存储空间是它的各个成员所占存储空间之和，而共用型变量所占存储空间为它的最大成员所占存储空间之和。还有一点需注意的是不能在定义共用体类型变量时对它初始化。

1．共用体类型的定义

定义的一般形式如下：

```
union 共用体名
{  成员列表 };
```

成员表列的一般形式如下：

类型说明符 成员名;

例如，有下述定义：

```
union  ic{  int x;
            char  c[2];
         }
```

该共用体类型有两个成员，*x* 和 *c* 它们占用相同的存储单元，本共用体定义的变量占两个字节。

2．共用体类型变量或数组的定义

（1）定义共用体类型的同时定义共用型变量。

例如：

```
union ic{/* 略*/}a,b,c[5];
```

（2）先定义一个共用体类型，然后使用该类型来定义共用体变量。

例如：

```
union ic{/* 略*/};
union  ic  a,b,c[5];
```

（3）定义一个无名称的共用体类型的同时定义共用体变量。

例如：

```
union {/* 略*/}a,b,c[5];
```

### 考点 2：共用体类型变量和数组的成员引用

引用格式如下：

共用型变量名.成员名　　或　　数组名[下标].成员名

在上述定义下，要引用共用型变量 *a* 的成员 c[1]，可以写成“ *a*.*c*[1] ”；要引用共用型数组元素 *c*[1]的成员 *x*，可以写成“ *c*[1].*x* ”。

**考点 3：指向共用体的变量指针变量的使用**

若某个变量指向了共用型变量，则使用指针变量引用该共用型变量成员的方法有两种：

(*指针变量名).成员名　　或　　指针变量->成员名

如有下述定义：

```
union  ic  a,b,c[5],*p1=&a,*p2=c;
```

则用指针变量 *p*1 给变量 *a* 的成员 *c*[1]赋'f'的语句可写成：

```
(*p1).c[1]= 'f';
```

或

```
p1->c[1]= 'f';
```

用指针变量 *p*2 给共用型数组元素 *c*[2]的 *x* 赋值 99 可写成：

```
(*(p2+2)).x=99;
```

或

```
(p2+2)->x=99;
```

## 14.3.2 题眼分析

### 一、选择题分析

【例 1】若有下面的说明和定义：

```
struct test
{   int ml; char m2; float m3;
    union uu {char ul[5]; int u2[2];} ua;
} myaa;
```

则 sizeof(struct test )的值是_____。（2002 年 4 月）

（A）12　　（B）16　　（C）14　　（D）9

题眼分析　结构体所占用的存储空间是其所有成员占用的存储空间之和，而共用体所占用的存储空间是成员中占用存储空间最大者的空间。共用体类型 uu 是结构体的成员，它所占的内存长度为最大成员的长度，即字符型数组 *u*1 的长度，即 1×5＝5。整型数据占 2 个字节，字符型数据占 1 个字节，单精度型数据占 4 个字节，myaa 为结构体变量，它所占的存储空间为各成员所占存储空间之和，即 2＋1＋4＋5＝12。

答案　A

【例 2】有以下程序

```
main()
{   union { unsigned int n;
            unsigned char c;
    }ul;
    u1.c='A';
    printf("%c\n",u1.n);
}
```

执行后输出结果是_____。（2003 年 4 月）

（A）产生语法错　　（B）随机值　　（C）A　　（D）65

题眼分析　在定义共用型的同时定义了一个共用型变量 *u*1，*u*1 占两个字节，有两个成员 *n* 和 *c*，两个成员的首地址是相同的。因此给 u1.*c* 赋一个'A'，其实就是给无符号整型成员 *u*1.n 的低字节赋了一个'A'，输出 u1.*n* 的时候是以字符型的形式输出，只输出它的低地址的一个字

节，故为'A'。

**答案**　C

**【例 3】以下程序的输出结果是_____。（2001 年 9 月）**

```
union myun
{   struct { int x, y, z; } u;
    int k;
} a;
main()
{   a.u.x=4; a.u.y=5; a.u.z=6; a.k=0;
    printf(%d\n",a.u.x);
}
```

（A）4　　（B）5　　（C）6　　（D）0

**题眼分析**　共用体类型 myun 中有两个成员：结构体成员 *u* 和整型成员 *k*。结构体中有 3 个 int 型成员 *x*、*y*、*z*，占用连续的不同的存储空间。变量 *a.u.x*、*a.k* 首地址相同，根据赋值的先后顺序，*a.k* 的值将覆盖 *a.u.x* 的值。因此，输出的 *a.u.x* 的值即 *a.k* 的值。

**答案**　D

**【例 4】若有以下说明和定义**

```
union dt
{   int a;  char b;     double c;}data;
```

**以下叙述中错误的是_____。（2005 年 4 月）**

（A）data 的每个成员起始地址都相同

（B）变量 data 所占的内存字节数与成员 *c* 所占字节数相等

（C）程序段：data.a=5;printf("%f\n",data.c);输出结果为 5.000000

（D）data 可以作为函数的实参

**题眼分析**　根据共用体的定义，很明显选项（A）和（B）是正确的说法。共用体类型的变量可以作为实参进行传递，也可以传送共用体变量的地址，因此选项（D）也是正确的。在选项（C）中，先给成员 a 赋值 5，由于成员 *a* 的类型是整型，占用两个字节，存储在共用体的前两个字节中，输出时是输出成员 *c* 的值，而成员 *c* 的类型是双精度型，占用 8 个字节，输出时应输出这内存中 8 个字节的内容，因此选项（C）是错误的。

**答案**　C

## 二、填空题分析

**【例】若有以下定义和语句，则 sizeof(a)的值是__（1）__，而 sizeof(a.share)的值是__（2）__。（1999 年 9 月）**

```
struct  date
{   int  day;int  month;int  year;
    union
    {   int  share1;
        float  share2;
    }share;
}a;
```

**题眼分析** 本题考核的考点是共用体变量和结构体变量所占内存的字节数。共用体变量在内存中所占的长度是各成员变量中最长的，而结构体变量在内存中所占的长度是各成员变量长度之和。因此，共用体 share 的长度为其单精度型成员 share2 所占的内存字节数为 4，变量 a 在内存中的长度为 2＋2＋2＋4＝10。

**答案** （1）10 （2）4

# 14.4 单元训练及参考答案

## 14.4.1 单元训练

### 一、选择题

1．在说明一个结构体变量时系统分配给它的存储空间是_____。

（A）该结构体中占用最大存储空间的成员所需存储空间

（B）该结构体中所有成员所需存储空间的总和

（C）该结构体中第 1 个成员所需存储空间

（D）该结构体中最后一个成员所需存储空间

2．共用体类型在任何给定时刻_____。

（A）所有成员一直驻留在结构中 （B）没有成员驻留在结构中

（C）部分成员驻留在结构中 （D）只有一个成员驻留在结构中

3．使用共用体 union 的目的是_____。

（A）将一组数据作为一个整体，以便于其中的成员共享同一存储空间

（B）将一组具有相同类型的数据作为一个整体，以便于其中的成员共享同一存储空间

（C）将一组相关数据作为一个整体，以便程序中使用

（D）将一组具有相同数据类型的数据作为一个整体，以便程序中使用

4．以下关于 typedef 的叙述不正确的是_____。

（A）用 typedef 可以定义各种类型名，但不能用来定义变量

（B）用 typedef 可以增加新类型

（C）用 typedef 只能将已存在的类型用一个新的名称来代表

（D）使用 typedef 便于程序的通用

5．若有以下说明和语句：

```
struct worker
{   int no;char  *name;}    work, *p=&work;
```

则以下引用方式不正确的是_____。

（A）work->no （B）(*p).no （C）p->no （D）work .no

6．以下程序执行后的结果是_____。

```
struct  tee
{   int x;char *s;}t;
```

```
func(struct tee t)
{   t.x=10; t.s="minicomputer"; return(0);}
main()
{   t.x=1; t.s="computer";
    func(t);
    printf("%d,%s\n",t.x,t.s);
}
```

（A）10,computer　　（B）1,minicomputer

（C）1,computer　　（D）10,minicomputer

7．**有如下定义：**

```
struct date {int year,month,day;};
struct worklist {char name[20]; char sex; struct date birthday;}personone;
```

**对结构体变量** personone **的出生年份进行赋值时，下面正确的赋值语句是_____。**

（A）year=1968　　（B）birthday.year=1968

（C）personone.birthday.year=1968　　（D）personone.year=1968

8．**在如下结构体定义中，不正确的是_____。**

（A）struct　student {int num; char name[10]; float score;};

（B）struct　{int num; char name[10]; float score;}stud[20];

（C）struct　student {int num; char name[10]; float score;}stud[20];

（D）struct　stud[20]{int num; char name[10]; float score;}

9．**以下函数定义中不正确的是_____。**

（A）struct tree func(s)
　　struct tree s[];
　　{……}

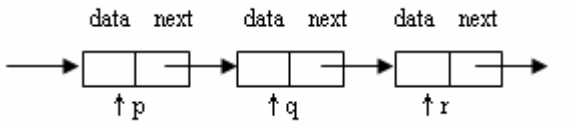
（B）int *func(s)
　　char s[];
　　{……}

（C）struct tree *func(s)
　　char **s;
　　{……}

（D）int *func(s)
　　char *s[10][];
　　{……}

10．**有以下结构体说明和变量定义，如图所示，指针** p、q、r **分别指向一个链表中的** 3 **个连续结点。**

```
struct node
{   int data;
    struct node *next;
} *p, *q, *r;
```




**现要将** *q* **和** *r* **所指结点的先后位置交换，同时要保持链表的连续，以下错误的程序段是_____。**

（A）r->next=q; q->next=r->next; p->next=r;

（B）q->next=r->next; p->next=r; r->next=q;

（C）p->next=r; q->next=r->next; r->next=q;

（D）q->next=r->next; r->next=q; p->next=r;

11．**以下程序的运行结果是_____。**

```
typedef union
{   long b[2]; int y[5]; char s[10]; }SS;
    SS sdh;
main()
{   printf("%d\n",sizeof(sdh)); }
```

（A）20　　（B）10　　（C）24　　（D）16

12．有如下定义

```
struct person{char name[9]; int age;};
struct person class[10]={ "Johu", 17, "Paul", 19, "Mary", 18, "Adam", 16 };
```

根据上述定义，能输出字母 M 的语句是_____。（2000 年 9 月）

（A）prinft("%c\n",class[3].name);　　（B）pfintf("%c\n",class[3].name[1]);

（C）prinft("%c\n",class[2].name[1]);　　（D）printf("%c\n",class[2].name[0]);

13．下列程序的输出结果是_____。

（A）5　　（B）6　　（C） 7　　（D）8

```
struct abc
{   int a, b, c; };
main()
{   struct abc s[2]={{1,2,3},{4,5,6}}; int t;
    t=s[0].a+s[1].b;
    printf("%d \n",t);
}
```

14．在说明一个共用体变量时系统分配给它的存储空间是_____。

（A）该共用体中的第 1 个成员所需存储空间

（B）该共用体中的最后一个成员所需存储空间

（C）该共用体中占用最大存储空间的成员所需存储空间

（D）该共用体中所有成员所需存储空间的总和

15．若程序中有如下的说明和定义：

```
struct exep
{ int x,y;}
 ...
struct exep x,y;
  ...
```

则会发生的情况是_____。

（A）编译时出错　　（B）能通过编译、连接、执行

（C）能通过编译但连接出错　　（D）能通过编译、连接，但不能执行

16．下列程序的输出结果是_____。

```
#include<stdio.h>
void  main()
{   struct com
    { int  a,b;}c[3]={1,2,3,4,5,6};
    printf("%d",c[0].b/c[0].a*c[1].a/c[1].b+c[2].a-c[2].b);
}
```

（A）0　　（B）1　　（C）3　　（D）6

17．变量 *a* 所占内存字节数是_____。

（A）4　　（B）5　　（C）6　　（D）8

```
union U
{   char st[4];
    int  i;
    long l;
};
struct A
{   int  c;
    union U u;
}a;
```

18．以下程序的输出结果是_____。

```
struct HAR
{   int x, y; struct HAR *p;} h[2];
main()
{   h[0].x=1;h[0].y=2; h[1].x=3;h[1].y=4;
    h[0].p=&h[1];h[1].p=h;
    printf("%d %d \n",(h[0].p)->x,(h[1].p)->y);
}
```

（A）12　　（B）23　　（C）14　　（D）32

19．下面程序的输出是_____。

```
main( )
{   struct cmpl{int x; float y;} cum[2] = {1,3.5,2,8.5};
    printf("%f\n",cum[0].y/ cum[0].x * cum[1].x);
}
```

（A）5.000000　　（B）1.00000　　（C）6.000000　　（D）7.00000

20．下面程序的输出是_____。

```
typedef union {  float x[2]; int y[4]; long z[8]; }ENDE;
ENDE  their;
main( )
{   printf("%d\n",sizeof(their));}
```

（A）32　　（B）16　　（C）8　　（D）24

21．typedef　int　BEGIN；的作用是_____。

（A）建立了一种新的数据类型　　（B）定义了一个整型变量

（C）定义了一个长整型变量　　（D）说明了一个新的数据类型标识符

22．设有以下定义：

```
typedef union
{   long i;int k[7];char c;}DATA;
struct date
{   int cat;  DATA cow;  float dog;}t;
DATE m;
```

则下列语句执行后输出结果是_____。

```
printf("%d",sizeof(struct date)+sizeof(m));
```

（A）25　　（B）34　　（C）18　　（D）8

23．设有以下语句：

```
struct st{int n; struct st *next;};
static struct st a[3]={5,&a[1],7,&a[2],9 ,'\0'},*p;
p=&a[1];
```

则值为 8 的表达式是_____。

（A）p++->n　　（B）p->n++　　（C）(*p).n++　　（D）++p->n

24．设有以下语句

```
typedef   struct  S
{   int   g;   char  h; }T;
```

则下面叙述中正确的是_____。

（A）可用 S 定义结构体变量　　（B）可以用 T 定义结构体变量

（C）S 是 struct 类型的变量　　（D）T 是 struct S 类型的变量

25．以下选项中不能正确把 *c*1 定义成结构变量的是_____。

（A）typedef struct
```
{    int   red;
     int   green;
     int   blue;
}COLOR;
COLOR c1;
```

（B）struct color c1
```
{    int   red;
     int   green;
     int   blue;
 };
```

（C）struct color
```
{    int   red;
     int   green;
     int   blue;
}c1;
```

（D）struct
```
{    int   red;
     int   green;
     int    blue;
}c1;
```

## 二、填空题

1．以下程序的执行结果是___（1）___。

```
typedef struct { long x[3]; int y[2]; char z[10]; } MYTYPE;
MYTYPE a;
main() { printf("%d\n",sizeof(a)); }
```

2．有如下定义：

```
struct { int x; int y; } s[2]={{1,5},{20,40}},*p=s;
```

则表达式++p->x 的结果是___（2）___，表达式(++p)->x 的结果是___（3）___。

3．以下程序的执行结果是___（4）___。

```
struct stru
{   int x; char c; } ;
main()
{   struct stru a={10,'f'}, *p=&a;
    func(p);
```

```
    printf("%d,%c\n",a.x,a.c);
}
func(struct stru *b)
{   b->x=10; b->c='h';     }
```

4．**以下程序的执行结果是__（5）__。**

```
typedef union
{   long x[3]; int a[2]; char c[10];   } MYTYPE;
MYTYPE a;
main() { printf("%d\n",sizeof(a)); }
```

5．**以下程序的执行结果是__（6）__。**

```
main()
{   union EXAMPLE
    {   struct
        {   int x; int y; int z; } in;
        int a; int b;
    }e;
    e.a=2; e.b=3;
    e.in.x=e.a*e.b; e.in.y=e.a+e.b; e.in.z=(e.a*e.b)+(e.a+e.b);
    printf("%d,%d,%d \n",e.in.x,e.in.y,e.in.z);
}
```

6．**以下程序的执行结果是__（7）__。**

```
main()
{   union a { char *name; int age; int income; } s;
    s.name="LiMing"; s.age=29; s.income=2000;
    printf("%d\n",s.age);
}
```

7．**有如下定义：**

```
    struct {int x; int y;}tab[2]={{1,2},{3,4}}, *p=tab;
```

**则表达式** *p->y* **的结果是__（8）__。**

8．**设有定义语句** struct {int a;float b;char c;} abc,*p_abc=&abc;**，则对结构体成员** *a* **的引用方法可以是__（9）__和__（10）__。**

9．**下面程序的输出结果是__（11）__。**

```
main()
{   union example
    {   struct
        {   int x; int  y;}in;
        int a;
        int b;
    }e;
    e.a=1; e.b=2;
    e.in.x=e.a*e.b;
    e.in.y=e.a+e.b;
    printf("%d,%d",e.in.x, e.in.y);
}
```

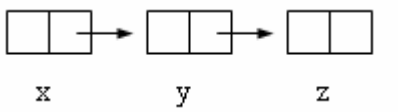
10．以下程序用以输出结构体变量 *bt* 所占的内存单元的字节数，请在括号内填上适当的内容。

```
struct pa
{   double i;
    char arr[10];
};
main()
{   struct pa bt;
    printf("bt size:%d\n",   (12)   );
}
```

11．设有以下定义

```
struct  ss
{   int info; struct ss *link; } x,y,z;
```

且已建立如下图所示链表结构：




请写出删除结点 *y* 的赋值语句 （13） 。

## 14.4.2 参考答案

### 一、选择题

| | | | | |
|---|---|---|---|---|
| 1． B | 2． D | 3． A | 4． B | 5． A |
| 6． C | 7． C | 8． D | 9． D | 10．A |
| 11．B | 12．D | 13．B | 14．C | 15．A |
| 16．A | 17．C | 18．D | 19．D | 20．A |
| 21．D | 22．B | 23．D | 24．B | 25．B |

### 二、填空题

（1）26
（2）2
（3）20
（4）10,h
（5）12
（6）9,18,99
（7）2000
（8）2
（9）abc.a
（10）*p->a* 或(**p*).*a*
（11）4,8
（12）sizeof(struct pa)
（13）*x*->link=*z*; 及其等价形式

# 第 15 章　位运算

## 15.1　位运算符

### 15.1.1　考点提炼

**📖考点 1：数在机器中的存放形式**

1．字和字节

在计算机系统中，内存储器是由许多被称为“字节”的单元组成的。每一个字节有一个地址。一个字节由 8 个二进制组成的。若干个字节组成一个“字”。程序运算中的数与中间结果一般都要存放在内存储器中，如在 C 语言中，用 4 个字节存放一个单精度数，用 8 个字节存放一个双精度数，用两个字节存放一个整型数，用一个字节存放一个字符。对于实数和整型，一般第 1 位用作符号位，为“0”表示正，为“1”表示负。

2．原码

原码的表示方法是将最高位作为符号位（正数为 0，负数为 1），其余各位代表数值本身的绝对值（需用二进制表示）。如 7 的原码是 0000 0111，–7 的原码是 1000 0111。注意在计算机系统中+0 和–0 不是同一个数，+0 的原码是 0000 0000，–0 的原码是 1000 0000。

3．补码

在计算机中数是以补码的形式存放的。正数的补码与它的原码相同。负数的补码是其绝对值的原码取反再加 1。如 7 的原码 0000 0111，对其取反加 1 得到 1111 1001。由此可以看到补码最高位为 1 时，该数为负数，为 0 时该数为正数。对负数的补码取反加 1 得到该数的绝对值。

**📖考点 2：位运算符**

C 语言中提供如表 15-1 所列的位运算符。

表 15-1　　位运算符

| 运算符 | 含义 | 优先级 |
|---|---|---|
| ~ | 按位求反 | 1（高） |
| << | 左移 | 2 |
| >> | 右移 | 2 |
| & | 按位与 | 3 |
| ^ | 按位异或 | 4 |
| \| | 按位或 | 5（高） |

说明：

（1）位运算符中除~以外，均为二目（元）运算符，即要求两侧各有一个运算量。

（2）运算量只能是整型或字符型的数据，不能为实型数据。

### 15.1.2 题眼分析

【例 1】设有定义语句：char *c*1=92 , *c*2=92 ; ，则以下表达式中值为零的是_____。（2004 年 9 月）

（A）*c*1^*c*2　　（B）*c*1&*c*2　　（C）~*c*2　　（D）*c*1|*c*2

题眼分析　选项（B）的运算符是位与，*c1* 和 *c2* 相等，相与的结果还是该数；选项（C）中，运算符“~”是按位求反，92 对应的二进制是 1011100，按位求反是 0100011，转换成十进制是 35；选项（D）中的运算“|”为位或，只要两个数的某一位不为 0，则运算结果不为 0；选项（A）中的运算符是“异或”，由于 *c*1 和 *c*2 相同，这两个数异或后结果为 0。

答案　A

【例 2】读程序段：

```
int a=-5;  a=a | 0337;       printf("%d,%o\n",a,a);
```

以上程序段输出的结果是_____。

题眼分析　在计算机中数是以补码的形式存放的。正数的补码与它的原码相同。负数的补码是其绝对值的原码取反再加 1。–5 补码形式为： 1111 1111 1111 1011，0337 的二进制数应是 0000 0000 1101 1111，这两个数位或后的机器码是 1111 1111 1111 1111，它是十进制数–1 的补码，转换为八进制为：177777。

答案　–1,177777

## 15.2 位运算符的运算功能

### 15.2.1 考点提炼

#### 考点 1：位运算符的含义及使用

1．位逻辑运算符

（1）位与运算符（&）。参加运算的两位都为 1，则结果为 1，否则结果为 0。一个数与 1 位与，结果为该数，与 0 位与，结果为 0。所以程序中通常使用它将变量的某些位清零。

（2）位或运算符（|）。参加运算的两个位只要有一个为 1，那么运算结果为 1。一个位与 1 位或，结果为 1，与 0 位或，结果不变。所以在程序中常用它将一个数的某些位置 1。

（3）异或运算符（^）。参加运算的两个位相同，结果为 0，不同结果为 1。一个位与 1 异或，结果把它取反，一个位与 0 异或，结果不变。所以在程序中通常使用它把某数的相应位取反。

（4）取反运算符（~）。~ 是一个单目运算符，用来对一个二进制按位取反。~运算符的优先级和所有的单目运算符的优先级一样，比双目运算符和三目运算符都高。

2．位移位运算符

（1）左移运算符（<<）。其作用是将一个数的二进制位左移若干位。高位左移溢出，舍

弃不用。一般来说左移了 x 位相当于将该数乘以 2 的 x 次方。但是这种情况只适用于舍弃的高位中不包含 1。左移运算符对于有符号数和无符号数的运算规则是一样的。

（2）右移运算符（>>）。其作用是将一个数的二进制位右移若干位。移出的位被舍弃，对于无符号数高位补 0，对于有符号的数，高位补符号位，即负数补 1，正数补 0。所以对于有符号数与无符号数的运算规则是不一样的。

注意：位运算符中除了~以外，均为二目运算符，运算量只能是整型或字符型的数据，不能为实型数据。对于不同长度的数据进行位运算时，系统将两数按右端对齐，根据参加运算的数来决定左侧所补的数是 1 还是 0，如果是正数则空位补 0，如果是负数则空位补 1，如果是无符号数则空位补 0。

3．位自反赋值运算

位运算符可以与赋值运算符结合成位自反赋值运算符。运算符的含义如下：

&=（自反位与运算）　如：*a*&=*b* 相当于 *a*=*a*&*b*。

|=（自反位或运算）　如：*a*|=*b* 相当于 *a*=*a*|*b*。

>>=（自反右移位运算）如：*a*>>=*b* 相当于 *a*=*a*>>*b*。

<<=（自反左移位运算）如：*a*<<=*b* 相当于 *a*=<<*b*。

^=（自反异或运算）　如：*a*^=*b* 相当于 *a*=*a*^*b*。

### 考点 2：位运算符的优先级

位运算符在表达式中的优先级可概括成如下 5 点：

（1）位反（~）运算符为单目运算符，优于所有的双目运算符和三目运算符。

（2）位移位运算符优先级相同，比算术运算符的优先级低，比关系运算符的优先级高。

（3）位逻辑运算符的优先级比关系运算符的优先级低，比逻辑运算符的优先级高。

（4）3 个位逻辑运算符的优先次序为：& 优于 ^　优于　|。

（5）位自反值运算符和赋值运算符、算术自反值运算符是同级的。

## 15.2.2　题眼分析

### 一、选择题分析

【例 1】有以下程序

```
main()
{   unsigned char a, b;
    a= 4|3;
    b= 4 & 3;
    printf("%d %d\n",a,b);
}
```

执行后输出结果是_____。（2004 年 4 月）

（A）7　0　　（B）0　7　　（C）1　1　　（D）43　0

题眼分析　4 的二进制可表示为“ 00000100 ”，3 的二进制可表示为“ 00000011 ”。两者“ 按位或 ”将得到“ 00000111 ”，即 7；两者都“ 按位与 ”将得到“ 00000000 ”，即 0。

**答案** A

**【例 2】设 char 型变量 *x* 中的值为 10100111，则表达式(2+*x*)^(~3)的值是_____。（2003 年 4 月）**

（A）10101001　　（B）10101000　　（C）11111101　　（D）01010101

**题眼分析** 表达式(2+*x*)的二进制表示为“10101001”，(~3)即把 3 按位取反得到二进制值为“11111100”，再把这两个二进制值按位加（异或），得到的结果为“01010101”。

**答案** D

**【例 3】有以下程序：**

```
main()
{   unsigned char a,b,c;
    a=0x3; b=a|0x8;c=b<<1;
    printf("%d%d\n",b,c);
}
```

**程序运行后的结果是_____。（2002 年 9 月）**

（A）–11 12　　（B）–6 –13　　（C）12 24　　（D）11 22

**题眼分析** 将 *a* 的值转换为二进制为：“0000 0011”，再与 0x8 的二进制“0000 1000”按位或，得到结果“0000 1011”赋值给 *b*，*b* 的值为 11，再将 *b* 左移一位得到：0001 0110，赋值给 *c*，*c* 的值为 22。

**答案** D

**【例 4】有以下程序**

```
main()
{   int c=35;   printf("%d\n",c&c);    }
```

**程序运行后的输出结果是_____。（2005 年 4 月）**

（A）0　　（B）70　　（C）35　　（D）1

**题眼分析** 十进制数 35 的二进制形式为 00100011，00100011 与 00100011 按位进行与操作结果还是 00100011。因此按十进制输出时还是 35。

**答案** C

**【例 5】有一下程序**

```
main()
{   int x=3,y=2,z=1;
    printf(“%d\n”,x/y&~z);
}
```

**程序运行后的输出结果是_____。（2003 年 9 月）**

（A）3　　（B）2　　（C）–1　　（D）0

**题眼分析** “/”运算符是算术运算符，它的优先级要比位运算符高；在位运算符中，“~”运算符是单目运算符，其优先级要高于“&”运算符。因此表达式“*x*/*y*&~*z*”等价于“(*x*/*y*)&(~*z*)”，结果就为 0。

**答案** D

**二、填空题分析**

**【例 1】** *b* **为任意** int **类型变量，运用位运算，能将变量** *b* **清零的表达式为_____。**

**题眼分析　要达到将** *b* **清零的目的，可把** *b* **同** 0 **相与，也可以与自身异或。**

**答案**　*b*&=0 **或** *b*^=*b* **及其等价形式**

**【例** 2**】程序段如下：**

```
int x=1,y=2;
if (x&y)  printf("***\n");
else    printf("$$$\n");
```

**以上程序段输出的结果是_____。**

**题眼分析　首先把** *x* **的值** 1 **和** *y* **的值** 2 **转换成二进制分别是** 00000001 **和** 00000010**，位与后得到结果** 00000000**，所以** x&y **值为零，表示逻辑假，因此执行是** else **后的语句。**

**答案**　$$$

# 15.3　单元训练及参考答案

## 15.3.1　单元训练

**一、选择题**

1．**以下运算中优先级最低的是_____。**

（A）&&　　（B）&　　（C）||　　（D）|

2．**在** C **语言中必须是整型或字符型的运算符是_____。**

（A）>>　　（B）&&　　（C）!　　（D）+

3．**表达式** *x*<*b* ||~*c*&*d* **的运算顺序是_____。**

（A）~，&，<，||　　（B）~，||，—，>

（C）~，&，||，<　　（D）~，<，&，||

4．**表达式** 19&23 **的值是_____。**

（A）0x17　　（B）0x13　　（C）0xf8　　（D）0xec

5．**读程序段：**

```
char a=56;
a=a&056; printf("%d,%o",a,a);
```

**以上程序输出的结果是_____。**

（A）56,70　　（B）0,0　　（C）40,50　　（D）62,76

6．**有下程序段：**

```
int a=3,b=4;a=a^b; b=a^b;a=a^b;
```

**执行了上述的语句段后** *a* **和** *b* **的值分别是_____。**

（A）*a*=3 *b*=3　　（B）*a*=4 *b*=4　　（C）*a*=3 *b*=4　　（D）*a*=4 *b*=3

7．**在位运算中，操作数每左移一位，结果相当于_____。**

（A）**操作数乘以** 2　　（B）**操作数除以** 2　　（C）**操作数乘以** 4　　（D）**操作数除以** 4

8．整型变量 *x* 和 *y* 的值相等，且均为非 0 值，则以下选项中，结果为 0 的表达式是_____。

（A）*x* || *y*　　（B）*x* | *y*　　（C）*x* & *y*　　（D）*x* ^ *y*

**二、填空题**

1．在 C 语言中，&作为单目运算符时表示的是__（1）__运算；作为双目运算符时表示的是__（2）__运算。

2．与 *a*^=*b* 等价的另一种书写形式是__（3）__。

3．与表达式 *x*&=*y*–*4* 等价的另一种书写形式是__（4）__。

4．设有二进制数 *a* 的值为 11001101，若通过 *a*&*b* 运算使 *a* 中的低 4 位不变，高 4 位清零，则 *b* 的二进制数是__（5）__。

5．设 *a* 是一个整数（两个字节），若要通过 *a*|*b* 使 *a* 的低八位为 1，高八位不变，则 *b* 的八进制数是__（6）__。

## 15.3.2　参考答案

**一、选择题**

1．C　　2．A　　3．D　　4．B　　5．C

6．D　　7．A　　8．D

**二、填空题**

（1）取地址　　（2）按位与

（3）a=a^b　　（4）x=x&(y–4)

（5）0000 1111　　（6）0377

# 第 16 章　文件

## 16.1　C 语言文件的概念

### 16.1.1　考点提炼

1．文件的概念

通常所说的文件是指存放在磁盘上的一组相关信息的集合。为了区分不同文件，可给每个文件一个标识，称为“磁盘文件名”。磁盘文件名由 3 个部分组成：[盘符][路径]<主文件名>[.<扩展名>]。如果使用的文件在当前盘，盘符可以省去，如果使用的文件在当前目录下，路径可以省去。

2．文本文件与二进制文件

磁盘文件按数据存放的格式分类，可分成“二进制文件”和“文本文件”两种。

二进制文件中数据都是以二进制的形式存放的。文件中存放的是数据对应字符的 ASCII 码，一个字符占一个字节。二进制形式存放数据占有存储空间较少，且不直观。文本文件占用的存储空间较多，比较直观。

3．顺序文件与随机文件

按文件的读写方式来分，可以把文件分为“顺序文件”和“随机文件”。

对顺序文件来说，读写必须从头开始。对随机文件来说，其读写的位置是任意的。可以对文件中任意位置的数据进行读写，因此在读写之前，需要定位到读写数据的位置。

4．磁盘文件与设备文件

通常把存放在磁盘上的文件称磁盘文件。

由于计算机中的输入/输出设备的作用也是输入/输出数据，其功能和文件的读取数据/写入数据相似，所以把输入/输出设备也当成文件来处理，称为设备文件。

设备文件主要有 3 个：标准的输入设备，通常指键盘；标准的输出设备，通常指显示器；标准的错误输出设备，通常指显示器。

C 语言规定，3 种标准的输入输出设备进行数据的读写操作，不必事先打开设备文件，操作后，也不必关闭设备文件，而由系统自动打开和关闭：在系统启动时打开这些设备，在系统关闭时关闭这些设备。

5．文件的打开、关闭与文件位置指针的概念

磁盘文件在使用之前必须“打开”，所谓“文件的打开”是指从磁盘文件读取数据到内存。文件使用完毕要“关闭”，“文件的关闭”是指把内存中的数据写回到“磁盘文件”。

磁盘文件打开后，将用一个“文件位置指针”指向磁盘文件中的第 1 个数据，当读取了这个数据后，“文件位置指针”自动移向下一个数据。当把数据写入某个文件时，“文件位置指针”总是自动指向下一个要写入数据的位置。“文件位置指针”随文件的打开而存在，随文件的关

闭而消失。

### 16.1.2 题眼分析

【例 1】以下叙述中不正确的是_____。（2003 年 4 月）

（A）C 语言中的文本文件以 ASCII 码形式存储数据

（B）C 语言中对二进制文件的访问速度比文本文件快

（C）C 语言中，随机读写方式不适用于文本文件

（D）C 语言中，顺序读写方式不适用于二进制文件

题眼分析　在 C 语言中文本文件是以 ASCII 码形式存放的，每个字符占一个字节，因此选项（A）所描述的是正确的。由于数据在计算机中是以二进制形式存放的，因此二进制文件中的数据可以直接读出而不需要像文本文件那样把 ASCII 码转换成二进制，因此速度较快，因此选项（B）所描述的也是正确的。在文本文件中，数据是以 ASCII 码形式存放的，用户很难判定一个数据到底占几个字节，所以不适合使用随机读写方式，因此选项（C）所描述的也是正确的。数据以二进制形式存放，占有的字节数是固定的，所以可以进行随机读写，当然也可以进行顺序读写，因此选项（D）所描述的是错误的，是答案。

答案　D

【例 2】下列关于 C 语言数据文件的叙述中正确的是_____。（2003 年 9 月）

（A）文件由 ASCII 码字符序列组成，C 语言只能读写文本文件

（B）文件由二进制数据序列组成，C 语言只能读写二进制文件

（C）文件由记录序列组成，可按数据的存放形式分为二进制文件和文本文件

（D）文件由数据流形式组成，可按数据的存放形式分为二进制文件和文本文件

题眼分析　在 C 语言中，数据文件可以是文本文件也可以是二进制文件，因此选项（A）和（B）都是错误的。记录只是 C 语言中数据文件的一种形式，因此选项（C）也是错误的。

答案　D

## 16.2 文件指针、打开文件和关闭文件

### 16.2.1 考点提炼

#### 考点 1：文件指针

为便于处理文件，C 语言规定了一类特殊的结构类型——文件类型，用以记录处理文件时所需要的信息。文件类型 FILE 在“stdio.h”文件中定义。在 C 程序中，进行与文件有关的操作时，通常使用 FILE 类型定义指针变量，称“文件型指针”，然后通过文件的打开操作建立“文件型指针”与磁盘文件的联系。在处理文件的时候，程序只需和指向该文件的“文件型指针”打交道，而不必直接和磁盘文件打交道。定义文件类型指针变量的一般形式为：

```
FILE   *指针变量名;
```

例如：

```
FILE *fp1,*fp2;
```

*fp*1 和 *fp*2 均定义为指向文件类型的指针变量，称为文件指针。

### 考点 2：文件的打开

fopen 函数用来打开一个文件，其调用的一般形式为：

```
文件指针名=fopen(文件名，使用文件方式)
```

其中，“文件指针名”必须是被说明为 FILE 类型的指针变量，“文件名”是被打开文件的文件名。“使用文件方式”是指文件的类型和操作要求。“文件名”是字符串常量或字符串数组。例如：

```
FILE *fphzk
fphzk=("c:\\hzk16',"rb")
```

其意义是打开 C 驱动器磁盘的根目录下的文件 hzk16，这是一个二进制文件，只允许按二进制方式进行读操作。两个反斜线“\\ ”中的第 1 个表示转义字符，第 2 个表示根目录。使用文件的方式共有 12 种，表 16-1 给出了它们的符号和意义。

表 16-1　　文件的使用方式

| 文件使用方式 | 意　义 |
|---|---|
| “r” | 只读打开一个文本文件，只允许读数据 |
| “w” | 只写打开或建立一个文本文件，只允许写数据 |
| “a” | 追加打开一个文本文件，并在文件末尾写数据 |
| “rb” | 只读打开一个二进制文件，只允许读数据 |
| “wb” | 只写打开或建立一个二进制文件，只允许写数据 |
| “ab” | 追加打开一个二进制文件，并在文件末尾写数据 |
| “r+” | 读写打开一个文本文件，允许读和写 |
| “w+” | 读写打开或建立一个文本文件，允许读写 |
| “a+” | 读写打开一个文本文件，允许读，或在文件末追加数据 |
| “rb+” | 读写打开一个二进制文件，允许读和写 |
| “wb+” | 读写打开或建立一个二进制文件，允许读和写 |
| “ab+” | 读写打开一个二进制文件，允许读，或在文件末追加数据 |

对于文件使用方式有以下几点说明：

（1）凡用“r”打开一个文件时，该文件必须已经存在，且只能从该文件读出。

（2）用“w”打开的文件只能向该文件写入。若打开的文件不存在，则以指定的文件名建立该文件，若打开的文件已经存在，则将该文件删去，重建一个新文件。

（3）若要向一个已存在的文件追加新的信息，只能用“a”方式打开文件。但此时该文件必须是存在的，否则将会出错。

（4）在打开一个文件时，如果出错，fopen 将返回一个空指针值 NULL。在程序中可以用这一信息来判别是否完成打开文件的工作，并做相应的处理。

（5）把一个文本文件读入内存时，要将 ASCII 码转换成二进制码，而把文件以文本方式

写入磁盘时，也要把二进制码转换成 ASCII 码，因此文本文件的读写要花费较多的转换时间。对二进制文件的读写不存在这种转换。

（6）标准输入文件（键盘）、标准输出文件（显示器）、标准出错输出（出错信息）是由系统打开的，可直接使用。

📖考点 3：文件的关闭

文件一旦使用完毕，应用关闭文件函数把文件关闭，以避免文件的数据丢失等错误。fclose 函数调用的一般形式是：

```
fclose(文件指针);
```

例如：

```
fclose(fp);
```

正常完成关闭文件操作时，fclose 函数返回值为 0。如返回非零值则表示有错误发生。文件的读写对文件的读和写是最常用的文件操作。

## 16.2.2 题眼分析

### 一、选择题分析

【例 1】以下叙述中错误的是_____。（2002 年 9 月）

（A）二进制文件打开后可以先读文件的未尾，而顺序文件不可以

（B）在程序结束时，应当用 fclose()函数关闭已打开的文件

（C）在利用 fread()函数从二进制文件中读数据时，可以用数组名给数组中所有元素读入数据

（D）不可以用 FILE 定义指向二进制文件的文件指针

题眼分析　顺序文件只能从头读写，二进制文件可以随机读写，答案（A）是正确的；文件在使用后应关闭，当然程序结束时，应当把打开的文件关闭，答案（B）是正确的；用 fread()函数可以一次性地读取同类型的很多数据，答案（C）也是正确的；在 C 语言中指向各种文件的文件指针都是通过 FILE 来定义的，故答案（D）是错误的。

答案　D

【例 2】若要打开 A 盘上 user 子目录下名为 abc.txt 的文本文件进行读、写操作，下面符合此要求的函数调用是_____。（2002 年 4 月）

（A）fopen("A:\user\abc.txt","r")　　（B）fopen("A:\\user\\abc.txt","r+")

（C）fopen("A:\user\abc.txt","rb")　　（C）fopen("A:\\user\\abc.txt","w")

题眼分析　此题考核的考点是文件名的表示以及文件读写模式。由于“\”是转义字符，所以在文件名中“\”用“\\”来表示。要打开文本文件进行读写，应使用读写模式“r+”。

答案　B

【例 3】有以下程序（2004 年 9 月）

```
#include <stdio.h>
main( )
{   FILE *fp1;
```

```
    fp1=fopen("f1.txt","w") ;
    fprintf(fp1, "abc");
    fclose(fp1);
}
```

**若文本文件** f1.txt **中原有内容为：**good **，运行以上程序后文件** f1.txt **中的内容应为____。**

**（**A**）**goodabc　　**（**B**）**abcd　　**（**C**）**abc　　**（**D**）**abcgood

**题眼分析　在本题中，使用“** w **”方式打开文件“** f1.txt **”。用“** w **”方式打开的文件只能向该文件写入。若打开的文件不存在，则以指定的文件名建立该文件，若打开的文件已经存在，则将该文件删去，重建一个新文件。因为文件“** f1.txt **”已经存在，程序先删除这个文件，再建一个空的文件“** f1.txt **”，而文件的内容是通过“** fprintf(fp1, "abc"); **”写入的，因此，该文件的内容是“** abc **”。**

**答案**　C

**【例** 4**】有以下程序**

```
#include
void WriteStr(char *fn,char *str)
{   FILE *fp;
    fp=fopen(fn,"w");fputs(str,fp);fclose(fp);
}
main()
{   WriteStr("t1.dat","start");
    WriteStr("t1.dat","end");
}
```

**程序运行后，文件** t1.dat **中的内容是_____。（**2005 **年** 4 **月）**

**（**A**）**start　　**（**B**）**end　　**（**C**）**startend　　**（**D**）**endrt

**题眼分析　参见【例** 3**】分析。**

**答案**　B

## 二、填空题分析

**【例** 1**】以下程序段打开文件后，先利用** fseek()**函数将文件位置指针定位在文件末尾，然后调用** ftell **函数返回当前文件位置指针的具体位置，从而确定文件长度，请填空。（**2001 **年** 9 **月）**

```
FILE *myf; long f1;
myf= _______("test.t","rb");
fseek(myf,0,SEEK_END); f1=ftell(myf);
fclose(myf);
printf("%d\n",f1);
```

**题眼分析　本题首先定义了一个文件型指针** myf**，然后读打开文件“** test.t **”，所以空格处应填打开文件函数名，为“** fopen **”。**

**答案**　fopen

**【例** 2**】以下程序的功能是：从键盘上输入一个字符串，把该字符串中的小写字母转换为大写字母，输出到文件** test.txt **中，然后从该文件读出字符串并显示出来，请填空。（**1998 **年** 9

月）

```
#include "stdio.h"
main()
{   FILE *fp; char str[100]; int i=0;
    if((fp=fopen("text.txt", __(1)__))==NULL)
    {   printf("can't open this file.\n");exit(0);}
    printf("input astring:\n");gest(str);
    while (str[i])
    {   if(str[i]>='a'&&str[i]<='z') str[i]= __(2)__;
        fputc(str[i],fp);
        i++;
    }
    fclose(fp);
    fp=fopen("test.txt", __(3)__);
    fgets(str,100,fp); printf("%s\n",str);
    fclose(fp);
}
```

**题眼分析　本题首先打开用来存放字符串的文件“** text.txt **”，由于要向文件中写入数据，所以读写模式应为“** w **”，故第（**1**）空处应填“** w **”。接着从键盘输入一个字符串存放到字符数组** str **中，然后通过一个** while **循环从** str **中第** 1 **个字符开始，依次判断字符是否为小写字母，若是则把它转换为大写字母。小写字母与大写字母的** ASCII **码值相差** 32**，所以第（**2**）空处应填“** str[*i*]–32 **”。转换过后应把字符写入到文件。当所有字符均写入到文件中后，关闭该文件。文件关闭后再打开该文件用于从中读取一个字符串，所以打开模式应该是“** r **”，所以第（**3**）空处应填：“** "r" **”。**

**答案**　（1）"w"　　（2）str[*i*] –32　　（3）"r"

# 16.3　文件的读写

## 16.3.1　考点提炼

### 📖考点 1：文件尾测试

**读取文件时，当文件中的数据全部读完后，文件位置指针将位于文件的结尾。此时如果读数据，将会出现错误。为了保证读写数据的正确性，需要进行文件尾测试，文件尾测试使用函数** feof()**，其格式如下：**

**格式：**int feof(FILE *fp)

**功能：测试** fp **指向的文件是否到达文件尾。若到达文件尾，返回值为非** 0**，否则返回值为** 0**。**

### 考点 2：调用 getc（fgetc）和 putc（fputc）函数进行输入和输出

1．读字符函数 fgetc

fgetc 函数的功能是从指定的文件中读一个字符，函数调用的形式为：

```
字符变量=fgetc(文件指针)；
```

例如：

```
ch=fgetc(fp);
```

其意义是从打开的文件 *fp* 中读取一个字符并送入 *ch* 中。

对于 fgetc 函数的使用有以下几点说明：

（1）在 fgetc 函数调用中，读取的文件必须是以读或读写方式打开的。

（2）读取字符的结果也可以不向字符变量赋值，例如：fgetc(fp);但是读出的字符不能保存。

（3）在文件内部有一个位置指针。用来指向文件的当前读写字节。在文件打开时，该指针总是指向文件的第 1 个字节。使用 fgetc 函数后，该位置指针将向后移动一个字节。因此可连续多次使用 fgetc 函数，读取多个字符。应注意文件指针和文件内部的位置指针不是一回事。文件指针是指向整个文件的，须在程序中定义说明，只要不重新赋值，文件指针的值是不变的。文件内部的位置指针用以指示文件内部的当前读写位置，每读写一次，该指针均向后移动，它不需在程序中定义说明，而是由系统自动设置的。

2．字符函数 fputc

fputc 函数的功能是把一个字符写入指定的文件中，函数调用的形式为：

```
fputc(字符量，文件指针)；
```

其中，待写入的字符量可以是字符常量或变量，例如：

```
fputc('a',fp);
```

其意义是把字符'a'写入 *fp* 所指向的文件中。

对于 fputc 函数的使用也要说明几点：

（1）被写入的文件可以用、写、读写，追加方式打开，用写或读写方式打开一个已存在的文件时将清除原有的文件内容，写入字符从文件首开始。如需保留原有文件内容，希望写入的字符以文件末开始存放，必须以追加方式打开文件。被写入的文件若不存在，则创建该文件。

（2）每写入一个字符，文件内部位置指针向后移动一个字节。

（3）fputc 函数有一个返回值，如写入成功则返回写入的字符，否则返回一个 EOF。可用此来判断写入是否成功。

### 考点 3：fscanf 函数和 fprintf 函数

fscanf 函数和 fprintf 函数与前面使用的 scanf 和 printf 函数的功能相似，都是格式化读写函数。两者的区别在于 fscanf 函数和 fprintf 函数的读写对象不是键盘和显示器，而是磁盘文件。

1．格式读函数 fscanf()

格式：int　fscanf (FILE　*fp，const char *format [, address, …])

功能：根据 format 中的格式从 *fp* 指向的文件中读取数据存入到相应的 address 指向的变量

中。

**说明：**

（1）fscanf()根据 frmat 所指向的格式字符串，从 *fp* 指向的文件中读取数据存放到由地址参数所指定的变量中，格式字符的数目必须与输入变量的地址个数相等。fscanf()执行成功则返回输入变量个数，如果没有变量被输入则返回 0。

（2）参数 frmat 指定输入格式。

（3）参数 address 为输入变量的地址列表。

2．格式写函数 fprintf()

**格式：**int　fprintf (FILE *fp,const char *frmat [, argument, …])

**功能：**根据格式字符串 frmat 把 argument 列表中的表达式值写到 *fp* 所指向的文件中。

**说明：**

（1）fprintf()接收一组表达式，每个表达式根据 frmat 格式字符串进行格式化，并输出格式化数据到 *fp* 所指向的文件中。输出表达式的个数与格式字符串中的格式字符的数目相等。若调用成功则返回输出的表达式的数目；错误则返回 EOF。

（2）参数 frmat 指定输出格式。

（3）参数 argument 为输出表达式列表。

### 考点 4：fgets 函数和 fputs 函数

1．读字符串函数 fgets

函数的功能是从指定的文件中读一个字符串到字符数组中，函数调用的形式为：

```
fgets(字符数组名，n，文件指针)；
```

其中的 *n* 是一个正整数。表示从文件中读出的字符串不超过 *n*–1 个字符。在读入的最后一个字符后加上串结束标志'\0'。例如：

```
fgets(str,n,fp);
```

其意义是从 *fp* 所指的文件中读出 *n*–1 个字符送入字符数组 str 中。

2．写字符串函数 fputs

fputs 函数的功能是向指定的文件写入一个字符串，其调用形式为：

```
fputs(字符串，文件指针);
```

其中字符串可以是字符串常量，也可以是字符数组名或指针变量，例如：

```
fputs("abcd", fp)；
```

其意义是把字符串“abcd”写入 *fp* 所指的文件之中。

### 考点 5：数据块读写函数 fread 和 fwrite

C 语言还提供了用于整块数据的读写函数。可用来读写一组数据，如一个数组元素，一个结构变量的值等。

1．数据块读写函数 fread

读数据块函数调用的一般形式为：

```
fread(buffer,size,count,fp);
```

其中 buffer 是一个指针，在 fread 函数中，它表示存放输入数据的首地址。在 fwrite 函数

中，它表示存放输出数据的首地址； size 表示数据块的字节数；count 表示要读写的数据块块数；*fp* 表示文件指针。例如：

```
fread(fa,4,5,fp);
```

其意义是从 *fp* 所指的文件中，每次读 4 个字节(一个实数)送入实数组 *fa* 中，连续读 5 次，即读 5 个实数到 *fa* 中。

2．数据块读写函数 fwrite

写数据块函数调用的一般形式为：

```
fwrite(buffer,size,count,fp);
```

## 16.3.2　题眼分析

### 一、选择题分析

【例 1】若 *fp* 已正确定义并指向某个文件，当未遇到该文件结束标志时函数 foef(fp)的值为_____。(2003 年 9 月)

(A) 0　　(B) 1　　(C) –1　　(D) 一个非 0 值

题眼分析　函数 foef(fp)是用来测试 *fp* 指向的文件是否到达文件尾。若到达文件尾，返回值为非 0，否则返回值为 0。

答案　A

【例 2】有以下程序

```
#include  <stdio.h>
main()
{   FILE *fp; int i,k=0,n=0;
    fp=fopen("dl.dat", "w");
    for(i=1;i<4;i++)  fprintf(fp,"%d",i);
    fclose(fp);
    fp=fopen("dl.dat","r");
    fcanf(fp, "%d%d",&k, &n);  printf("%d%d\n",k, n) ;
    fclose(fp);
}
```

执行后输出结果是_____。(2004 年 4 月)

(A) 1　2　　(B) 123　0　　(C) 1　23　　(D) 0　0

题眼分析　本题首先以创建方式打开文件“d1.dat”，通过 for 循环，3 次调用 fprintf()把 1、2、3 写入文件“dl.dat”。由于在写的时候并没有用空格隔开，因此文件“d1.dat”中的内容为“123”。接着，把该文件关闭再以读方式打开，文件位置指针指向文件头，再通过 fscanf()函数从中读取数据，这时会将“123”视为一个数并赋值于 *k*，而 *n* 并没有读入数，仍为 0。

答案　B

【例 3】有以下程序：

```
#include<stdio.h>
main()
{   FILE  *fp;int i=20,j=30,k,n;
    fp=fopen("d1.dat","w");
```

```
        fprintf(fp,"%d\n",i);fprintf(fp,"%d\n",j);
        fclose(fp);
        fp=fopen("d1.dat","r");
        fscanf(fp,"%d%d",&k,&n);
        printf("%d  %d",k,n);
    }
```

**程序运行后的输出结果是_____。**（2002 年 9 月）

（A）20　30　　（B）20　50　　（C）30　50　　（D）30　20

**题眼分析**　本题首先以创建方式打开文件“ d1.dat ”，两次调用 fprintf()把 *i* 和 *j* 的值写到文件“ d1.dat ”中，文件“ d1.dat ”中的内容为 20<回车>30<回车>。然后把该文件关闭再以读方式打开，文件位置指针指向文件头，再通过 fscanf()函数从中读取两个整数到 *k* 和 *n* 中，由于格式符之间无间隔，因此输入数据可以用回车隔开，故输入的 *k* 的值为 20，n 的值为 30。

**答案**　A

**【例 4】在 C 程序中，可把整型数以二进制形式存放到文件中的函数是_____。**（2000 年 4 月）

（A）fprintf()函数　　（B）fread()函数　　（C）fwrite()函数　　（D）fputc()函数

**题眼分析**　答案（A）是格式化写函数，它的作用是把格式化数据写到文件中去，可以写整型数，但通常都是以文本的方式写。答案（B）是数据块读函数，其作用是从文件中读取若干个二进制字节，不符合要求；答案（C）是数据块写函数，它可以把数据（包含整型数据）以二进制方式写入到文件中，所以符合本题题意；答案（D）是字符写函数，显然不符合题意。

**答案**　C

## 二、填空题分析

**【例 1】以下程序用来统计文件中字符个数，请填空。**（2002 年 4 月）

```
#include "stdio.h"
main()
{   FILE *fp; long num=0L;
    if((fp=fopen("fname.dat","r"))==NULL)
        {   pirntf("Open error\n"); exit(0);}
    while(_______){ fgetc(fp); num++;}
    printf("num=%1d\n",num-1);
    fclose(fp);     }
```

**题眼分析**　本题中统计文件中字符个数的算法可描述如下：首先判断文件位置指针是否指向了文件尾，如果不是则读出一个字符，同时字符的个数加 1；再判断文件位置指针是否位于文件尾，如果不是则读出一个字符，同时字符的个数加 1，……，如此循环直到文件位置指针位于文件尾为止。本题首先以读文件的方式打开了文件“ fname.dat ”，如果打开成功则把返回的文件型指针赋值给 *fp*，然后通过循环求文件中的字符数，首先应是判断文件位置指针是否位于文件尾，如果不是在循环中读取字符，字符数加 1。所以下划处应填循环条件“ 文件位置指针不是处于文件尾即“ !feof(fp) ”。

**答案**　!feof(fp)

**【例 2】已有文本文件 test.txt，其中的内容为：Hello,everyone!。以下程序中，文件 test.txt**

已正确为“读”而打开，由文件指针 *fr* 指向该文件，则程序的输出结果是_____。（2003 年 4 月）

```
#include  <stdio.h>
main()
{   FILE *fr, char str[40];
    ……
    fgets(str,5,fr);
    printf("%s\n",str);
    fclose(fr);     }
```

**题眼分析**　文件的字符串读写函数 fgets 有 3 个参数，第 3 个参数是文件指针指向要读取数据的文件，第 2 个参数是一个整数（假设为 *n*），表示从文件中读取 *n*–1 个字符并在其后加一个'\0'，第 1 个参数为存放读取的字符串的内存区的起始地址，读取的数据保存在其中。可见本题的输出结果为：Hell。

**答案**　Hell

# 16.4　文件定位函数

## 16.4.1　考点提炼

文件的定位主要用于文件的随机读写，即先把文件位置指针移到指定位置，然后再读写数据。注意：“文件位置指针”和“文件指针”是两个完全不同的概念。

**考点 1：文件的随机定位函数** fseek()

**格式：**int　fseek (FILE *fp, long offset, int from)

**功能：移动文件位置指针到指定位置。**

**说明：**

（1）fseek()**把文件位置指针移动到与** from **所指定的文件位置距离** offset **个字节处，如果指针移动成功，则返回** 0**，出错时返回非** 0**。**

（2）**参数** offset **为字节偏移量，为长整型数据，正数代表前进，负数代表后退。**

（3）**参数** form **代表移动的开始位置。其值可以是** 0、1、2 **中的一个，代表文件开始位置** SEEK_SET（0）**，文件当前位置** SEEK_CUR（1）**和文件末尾** SEEK_END（2），SEEK_SET、SEEK_CUR **和** SEEK_END **这** 3 **个符号常量定义在头文件“** stdio.h **”中。**

**考点 2：当前位置测试函数** ftell()

**格式：**long　ftell (FILE *fp)。

**功能：得到** *fp* **指向的文件的文件位置指针位置。**

**说明：**ftell()**在调用成功后返回当前指针位置，出错时返回**–1L。

**考点 3：文件头定位函数** rewind()

**格式：**void　rewind (FILE *_fp_)

**功能：将文件位置指针定位于文件的开始头。**

**说明：该函数的作用是把文件位置指针返回到文件开始处，清除文件结束标志和出错标志，无返回值。**

## 16.4.2　题眼分析

### 一、选择题分析

**【例 1】有以下程序（提示：程序中** fseek(*fp*, –2L*sizeof(int),SEEK_END);**语句的作用是使位置指针从文件末尾向前移** 2*sizeof(int)**字节）**

```
#include  <stdio.h>
main()
{   FILE  *fp;  int i，a[4]={1,2,3,4},b;
    fp=fopen("data.dat ","wb");
    for(i=0;i<4;i++)fwrite(&a[i], sizeof(int),1,fp);
    close(fp)
    fp= fopen("data.dat","rb");
    fseek(fp,-2L* sizeof(int), SEEK_END);
    fread(&b, sizeof(int),1,fp);  /*从文件中读取sizeof(int)字节的数据到变量b
中/*
    fclose(fp);
    printf("%d\n"，b);
}
```

**执行后输出结果是_____。（2004 年 4 月）**

（A）2　　（B）1　　（C）4　　（D）3

**题眼分析**　**本题首先以创建二进制文件方式打开“** data.dat **”，通过** for **循环，**4 **次调用数据块写入函数** fwrite()**把“** 1 **”、“** 2 **”、“** 3 **”、“** 4 **”的二进制形式写入文件“** data.dat **”，每次写入** sizeof(int)**字节。接着关闭文件，并以读取二进制文件方式再次打开‘** data.dat **’文件，通过‘** fseek(*fp*, –2L*sizeof(int),SEEK_END); **”语句使文件位置指针从文件末尾向前移** 2*sizeof(int)**字节，即将文件位置指针指向“** 3 **”的前面。最后通过数据块读取函数** fread **读** sizeof(int)**个字节至变量** b**，因此变量** b **的值为** 3**。**

**答案**　D

**【例** 2**】若** fp **为文件指针，且文件已正确打开，**

```
fseek(fp,0,SEEK_END);
i=ftell(fp);
printf("i=%d\n", i);
```

**以下语句的输出结果为_____。**

（A）*fp* **所指文件的记录长度**　　（B）*fp* **所指文件的长度，以字节为单位**

（C）*fp* **所指文件的长度，以比特为单位**　　（D）*fp* **所指文件当前位置，以字节为单位**

**题眼分析**　fseek 函数的作用是使文件定位到文件末尾，ftell 函数返回文件位置指针的位置，以字节数的形式。显然文件末尾的文件位置指针的位置就是文件的长度。

**答案**　B

**【例 3】以下与函数** fseek(fp,0L,SEEK_SET)**有相同作用的是_____。（2005 年 4 月）**

（A）feof(fp)　　（B）ftell(fp)　　（C）fgetc(fp)　　（D）rewind(fp)

**题眼分析**　在本题中，fseek 函数的作用是使文件定位到文件开始位置，而这一工作可由 rewind 函数来完成，因此选项（D）是正确的。

**答案**　D

**二、填空题分析**

**【例】在对文件进行操作的过程中,若要求文件的位置回到文件的开头，应当调用的函数是_____函数。（1998 年 5 月）**

**题眼分析**　文件头定位函数名为 rewind()，故应填“ rewind() ”。

**答案**　rewind()

# 16.5　单元训练及参考答案

## 16.5.1　单元训练

**一、选择题**

1．**为读而打开二进制文件** c:\aa.dat **的正确写法是_____。**

（A）fopen("c:\\aa.dat", "rb")　　（B）fp= fopen("c:\aa.dat", "r")

（C）fopen("c:\\aa.dat", "wb")　　（D）fp= fopen("c:\aa.dat", "w")

2．**当已存在一个** abc.txt **文件时，执行函数** fopen("abc.txt" , "r+")**的功能是_____。**

（A）**打开** abc.txt **文件，覆盖原有的内容**

（B）**打开** abc.txt **文件，可以读取和写入新的内容**

（C）**打开** abc.txt **文件，只能写入数据，但不能读出数据**

（D）**打开** abc.txt **文件，只能读取原有内容，但不能写和数据**

3．fopen()**函数的第** 2 **个参数取值**"r"**和**"w"**时，它们之间的差别是_____。**

（A）"r"**可向文件写入，**"w"**不但可以向文件写入而且还可以读出**

（B）"r"**用于从文件读出数据，**"w"**用于向文件中写入数据**

（C）**当文件不存在的时候，**"r"**将创建一个文件并读出，**"w"**将创建文件并写入**

（D）**文件不存在时，**"r"**建立新文件**,，"w"**出错**

4．fopen()**函数的第** 2 **个参数取值**"w+" **和**"a+"**时都可以写入数据，它们之间的差别是_____。**

（A）"w+"**时可在中间插入数据，而**"a+"**时只能在末尾追加数据**

（B）"w+"**时和**”a+”**时只能在末尾追加数据**

（C）在文件存在时，"w+"时清除原文件数据，而"a+"时保留原文件数据

（D）"w+"时不能在中间插入数据，而"a+"时只能在末尾追加数据

5．若用 fopen()函数打开一个新的二进制文件，该文件可以读也可以写，则文件打开模式是_____。

（A）"ab+"　　（B）"wb+"　　（C）"rb+"　　（D）"ab"

6．若用 fopen()函数打开一个已存在的文本文件，保留该文件原有数据且可以读也可以写，则文件打开模式是_____。

（A）"r+"　　（B）"w+"　　（C）"a+"　　（D）"a"

7．使用 fseek 函数可以实现的操作是_____。

（A）改变文件位置指针的当前位置　　（B）文件的顺序读写

（C）文件的随机读写　　（D）以上都不是

8．以下不能将文件位置指针重新移到文件开头位置的函数是_____。

（A）rewind(*fp*)　　（B）fseek(*fp*,0,SEEK_SET)

（C）fseek(*fp*, –(long)ftell(*fp*),SEEK_CUR)　　（D）fseek(*fp*,0,SEEK_END)

9．检测指向的文件位置指针在文件头的条件是_____。

（A）*fp*==0　　（B）ftell(*fp*)==0

（C）fseek(*fp*,0,SEEK_SET)　　（D）feof(*fp*)

## 二、填空题

1．下列程序实现的功能是从终端上读入的 10 个整数以二进制方式写入一个名为“bi.bat”的新文件中，请填空。

```
main()
{   int i,j; FILE *fp;
    if ( (fp=fopen("bi.bat" , __(1)__))= =NULL)    exit(0);
    for(i=0;i<10;i++)
    {   scanf("%d",&j);fwrite(__(2)__,sizeof(int),1,fp ); }
    fclose(fp);     }
```

2．设 D 盘根目录下有一个名为“aaaa.dat”的二进制文件，其中连续存放了 100 个整数。编程实现下面要求：

（1）读取其中的奇数编号（从 0 开始编号）的整数并输出；

（2）读取并输出第 99 个整数和第 100 个整数。请填空。

```
#include "stdio.h"
main()
{   FILE  *fp; int i,x;
    if((fp=fopen(__(3)__,"rb"))==null)
    {   printf("Cannot open file d:\\aaaa.txt");  exit(0);   }
    for(i=1;i<=99;i+=2)
    {   fseek(fp,(long)(i*sizeof(int)),SEEK_SET);
        fread(&x,sizeof(int),1,fp);printf("%d...%d\n",i,x);    }
    fseek(fp,-2L*sizeof(int),SEEK_END);
    fread(__(4)__);
```

```
        printf("%d...%d\n",99,x);
        fseek(fp,-1L*sizeof(int),SEEK_END);
        fread(&x,sizeof(int),1,fp);
        printf("%d...%d\n",100,x);
        fclose(fp);
}
```

3．**下列程序实现的功能是将用户从键盘上随机输入的** 30 **个学生的学号、姓名、数学成绩、计算机成绩及总分写入数据文件** score **中，假设** 30 **个学生的学号从** 1~30 **连续。输入时不必按学号顺序进行，程序自动按学号顺序将输入的数据写入文件。请填空。**

```
#include <stdio.h>
main()
{   FILE *fp;
    struct st
    {   int number;  char name[20];
        float math;   float computer;
        float total;
    } student;
    int i,j;
    if ((fp=fopen("score" , ___(5)___))= =NULL)
    {   printf("file open error\n"); exit(0); }
    for (i=0;i<30;i++)
    {   scanf("%d,%20s,%f,%f",
        &student.number,student.name,&student.math,&student.computer);
        student.total=student.math+student.computer;
        j=student.number-1;
        fseek(fp, ___(6)___(long)j*sizeof(struct st),SEEK_SET);
        if (fwrite(&student, ___(7)___,1,fp)!=1)  printf("write file error\n");
    }
    fclose(fp);
}
```

## 16.5.2　参考答案

### 一、选择题

| | | | | |
|---|---|---|---|---|
| 1．A | 2．B | 3．B | 4．C | 5．B |
| 6．C | 7．A | 8．D | 9．B | |

### 二、填空题

（1）"wb"　　（2）&j

（3）"d:\\aaaa.dat"　　（4）&x,sizeof(int),1,*fp*

（5）"wb+"　　（6）(long)j*sizeof(struct st)

（7）sizeof(student)

# 第 17 章　上机考试专题辅导

## 17.1　上机考试试题类型简介

从 2004 年 9 月，二级 C 语言的考试类型由 DOS 操作题、程序改错题和程序设计题，改为程序填空题、程序改错题和程序设计题。下面分别对这 3 种题型做简要介绍。

### 17.1.1　程序填空题

程序填空题是新增加的题型，主要是考查考生程序阅读理解能力。对于程序填空题，题目给出程序的功能和源代码，在源代码中抽去 2～3 语句或表达式，需考生来填写。做程序填空题时不能在程序中增加语句或删除语句，也不得更改程序的结构，只需在程序的下划线处填入正确的内容并把下划线删除，使程序得出正确的结果即可。

程序填空题基本上和笔试的填空题相似，主要是搞清解题思路和有关变量的含义。在解题的方法和注意事项基本上和笔试填空题一样，这里就不再赘述。

### 17.1.2　程序改错题

1．关于程序改错题

程序改错题是考查考生的程序调试能力。对于程序改错题，题目中给出程序的功能和源代码，在源代码中有 2～3 处错误不等，这些错误均在以“ /**********found**********/ ”的标志下方。有的题目明确指出错误紧跟下面第 1 行，而有的没有明确指出来。考生应重点注意下面的第 1 条语句，这条语句有可能是一个复合语句，分成几行。

需要注意，有时候几条语句出现错误，实际上是一条错误引起的连锁反应，这时要抓住要害，由前向后修改，当按 Ctrl+F9 编译出现多处错误时，千万不要紧张或错误地以为考题有误，往往修改一个错误后，其他错误就自动消失。

在做程序改错题时，要按照源程序的思路来修改。总的来说，需要修改的地方很少，有时仅仅需要增加或删除一个符号。因此，切忌打破原程序结构，特别注意不要增加或删除语句（题目中明确指出要增删语句时除外）。

2．错误的分类

一般情况下，错误主要分为语法错误和逻辑错误两大类。

（1）语法错误。对于语法错误，用编译器很容易解决。所以，程序改错题的第 1 步是先进行编译，解决这类语法错误。

在程序改错题中，常见的有以下几类语法错误：①丢失分号，或分号误写成逗号。②关键字拼写错误；如本来小写变成大写。③语句格式错误，例如 for 语句中多写或者少写分号。④表达式声明错误，例如：少了()。⑤函数类型说明错误。与 main()函数中不一致。⑥函数形参

类型声明错误。例如，少*等。⑦运算符书写错误，例如：/写成了\。

（2）逻辑错误。逻辑错误又叫语义错误，这类错误是和程序功能紧密相关，用编译器一般不能发现这类错误。

对于逻辑错误可以按这样的步骤进行查找：①先读试题，看清题目的功能要求。②通读程序，看懂程序中算法的实现方法。③细看程序，发现常见错误点。

下面列出了一些常用逻辑错误，以供参考：①变量初值错误。②循环次数不对。③下标越界。④运算类型不匹配。

### 17.1.3　程序设计题

1．关于程序设计题

程序设计是考查考生的程序编制能力。对于程序设计题，题目中给出程序的功能和部分源代码。在源代码中，给出程序主函数 main 和一些辅助函数。对于这部分源程序，一定不要修改，增加或删除行，也不能更改程序结构。考生只需要在函数 fun 的花括号中填入实现程序功能的若干语句。

2．程序设计题的解题思路

要做好程序设计题，考生应在平时就要多做一些典型算法的习题，掌握这些算法的实现过程。下面给出做程序设计题的思路及一些注意事项以供参考。

（1）审题。在做程序设计题的第 1 步是审题。在审题时，要认真阅读试题说明，收集有关信息。这些信息包括：①要实现的功能，如“编写程序，实现矩阵的转置（即行列互换）”。②使用的算法，如“使用冒泡法进行排序”。③算法一些术语的解释，如“回文数是指其各位数字左右对称的整数”等。④限制条件，如“不得使用 C 语言提供的字符串函数”等。

（2）清晰地理解函数 fun 功能和实现。理解 fun 函数功能和实现主要包括：①函数 fun 的功能。②函数 fun 是否有返回值及返回值的类型。③主函数 main 与函数 fun 之间参数传递方式，是通过“值传递”还是通过“指针（地址）传递”。④运行结果。

（3）实现函数 fun。在实现函数 fun 时，需要注意以下几点：①根据 fun 函数的功能及考点，决定采用的算法，此时要联想平时做过的同类型的习题。②“扬长避短”。如果题目中没有规定算法，可以有几种算法或者有几种实现语句可选，则选用自己最擅长的。③定义合适的临时变量，并赋初值。根据函数 fun 的功能、主函数调用情况及程序执行结果，要定义合适的临时变量，并注意根据程序的需要及时、合理的给变量赋初值。④良好的编程序风格。良好的编程序风格有利于程序的阅读和错误的排除，例如，使用具体含义的变量名；利用空格、空行、缩进等使程序层次清晰；避免使用复杂或难以理解的表达式和语句（如自增自减运算符、条件运算符、逗号表达式等）；把握运算顺序，多加圆括号。

（4）调试运行程序。在调试运行程序时，需要注意以下事项：①先调试程序，如果直接运行，可能出现无限循环或死机现象。调试程序可以发现语法错误。调试时需要有意识地注意循环变量是否赋初值、循环界限、循环变量是否递增或递减，这样可以有效地防止无限循环情况的发生。②运行程序。首先用题目给的输入数据，观察运行结果是否与题目所给的相同。如果不同，可能是函数返回类型或通过形参返回时出现错误，也可能是程序逻辑或算法不正确。然后用几组特殊的值判断结果是否正确。③利用好 TC 的调试工具。如把 F7、F8 和 Ctrl+F4

组合运用。

## 17.2 常考题型提炼

分析近年来的机考题，发现试题基本上变化不大，每年的试题重复率很高。机考题分为几大类，分别是：数的转换与计算、数列与级数求和、矩阵运算、数组处理、排序、字符串运算、结构体及链表、文件操作、数学问题和实际应用等。下面将对上述几类问题做概括讲解。

### 17.2.1 题型 1：数的转换与计算

数的转换与计算通常有以下几类问题：数的按位分离及合并、数制的转换、素数问题、四舍五入问题、整除及奇偶判断问题。

1．数的按位分离及合并

【例 1】函数 fun 的功能是：将两个两位数的整数 *a*、*b* 合并形成一个整数放在 *c* 中。合并的方式是：将 *a* 的十位和个位数依次放在 *c* 数的千位和十位上，*b* 数的十位和个位数依次放在 *c* 数的个位和百位上。

例如：当 *a*=45，*b*=12。调用该函数后，*c*=4251

```
#include <conio.h>
#include <stdio.h>
void fun(int a, int b, long *c)
{
}
main()
{   int a,b; long c;
    printf(" input a, b: ");
    scanf("%d%d", &a,&b);
    fun(a,b,&c);
    printf(" the result is :%ld\n", c);
}
```

题眼分析　本题的关键在于如何表示出个、十、百、千位数。程序实现思路是：(1) 按位拆分这两个正整数。对于一个两位的整数，用 10 对它求余（%）得到个位数上的数，将它除以（/）10 得到十位数上的数。(2) 按位合并。每位上的数字与该位的位数相乘，然后各位相加，对于一个 4 位数 $a_3a_2a_1a_0$ 其合并方式为：$a_3a_2a_1a_0=a_3*1000+a_2*100+a_1*10+a_0$。

答案

```
void fun(int a, int b, long *c)
{   int i,j,k,n;
    i=a%10; j=a/10; k=b%10; n=b/10;
    *c=j*1000+k*100+i*10+n;
}
```

2．素数问题

【例 2】函数 fun 的功能是：将所有大于 1 小于整数 *m* 的非素数存入 *xx* 所指数组中，非

素数的个数通过 *k* 传回。

例如，若输入 17，则应输出 9 和 4 6 8 9 10 12 14 15 16。

```
#include <conio.h>
#include <stdio.h>
void fun( int m, int *k, int xx[] )
{
}
main()
{   int m, n,zz[100];
    printf("\n Please enter an integer number between 10 and 100: ");
    scanf( "%d",&n);
    fun(n,&m,zz);
    printf("\n\n There are %d non-prime numbers less than %d: ", m,n);
    for(n=0; n<m; n++)
        printf("\n %4d",zz[n]);
}
```

题眼分析　本题的关键是求素数的算法。素数是指那些大于 1，且只能被 1 和其本身整除的数。为了判断一个数 *n* 是否是素数，最简单的方法是用 2～*n*–1[或 sqrt(*n*)或 *n*/2]逐个去除 *n*，只要能被一个数整除，*n* 就不是素数，若不能被任何一个数整除，*n* 才是素数。

程序实现思路是：(1) 用一个外循环遍历从 2 到 *m* 之间的整数；(2) 用一个内循环判断是否是素数；(3) 如果存在一个数能被 2 到 *i*–1 的数整除，说明此数为非素数，应该保存在形参数组 xx[]中。

答案

```
void fun( int m, int *k, int xx[] )
{   int i,j;
    *k=0;
    for( i=2; i<m; i++)          /*遍历从 2 到 m 之间的整数*/
        for( j=2; j<i; j++)      /*用于判断是否是素数*/
            if( i%j==0)          /*余数为 0 则表示 i 不是素数*/
            {   xx[(*k)++]=i;    /*将 i 存入 xx 数组中*/
                break;           /*它提前结束循环，不用亦可。*/
            }
}
```

3．整除及奇偶判断问题

【例 3】编写函数 fun()，它的功能是求 *n* 以内(不包括 *n*)同时能被 5 与 11 整除的所有自然数之和的平方根 *s*，并作为函数值返回。

例如：*n* 为 1000 时，函数值应为 *s*=96.979379。

```
#include   <conio.h>
#include   <math.h>
#include   <stdio.h>
double  fun(int n)
{
}
```

```
main()
{
    clrscr();
    printf("s=%f\n", fun(1000));
}
```

**题眼分析　本题的解题思路是：(1) 逐个取得从 0～n 之间的所有数；(2) 对每次取得的数进行条件判断，条件是既能被 5 整除同时又能被 11 整除（注意：这两个条件要求同时成立，因此用到了"&&"运算符。满足条件，该数就被累加到 *s* 中去）；(3) 求出所有符合条件的数后，用 sqrt()函数(包含于头文件<math.h>中)对 *s* 求平方根。**

**答案**

```
double  fun(int n)
{   double  s=0.0;
    int  i;
    for(i=0; i<n;i++)                /*从 0～n 中找到*/
        if(i%5==0&&i%11==0)          /*既能被 5 整除同时又能被 11 整除的数*/
            s=s+i;                   /*将这些数求和*/
    s=sqrt(s);                       /*对 s 求平方根*/
    return  s;                       /*函数返回值*/
}
```

**补充说明　对于整除及奇偶判断问题，使用求余运算（%），当一个整数能被另一个整数整除，则求余运算结果为 0。若要进行偶数判断，只需要对这个数对 2 求余，为 1 时表示原数为奇数，否则为偶数。**

**4．四舍五入问题**

**【例 4】函数 float fun(double h)的功能是对变量 *h* 中的值保留两位小数，并对第 3 位进行四舍五入（规定 *h* 中的值为正数）。**

**例如：若 *h* 的值为 8.32433，则函数返回 8.32，若 *h* 的值为 8.32533，则函数返回 8.33。**

```
#include   <stdio.h>
#include   <conio.h>
double  fun(double h)
{
}
main()
{   float a;
    clrscr();
    printf("Enter a:  ");
    scanf("%f", &a);
    printf("The original data is :  ");
    printf("%f \n\n", &a);
    printf("The result : %f\n", fun(a));
}
```

**题眼分析　本题的解题思路是：(1) 将 *h* 乘以 1000；(2) 对此数加 5 再与 10 整除，其个位将被舍弃；(3) 把所得的整数强制转换成 float 型再除以 100 就得到最后结果。**

**答案**

```
double  fun(double h)
{   long int t;
    float s;
    h=h*1000;
    t=(h+5)/10;
    s=(float)t/100.0
    return(s);
}
```

**补充说明**　例题中是保留两位，如果要保留 n 位，操作步骤如下：(1) 将待转换的实数乘以 10 的 $n$+1 次方；(2) 对此数加 5 再与 10 整除，其个位将被舍弃；(3) 把所得的整数强制转换成 float 型再除以 10 的 $n$ 次方就得到最后结果。

### 17.2.2　题型 2：数列与级数求和

**1．数列**

**【例 1】请编写函数 fun()，它的功能是求 Fibonacci 数列中小于 $t$ 的最大的一个数，结果由函数返回。其中 Fibonacci 数列 F($n$)的定义为**

$$F(0)=0，F(1)=1$$

$$F(n)=F(n-1)+F(n-2)$$

**例如：$t$=1000 时 ，函数值为 987。**

```
#include   <conio.h>
#include   <math.h>
#include   <stdio.h>
int  fun(int t)
{
}
main()
{
    int  n;
    clrscr();
    n=1000;
    printf("n=%d,  f=%d\n",n, fun(n));
}
```

**题眼分析**　这是一个典型的数列问题，解决这类问题的关键是搞清楚数列前后项的关系，并注意保存已运算的结果。

在本题中，根据所给数列定义，不难发现该数列最终的结果是由两个数列之和组成。程序实现的思路是：(1) 定义 3 个变量 $a$、$b$、$c$，把变量 $c$ 看成是前两项之和(即第 $n$ 项)，而变量 $a$ 始终代表第 $n$–2 项，变量 $b$ 始终代表第 n–1 项；(2) 通过循环，不断地重新赋值来实现数列的计算；(3) 当变量 c 大于指定比较的数时，退出循环；(4) 由于这时的变量 $c$ 的值已经不满足要求了，而它的前一个数是要求的数，因此使用它的前一个数作为函数返回值。

**答案**

```
int  fun(int t)
```

```
{
    int  a=1,b=1,c=0,i;      /*a 代表第 n-2 项，b 代表第 n-1 项，c 代表第 n 项*/
    /*如果求得的数 c 比指定比较的数小，则计算下一个 Fibonacci 数，对 a,b 重新置数*/
    do
    {
        c=a+b;
        a=b;
        b=c;
    }while (c<t);   /*如果求得的数 c 比指定比较的数大时，退出循环*/
    c=a;            /*此时数 c 的前一个 Fibonacci 数为小于指定比较的数的最大的数*/
    return  c;
}
```

**2．级数求和**

**【例 2】下列给定程序中，函数 fun()的功能是根据整型形参 *m*，计算如下公式的值。**

$y=1-1/(2\times2)+1/(3\times3)-1/(4\times4)+\cdots+(-1)^{(m+1)}/(m\times m)$

**例如：m 中的值为 5，则应输出 0.838611。**

```
#include  <conio.h>
#include  <stdio.h>
double fun(int  m)
{   double  y=1.0;
/**********************found**********************/
    int j=1;
    int  i;
    for(i=2; i<=m; i++)
    {
        j=-1*j;
/**********************found**********************/
    y+=1/(i * i);
    }
    return(y);
}
main()
{
    int  n=5;
    clrscr();
    printf("\nThe result is %lf\n" ,fun(n));
}
```

**题眼分析　这是一个典型的数列求和问题，在解这类问题时需要注意几点：(1) 用于累加或累乘的变量要先赋初值；(2) 运算结果往往是实型数，在定义变量和运算时必须要注意类型；(3) 注意级数每一项的符号是否交替变化，如果前一项是正数，而后一项是负数，通常使用一个变量实现；(4) 做这类题目时，一般应分析公式的特点，把它拆分成几个独立的单元，分别求值，然后组合。**

**在本题中，变量 *j* 的作用就是实现每一项的符号的交替变化。由于程序中使用的循环变量是整数，它在和整数进行算术运算时，结果也是整数。因此，本题错误之一是变量 *j* 的类型定**

义错误；错误之二是在求和时没有体现每一项的符号的交替变化。

答案（1）将 int j=1;修改为 double j=1.0;

（2）将 y+=1/(i * i); 修改为 y+=j/(i * i);

### 17.2.3　题型 3：矩阵运算

矩阵问题其实就是对二维数组(aa[*M*][*N*])，bb[*M*][*M*]）处理，主要包括矩阵的转置、矩阵的加减乘除运算、半三角元素运算及求周边元素的和或平均值。

1．矩阵行(或列)互换

【例 1】给定程序中，函数 fun 的功能是：将 *N*×*N* 矩阵中元素的值按列右移 1 个位置，右边被移出矩阵的元素绕回左边。例如，*N*=3，有下列矩阵

1　　2　　3
4　　5　　6
7　　8　　9

计算结果为

3　　1　　2
6　　4　　5
9　　7　　8

```
#include   <stdio.h>
#define   N   4
void fun(int  (*t)[N])
{  int  i, j, x;
/**********found**********/
   for(i=0; i<__(1)___; i++)
     {
/**********found**********/
       x=t[i][__(2)___] ;
       for(j=N-1; j>=0; j--)
         t[i][j]=t[i][j-1];
/**********found**********/
       t[i][__(3)___]=x;
     }
}
main()
{  int  t[][N]={21,12,13,24,25,16,47,38,29,11,32,54,42,21,33,10}, i, j;
   printf("The original array:\n");
   for(i=0; i<N; i++)
   {  for(j=0; j<N; j++)  printf("%2d  ",t[i][j]);
      printf("\n");
   }
   fun(t);
   printf("\nThe result is:\n");
   for(i=0; i<N; i++)
```

```
    {  for(j=0; j<N; j++) printf("%2d  ",t[i][j]);
       printf("\n");
    }
}
```

**题眼分析　本题的目的是进行矩阵的列互换，关键在于对每一行的最后的一个元素的处理，要将它先保存起来，在其他元素移动完成后，将其放入该行第 1 个位置。**

**程序的实现思路是：(1) 使用一个外循环来控制行，很显然是从第 0 行开始，到第 $N-1$ 行结束，因此循环可以写成 for($i$=0;i<$N$;$i$++)；(2) 保存每一行的最后一个元素到临时变量 $x$ 之中；(3) 使用一个内循环来完成元素的移动，移动的方法是：将这一行第 $N-2$ 个元素移动到 $N-1$ 的位置，$N-3$ 移动到 $N-2$ 的位置，依此类推；(4) 将临时变量 $x$ 的值赋值给该行的第 0 个元素。**

**答案　(1) $N$　　(2) $N-1$　　(3) 0**

**2．半三角元素运算**

**【例 2】程序定义了 $N\times N$ 的二维数组，并在主函数中自动赋值。请编写函数 fun(int $a$[][$N$])，函数的功能是使数组左下三角元素中的值全部置成 0。例如：$a$ 数组的值为**

$$a=\begin{vmatrix}1&9&7\\2&3&8\\4&5&6\end{vmatrix}$$

**则返回主程序后 $a$ 数组中的值应为**

$$\begin{vmatrix}0&9&7\\0&0&8\\0&0&0\end{vmatrix}$$

```
#include <stdio.h>
#include <stdlib.h>
#define  N  5
int  fun ( int a[][N] )
{
}
main ( )
{  int  a[N][N], i, j;
   printf("***** The array *****\n");
   for ( i =0;  i<N; i++ )
   {  for ( j =0; j<N; j++ )
     {  a[i][j] = rand()%10; printf( "%4d", a[i][j] ); }
        printf("\n");
   }
   fun ( a );
   printf ("THE  RESULT\n");
   for ( i =0;  i<N; i++ )
   {  for ( j =0; j<N; j++ ) printf( "%4d", a[i][j] );
      printf("\n");
   }
   NONO( );
}
NONO( )
{/* 本函数用于打开文件，输入数据，调用函数，输出数据，关闭文件。 */
```

```
 /* 省略*/
}
```

**题眼分析**　本题的目的是使矩阵的左下三角元素中的值全部置成 0，关键在于对行列下标上限的确定。

**程序的实现思路是：**（1）使用一个外循环来控制行，循环可以写成 for(*i*=0;*i*<*N*;i++)；（2）使用一个内循环来控制左下三角，控制方法是让循环变量 *j* 从 0 到 *i*，即 for(*j*=0;*j*<=*i*;*j*++)；（3）将 *a*[*i*][*j*]置 0。

**答案**

```
int  fun ( int a[][N] )
{   int i,j;
    for(i=0; i<N; j++)
        for(j=0; j<=i; j++)
            a[i][j]=0;
}
```

3．矩阵的转置

【例 3】编写程序，实现矩阵（3 行 3 列）的转置（即行列互换）。

例如，若输入下面的矩阵：

100　200　300
400　500　600
700　800　900

程序输出：

100　400　700
200　500　800
300　600　900

注意：部分源程序给出如下。

请勿改动主函数 main 和其他函数中的任何内容，仅在函数 fun 的花括号中填入你编写的若干语句。

```
#include <stdio.h>
#include <conio.h>
int fun(int array[3][3])
{
}
main()
{   int i,j;
    int array[3][3]={{100,200,300},{400,500,600},{700,800,900}};
    clrscr();
    for(i=0;i<3;i++)
    {   for(j=0;j<3;j++)
            printf("%7d",array[i][j]);
        printf("\n");
    }
    fun(array);
```

```
    printf("Converted array:\n");
    for(i=0;i<3;i++)
    {   for(j=0;j<3;j++)
        printf("%7d",array[i][j]);
        printf("\n");
    }
}
```

**题眼分析　本题的目的是进行矩阵的转置，关键在于进行行列下标转换的算法。由矩阵的对称性，不难看出在进行行列互换时 $a[j][i]$正好是与 $a[j][i]$互换，因而只要让程序遍历矩阵的右上角即可，可以通过两个循环来实现。**

**程序的实现思路是：**(1)**使用一个外循环来控制行，循环可以写成** for($i$=0;$i$<2;$i$++)**；**(2)**使用一个内循环来控制右上角，控制方法是让循环变量** $j$ **从** $i$+1 **到** 2**，即** for($j$=$i$+1;$j$<3;$j$++)**；**(3)**实现** $a[i][j]$**与** a[$j$][$i$]**互换。**

**答案**

```
int fun(int array[3][3])
{   int i,j,t;
    for(i=0;i<2;i++)                /*控制行 */
        for(j=i+1;j<3;j++)          /*控制左上角 */
        {   t=array[i][j];array[i][j]=array[j][i]; array[j][i]=t;}
                                    /* a[j][i]与 a[j][i]互换*/
}
```

**4．求矩阵周边元素的和或平均值**

**【例** 4**】函数** fun **的功能是求出二维数组周边元素之和，作为函数值返回。二维数组中的值在主函数中赋予。**

**例如：二维数组中的值为**

$$a=\begin{bmatrix}1&3&5&7&9\\2&9&9&9&4\\6&9&9&9&8\\1&3&5&7&0\end{bmatrix}$$

**则函数值为** 61**。**

```
#include <conio.h>
#include <stdio.h>
#define M 4
#define N 5
int fun(int a[M][N])
{
}
main()
{   int aa[M][N]={{1,3,5,7,9},
                  {2,9,9,9,4},
                  {6,9,9,9,8},
                  {1,3,5,7,0}};
```

```
    int i,j,y;
    clrscr();
    printf("The original data is :\n");
    for(i=0;i<M;i++)
    {   for(j=0;j<N;j++) printf("%6d",aa[i][j]);
        printf("\n");
    }
    y=fun(aa);
    printf("\nThe sum: %d\n",y);
    printf("\n");
}
```

**题眼分析　本题的目的是求矩阵周边元素的值，可以通过两个循环来实现。特别要注意的是，不能重复累加四角元素。程序的实现思路是：(1) 求第 1 行和第 $M$ 行所有元素的之和；(2) 求第 1 列和第 $N$ 列除两端元素以外（即矩阵四角）所有元素的和。**

**答案**

```
/*注：该题的第 1 个 for()循环是计算矩阵的最上一行和最下一行的总和，第 2 个 for()是计算除两端元素以外（即矩阵四角）的最左一列和最右一列的元素的和，最后 sun 就是周边元素的和。*/
int fun(int a[M][N])
{   int sum=0,i;
    for(i=0;i<N;i++)
        sum+=a[0][i]+a[M-1][i];
    for(i=1;i<M-1;i++);
        sum+=a[i][0]+a[i][N-1];
    return sum ;
}
```

### 17.2.4　题型 4：数组运算

**1．找最大（小）数问题**

**【例 1】函数 fun()的功能是：求出一个 2×$M$ 整型二维数组中最大元素的值，并将此值返回调用函数。**

```
#include <stdio.h>
#define M 4
fun (int a[][M])
{
}
main( )
{  int arr[2][M]={5,8,3,45,76,-4,12,82} ;
   printf("max =%d\n", fun(arr)) ;
   NONO( ) ;
}
NONO ( )
{   /* 本函数用于打开文件，输入数据，调用函数，输出数据，关闭文件。 */
    /* 本函数省略*/
```

```
}
```

**题眼分析　程序的实现思路是：(1) 首先定义一个临时变量 max，并把数组第 1 个元素的值赋给 max 作为比较初值；(2) 通过一个双重循环，遍历整个数组每个元素，在循环中不断修改 max 值，使它为当前最大的值；(3) 循环结束时，max 就是该二维数组的最大值，把它作为返回值返回。**

**答案**

```
fun (int a[][M])
{   int i,j;
    int max=a[0][0];
    for(i=0;i<2;i++)
        for(j=0;j<M;j++)
            if(max<a[i][j])  max=a[i][j];
    return max;
}
```

**补充说明　求最大值（或最小值）时，首先要定义一个临时变量(如 temp)，一般把数组第 1 个元素的值赋给 temp 作为比较初值，并在循环中改变 temp 的值，使 temp 是当前最大（或最小）的值。循环结束时，temp 就是该数组的最大值（或最小值）。**

**【例 2】学生的记录由学号和成绩组成，*N* 个学生的数据已在主函数中放入结构体数组 *s* 中，请编写函数 fun()，它的功能是：把学生最高的的学生成绩放在 *h* 指定的数组中。（注意：分数最高的学生可能不只一个，函数返回分数最高学生的人数。）**

```
#include <stdio.h>
#define N 16
typedef struct
{   char num[10];
    int s;
} STREC;
int fun ( STREC *a, STREC *b )
{
}
main ()
{   STREC s[N]= {{"GA05",85}, {"GA03",76}, {"GA02",69}, {"GA04",85},
                 {"GA01",91}, {"GA07",72}, {"GA08",64}, {"GA06", 87},
                 {"GA015",85}, {"GA013",91}, {"GA012",64}, {"GA014",91},
                 {"GA011",77}, {"GA017",64}, {"GA018",64}, {"GA016",72}
                };
    STREC h[N];
    int i, n; FILE *out;
    n=fun ( s, h );
    printf ("The %d highest score :\n", n);
    for (i=0; i<n; i++)
        printf ("%s %4d\n", h[i]. num, h[i]. s);
    printf ("\n");
    out=fopen ("out15.dat", "w");
    fprintf (out, "%d\n", n);
```

```
    for (i=0; i<n; i++)
        fprintf (out, "%s %4d\n", h[i]. num, h[i]. s);
    fclose (out );
}
```

**题眼分析　这也是一个找数组最大数（最小数）问题，解决本问题的关键是结构体的引用问题和结构体的复制。程序的实现思路是：(1) 通过一个循环，在数组中找到最大值 max；(2) 再通过一个循环，遍历整个数组每个元素，找成绩为 max 的学生，并将这些学生名字复制到数组 *b* 中。**

**答案**

```
int fun ( STREC *a, STREC *b )
{   int i;
    int j=0,max=a[0].s;
    for(i=0;i<N;i++)                /*找最高的学生成绩*/
        if(a[i].s>max)  max=a[i].s;
    for(i=0;i<N;i++)                /*将具有最高的成绩学生记录放到数组 b 中*/
        if(a[i].s==max) b[j++]=a[i];
    return j;                       /*j 存放的是学生数，作为函数返回*/
}
```

**2．累加和、求平均值累积**

**【例 3】下列给定程序中，函数 fun()的功能是：按顺序给 *s* 所指数组中的元素赋予从 2 开始的偶数，然后再按顺序对每 5 个元素求一个平均值，并将这些值依次存放在 *w* 所指的数组中。若 *s* 所指数组中元素的个数不是 5 的倍数，多余部分忽略不计。例如，*s* 所指数组有 14 个元素，则只对前 10 个元素进行处理，不对最后的 4 个元素求平均值。**

```
#include  <stdio.h>
#define SIZE 20
fun (double *s,double *w)
{   int k,i; double sum;
    for(k=2,i=0;i<SIZE;i++)
    {   s[i]=k;k+=2;}
        sum=0.0;
        for(k=0,i=0;i<SIZE;i++)
        {   sum+=s[i];
            if((i+1)__(1)__5==0)
            {   w[k]=sum/5; sum=0;k++; }
        }
          __(2)__ k;
}
main()
{   double a[SIZE],b[SIZE/5];
    int i, k;
    k=fun(a,b);
    printf("The original data:\n");
    for(i=0;i<SIZE;i++)
    {
```

```
        if(i%5==0) printf("\n");
        printf("%4.0f",a[i]);
    }
    printf("\n\nThe result:\n");
    for(i=0;i<k;i++) printf("%6.2f", (3) );
    printf("\n\n");
}
```

**题眼分析　这是一个求和问题，通过程序很清楚程序的实现思路，这里不再赘述。在空（1）处，根据题目的意思，这里是执行按顺序对每 5 个元素求一个平均值的操作，所以应该使用取余符号“%”，如果是 5 的倍数，则该式子的值为零。空（2）处，根据 C 语言的规定，除了使用关键字 void 的任何一个子函数都应该有返回值，所以应该使用关键字 return 把变量 *k* 的值返回主函数。空（3）处，由题目的意思可知这里是把存在数组 b 中的内容依次循环输出，所以后面的变量名应该使用 b[i]。**

**答案　（1）%　　（2）return　　（3）b[i]**

**补充说明　累加、求平均值和累积是一类问题，这类问题比较简单。做这类题目时需要注意以下几点：（1）定义用于保存累加或累积的变量的类型；（2）必须先给保存累加和累积的变量赋初值，累加时赋 0，累积时赋 1；（3）累加或累积时，注意保存累加和累积的结果。**

### 17.2.5　题型 5：排序

**关于排序的算法，在《新编二级公共基础知识题眼分析与全真训练》的第 1 章中已经介绍了多种排序算法，限于篇幅，这里就不再重复了。这里强调两点：（1）注意排序方法的选择，有的题目指定必须使用的排序方法，有的没有指定。对于没有指定排序方法，一般使用冒泡法或直接插入法；（2）要注意是升序排序还降序排序。**

**【例】请编写函数 fun，对长度为 7 个字符的字符串，除首、尾字符外，将其余 5 个字符按降序排列。例如，原来的字符串为 CEAedca，排序后输出为 CedcEAa。**

```
#include <string.h>
#include <conio.h>
#include <stdio.h>
int fun(char *s,int num)
{
}
main()
{    char s[10];
    clrscr();
    printf("输入 7 个字符的字符串:");
    gets(s);
    fun(s,7);
    printf("\n%s",s);
}
```

**题眼分析　由于题目并没有指定排序方法，可采用的冒泡排序法进行降序排序。算法实现思路是：用外 for()循环从字符串的前端往后端走动，每走动一个字符都用内嵌的 for()循环在**

该字符后找出最小的字符与该字符进行换位。直到外 for()循环走到最后一个字符。此外，该题还要注意把首尾字符除去，即在最外层 for()循环中从 1 开始，到 num–2 结束。

**答案**

```
int fun(char *s,int num)
{   int i,j,t;
    for(i=1;i<num-2;i++)
        for(j=i+1;j<num-1;j++)
            if(s[i]<s[j])
            {   t=s[i];
                s[i]=s[j];
                s[j]=t;
            }
}
```

### 17.2.6　题型 6：字符串运算

字符串运算是二级 C 语言常考的类型，主要包括：字符 ASCII 码值应用（字符排序、比较字符串大小、大小写转换、删除指定 ASCII 码的字符等）、字符串常用库函数的实现（如求字符串长度、字符串的比较、字符串连接、字符串复制等）、串匹配、字符串处理（字符查找及删除指定字符、字符串逆置、回文数、数字字符串转换成长整型数等）。

1．字符 ASCII 码值应用

【例 1】函数 fun 的功能是进行字母转换。若形参 *ch* 中是小写英文字母，则转换成对应的大写英文字母；若 *ch* 中是大写英文字母，则转换成对应的小写英文字母；若是其他字符则保持不变；并将转换后的结果作为函数返回。

```
#include   <stdio.h>
#include   <ctype.h>
char fun(char  ch)
{
/**********found**********/
  if ((ch>='a')__(1)___(ch<='z'))
    return  ch -'a' + 'A';
  if ( isupper(ch) )
/**********found**********/
    return  ch +'a'-__(2)___ ;
/**********found**********/
  return __(3)___;
}
main()
{  char  c1, c2;
  printf("\nThe result  :\n");
  c1='w';   c2 = fun(c1);
  printf("c1=%c    c2=%c\n", c1, c2);
  c1='W';   c2 = fun(c1);
  printf("c1=%c    c2=%c\n", c1, c2);
```

```
    c1='8';   c2 = fun(c1);
    printf("c1=%c    c2=%c\n", c1, c2);
}
```

**题眼分析　这一题是一个大小写转换的题，比较简单。空（1）处是要填入判断是小写字母的表达式，显然应填入&&。空（2）处是将大写字母转换成小写字母的表达式，应填入'A'。空（3）处要填入当字符不是大写或小写字母时函数返回值，根据题意，应返回字符本身，即** ch。

**答案**　（1）&&　　（2）'A'　　（3）ch

**补充说明：在做大小写字母转换一类问题时，需要掌握以下内容。**

**（1）判断一个字母 *ch* 是大写还是小写字母：判断方法有两种，一种是用库函数，但必须在程序中包含** ctype.h**。判断大写字母的库函数是** isupper(ch)**，判断小写字母的库函数是** islower(ch)**。另一种是条件表达式，判断大写字母的条件表达式是**(ch>='A')&&(ch<='Z')**，判断小写字母的条件表达式是**(ch>='a')&&(ch<='z')**。**

**（2）大小写转换规则：若 *ch* 是一个大写字母，转换成小写字母的表达式是 *ch*+32 或** ch+'a'–'A'**。若 *ch* 是一个小写字母，转换成大写字母的表达式是 *ch*–32 或 *ch*–'a'+'A'。**

**2．字符串常用库函数的实现**

**【例 2】编写一个函数** fun**，它的功能是实现两个字符串的连接（不使用库函数** strcat**），即把** p2 **所指的字符串连接到** p1 **所指的字符串后。**

**例如，分别输入下面两个字符串：**

FirstString--
SecondString

**程序输出：**

FirstString--SecondString

```
#include <stdio.h>
void fun(char p1[], char p2[])
{
}
main()
{    char s1[80], s2[40] ;
     printf("Enter s1 and s2:\n") ;
     scanf("%s%s", s1, s2) ;
     printf("s1=%s\n", s1) ;
     printf("s2=%s\n", s2) ;
     printf("Invoke fun(s1,s2):\n") ;
     fun(s1, s2) ;
     printf("After invoking:\n") ;
     printf("%s\n", s1) ;
     NONO() ;
}
NONO ( )
{   /* 本函数用于打开文件，输入测试数据，调用 fun 函数，输出数据，关闭文件。*/
    /*本函数省略*/
```

```
}
```

**题眼分析　本题的关键是把指针定位到第 1 个字符串的末尾，然后再把第 2 个字符串的字符依次拷贝到第 1 个字符串的末尾。本题实现的思路是：(1)用指针遍历第 1 个字符串，把指针定位到串尾标志符'\0'处；(2) 遍历第 2 个字符串，依次把字符拷贝到第 1 个字符串的末尾；(3) 为第 1 个字符串赋结尾标志字符'\0'。**

```
void fun(char p1[], char p2[])
{   int i=0,n=0;          /*i 是遍历第 2 个字符串的循环变量，n 是第 1 个字符串的长度*/
    char *p=p1,*q=p2;     /*定义两个字符指针，分别指向两个字符串的首地址*/
    while (*p)            /*遍历第 1 个字符串，使指针 p 指向串尾，同时计算其他长度*/
    {   p++;
        n++;
    }
    i=n;                  /*i 为第 1 个字符串的结尾标志符'\0'的位置*/
    while(*q)             /*遍历第 2 个字符串，依次把字符拷贝到第 1 个字符串的末尾*/
    {   p1[i]=*q;
        q++;
        i++
    }
    p[i]='\0';            /*为第 1 个字符串赋结尾标志字符'\0'*/
}
```

### 3．字符串匹配

**【例 3】给定程序 MODI1.C 中函数 fun 的功能是：统计 substr 所指子字符串（字串）在 str 所指字符串中出现的次数。**

**例如，字符串为 aaas lkaaas，子字符串为 as，则应输出 2。**

```
#include <stdio.h>
/************found************/
fun (char *str,char *substr)
{  int i,j,k,num=0;
/************found************/
   for(i = 0, str[i], i++)
     for(j=i,k=0;substr[k]==str[j];k++,j++)
       if(substr[k+1]=='\0')
       {  num++;
          break;
       }
   return num;
}
main()
{
  char str[80],substr[80];
  printf("Input a string:") ;
  gets(str);
  printf("Input a substring:") ;
  gets(substr);
```

```
    printf("%d\n",fun(str,substr));
}
```

**题眼分析　这是一个字符串匹配的问题，**fun **函数的思路是：（**1**）通过外循环遍历母字符串（母串）**str**；（**2**）内循环用于遍历子串** subst**，并与母串的字符比较；（**3**）如果相等则继续比较，如如果遍历完子串的所有字符，计数** num **加** 1**，并跳出内循环；（**4**）母串遍历完成后返回总个数** num**。**

**在本题中，错误有两个。（**1**）根据** fun **函数返回子串在母串中出现的次数，应该返回一个** int **型值，因此** fun（char *str,char *substr）**应改为** int fun（char *str,char *substr）**；（**2**）**for **循环语句括号中是三条语句而不是表达式，应以“；”结尾，所以** for(i = 0, str[i], i++)**应改为“** for(i = 0; str[i]; i++) **”。**

**答案　（**1**）**fun（char *str,char *substr）**应改为** int fun（char *str,char *substr）

（**2**）**for(i = 0, str[i], i++)**应改为“** for(i = 0; str[i]; i++) **”**

**补充说明：对于字符串匹配的问题，实现思路是：（**1**）使用一个外循环遍历母字符串** str **中的所有字符；（**2**）使用内循环遍历子串** substr **中的所有字符，如果所有字符都相等，则计数器加** 1**。**

**解答这类问题的关键在于：（**1**）使指针依次向后移动指向下一个特比较的字符。实现方法是：当没有相同字符时候，指向母字符串的时候继续后移指向下一个字符，而指向子字符串的指针要回到第** 1 **个字符与下一个母串字符比较；当字符相等的时候，母串和子串都指向下一个字符继续比较，如果子串没有结束就遇到不等字符，则此次比较失败，母串指向下一个字符，而子路重新指向串首等待下一轮比较。（**2**）使用字符结束标志** '\0' **来判断一个字串是否比较完成。**

4．字符串处理

**【例** 4**】请编写函数** fun**，函数的功能是：在字符串中所有数字字符前加一个**$**字符。**

**例如，输入：**A1B23CD45**，则输出为：**A$1B$2$3CD$4$5**。**

```
#include <stdio.h>
void fun( char *s)
{
}
main()
{   char s[80];
    printf("enter a string:");
    scanf("%s", s);
    fun(s);
    printf("the result: %s\n", s);
}
```

**题眼分析　程序的实现思路是：（**1**）用一个** while()**循环来控制原字符串从头走到尾的遍历；（**2**）在遍历过程中判断当前字符是否是数字，若是则在新串中先连一个**'$'**然后再连原字符，否则直接连原字符，这里需要注意指针和下标的变化；（**3**）把新串拷贝到** s **所指的地址中，注意不能用** s=a **；若用了，则实参数组还是原字符串。**

**答案**

```
void fun( char *s)
```

```
{   char a[100];
    int i=0;
    while(*s)
        if(*s>='0'&&*s<='9')
        {   a[i++]='$';a[i++]=*s++;}
        else    a[i++]=*s++;
    a='\0';
    strcpy(s,a);
}
```

### 17.2.7　题型 7：链表处理

**链表处理有以下几类问题：链表的遍历（包括求和、求平均数、打印输出等）、链表的建立和结点的插入、链表结点的删除等问题。**

**1．链表遍历**

**【例 1】N 名学生的成绩已在主函数中放入一个带头结点的链表结构中，h 指向链表的头结点。请编写函数 fun，它的功能是：求出平均分，由函数值返回。**

**例如：若学生的成绩是 85，76，69，85，91，64，87；则平均分应当是 78.625。**

```
#include <stdio.h>
#include <stdlib.h>
#define   N   8
struct  slist
{  double   s;
   struct slist  *next;
};
typedef  struct slist  STREC;
double  fun( STREC *h  )
{
}
STREC * creat( double *s)
{ STREC  *h,*p,*q;   int  i=0;
  h=p=(STREC*)malloc(sizeof(STREC));p->s=0;
  while(i<N)
  { q=(STREC*)malloc(sizeof(STREC));
    q->s=s[i]; i++;  p->next=q; p=q;
  }
  p->next=0;
  return  h;
}
outlist( STREC *h)
{ STREC  *p;
  p=h->next; printf("head");
  do
  { printf("->%4.1f",p->s);p=p->next;}
  while(p!=0);
```

```
  printf("\n\n");
}
main()
{  double  s[N]={85,76,69,85,91,72,64,87},ave;
   STREC  *h;
   h=creat( s );   outlist(h);
   ave=fun( h );
   printf("ave= %6.3f\n",ave);
   NONO();
}
NONO()
{/* 本函数用于打开文件，输入数据，调用函数，输出数据，关闭文件。 */
 /*省略*/
}
```

**题眼分析　这是一个链表遍历的问题，程序实现思路是：（1）定义求和存放的临时变量，并赋初值；（2）定义用于循环的指针变量 p，并使该指针指向头结点的下一个结点 *p*->next，因为头结点没有数据；（3）累加成绩后并使指针指向下一个结，当指针 *p* 为空时，退出循环；（4）返回平均分。**

**答案**

```
double fun ( STREC *h )
{   double  ave=0.0;
    STREC *q;
    q=h->next;
    while(q)
    {   sum+=q->s;
        q=q->next;
    }
    return ave/N;
}
```

**补充说明：链表遍历的问题就是要从链头至链尾访问链表的每一个结点，以实现对链表的输出、查找、统计等工作。在做这类问题时需要注意以下几点：（1）用于循环的指针变量指向下一个结点的方法；（2）链表结束的判断；（3）链表是否具有头结点的处理。**

**2．链表的建立和结点的插入**

**【例 2】请完成函数** struct node *fun **()，它的功能是：建立一个带头结点的单向链表，新产生的结点总是插入在链表的末尾。单向链表的头指针作为函数值的返回。**

```
struct node{
            char data;
            struct node *next;
};
struct node *fun()
{   struct node *t1,*t2,*t3;
    char ch;
    t1= (struct node *)malloc(sizeof(struct node));
    t3=t2=t1;
```

```
    ch=getchar();
    while(ch!='\n')
    {   t2=__(1)__malloc(sizeof(struct node)};
        t2->dat=ch;
        t3->next= __(2)__;
        t3=t2;
        ch=getchar();
    }
    p->next='\0';
    __(3)__;
}
```

**题眼分析　这是一个建立链表问题，程序实现思路是：（1）首先定义 3 个指针 *t1*，*t2*，*t3*，它们的作用是分别指向链表的头结点、待插入结点（即链尾）；（2）生成头结点，使 *t1*，*t2*，*t3* 都指向该结点；（3）读入数据（字符）进入循环；（4）在循环中，首先生成一个结点，为该结点分配空间，并用 *t2* 指向该结点，接着将该结点插入到 *t3* 所指结点之后，最后修改 *t3*，使其再次指向下一次插入位置；（4）再次读入数据，若为回车字符，退出循环，否则继续插入结点；（5）链表建成以后，返回头指针。**

**综上所述，第 1 个空处是生成结点操作，在动态申请函数时要进行强制类型转换，使它的返回值有意义，因此应填写“**(struct node *)**”；第 2 个空处是结点的插入，使 *t3* 的下一结点为 *t2*，因此应填写“***t2***”；第 3 个空处是函数返回值，题目的要求是返回链表的表头指针，因此应填写“***t1***”。**

**答案**　（1）(struct node *)　　（2）t2　　（3）return(t1)

**补充说明：链表的建立和结点的插入是链表最常见的运算。在插入结点时需要注意以下几点：（1）插入结点时需要定义两个指针，一个指针 q 用来指向将要待插入结点，另一个指针 p 指向插入位置的前一结点；（2）插入时需要修改指针 p 和 q 的指针域，需要注意修改顺序，正确的顺序是“**q->next=p->next;p->next=q;**”，这个顺序不能颠倒。具体插入过程，可参阅第 14 章的图 14-2 所示链表结点插入过程。**

**3．链表结点删除**

**【例 3】给定程序中已建立一个带有头结点的单向链表，链表中的各结点按数据域递增有序链接。函数 fun 的功能是：删除链表中数据域值相同的结点，使之保留一个。**

```
#include   <stdio.h>
#include   <stdlib.h>
#define   N   8
typedef  struct list
{  int  data;
   struct list  *next;
} SLIST;
void  fun( SLIST *h)
{  SLIST  *p, *q;
   p=h->next;
   if (p!=NULL)
   {  q=p->next;
```

```
      while(q!=NULL)
      {  if (p->data==q->data)
         {  p->next=q->next;
/**********found**********/
           free(  (1)  );
/**********found**********/
           q=p->  (2)  ;
         }
         else
         { p=q;
/**********found**********/
           q=q->  (3)  ;
         }
      }
   }
}
SLIST *creatlist(int  *a)
{  SLIST  *h,*p,*q;     int  i;
   h=p=(SLIST *)malloc(sizeof(SLIST));
   for(i=0; i<N; i++)
   {  q=(SLIST *)malloc(sizeof(SLIST));
      q->data=a[i];  p->next=q;  p=q;
   }
   p->next=0;
   return h;
}
void outlist(SLIST  *h)
{  SLIST  *p;
   p=h->next;
   if (p==NULL)  printf("\nThe list is NULL!\n");
   else
   {  printf("\nHead");
      do { printf("->%d",p->data);  p=p->next;   } while(p!=NULL);
      printf("->End\n");
  }
}
main( )
{  SLIST  *head;     int  a[N]={1,2,2,3,4,4,4,5};
   head=creatlist(a);
   printf("\nThe list before deleting :\n");  outlist(head);
   fun(head);
   printf("\nThe list after deleting :\n");  outlist(head);
}
```

**题眼分析　这是一个删除链表结点问题，程序实现思路是：（1）定义两个指针变量 $p$，$q$，分别用来指向被删除结点的前一结点和被删除结点的本身；（2）初始时，指针 $p$ 指向头结点**

的后一个结点，即第 1 个数据结点，若该结点非空，使指针 *q* 指向第 2 结点；（3）当指针 q 的所指结点非空时，进入循环；（4）在循环中，首先判断 p 和 q 两个指针所指的结点数据域是否相同；（5）若相等，删除 *p* 所指结点，删除方法是：先将指针 *p* 所指结点下一个结点为指针 *q* 所指结点下一个结点（即 *p*->next=*q*->next），使 *q* 指向结点从链表脱离，接着释放该结点。删除后，使指针 *p* 再次指向 *p* 所指结点的下一结点，进入下一轮循环；（6）若不相等，指针 *p* 和 *q* 都向后移动，进入下一轮循环。综上所述，第 1 个空是释放是结点，应释放 *q* 所指的结点，应填入“ *q* ”；第 2 个空是使指针 *p* 再次指向 *p* 所指结点的下一结点，应填入“ next ”；第 3 个空是使指针 *q* 向后移动，应填入“ next ”。

**答案**　（1）q　　（2）next　　（3）next

**补充说明：**同样，删除链表结点也是链表最常见的运算之一。删除链表结点时需要注意以下几点：（1）删除结点时需要定义两个指针，一个指针 *q* 用来指向将要被删除的结点，另一个指针 *p* 指向将要被删除结点的前一结点。（2）修改链表结点指针域和释放结点空间的顺序。正确的顺序是先要修改被删除结点的前一结点（p 所指的结点）的指针域，使其指向将要被删除结点的下一个结点（即 *q*->nex 所指的结点），这个顺序不能颠倒。（3）要释放结点，并注意释放结点的方法。考生可参考第 14 章的图 14-3 所示链表结点删除过程。

### 17.2.8　题型 8：其他

除上面介绍的几种题型外，在机考题还有可能出现文件操作、二分法查找等问题。

1．文件操作

【例 1】给定程序的功能是调用 fun 函数建立班级通讯录。通讯录中记录每位学生的编号、姓名和电话号码。班级人数和学生的信息从键盘读入，每个人的信息作为一个数据块写到名为 myfile5.dat 的二进制文件中。

```
#include   <stdio.h>
#include   <stdlib.h>
#define   N   5
typedef  struct
{  int  num;
   char  name[10];
   char  tel[10];
}STYPE;
void check();
/**********found**********/
int fun(   (1)    *std)
{
/**********found**********/
    (2)     *fp;
   int  i;
   if((fp=fopen("myfile5.dat","wb"))==NULL)
     return(0);
   printf("\nOutput data to file !\n");
   for(i=0; i<N; i++)
```

```
/**********found**********/
     fwrite(&std[i], sizeof(STYPE), 1, __(3)__);
  fclose(fp);
  return (1);
}
main()
{  STYPE  s[10]={ {1,"aaaaa","111111"},{1,"bbbbb","222222"},
                  {1,"ccccc","333333"},{1,"ddddd","444444"},
                  {1,"eeeee","555555"}};
  int  k;
  k=fun(s);
  if (k==1)
  {  printf("Succeed!");  check();  }
  else
     printf("Fail!");
}
void check()
{  FILE  *fp;    int  i;
  STYPE  s[10];
  if((fp=fopen("myfile5.dat","rb"))==NULL)
  { printf("Fail !!\n"); exit(0); }
  printf("\nRead file and output to screen :\n");
  printf("\n   num    name     tel\n");
  for(i=0; i<N; i++)
  {  fread(&s[i],sizeof(STYPE),1, fp);
     printf("%6d    %s    %s\n",s[i].num,s[i].name,s[i].tel);
  }
  fclose(fp);
}
```

**题眼分析　这是一个文件操作问题，函数 fun 实现思路是：（1）通过参数传递得到一个结构体数组首地址；（2）定义文件指针，并以二进制写方式打开文件；（3）通过循环，将结构体数组中元素依次写入文件；（4）文件后关闭文件，并返回 1。**

**空（1）处，需要填写函数的形参的类型，从 main 函数的调用可以看出，形参是一个用户自定义结构体的指针，因此应填写“ STYPE ”；空（2）处是一个文件指针的定义语句，因此应填写“ FILE ”；空（3）处是一个块写函数 fwrite 的调用语句，是将一个结构体数组元素写入文件，根据函数 fwrite 的调用方式，此处应填写“ *fp* ”。**

**答案**　（1）STYPE　　（2）FILE　　（3）*fp*

**2．方程求解**

**【例 2】编写函数 fun，它的功能是：利用以下所示的简单叠代方法求方程：**$\cos(x)-x=0$ **的一个实根。**

$$x_{n+1}=\cos(x_n)$$

**叠代步骤如下：**

**（1）取** $x1$ **初值为 0.0；**

（2）$x0 = x1$，把 $x1$ 的值赋给 $x0$；

（3）$x1 = \cos(x0)$，求出一个新的 $x1$；

（4）若 $x0 - x1$ 的绝对值小于 0.000001，执行步骤（5），否则执行步骤（2）；

（5）所求 $x1$ 就是方程 $\cos(x) - x = 0$ 的一个实根，作为函数值返回。

程序将输出结果 Root=0.739085。

```
#include <conio.h>
#include <math.h>
#include<stdio.h>
float fun()
{
}
main()
{   clrscr()
    printf("Root=%f\n",fun());
}
```

**题眼分析**　这是一个通过叠代法求方程解的问题。函数 fun 实现思路在说明中已经说得很清楚了，只需把算法用合适语句表达出来就可以完成求解，这里就不再赘述。做这类问题最好使用 do-while 循环，需要注意的是循环中止条件。

**答案**

```
float fun()
{   double x1=0.0;
    do
    {   x0=x1; x1=cos(x0); }
    while(fabs(x0-x1)>=0.000001);
    return x;
}
```

# 第 18 章　笔试模拟试题及参考答案

## 18.1　笔试模拟试题

### 18.1.1　笔试模拟试题（一）

**一、选择题[（1）～（10），每小题 2 分，（11）～（50）每题 1 分，共 60 分]**

1．算法的空间复杂度是指_____。

（A）算法程序的长度

（B）算法程序的指令条数

（C）算法程序所占存储空间

（D）算法程序在执行过程中所需要的存储空间

2．下面关于线性表的叙述中，错误的为_____。

（A）顺序表使用一维数组实现的线性表

（B）在链表中，每个结点只有一个链域

（C）顺序表的空间利用率高于链表

（D）顺序表必须占用一片连续的存储单元

3．下述说法不正确的是_____。

（A）栈是一种运算受限的线性结构　　（B）栈是一种后进先出的线性结构

（C）栈可以是线性结构也可以是非线性结构　（D）栈可以用数组或链表来实现

4．深度为 5 的二叉树至多有_____个结点。

（A）16　　（B）32　　（C）31　　（D）10

5．设待排序关键码序列为（25，18，9，33，67，82，53，95，12，70），要按关键码值递增的顺序排序，采取以第 1 个关键码为分界元素的快速排序法，第 1 趟排序完成后关键码 33 被放到了第_____位置。

（A）3　　（B）5　　（C）7　　（D）9

6．结构化程序设计主要强调的是_____。

（A）程序的规模　　（B）程序的效率

（C）程序设计语言的先进性　　（D）程序易读性

7．黑盒测试也称为功能测试。黑盒测试不能发现_____。

（A）终止性错误　　（B）输入是否正确接收

（C）界面是否有误　　（D）是否存在冗余代码

8．在数据流图中，——（双杠）表示_____。

（A）存储　　（B）外部实体　　（C）数据流　　（D）加工

9．数据的逻辑独立性是指_____。

（A）存储结构与物理结构的逻辑独立性　　（B）数据与存储结构的逻辑独立性

（C）数据与程序的逻辑独立性　　（D）数据元素之间的逻辑独立性

10．在下列实体类型的联系中，一对多联系的是_____。

（A）学校与课程的学习联系　　（B）父亲与孩子的父子关系

（C）省与省会的关系　　（D）顾客与商品的购买关系

11．以下叙述中正确的是_____。

（A）C 语言的源程序不必通过编译就可以直接运行

（B）C 语言中的每条可执行语句最终都将被转换成二进制的机器指令

（C）C 源程序经编译形成的二进制代码可以直接运行

（D）C 语言中的函数不可以单独进行编译

12．下列符号串中，属于 C 语言合法标识符的是_____。

（A）ex-1　　（B）for　　（C）_cook　　（D）951_

13．当 $a$=1,$b$=3,$c$=5,$d$=6 时，执行下面一段程序后，$x$ 的值为_____。

```
if(a<b)
  if(c>d)  x=1;
  else
    if(a<c)
      if(b>d)  x=2;
      else x=3;
    else x=3;
    else x=6;
```

（A）1　　（B）2　　（C）3　　（D）6

14．若有定义：int　a=67；　char　b='A'；则表达式“a < b”的结果为_____。

（A）0　　（B）1

（C）任何一个非零的整数　　（D）两个变量不能比较

15．假定 $x$,$y$ 为 int 类型，则执行以下程序段后 $x$ 的值为_____。

```
x=1;  y=10;
while (x<6 )
{  y-=x;
   if (y<x)  break;    x++;
}
```

（A）3　　（B）4　　（C）5　　（D）6

16．有如下程序

```
main()
{   int y=3,x=3,z=1;
    printf("%d %d\n",(++x,y++),z+2);
}
```

运行该程序的输出结果是_____。

（A）3 4　　（B）4 2　　（C）4 3　　（D）3 3

17．设 $p$ 为 int 型变量,则下面 for 循环语句的执行结果是_____。

```
for(p=1;p<=10;p++)
  {  if (p%3)  p++;
     ++p;  printf("%d",p);
  }
```

（A）35811　　（B）36912　　（C）2468　　（D）258

18．设有下列定义 char s[ ]={ "12345"}, *p=s, 则下列表达式中不正确的是_____。

（A）p+1　　（B）*(s+2)　　（C）p="ABC"　　（D）*s="ABC"

19．设有下列定义

```
static  int  x, *p=&x, *q ;
q=p; scanf ("%d,%d",p,q);
```

若输入 3,4 则 x 的值为_____。

（A）3　　（B）4　　（C）0　　（D）无法确定

20．以下叙述中不正确的是_____。

（A）在不同的函数中可以使用相同名字的变量

（B）函数中的形式参数是局部变量

（C）在一个函数内定义的变量只在本函数范围内有效

（D）在一个函数内的复合语句中定义的变量在本函数范围内有效

21．有以下程序

```
main()
{   char str[]="xyz",*ps=str;
    while(*ps)ps++;
    for(ps--;ps-str>=0;ps--)puts(ps);
}
```

执行后输出结果是_____。

（A）
```
yz
xyz
```

（B）
```
 z
yz
```

（C）
```
 z
yz
xyz
```

（D）
```
 x
xy
xyz
```

22．以下程序的输出结果是_____。

```
main()
{   int  i,x,a[10],b[3];
    x=5;
    for (i=0;i<10;i++)  a[i]=i;
    for (i=0;i<3;i++)  b[i]=a[i*(i+1)];
    for (i=0;i<3;i++)  x=b[i]*2;
    printf("%d\n",x);
}
```

（A）12　　（B）21　　（C）22　　（D）23

23．有以下程序

```
main()
{   int v[]={1,3,5,7,2,4,6,8};
    int i, *p;  p=v;
    for(i=0;i<8;i++) if(*(p+i)==i+1)  printf("%d",*(p+i));
```

```
}
```

输出结果是_____。

（A）1　　（B）18　　（C）35　　（D）16

24．已定义以下函数

```
fun(char *p2,char *p1)
{   while((*p2=*p1)!= '\0') {p1++;p2++}}
```

函数的功能是_____。

（A）将 *p*1 所指字符串复制到 *p*2 所指内存空间

（B）将 *p*1 所指字符串的地址赋给指针 *p*2

（C）对 *p*1 和 *p*2 两个指针所指字符串进行比较

（D）检查 *p*1 和 *p*2 两个指针所指字符串中是否有'\0'

25．下列程序执行后的输出结果是_____。

（A）6　　（B）8　　（C）10　　（D）12

```
#define Max(x)  x*(x-1)
main()
{   int a=1,b=2;  printf("%d \n",Max(1+a+b));}
```

26．以下程序运行后，输出结果是_____。

```
main( )
{   char  *p="abcde";
    p+=3;   printf ("%c ",p);
}
```

（A）de　　（B）d　　（C）字符 d 的地址　　（D）出错

27．若有以下定义：

char　s[10]={ 'a', 'b', 'c', '0', '\0', '2', '\0x32', '\0'}; 执行语句 printf("%d",strlen(s));的结果是_____。

（A）3　　（B）4　　（C）8　　（D）10

28．以下数据类型中不是构造类型的是_____。

（A）数组型　　（B）指针型　　（C）结构型　　（D）共用型

29．如果函数定义时，形式参数是实型变量，则调用该函数时，实际参数不可以是_____。

（A）实型常量　　（B）字符型变量

（C）实型表达式　　（D）指向实型变量的指针变量前面加“*”

30．设有以下宏定义：

```
#define N 4
#define Y(n)  ((N+1)*n)
```

则执行语句：*z*=2 *(N+Y(5+1));后，*z* 的值为_____。

（A）出错　　（B）60　　（C）48　　（D）54

31．若 int i=10;执行下列程序后，变量 *i* 的正确结果是_____。

```
switch(i)
{  case  8 : i+=1;
   case  10 : i+=1;
   case  1 : i+=1;
```

```
    default  : i+=1;
  }
```

（A）10　　（B）11　　（C）12　　（D）13

32．定义如下变量和数组:

```
int i;
int x[3][3]={1,2,3,4,5,6,7,8,9};
```

则下面语句的输出结果是_____。

```
for(i=0;i<3;i++) printf("%d",x[i][2-1]);
```

（A）1 5 9　　（B）1 4 7　　（C）2 5 8　　（D）3 6 9

33．设有以下定义:

```
int a[4][3]={1,2,3,4,5,6,7,8,9,10,11,12};
int (*prt)[3]=a,*p=a[0]
```

则下列能够正确表示数组元素 a[1][2]的表达式是_____。

（A）*((*prt+1)[2])　　（B）*(*(p+5))　　（C）(*prt+1)+2　　（D）*(*(a+1)+2)

34．以下函数调用语句中含有_____参数。

```
excc((v1,v2),(v2,v3,v4),v6)
```

（A）3　　（B）4　　（C）5　　（D）6

35．若有以下说明，则_____不是对 strcpy 库函数的正确的调用。

```
char  *str1="copy",str2[10],*str3="hijkl",*str4,*str5="abcd"
```

（A）strcpy(str2,str1);　　（B）strcpy(str3,str1);

（C）strcpy(str4,str1);　　（D）strcpy(str5,str1);

36．以下程序的运行结果是_____。

```
# include < stdio.h>
main( )
{  int a = 1, b = 2, c;
   c = max (a, b);
   printf ("max is % d\n", c);
}
max ( int x, int y)
{  int z;
   z = ( x>y)? x: y;
   return (z);
}
```

（A）2　　（B）MAXIS2　　（C）max is 2　　（D）maxis2

37．下列程序运行结果是_____。

```
swap ( int * ptl, int *pt2)
{   int  p;
    p=*ptl; *ptl = * pt2;  * pt2=p;
 }
 main()
 {  int a = 5, b = 7, * p1, *p2;
    p1 = &a; p2 = &b;
    swap (p1, p2);
```

```
    printf ("* p1=%d, * p2 = %d\n", * p1, *p2);
    printf (" a=%d, b=%d\n", a, b);
}
```

（A）*p1=7,*p2=5,
a=5,b=7

（B）*p1=7,*p2=5
a=7,b=5

（C）*p1=7*p2=5
a=7b=5

（D）*p1=5,*p2=7,
a=7,b=5

38．**下面程序运行结果是_____。**

```
main ( )
{  int a[6], i;
   for(i=1;i<6;i++)
   {  a[i]=9*(i-2+4*(i>3)%5);
      printf("%2d", a[i]);
   }
}
```

（A）–9 0 9 5 4 3 6　（B）–18 –9 0 9 54　（C）–9 0 95463　（D）–9095463

39．**下面程序运行结果是_____。**

```
main ( )
{  int a, b, c, d, x;
   a=c=0;
   b=1;  d=20;
   if (a)  d=d-10;
   else   if ( ! b )
          if ( ! c ) x = 15;
          else x = 25;
   printf ("%d\n", d);
}
```

（A）20　（B）25　（C）15　（D）10

40．**下面程序运行结果是_____。**

```
# include <stdio.h>
main( )
{  int  i;
   for ( i =1; i<=5; i ++)
   {  if (i%2)  printf (" * " );
      else    continue;
      printf ("#");
   }
   printf ("$\n" );
}
```

（A）*#*#$　（B）#*#*#*$　（C）*#*#*#$　（D）***#$

41．**以下程序的输出结果是_____。**

```
main( )
{  int i, x[3][3]={9,8,7,6,5,4,3,2,1},*p=&x[1][1];
   for(i=0;i<4;i+=2)printf("%d ", p[i]);
```

```
}
```

（A）5 2　　（B）5 1　　（C）5 3　　（D）9 7

42．以下程序的运行结果是_____。

```
main()
{ char b[10]={'1','2','3','4','5','6','7','8','9', '\0'},*p=b;
  int k;
  k=8; p=b+k;
  printf("%s \n",p-3);
}
```

（A）6　　（B）6789　　（C）'6'　　（D）789

43．以下程序的运行结果是_____。

```
#include "stdio.h"
main( )
{ int a[]={1,2,3,4,5,6,7,8,9,10,11,12};
  int *p=a+5,*q=NULL;
  *q=*(p+5);
  printf("%d %d\n",*p,*q);
}
```

（A）运行后报错　　（B）6 6　　（C）6 12　　（D）5 5

44．以下程序的输出结果是_____。

```
# include <string.h>
main( )
{  char  * a = "abcdefghi"; int k;
   fun(a); puts(a);
}
fun ( char * s)
{  int  x, y;  char   c;
   for ( x=0, y=strlen(s)-1; x<y; x++,y--)
   { c=s[y];s[y]=s[x];s[x]=c;}
}
```

（A）ihgfedcba　　（B）abcdefghi　　（C）abcdedcba　　（D）ihgfefghi

45．以下程序的输出结果是_____。

```
main ( )
{ int  n[3][3] , i,  j;
  for (i=0; i< 3 ; i++)
    for (j=0; j<3; j++)  n[i][j]=i+j;
  for ( i=0; i<2; i++)
    for (j=0; j<2; j++)  n[i+1][j+1]+=n[i][j];
  printf("%d \n",n[i][j]);
}
```

（A）14　　（B）0　　（C）6　　（D）值不确定

46．以下程序的输出结果是_____。

```
main( )
{ union{  char  i[2];  int  k;  } r;
```

```
    r.i[0]=2;       r.i[1] = 0;
    r.k=r.i[0]+r.i[1]*2;
    printf("%d \n", r.k);
  }
```

（A）2　　（B）1　　（C）0　　（D）不确定

47．设有以下定义和语句，则输出的结果是(用 small 模式编译，指针变量占两个字节)_____。

```
struct   date
{  long  * cat;
   struct  date   *next;
   double  dog;
} too;
printf ("%d", sizeof(too));
```

（A）20　　（B）16　　（C）14　　（D）12

48．以下程序的输出结果是_____。

```
main( )
{   int  w=5; fun(w); printf("\n");  }
fun (int k)
{   if (k>0) fun(k-1); printf("%d", k);}
```

（A）5　4　3　2　1　　（B）0　1　2　3　4　5

（C）1　2　3　4　5　　（D）5　4　3　2　1　0

49．以下程序的输出结果是_____。

```
int   a=1;
fun(int  k)
{ static  int  a=5;
  a+=k;  printf("%d ",a);
  return(a);
}
main( )
{ int  b=3;  printf("%d \n", fun(b+fun(a))); }
```

（A）6 9 9　　（B）6 6 9　　（C）6 15 15　　（D）6 6 15

50．以下程序的输出结果是_____。

```
main( )
{  char  *p="12134211";   int z[4]={0,0,0,0},j,i;
   for (j=0; p[j];j++)
   { switch (p[j])
     { case  '1': i=0;
       case  '2': i=1;
       case  '3': i=2;
       case  '4': i=3;
     }
     z[i]++;
       }
   for(j=0;j<4;j++) printf("%d ",z[j]);
```

```
}
```

（A）4 2 1 1　　（B）0 0 0 8　　（C）4 6 7 8　　（D）8 8 8 8

## 二、填空题（每题 2 分，共 40 分）

1．在计算机中，可以采用【1】结构来表示算术表达式。

2．在长度为 n 的有序线性表中进行二分法查找，在最坏情况下，需比较的次数为【2】。

3．类是具有共同属性、共同操作方法的对象的集合，所以类是对象的【3】。

4．软件工程概念的出现源自【4】。

5．关系模型允许定义三类数据约束，它们是【5】约束、参照完整性约束以及用户定义的完整性约束。

6．若有说明 int i,j,k;则表达式 i=10,j=20,k=30,k*=i+j 的值为【6】。

7．执行程序：

```
main()
  {  int  x,i,j,k;
     scanf("%d",&x);
     i=x%10 ;  x=x/10 ;  j=x%10 ;  k=x/10;
     printf ("%d" , i*100+j*10+k);
  }
```

当输入 234 时，输出结果为：【7】_。

当输入 2345 时，输出结果为：【8】_。

8．若有定义语句 char a=5; int flag ; 则执行下列语句后 flag 的值是【9】。

```
if (10>a>1)  flag=1;
else  flag=0;
```

9．下列程序段的循环次数【10】。

```
n=0;  i=7;
do
n=2*n+1;
while (n<=i);
```

10．设 char a[ ]= "abc"，b[20]= "abcd"，c[20]；执行语句 strcpy(c,strcat(b,a))后 *c* 中的内容为【11】。

11．设下列定义语句，则表达式‘ p->name[2] ’的值是【12】；“ (*p).age ”的值是【13】。

```
struct  stud
{  char  name[20];
   int  age;
   char  sex;
}
 ……
struct stud  x={"zhang",20, 'm'},*p=&x;
```

12．以下程序是选出能被 3 整除且至少有一位是 5 的两位数，打印出所有这样的数及其个数。请选择填空。

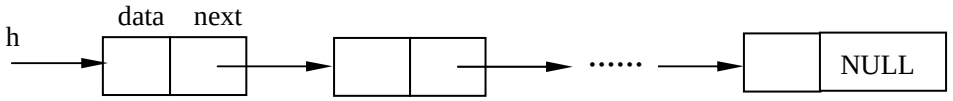
```
 sub( int k, int n )
 {  int a1, a2;
```

```
   【14】
   【15】
   if ((k%3==0&&a2==5)||(k%3== 0 && a1 == 5))
    {  printf (" %d ", k);
       n++;
       return n;
    }
   else   return -1;
}
main( )
{  int  n = 0, k, m;
   for ( k=10; k<=99; k ++)
   {  m=sub(k,n);
      if( m!= -1)   n=m;
   }
   printf ( "n=%d\n", n);
}
```

13．以下程序将一个名为 fl.dat 的文件拷贝到一个名为 f2.dat 的文件中。请填入正确答案。

```
# include<stdio.h>
main( )
{  char c; FILE * fpl,*fp2;
   fp1=fopen("f1,dat",【16】);
   fp2=fopen("f2.dat",【17】);
   c=getc(fp1);
   while(c! =EOF)
   {  【18】(c,fp2); c=fgetc(fp1); }
   fclose (fp1); fclose (fp2);
   return;
}
```

14．以下函数 creatlist 用来建立一个带头结点的单链表,链表的结构如下图所示，新的结点总插入在链表的末尾。链表的头指针作为函数值返回，链表最后一个结点的 next 域放入 NULL 作为链的结束标志。data 为字符型数据域，next 为指针域，读入时字符以#表示输入结束(# 不存入链表)。请填空




```
struct node
{    char data
      struct node *next;
};
  ……
【19】creatlist( )
{  struct node  *h,*s,*r;  char  ch;
```

```
    h=(struct node *)malloc(sizeof(struct  node));
    r=h;
    ch=getchar( );
    while(ch != '# ' )
    {  s=(struct node *)malloc(sizeof(struct  node));
       s->data=ch; r->next=s; r=s;
       ch=getcgar( );
       r->next=__【20】__;}
    return  h;
}
```

### 18.1.2 笔试模拟试题（二）

**一、选择题**[（1）～（10），每小题 2 分，（11）～（50）每题 1 分，共 60 分]

1．在数据结构中，根据各数据元素之间前后件关系的复杂程序，一般将数据结构分成_____两类。

（A）动态结构和静态结构　　（B）紧凑结构和非紧凑结构

（C）线性结构和非线性结构　　（D）内部结构和外部结构

2．下面关于队列的叙述中正确的是_____。

（A）在队列中只能插入数据　　（B）在队列只能删除数据

（C）队列是先进先出的线性表　　（D）队列是先进后出的线性表

3．已知一棵二叉树的前序遍历为 ABDECF，中序遍历为 DBEAFC，则对该树进行后序遍历得到的序列为_____。

（A）DEBAFC　　（B）DEFBCA　　（C）DEBCFA　　（D）DEBFCA

4．已知一个有序表为（14,21,27,39,45,53,66,80,91,119,150），当使用二分法查找值为元素 91 的元素时，查找成功的比较次数为_____。

（A）1　　（B）2　　（C）3　　（D）5

5．在第 1 趟排序之后，一定能把数据表中最大或最小元素放在其最终位置上的排序算法是_____。

（A）冒泡排序　　（B）插入排序　　（C）快速排序　　（D）希尔排序

6．在面向对象技术中，对象封装了属性和_____。

（A）消息　　（B）参数　　（C）地址　　（D）方法

7．软件测试的目的是_____。

（A）证明软件系统中存在错误

（B）找出软件系统中存在的所有错误

（C）尽可能多地发现软件系统中的错误和缺陷

（D）证明软件的正确性

8．进行需求分析可使用多种工具，但_____是不适用的。

（A）数据流图　　（B）判定表　　（C）PAD 图　　（D）数据词典

9．设 *R* 和 *S* 为 3 个关系，_____中的符号分别代表选择、投影、乘积的关系代数运算。

（A）$f\Omega(R)$、$\pi_A(R)$、$R\times S$　　（B）$E_A(R)$、$V_A(S)$、$R*S$

（C）$R_i$占、$R\cup S$、$R\times S$　　（D）$\pi_A(R)$、$f\Omega(R)$、$R\times S$

10．在关系数据库管理系统中，创建的视图在数据库三层结构中属于_____

（A）外模式　　（B）存储模式　　（C）内模式　　（D）概念模式

11．以下运算符中优先级最高的是_____。

（A）>=　　（B）%=　　（C）&&　　（D）++

12．在下列符号中，不属于转义字符的是_____。

（A）\　　（B）\x12　　（C）\013　　（D）\05

13．浮点型变量 *f*，能实现对 *f* 中的值在小数点后第 3 位进行四舍五入的表达式是_____。

（A）f = (f * 100 + 0.5)/100.0　　（B）（f * 100 + 0.5）/100

（C）f = (int)(f * 100 + 0.5)/100.0　　（D）f = (f/100 + 0.5) * 100

14．执行语句“x=(a=5,b=a--)”后，*x,a,b* 的值依次为_____。

（A）5,5,4　　（B）5,4,4　　（C）5,4,5　　（D）4,5,4

15．设整型变量 *k,p,x,y,m,n* 均为 1，执行“(m=x>y)&&(n=k>p)”后 *m,n* 的值是_____。

（A）0,0　　（B）0,1　　（C）1,0　　（D）1,1

16．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

17．使用“scanf("a=%d,b=%d",&a,&b)”,要使 *a,b* 均为 125，正确的输入是_____。

（A）125,125　　（B）125 125 (空格分开)

（C）a=125,b=125　　（D）a=125 b=125(空格分开)

18．假定所有变量均已正确定义，下列程序段运行后 *x* 的值是_____。

```
a=b=c=x=y=0;
if(b)  x--;
else  if(c)  y=1;
      if(a)  x=4;
      else  x=3;
```

（A）1　　（B）0　　（C）4　　（D）3

19．以下程序段中，变量 *n* 是计算外循环体的执行次数的，程序执行后 n 的值为_____。

```
main()
{  int i,j,n=0;
   for( i=4; i; i--)
     for(j=0;j-5;j++)
         n++;
   printf("%d",n);
    }
```

（A）20　　（B）24　　（C）25　　（D）30

20．若有以下说明，则数值为 5 的表达式是_____。

```
int a[12]={1,2,3,4,5,6,7,8,9,10,11,12};
char c='a',e,g;
```

（A）a[g–c]　　（B）a[5]　　（C）a['d'–'c']　　（D）a['e'–c]

21．*t* 为 int 类型，进入下面的循环之前，*t* 的值为 0

```
while( t=l )
{   ……   }
```

则以下叙述中正确的是_____。

（A）循环控制表达式的值为 0　　（B）循环控制表达式的值为 1

（C）循环控制表达式不合法　　（D）以上说法都不对

22．语句“printf("%d\n",strlen("asd\n\x12\1\\"));”的输出结果是_____。

（A）9　　（B）7　　（C）5　　（D）8

23．若有说明：int i,j=9,*p1=&j;则与 i=j;等价的语句是_____。

（A）i=*p1　　（B）*p1=*&j　　（C）i=&j　　（D）i=**p1

24．若有以下定义，则不能表示 *a* 数组元素的表达式是_____。

```
int a[10]={5,1,3,4,2,6,7,8,9,11},*p=a;
```

（A）*p　　（B）a[10]　　（C）*a　　（D）a[p–a]

25．若有以下定义，则数值 4 的表达式是_____。

```
int a[3][4]={{0,1},{2,4},{5,8}},(*p)[4]=a;
```

（A）*a[0]+2　　（B）p++,*(p+1)　　（C）a[2][2]　　（D）p[1][1]

26．以下能正确定义数组并正确赋初值的语句是_____。

（A）int N=5,b[N][N];　　（B）int a[1][2]={{1},{3}};

（C）int c[2][]={{1,2},{3,4}};　　（D）int d[3][2]={{1,2},{34}};

27．设有定义语句“struct { int x;int y;} d[2]={{1,3},{2,7}};”，则 printf("%d\n",d[0].y/d[0].x*d[1].x);的输出是_____。

（A）0　　（B）1　　（C）3　　（D）6

28．若有如下定义，则 printf("%d\n",sizeof(he));的输出是_____。

```
typedef union { long x[2];int y[4];char z[8];} MYTYPE;
MYTYPE he;
```

（A）32　　（B）16　　（C）8　　（D）24

29．输入“12345,xyz”，下列程序输出的结果是_____。

```
main()
{    int x;  char y ;
     scanf("%3d%c",&x,&y);
     printf("%d,%c",x,y); }
```

（A）123,xyz　　（B）123,4　　（C）123，x　　（D）12345,xyz

30．有如下程序

```
main( )
{   int a[3][3]={{1,2},{3,4},{5,6}},i,j,s=0;
    for(i=1;i<3;i++)
        for(j=0;j<=i;j++) s+=a[i][j];
    printf("%d\n",s);
}
```

该程序的输出结果是_____。

（A）18　　（B）19　　（C）20　　（D）21

31．写出下列程序的运行结果_____。

```
#include <stdio.h>
main( )
{  int  i,j=4;
   for(i=j;i<=2*j;i++)
   { switch (i/j)
    {  case  0:
       case  1:printf("$");  break;
       case  2:printf("*");
    }
  }
}
```

（A）$*$$*　　（B）$$$$*　　（C）**$$*　　（D）$**$*

32．写出下列程序的执行结果_____。

```
main( )
{  int a[10],i,k = 0;
   for( i=0;i<10;i++) a[i]=i;
   for(i=1;i<4;i++) k+=a[i]+i;
   printf("%d\n",k);
}
```

（A）10　　（B）12　　（C）11　　（D）14

33．写出下列程序的执行结果_____。

```
main( )
{   int a[ ]={2,4,6},*pr=&a[0],x=6,y,z;
    for(y=0;y<4;y++)
    z=(*(pr+y)<x)? * (pr+y):x;
    printf("%d\n",z);
}
```

（A）6　　（B）0　　（C）4　　（D）2

34．写出下列程序的运行结果_____。

```
main()
{ void swap1();
  void swap2();
  int a=3,b=4;
  swap1(a,b);
  printf("%d,%d\n",a,b);
  a=3;b=4;
  swap2(&a,&b);
  printf("%d,%d",a,b);
}
void swap1(x,y)
int x,y;
{  int t; t=x ; x=y ; y=t;}
```

```
void  swap2(x,y)
int *x,*y ;
{  int t; t=*x; *x=*y ; *y=t ;}
```

（A） 3,4
4,3

（B） 4,3
3,4

（C）3 4
4 3

（D）4 3
3 4

35．写出下列程序的运行结果_____。

```
main()
{  int k=0;char c='A';
   do
    { switch (c++)
      {  case  'A':k++;break;
         case  'B':k--;
         case  'C':k+=3;break;
         case  'D':k=k%2;continue;
         case  'E':k=k*6;break;
         default :k=k/4;
      }
     k++;
    }
  while (c<'G');
  printf("k=%d\n",k);
}
```

（A）2　　（B）k=2　　（C）k=3　　（D）k=6

36．写出下列程序的运行结果_____。

```
void prt(x,y,z)
int  *x,*y,*z;
{   printf("%d,%d,%d\n",++*x,++*y,*(z++)); }
int a=12,b=20,c=30;
main()
{   prt(&a,&b,&c);
    prt(&a,&b,&c);
}
```

（A）21,13,30
14,21,30

（B）13,21,30
14,22,30

（C）13 21 30
14 21 30

（D）30 21 13
14 22 30

37．写出下列程序的运行结果_____。

```
main ()
{  int y=10,i=0,j,a[8];
   do
   {  a[i]=y%2;i++;
      y=y/2;
   } while(y>=1);
   for(j=i-1;j>=0;j--)
```

```
      printf(" %d",a[j]);
    printf("\n");
}
```

（A）1010　　（B）1001　　（C）1101　　（D）1 0 11

38．**写出下列程序的运行结果_____。**

```
main()
{  int i;
   int a[5]={2,3};
   for(i=2;i<5;i++)
      a[i]=a[i-2]+a[i-1];
   for(i=0;i<5;i++)
     {  if(i%5==0)  printf("\n");
        printf("%-3d",a[i]);
     }
}
```

（A）2 3 5 8 13　　（B）2 3 8 13 15　　（C）235813　　（D）3 5 8 13 15

39．**写出下列程序的运行结果_____。**

```
 main( )
 {   static char a[]="abcdefGH",b[]="abCDefGh";
     char *pt1,*pt2;  int k;
     pt1=a; pt2=b;
     for(k=0;k<=7;k++) if(*(pt1+k)==*(pt2+k))
     printf("%c",*(pt1+k));
     printf("\n");
 }
```

（A）abcef　　（B）abefG　　（C）abefg　　（D）ABEFG

40．**下面程序的输出是_____。**

```
main( )
{  int a[8],i,k=0;
   for(i=0;i<8;i++) a[i]=i;
   for(i=1;i<5;i++) k+=a[i]+i;
   printf("%d\n",k);
}
```

（A）12　　（B）20　　（C）21　　（D）22

41．**下面程序的输出是_____。**

```
main( )
{  int j,a[]={1,3,5,7,9,11,13,15},*p=a+5;
   for(j=5;j>0;j--)
     { switch(j)
        {  case  3:
           case  1:*(p++);break;
           case  2:*(--p);
           defealt :*p++;
        }
```

```
    }
   printf("%d",*p);
}
```

（A）14　　（B）13　　（C）15　　（D）12

42．写出下列程序的执行结果_____。

```
int a=5,c=2;
main()
{  void s1(); int a=3,b;
   b=a+c; a=a+c;
   s1(a,b);
   printf("%d,%d,%d\n",a,b,c);
}
void  s1(a,b)
int a,b;
{  int c=4;
   a=a+c;  c=a+b;
   printf("%d,%d,%d\n",a,b,c);
}
```

（A）11,8,9　　（B）9,5,14　　（C）5,5,2　　（D）5,2,5
8,8,3　　5,5,2　　9,5,14　　9,14,5

43．写出下列程序的运行结果_____。

```
main()
{  int a=5,b=7,c=3;
   int *p1=&a,*p2=&b,*p3=&c;
   fun1(p1,p2,p3);
   printf("%d,%d,%d\n",a,b,c);
}
int  fun1(int *a,int *b,int  *c)
{  int *temp;
   temp=a,a=b,b=temp;
   *temp=*b;*b=*c,*c=*temp;
}
```

（A）3,7,3　　（B）7,3,7　　（C）3,3,7　　（D）7,7,3

44．程序段：

```
int x=-2;
printf("%d,%u,%o",x,x,x);
```

输出结果为_____。

（A）–2, –2, –2　　（B）–2,32767, –177777
（C）–2,32768,177777　　（D）–2,65534,177776

45．写出下列程序的执行结果_____。

```
int p=2;
main( )
{   int s, j, sum( );
    for (j=0;j<=5;j++) s=sum(j);
```

```
    printf("s=%d\n",s);
 }
int sum(int k)
{   static int x=1; int y=1;
    p++;y++;
    return (x+=k+p+y);
}
```

（A）s=67　　（B）s=61　　（C）s=63　　（D）s=56

46．字符 0 的 ASCII 码的十进制数为 48，且数组的第 0 个元素在低位，则以下程序的执行结果是_____。

```
#include<stdio.h>
main()
{  union
   {   int i[4];
       long k;
       char c[6];
    }a,*s=&a;
   s->i[0]=0x39;
   s->i[1]=0x38;
   printf("%c\n",s->c[0]);
}
```

（A）5　　（B）3　　（C）6　　（D）9

47．写出下列程序的运行结果_____。

```
main()
{   int a=3,y;
    y=fn(a,a++);
    printf("%d",y);
}
int  fn(x,c)
int x,c;
{  int b;
   if(x<c)  b=1;
   else if(x==c) b=0;
   else b=-1;
   return(b);
}
```

（A）3　　（B）–1　　（C）1　　（D）0

48．写出下列程序的输出结果_____。

```
main( )
{  int   n; char  ch[81],*pt;
   pt=ch;
   scanf("%d",&n);
   fun(n,ch);
   puts(pt); printf("\n");
 }
```

```
fun(int  j,char *s)
{  char  c;
   int  k,i=10;
   while(j!=0)
   {  k=j%i; *s=k+'0';
      s++; *s='*';
      s++; j=(j-k)/i;
    }
   s='\0';
}
```

**输入为**: 234

（A）4*3*2*　　（B）432　　（C）4*32*　　（D）4*3*2

49．**程序段如下：**

```
char  c[20],d[20];gets(c);
```

**现要把** *c* **数组中的内容复制到** *d* **数组中，将会正确的语句是**_____。

（A）d[10]=c[10]　　（B）d=c　　（C）strcpy(d,c)　　（D）strcat(d,c)

50．**如果在用户的程序中使用** C **语言库函数中的数学函数时，应在该源文件中使用的** include **命令是**_____。

（A）#include"string.h"　　（B）#include"math.h"

（C）#include"ctype.h"　　（D）#include"stdio.h"

## 二、填空题（每题 2 分，共 40 分）

1．**队列是限定只能在表的一端进行插入和在另一端进行删除操作的线性表。允许插入的一端称做**【1】。

2．**具有** 80 **个结点的完全二叉树的深度为**【2】。

3．**源程序文档化要求程序应加注释。注释一般分为**【3】**和功能性注释。**

4．**常用的软件结构设计工具是**【4】。

5．**一个教师能开多门课程，一门课程有许多教师会开，则实体“课程”与实体“教师”间的联系属于**【5】。

6．**在** for(k=0;k<4;k++)　printf("*")**；中，表达式“** k=0 **”执行了**【6】**次，表达式**“k++”**执行了**【7】**次。**

7．**有定义如下：**int　b[7]={3,1,2}**；则** b[5]**的值为**【8】。

8．**设有说明** char str[20]**；如果想从终端上把以下字符：**This　is　a　book.**送到数组** str **中，使用的完整语句为**【9】。

9．**若有以下定义和语句：**

```
int  a[4]={0,1,2,3},*p;
p=&a[1];
```

**则**++(*p)**的值是**【10】。

10．**用“选择排序法”对一维数组中的整数进行排序，使其元素的值是按从小到大的顺序排列。**

```
   main()
{  int n,i,k,temp,min_k,a[50];
   scanf("%d",&n);
   for (i=0;i<__【11】__;i++) scanf("%d",&a[i]);
   for(k=0;k<n-1;k++)
     {  __【12】__;
        for(i=k;i<n;i++)
          if(a[i]<a[min_k]) __【13】__
         temp=a[min_k],a[min_k]=a[i],a[i]=temp;
        }
     for(i=0;i<n;i++) printf("%d  ",a[i]);
     printf("\n");
   }
```

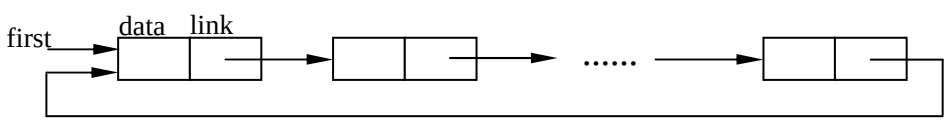
11．以下函数是把 *b* 字符串连接到 *a* 字符串的后面，并返回 *a* 中新字符串的长度。

```
strcen(char  a[ ],char  b[ ])
{  int   num=0,n=0;
   while(a[num++])!=__【14】__) ;
   while(b[n])  {a[num]=b[n];num++;__【15】__;}
   a[num]=b[n];
   return(num);
}
```

12．下面程序的功能是将一个磁盘中的二进制文件复制到另一个磁盘中，两个文件名随命令行一起输入，输入时原有文件的文件名在前，新复制文件的文件名在后。请在横线处填入适当内容。

```
# include < stdio.h>
main( int argc, char * argv[])
{   FILE * old, * new;
    char  ch ;
    if ( argc!=3)
      {   printf (" You fougot to enter a filename\n ");  exit (0);  }
    if (( old = fopen(__【16】__) = = NULL )
      {   printf (" cannot open infile\n"); exit (0);}
    if (( new = fopen(__【17】__) = = NULL)
      {   printf (" cannot open outfile\n");  exit (0); }
    while ( ! feof ( old )) fputc (__【18】__, new );
    fclose ( old );
    fclose ( new );
}
```

13．下面 min3 函数的功能是：计算单向循环链表 first 中每 3 个相邻结点数据域中值的和，返回其中最小的值。请填空。




```
struct node
```

```
{  int data;
   struct node  * link;
};
int min3 (struct node  * first)
{   struct node * p = first;
    int m,m3;
    m3=p->data+p->link->data+p->link->link->data;
    for(p=p->link;p!=first;p=__【19】__)
      {  m=p->data+p->link->data+p->link->link->data;
         if (__【20】__)
            m3 = m;
      }
    return (m3);
}
```

### 18.1.3 笔试模拟试题（三）

**一、选择题**[（1）～（10），每小题 2 分，（11）～（50）每题 1 分，共 60 分]

1．队和栈的主要区别是_____

（A）逻辑结构不同　　（B）存储结构不同

（C）所包含的运算个数不同　　（D）限定插入和删除的位置不同

2．在一颗非空二叉树中，叶子结点的总数比度为 2 的结点总数多_____个。

（A）–1　　（B）0　　（C）1　　（D）2

3．假设一棵二叉树的后序遍历序列为 DGJHEBIFCA，中序遍历序列为 DBGEHJACIF，则其前序遍历序列为_____。

（A）ABCDEFGHIJ　　（B）ABDEGHJCFI

（C）ABDEGHJFIC　　（D）ABDEGJHCFI

4．对采用二分查找法进行查找运算的查找表，要求按_____方式进行存储。

（A）顺序存储　　（B）链式存储

（C）顺序存储且结点按关键字有序　　（D）链式存储且结点按关键字有序

5．对序列（7，19，24，13，31，8，82，18，44，63，5，29）进行一趟排序后得到的结果为（7，18，24，13，5，8，82，19，44，63，31，29），则可判定使用的排序方法是：

（A）希尔排序　　（B）插入排序　　（C）快速排序　　（D）选择排序

6．软件是一种_____的产品。

（A）易复制　　（B）易损坏　　（C）易开发　　（D）易使用

7．在编制程序时，应采纳的原则之一是_____。

（A）不限制 GOTO 语句的使用　　（B）减少或取消注解行

（C）程序越短越好　　（D）程序结构应有助于读者理解

8．_____是调试程序的主要工作之一。

（A）调度　　（B）证明程序正确　　（C）人员安排　　（D）排错

9．下述说法中没有体现数据库系统特点的是_____。

（A）数据面向应用程序　　（B）数据结构化

（C）数据冗余小　　（D）数据共享性高

10．建立 E-R 模型的工作属于数据库生命周期中的_____。

（A）分析阶段　　（B）设计阶段　　（C）编码阶段　　（D）测试阶段

11．下列选项中，不能用作标识符的是_____。

（A）_1234_　　（B）_1_2　　（C）int_2_　　（D）2_int_

12．下列数据中，不属于常量的是_____。

（A）176L　　（B）'\011'　　（C）4.5L　　（D）"aDc"

13．若有定义：int　a=120；　char　b='a'；

则表达式“a < b”的结果为_____。

（A）　0　　（B）两个变量不能比较

（C）　任何一个非零的整数　　（D）1

14．若有定义：int x, y; char a, b, c; 并有以下输入数据(此处∪代表空格)：

1∪2↙

A∪B∪C↙

则能给 *x* 赋整数 1，给 *y* 赋整数 2，给 *a* 赋字符 A，给 *b* 赋字符 B，给 *c* 赋字符 C 的正确程序段是_____。

（A）scanf("x=%d y=%d"，&x,&y);a=getchar();b=getchar();c=getchar();

（B）scanf("%d　%d"，&x,&y);a=getchar();b=getchar();c=getchar();

（C）scanf("%d%d%c%c%c",&x,&y,&a,&b,&c);

（D）scanf("%d%d%c%c%c%c%c%c",&x,&y,&a,&a,&b,&b,&c,&c);

15．设 int a=6,b;则执行 b=a&&0；语句后，*b* 的结果是_____。

（A）0　　（B）1　　（C）2　　（D）3

16．下列程序执行后的输出结果是_____。

（A）G　　（B）H　　（C）I　　（D）J

```
main()
{   int x='f'; printf("%c \n",'A'+(x-'a'+1)); }
```

17．语句：printf("%03d,%-3d\n",4,5)；的输出为_____。

（A）004,5　　（B）004,　5　　（C）4,5　　（D）4,　5

18．表达式 1&&3&&0&&7 的值为_____。

（A）1　　（B）3　　（C）0　　（D）7

19．若有运算符：>、+=、<<、%、sizeof，则按它们优先级（由低至高）的正确排列次序为_____。

（A）+= →<< → > →% → sizeof

（B）<< → += → > →% → sizeof

（C）+= → > → << →sizeof → %

（D）+= → > → << → % → sizeof

20．若有以下类型说明语句：

```
char w;int x;float y;double z;
```

则表达式，*w* * *x* + *z*-*y* 的结果为_____类型。

（A）float　　（B）char　　（C）int　　（D）double

21．有如下函数调用语句：

```
func(rec1,rec2+rec3,(rec4,rec5));
```

该函数调用语句中，含有的实参个数是_____。

（A）3　　（B）4　　（C）5　　（D）有语法错

22．若 *a*、*b*、*m*、*n* 均为 int 型变量，则执行下面语句后的 *b* 值为_____。

```
m = 20; n = 6;
x= (--m == n++) ? --m:++ n;
b= m++;
```

（A）11　　（B）6　　（C）19　　（D）10

23．若有 char a[4]= {'A','\0','D'},*p=a, i=2, j=4;则下列表达式值不是 1 的有_____。

（A）strlen (a);　　（B）j||j;　　（C）* (p+2);　　（D）––i;

24．下列程序段的输出结果是_____。

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

```
void fun(int *x, int *y)
{   printf("%d %d", *x, *y); *x=3; *y=4;}
main()
{   int x=1,y=2;
    fun(&y,&x); printf("%d %d",x, y);
}
```

25．有以下程序

```
main()
{   int p[7]={11,13,14,15,16,17,18},i=0,k=0;
    while(i<7&&p[i]%2){    k=k+p[i]; i++;  }
    printf("%d\n",k)
}
```

执行后输出结果是_____。

（A）58　　（B）56　　（C）45　　（D）24

26．若 x 为 int 型变量，则以下函数 fun_____。

```
fun ( x )
int x;
{printf ("%d\n",x);}
```

（A）返回值为 void　　（B）无法确定返回值

（C）返回值为无类型　　（D）没有返回值

27．请读程序：

```
f (char *s)
{  char *p=s;
   while( *p !='\0') p++;
   return(p-s);
```

```
}
main()
{ printf("%d\n",f("abcdefhjkljl"));}
```

**上面输出结果是_____。**

（A） 3　　（B） 6　　（C） 8　　（D） 12

28．**请读程序：**

```
#include<stdio.h>
main()
{   int n[2],i,j;
    for(i=0,j=0;i<2;i++)   n[j]=n[i]+1;
    printf("%d\n", n[j]);
}
```

**上面程序的输出结果是_____。**

（A）**不确定的值**　　（B）203　　（C）2 0 3　　（D）201

29．**若有以下说明，且** *0<=j<4*，**则_____是错误的赋值。**

```
int a[5][10],*pt,*qt[5];
```

（A）*pt=a　　（B）qt[j]=a[j]　　（C）pt=a[j]　　（D）qt[j]=&a[2][0]

30．**若有以下说明：**int a[5][9]；**则_____是对数组元素** a[*i*][*j*]**的错误引用（此处** 0<=*i*<4,0<=*j*<9**）。**

（A）*(&a[0][0]+9*i+j)　　（B）*(a+i)(j)

（C）*(*(a+i)+j)　　（D）*(a[i]+j)

31．**若有以下说明，则对** *a* **数组元素地址的正确引用的是_____。**

```
int p[6],*k=p;
```

（A）&p[6]　　（B）k+2　　（C）p++　　（D）&p

32．**若有以下说明和语句，则对** up **中** sex **域的正确引用方式是_____。**

```
struct pupil
{  char name[20];
   int sex;
 }  up,*p;
 p=&up;
```

（A）p.up.sex　　（B）p->up.sex　　（C）(*p).up.sex　　（D）(*p).sex

33．fputc **函数的作用是向指定的文件写入一个字符，该文件的打开方式是_____。**

（A）**只写**　　（B）**只读**　　（C）**读或读写**　　（D）B **和** C **都正确**

34．**下列程序的输出结果为_____。**

```
main()
{  int x=1;
   while (x<=6)
      if(x++%2!=0)   continue;
      else  printf("%d",x);
   printf("\n");
}
```

（A）3578　　（B）357　　（C）573　　（D）375

35．下列程序的输出结果为_____。

```
main()
{   int k=0,i, s[ ]={1,-9,7,2,-10,3};
    for(i=0;i<6;i++) if(s[i]>s[k])  k=i;
    printf("\n%d\n",k);
}
```

（A）4　　（B）2　　（C）3　　（D）1

36．下列程序的输出结果为_____。

```
#include"stdio.h"
main()
{  int  i=0, j=0;
   char  a[80], b[80];
   gets(a);
   while( a[i]!='\0')
      {  if(a[i]>='a'&&a[i]<='z') { b[j]=a[i]; j++; }
         i++;
      }
   puts(b);
}
```

**当程序运行时输入：**the number is 123.

（A）Thenumberis　　（B）thenumberis

（C）The number is　　（D）the number is

37．下列程序的输出结果为_____。

```
int c=6;
void test(x,y)
int *x,y;
{   *x=3*(*x);y=*x+y;c=y%(*x);
    printf("x=%d,y=%d,c=%d\n",*x,y,c);
}
main()
{  int a=1,b=4;
   test(&a,b) ;
   printf("a=%d,b=%d,c=%d\n",a,b,c);
}
```

（A）x=1,y=7,c=3
a=3,b=4,c=1

（B）x=3,y=7,c=1
a=4,b=3,c=1

（C）x=3,y=7,c=1
a=3,b=4,c=1

（D）x=3,y=4,c=1
a=3,b=7,c=1

38．下列程序的输出结果为_____。

```
main()
{   int i,b,c,s[ ]={1,5,-3,-21,7,13,8},*pb,*pa;
   b=c=1; pb=pa=s;
   for (i=0;i<7;i++)
     {  if (b<*(s+i))  {b=*(s+i); pb=&s[i]; }
```

```
        if (c>*(s+i))  {c=*(s+i); pa=&s[i]; }
      }
      i=*s;*s=*pb; *pb=i; i=*(s+5); *(s+5)=*pa; *pa=i;
   for(i=0;i<7;i++) printf("%d  ",s[i]);
}
```

（A）13 5 –3 1 7 –21 8　　（B）135–317–21 8

（C）13 5 –3 1 7 8 –21　　（D）13 –3 5 1 7 –21 8

39．**已知** char s1[10] , *s2="abc\0def"; **则执行语句** strcpy(s1，s2);**之后，变量** strlen(s1)**的值是**_____。

（A）7　　（B）3　　（C）4　　（D）8

40．**若已定义：**int a[9],*p=a;**并在以后的语句中未改变** *p* **的值， 不能表示** a[2]**地址的表达式为**_____。

（A）p+2　　（B）a+2　　（C）a++　　（D）++(++p)

41．**执行下述程序片段后，变量** *m* **的值是**_____。

```
main()
{  int  a[ ]={7,4,6,3,11,5,9,1};
   int  m=1,k,*pr=&a[2];
   for(k=0;k<5;k++) m=*(pr+k)<m?*(pr+k):m;
   printf("%d",m);
}
```

（A）1　　（B）3　　（C）6　　（D）4

42．**已定义** *a*,*b* **为整型，且** *a*,*b* **分别赋值为** 1 **和** 0**，语句：**printf("%d",(a*=2)&&(b+=–2));**的输出结果是**_____。

（A）**无输出**　　（B）**结果不确定**　　（C）–1　　（D）1

43．**下面程序把数组元素中的最小值放入** a[0]**中。则在** if **语句中条件表达式应该是**_____。

```
main ( )
{   int a[10] ={6,7,2,9,1,10,5,8,4,3},*p = a,i;
    for (i=0;i<10;i++,p++)
       if (_____)  *a=*p;
       printf("%d",*a);
}
```

（A）p>a　　（B）*p<a[0]　　（C）*p<*a[0]　　（D）*p[0]<*a[0]

44．**以下程序的输出结果是**_____。

```
main()
{   int  a[4][4],i,j;
    for(i=0;i<4;i++)
      for(j=0;j<4;j++)  a[i][j]=i+j;
    for(i=0;i<3;i++)
      for(j=0;j<3;j++)  a[i+1][j+1]+=a[i][j];
    printf("%d\n",a[i][j]);
}
```

（A）14　　（B）0　　（C）12　　（D）值不确定

45．以下程序的输出结果是_____。

```
main( )
 {  char  c[3][4]={"asd","GHJ","78"},  *p[3];
    int  i;
    for(i=0;i<3;i++) p[i]=c[i];
    for(i=0;i<3;i++) printf("%s",p[i]);
 }
```

（A）GHJ78　　（B）asdGHJ78　　（C）asdGHJ　　（D）ASDGHJ78

46．以下程序的输出结果是_____。

```
#include<string.h>
main( )
{  char  *p1, *p2, ch[20]="AbCdEfG123456";
   p1="abcd"; p2="efgh";
   strcpy(ch+1, p2+1); strcpy(ch+3,p1+3);
   printf("%s\n",ch);
}
```

（A）Afgd123456　　（B）Abfhd　　（C）Afghd　　（D）Afgd

47．以下程序的输出结果是_____。

```
main()
{   int  i;
    for(i=0;i<5;i++)
      {  if(i%2) { printf("$"); continue; }
         printf("#");
      }
    printf("\n");
}
```

（A）#$#$#　　（B）#####　　（C）$$###$　　（D）#$#$$

48．执行以下程序后，*y* 的值是_____。

```
main()
{  char ch[]={'a','f','b','d','h','c'},*p;
   int y=1,j;
   p=&ch[1];
   for(j=0;j<3;j++) y+=*(p+j);
   printf("%d\n",y);
}
```

（A）305　　（B）301　　（C）278　　（D）347

49．下面程序的输出结果是_____。

```
#include<stdio.h>
#include<string.h>
main()
{   char *p1="abc12",*p2="ABC34";
    char str[30]="EFG";
    strcpy(str+1,strcat(p1,p2));
```

```
    printf("%s\n",str);
  }
```

（A）abc12ABC34　　（B）EFGabc12ABC34

（C）Eabc12ABC34　　（D）EabcABC34

50．下面程序的输出结果是_____。

```
#include<stdio.h>
fun( int b[],int n)
{   int i,k=1;
    for(i=0;i<=n;i++) k*=b[i];
    return k;
}
main()
{  int t,n[]={4,6,3,8,9,7,5};
   t=fun(n,4);
   printf("%d\n",t);
}
```

（A）5184　　（B）5728　　（C）3467　　（D）5183

## 二、填空题（每题 2 分，共 40 分）

1．线性表是最简单的一种数据结构，有顺序和链接两种存储方式。线性表按链接方式存储时，每个结点包括___【1】___两部分。

2．已知一棵含有 n 个结点的树中，只有度为 k 的结点和度为 0 的叶子结点，则该树中含有的叶子结点个数为___【2】___。

3．在面向对象方法中，类之间共享属性和操作机制称为___【3】___。

4．数据流图的类型有变换型和___【4】___。

5．用二维表数据来表示实体及实体之间联系的数据模型称为___【5】___。

6．执行下列语句后，*a* 的值是___【6】___。

```
int  a=11 ; a+=a-=a*a;
```

7．在执行 static int b[ ] ={1,2,3,4,5,6}，*p=b;语句后，表达式(*++p)++ 的值是___【7】___。

8．有如下定义：

```
struct {int x; int y;}tab[2]={{1,6},{7,9}}, *p=tab;
```

则表达式 p->y 的结果是___【8】___。

9．设 static int a[ ][4]={1,3,5,7,9,11,12,14,16,18,20,23}; int *p=&a[1][2]; 则*(p+2) 的值是___【9】___。

10．写出下列程序的执行结果___【10】___。

```
int a=4,c=3;
main()
{ void  f1();
  int a=5,b;
  b=a+c; a=a+c;
  f1(a,b);
  printf("%d,%d,%d\n",a,b,c);
```

```
}
void  f1(a,b)
int a,b;
{  int c=3;
   a=a+c; c=a+b;
   printf("%d,%d,%d\n",a,b,c);
}
```

11．此程序是用来求 3×2 矩阵的转置，矩阵由键盘输入，将其转置输出。

```
main( )
{  int  a[3][2], b[2][3];
   int  i,j;
   for(i=0;i<3;i++)
     for(j=0;j<2;j++)
       scanf("%d", 【11】 );
   for(i=0;i<3;i++)
      for(j=0;j<2;j++)
        b[j][i]= 【12】 ;
   for(i=0;i<2;i++)
     {  printf("\n");
        for(j=0;j<3;j++)
          printf("%4d",b[i][j]);
     }
}
```

12．此程序的作用是：在 main 函数调用 abcd 函数来交换两个变量的值。

```
main( )
{  void  abcd ( );
   float  x=10, y=20;
   abcd (  【13】  );
   printf("%f , %f\n",x,y);}
void  abcd ( x , y)
  【14】 ;
{ float  temp;
  temp =*x;   *x=*y; *y= temp;}
```

13．此程序用于从键盘输入一个以'\n'为结束标志的字符串，将它存入文本文件 abc.txt 中。

```
#include "stdio.h"
main( )
{  FILE *fp; char ch;
   if ( ( ( fp= 【15】 ) )==NULL)
      { printf("can not open the file "); exit(0);}
   ch=getchar( );
   while ( 【16】 )
      { fputc( ch, fp);  ch=getchar( ); }
   【17】 (fp);
}
```

14．以下程序段用以统计链表中元素的个数。其中 head 指向链表头结点，sum 用来统计

**结点个数，请填空。**

```
struct link
{  char  data;
   struct link  *next;
};
struct link *p, *head;
……
int  sum=0;
p=head->next;
while(  【18】  )
  { 【19】 ;
     p= 【20】 ;
  }
……
```

# 18.2　笔试模拟试题参考答案

## 18.2.1　笔试模拟试题（一）答案

**一、选择题[（1）～（10），每小题 2 分，（11）～（50）每题 1 分，共 60 分]**

| | | | | |
|---|---|---|---|---|
| 1．D | 2．B | 3．C | 4．C | 5．D |
| 6．D | 7．D | 8．A | 9．C | 10．B |
| 11．B | 12．C | 13．C | 14．A | 15．B |
| 16．D | 17．B | 18．D | 19．B | 20．D |
| 21．C | 22．A | 23．B | 24．A | 25．B |
| 26．D | 27．B | 28．B | 29．B | 30．B |
| 31．D | 32．C | 33．D | 34．A | 35．C |
| 36．C | 37．B | 38．C | 39．A | 40．C |
| 41．C | 42．B | 43．A | 44．A | 45．C |
| 46．A | 47．D | 48．B | 49．C | 50．B |

**二、填空题（每题 2 分，共 40 分）**

| | | | |
|---|---|---|---|
| **【1】树** | 【2】$\log_2 n$ | **【3】抽象** | **【4】软件危机** |
| **【5】实体完整性** | 【6】900 | 【7】432 | 【8】563 |
| 【9】0 | 【10】4 | 【11】abcdabc | 【12】*a* |
| 【13】20 | 【14】a2=k%10; | 【15】a1=k/10; | 【16】"r" |
| 【17】"w" | 【18】fputc | 【19】struct node　* | |

【20】NULL 或'\0'或'0'

### 18.2.2 笔试模拟试题（二）答案

**一、选择题[（1）～（10），每小题 2 分，（11）～（50）每题 1 分，共 60 分]**

| | | | | |
|---|---|---|---|---|
| 1．C | 2．C | 3．D | 4．B | 5．A |
| 6．D | 7．C | 8．C | 9．A | 10．A |
| 11．D | 12．A | 13．C | 14．C | 15．B |
| 16．C | 17．C | 18．D | 19．A | 20．D |
| 21．B | 22．B | 23．A | 24．B | 25．D |
| 26．D | 27．D | 28．C | 29．B | 30．A |
| 31．B | 32．B | 33．A | 34．A | 35．B |
| 36．B | 37．A | 38．A | 39．B | 40．B |
| 41．C | 42．B | 43．A | 44．D | 45．B |
| 46．D | 47．B | 48．A | 49．C | 50．B |

**二、填空题（每题 2 分，共 40 分）**

| | | | |
|---|---|---|---|
| 【1】队尾 | 【2】7 | 【3】序言性注释 | 【4】结构图 |
| 【5】多对多 | 【6】1 | 【7】4 | 【8】0 |
| 【9】gets(str); | 【10】2 | 【11】*n* | 【12】min_k=k; |
| 【13】min_k=i; | 【14】'\0' | 【15】*n*++ | 【16】argv[1], "rb" |
| 【17】argv[2], "wb" | 【18】fgetc(old) | 【19】p->link | |

【20】m<m3 或 m3>m 或 m<=m3 或 m3>=m

### 18.2.3 笔试模拟试题（三）答案

**一、选择题[（1）～（10），每小题 2 分，（11）～（50）每题 1 分，共 60 分]**

| | | | | |
|---|---|---|---|---|
| 1．D | 2．C | 3．B | 4．C | 5．A |
| 6．A | 7．D | 8．D | 9．A | 10．B |
| 11．D | 12．C | 13．A | 14．D | 15．A |
| 16．A | 17．A | 18．C | 19．D | 20．D |
| 21．A | 22．C | 23．C | 24．A | 25．D |
| 26．B | 27．D | 28．A | 29．A | 30．B |
| 31．B | 32．D | 33．A | 34．B | 35．B |
| 36．B | 37．C | 38．A | 39．B | 40．C |

41．A　　42．D　　43．B　　44．C　　45．B
46．D　　47．A　　48．B　　49．C　　50．A

## 二、填空题（每题 2 分，共 40 分）

【1】数据域和指针域　　【2】((k−1)×n+1)/k　　【3】继承　　【4】事务型

【5】关系模型　　【6】−220　　【7】2　　【8】6

【9】16　　【10】11,8,19 <回车>8,8,3　　【11】&a[i][j]

【12】a[i][j]　　【13】&x,&y　　【14】float *x,*y

【15】fopen("abc.txt","w")　　【16】ch!= '\n'　　【17】fclose　　【18】p!=NULL

【19】sum++　　【20】(*p).next　或 p->next

# 第 19 章　上机模拟试题及参考答案

## 19.1　上机考试模拟试题

### 19.1.1　上机考试模拟试题（一）

**一、程序填空题**

**函数** fun **的功能是：计算**

$$f(x)=1+x+\frac{x^2}{2!}+\Lambda+\frac{x^n}{n!}$$

**直到** $\left|\frac{x^n}{n!}\right|<10^{-6}$**。若** $x$=2.5**，函数值为：**12.182494**。**

**注意：部分源程序给出如下。**

**请勿改动主函数** main **和其他函数中的任何内容，仅在下划线上填入所需的内容。**

```
#include   <stdio.h>
#include   <math.h>
double fun(double  x)
{  double  f, t; int n;
/**********found**********/
   f = 1.0+___1___;
   t = x;
   n = 1;
   do {
       n++;
/**********found**********/
       t *= x/___2___;
/**********found**********/
       f += ___3___;
   } while (fabs(t) >= 1e-6);
   return  f;
}
main()
{  double  x, y;
   x=2.5;
   y = fun(x);
   printf("\nThe result is :\n");
```

```
    printf("x=%-12.6f    y=%-12.6f \n", x, y);
}
```

## 二、程序改错题

**给定程序** MODI1.C **中函数** fun **的功能是：从低位开始取出长整型变量** *s* **中奇数位上的数，依次构成一个新数放在** *t* **中。高位仍在高位，低位仍在低位。**

**例如，当** *s* **中的数为：**7654321 **时，**t **中的数为：**7531

**请改正程序中的错误，使它能得到正确结果。**

**注意：不要改动** main **函数，不得增行或删行，也不得更改程序的结构。**

```
#include <stdio.h>

/************found************/
void fun (long  s, long t)
{   long   sl=10;
    *t = s % 10;
    while ( s > 0)
    {   s = s/100;
        *t = s%10 * sl + *t;
/************found************/
 sl = sl*100;
    }
}

main()
{  long s, t;
   printf("\nPlease enter s:"); scanf("%ld", &s);
   fun(s, &t);
   printf("The result is: %ld\n", t);
}
```

## 三、程序设计题

**请编写一个函数** fun()，**它的功能是计算并输出给定整数** *n* **的所有因子(不包括** 1 **与自身)的平方和（规定** *n* **的值不大于** 100）。

**例如：主函数从键盘给输入** *n* **的值为** 56，**则输出为** sum=1113。

**注意：部分源程序给出如下。**

**请勿改动主函数** main **和其他函数中的任何内容，仅在函数** fun **的花括号中填入所编写的若干语句。**

```
#include  <stdio.h>
long fun(int n)
{

}
```

```
main()
{
   int  n;
   long sum;
   printf("Input  n:");
   scanf("%d", &n);
   sum=fun(n);
   printf("sum=%ld\n", sum);
}
```

### 19.1.2 上机考试模拟试题（二）

**一、程序填空题**

**下列给定程序中，函数** fun()**的功能是：将** *s* **所指字符串中的字母转换为按字母序列的后续字母(但** Z **转化为** A**，**z **转化为** a)**，其他字符不变。**

**注意：部分源程序给出如下。**

**请勿改动主函数** main **和其他函数中的任何内容，仅在下划线上填入所需的内容。**

```
#include  <stdio.h>
#include  <ctype.h>
#include  <conio.h>
void fun(char *s)
{while(__1__)
  { if(*s>='A'&&*s<='Z'||*s>='a'&&*s<='z')
     {if(*s=='Z')  *s='A';
      else if(*s=='z')  *s='a';
      else   *s+=__2__;
     }
    __3__;
  }
}
main()
{  char s[80];
   printf("\n Enter a string with length<80:\n\n"); gets (s);
   printf("\n The string:\n\n"); puts(s);
   fun(s);
   printf("\n\n The Cords :\n\n"); puts(s);
}
```

**二、程序改错题**

**给定程序** MODI1.C **中函数** fun **的功能是：根据形参** *m* **的值（**$2 \le m \le 9$**），在** *m* **行** *m* **列的二维数组中存放如下所示规律的数据，由** main **函数输出**

**例如，若输入** 2　　**若输入** 4

**则输出：**　　**则输出：**

```
1   2
2   4
```

```
1   2   3   4
2   4   6   8
3   6   9   12
4   8   12  16
```

**请改正程序中的错误，使它能得到正确结果。**

**注意：不要改动** main **函数，不得增行或删行，也不得更改程序的结构。**

```
#include <conio.h>
#define  M 10
int  a[M][M] = {0} ;

/**************found**************/
fun(int **a, int m)
{  int j, k ;
   for (j = 0 ; j < m ; j++ )
       for (k = 0 ; k < m ; k++ )
/**************found**************/
        a[j][k] = k * j ;
}

main ( )
{  int  i, j, n ;
   printf ( " Enter n : " ) ;  scanf ("%d", &n ) ;
   fun ( a, n ) ;
   for ( i = 0 ; i < n ; i++)
   {   for (j = 0 ; j < n ; j++)
         printf ( "%4d", a[i][j] ) ;
       printf ( "\n" ) ;
   }
}
```

## 三、程序设计题

**请编一个函数** fun(int *a,int n,int *odd,int *even)**，函数的功能是分别求出数组中所有奇数之和以及所有偶数之和。形参** *n* **给了数组中数据的个数：利用指针** odd **返回奇数之和，利用指针** even **返回偶数之和。**

**例如：数组中的值依次为：**1**，**8**，**2**，**3**，**11**，**6**；则利用指针** odd **返回奇数之和** 24**；利用指针** even **返回偶数之和** 8**。**

**注意：部分源程序给出如下。**

**请勿改动主函数** main **和其他函数中的任何内容，仅在函数** fun **的花括号中填入所编写的若干语句。**

```
#include <stdio.h>
#include <conio.h>
#define N 20
fun(int *a,int n,int *odd,int *even)
```

```
{

}
main()
{   int a[N]={1,9,2,3,11,6},i,n=6,odd,even;
    clrscr();
    printf("The original data is:\n");
    for(i=0;i<n;i++) printf("%5d",*(a+i));
    printf("\n\n");
    fun(a,n,&odd,&even);
    printf("The sum of odd numbers:%d\n",odd);
    printf("The sum of even number:%d\n",even);
}
```

### 19.1.3 上机考试模拟试题（三）

**一、程序填空题**

**给定程序的功能是：调用函数 fun 将指定源文件中的内容复制到指定的目标文件中，复制成功时函数返回值为 1，失败时返回值为 0。在复制的过程中，把复制的内容输出到终端屏幕。主函数中源文件名放在变量 sfname 中，目标文件名放在变量 tfname 中。**

**注意：部分源程序给出如下。**

**请勿改动主函数 main 和其他函数中的任何内容，仅在下划线上填入所需的内容。**

```
#include   <stdio.h>
#include   <stdlib.h>
int fun(char  *source, char  *target)
{  FILE  *fs,*ft;     char  ch;
/**********found**********/
   if((fs=fopen(source, ___1___))==NULL)
      return 0;
   if((ft=fopen(target, "w"))==NULL)
      return 0;
   printf("\nThe data in file :\n");
   ch=fgetc(fs);
/**********found**********/
   while(!feof(___2___))
   {  putchar( ch );
/**********found**********/
      fputc(ch,___3___);
      ch=fgetc(fs);
   }
   fclose(fs);  fclose(ft);
   printf("\n\n");
   return  1;
```

```
}
main()
{  char  sfname[20] ="myfile1",tfname[20]="myfile2";
   FILE  *myf;     int  i;     char  c;
   myf=fopen(sfname,"w");
   printf("\nThe original data :\n");
   for(i=1; i<30; i++){ c='A'+rand()%25;fprintf(myf,"%c",c);
   printf("%c",c); }
   fclose(myf);printf("\n\n");
   if (fun(sfname, tfname))  printf("Succeed!");
   else  printf("Fail!");
}
```

## 二、程序改错题

**给定程序** MODI1.C **中函数** fun **的功能是：求出以下分数序列的前** n **项之和。和值通过函数值返回到** main **函数。**

$$\frac{2}{1},\frac{3}{2},\frac{5}{3},\frac{8}{5},\frac{13}{8},\frac{21}{13},\cdots$$

**例如，若** $n$=5**，则应输出：**8.391667

**请改正程序中的错误，使它能得到正确结果。**

**注意：不要改动** main **函数，不得增行或删行，也不得更改程序的结构。**

```
#include <stdio.h>

/************found************/
fun (int  n )
{  int  a, b, c, k;  double  s;
   s = 0.0;  a = 2;  b = 1;
   for ( k = 1; k <= n; k++ ) {
/************found************/
     s = s + (Double)a / b;
      c = a;  a = a + b; b = c;
   }
   return s;
}
main( )
{  int   n = 5;
   printf( "\nThe value of  function is: %lf\n",  fun (  n ) );
}
```

## 三、程序设计题

**假定输入的字符串中只包含字母和*号。请编写函数** fun**，它的功能是：除了字符串前导的*号之外，将串中其他*号全部删除。在编写函数时，不得使用** C **语言提供的字符串函数。**

**例如，字符串中的内容为：******A*BC*DEF*G*********，删除后，字符串中的内容应当是：**

****ABCDEFG。

**注意：部分源程序给出如下。**

**请勿改动主函数** main **和其他函数中的任何内容，仅在函数** fun **的花括号中填入所编写的若干语句。**

```
#include <stdio.h>
void  fun( char *a )
{


}
main()
{  char  s[81];
   printf("Enter a string:\n");gets(s);
   fun( s );
   printf("The string after deleted:\n");puts(s);
   NONO();
}
NONO()
{/* 本函数用于打开文件，输入数据，调用函数，输出数据，关闭文件。 */
  FILE *in, *out ;
  int i ; char s[81] ;
  in = fopen("K:\\K01\\24000102\\in.dat","r") ;
  out = fopen("K:\\K01\\24000102\\out.dat","w") ;
  for(i = 0 ; i < 10 ; i++) {
    fscanf(in, "%s", s) ;
    fun(s) ;
    fprintf(out, "%s\n", s) ;
  }
  fclose(in) ;
  fclose(out) ;
}
```

# 19.2 上机考试模拟试题参考答案

## 19.2.1 上机考试模拟试题（一）答案

### 一、程序填空题

（1）x　　（2）n　　（3）t

### 二、程序改错题

（1）**将** void fun (long　s, long t)**修改为** void fun (long　s, long *t)

（2）将 sl = sl*100;修改为 sl = sl*10;

### 三、程序设计题

```
long fun(int n)
{
   int i;
   long s=0;
   for(i=2;i<=n-1;i++)  /*从2~n-1中找n的所有因子*/
     if(n%i==0)
        s+=i*i;          /*将所有因子求平方加*/
   return s;           /*将平方和返回*/
}
```

## 19.2.2　上机考试模拟试题（二）答案

### 一、程序填空题

（1）*s　　（2）1　　（3）s++

### 二、程序改错题

（1）将 fun(int **a, int m)修改为 void fun(int (*a)[M], int m)
（2）将 a[j][k] = k * j;修改为 a[j][k] = (k +1)*(j+1);

### 三、程序设计题

```
fun(int *a,int n,int *odd,int *even)
{   int i; *even=0;*odd=0;
   for(i=0;i<n;i++)          /*一步一步地找元素*/
     if(!(a[i]%2))           /*判断是否是奇数，%运算是求余运算*/
        *even+=a[i];         /*余为0时表示原数为偶数*/
     else
         *odd+=a[i];         /*余为1时表示原数为奇数*/
}
```

## 19.2.3　上机考试模拟试题（三）答案

### 一、程序填空题

（1）"r"　　（2）fs　　（3）ft

### 二、程序改错题

（1）将 fun ( int　n )修改为 double fun ( int　n )
（2）将 s = s + (Double)a / b;修改为 s = s + (double)a / b; 或 s=s+1.0*a/b;

## 三、程序设计题

```
    void  fun( char *a )
    {    int i=0;
         char *p=a;
         while (*p&&*p==’*’)     /*复制串中前导“*”复制到原串*/
         {   a[i]=*p;
             i++
             p++;
         }
         while (*p)              /*将串中和串尾的非“*”字符复制到原串*/
         {
             if(*p!=’*’)
             {   a[i]=*p;
                 i++;
             }
             p++;
         }
         a[i]=’\0’           /*为修改后的字符串赋字符串结尾标记*/
}
```