第 3 章

C 语言程序流程控制

本章要点

- 理解 C 语言程序基本特点。
- 掌握 C 语言程序算法描述方法。
- 掌握结构化程序的三种基本结构。
- 掌握 C 语言程序简单语句、条件语句和循环语句。
- 掌握 C 语言程序的分析与设计算法的过程。

在前面章节中，读者详细地完成了 C 语言最基本知识的学习。从本章开始，将要把上面介绍过的数据类型的知识放到 C 语言程序的流程中进行应用。

C 语言程序是一种结构化程序，共有三种基本结构：顺序结构、选择结构和循环结构。

顺序结构是最简单的一种结构，它按语句出现的先后顺序依次执行。

选择结构是一种分支结构，在选择结构执行时，首先要进行条件判断，当条件成立时执行相关的语句，否则执行其他语句。

循环结构又称为重复结构，循环结构在给定条件成立的情况下，反复执行同一段程序，直到条件不成立为止。

3.1 C 语言程序概述

3.1.1 C 语言程序基本特点

在编写 C 语言程序时，必须满足程序的基本原则。

(1) 确定性：流程中每条语句都是含义必须明确，不能存在二义性。

(2) 有穷性：一个程序所包含的操作步骤必须是有限的。

(3) 有效性：程序中的每一个步骤都应当被执行到，并能得到确定的结果。

(4) 必有输出语句：通过输出了解程序执行的情况及最后的结果。

(5) 输入可有可无：在需要由计算机输入设备输入数据时，则书写输入语句，若不需要输

入数据，可以不写输入语句。

3.1.2　C 语言程序算法

算法的概念：算法就是解决问题的方法和步骤。算法是程序设计的灵魂，在算法中合理运用数据，就实现了程序设计。每一个问题都有属于自己的解决办法。在本章将要介绍几类问题的分析和解决的办法。

C 语言中算法书写有几种常用方法：自然语言、传统流程图和伪代码等方法。

自然语言描述算法的方法一般只适合于比较简单的算法。伪代码是介于自然语言和 C 语言程序之间的一种描述算法的方法，也是一种便捷的描述算法的方法，一般只适合于比较简单的算法描述。

传统流程图采用不同的几何图形来代表不同性质的操作，用流程线来指出算法执行的方向。用流程图表示的算法简单直观，容易转换成相应的程序语言，适合于简单和复杂的算法。

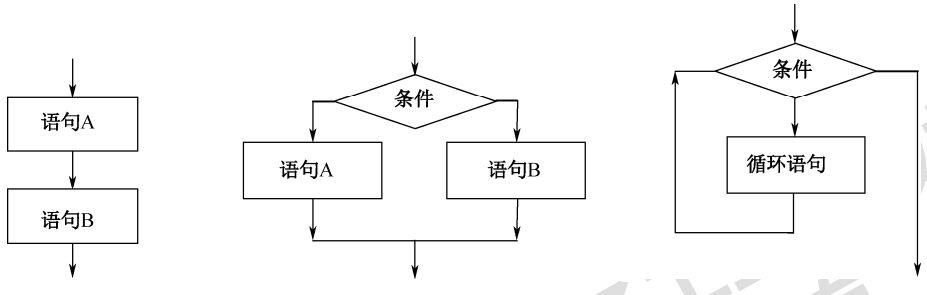
传统流程图表示算法时，常用到的流程图符号见表 3-1。

表 3-1　常用的流程图符号

流程图符号	说　明	作　用
	椭圆形的起止框	表示程序的开始与结束
	平行四边形的输入/输出框	表示程序的输入/输出操作
	矩形的执行框	表示程序的数据处理步骤
	棱形的判断框	表示程序中的条件判断
	圆形的连接点	表示程序中的接口
	流程线	表示程序执行的方向

上述这些描述算法的方法在多种程序设计语言中，可以说是通用的方法。

用传统流程图的方式表示出来的三种基本结构，如图 3-1 所示。



（a）顺序结构　（b）选择结构　（c）循环结构

图 3-1　C 语言的三种基本结构

在编写复杂的算法时，常用的是基本流程结构的组合，而不是单纯的一种基本结构。

3.2 顺序结构程序设计和简单语句

顺序结构的程序按照语句书写的先后顺序依次执行。每一条语句都会执行一次，是一种很简单的程序结构，书写顺序结构程序时一定要注意语句的确定性和有效性。

C 语言程序是由若干条语句构成的，根据这些语句所实现的功能不同，可以把这些语句分为声明语句、表达式语句、复合语句、返回语句、输入语句和输出语句等。

3.2.1 声明语句

声明语句用于定义变量，写在 C 语言程序开始的位置，声明语句的作用是向内存申请足够程序需要的存储单元。C 语言中的变量不经过声明是不允许使用的。一般地，定义变量时完成：

（1）变量数据类型的确定，需要考虑程序中需要什么数据类型变量，需要几种，每种需要几个；

（2）变量名称的确定，根据标识符的命名规则，定义与变量作用贴近的变量名；

（3）给变量赋值，可以将变量的初始值在声明语句中赋给变量。

以上三个作用前两个是必定存在的，第三个可根据程序运行的需要给变量赋初值。

声明语句格式 1：数据类型 变量名 1[，变量名 2，变量名 3，…，变量名 n];

说明

声明若干个变量，数据类型与变量名间用空格分隔，每个变量用逗号分隔，语句以分号结尾。

声明语句格式 2：数据类型 变量名 1=变量值 1[，变量名 2=变量值 2，变量名 3，…，变量名 n];

说明

声明变量的同时给变量赋值。

例如：已知三角形的三条边的长分别是 3 厘米、4 厘米和 5 厘米，编写程序计算三角形的面积。

分析：该例中根据三角形三条边计算三角形的面积的方法适合使用海伦公式，使用该公式时三条边可以分别用常用的变量 a、b、c 来表示，数据类型为 double 类型，它们都有初始值，可以在声明的同时给三个变量赋初值 a=3、b=4、c=5，计算面积的中间值 s=(a+b+c)/2，数据类型为 double 类型; 面积 area 数据类型为 double 类型。该例的声明语句就应该写为：double a=3,b=4,c=5,s,area;

3.2.2 表达式语句

任何表达式后面加分号就成为表达式语句，也可称作赋值语句。

例如：area=sqrt(s*(s-a)*(s-b)*(s-c));/* 计算三角形的面积*/

average=(a+b+c) /3; /* 计算三个数的平均值*/

ch=ch+32; /* 把大写字母转换成小写字母*/

执行表达式语句就是计算表达式的值。

3.2.3　输入语句

输入语句完成的是程序中所需要用户输入数据的操作，前面第 2 章已经全面地学习了标准格式输入语句的写法，而 C 语言还有其他的输入语句。

① 单字符输入函数 getchar();被定义在头文件<stdio.h>中。

作用：从键盘输入单个字符，将键盘输入的字符赋给一个字符变量，构成赋值语句。

格式：字符变量=getchar();

例如：char ch;

ch=getchar();

② 字符串输入函数 gets();被定义在头文件<stdio.h>中。

作用：从键盘输入一个字符串，读到回车符为止，gets()不会将回车符读入到字符串中。

格式：gets(数组名);/*因要输入多个字符，所以单个字符变量是不够用的*/

例如：char ch[10];

gets(ch);

3.2.4　输出语句

输出语句完成的是程序中用户需要输出结果的操作，前面第 2 章已经全面地学习了标准格式输出语句的写法，而 C 语言还有其他输出语句。

① 单字符输出函数 putchar();被定义在头文件<stdio.h>中。

作用：向终端输出单个字符。

格式：putchar(字符变量);

例如：char ch;

ch=getchar();

putchar(ch);

② 字符串输出函数 puts();被定义在头文件<stdio.h>中。

作用：向终端输出一个字符串。

格式：puts(数组名);/*字符串操作以数组为基础*/

例如：char ch[10];

gets(ch);

puts(ch);

例 3-1：输入、输出语句实例。

【参考代码】

```
#include<stdio.h>
void main()
{
    char ch1,ch2[10];
    printf("请输入一个字符串：");
    gets(ch2);
    printf("你输入的字符串为：");
    puts(ch2);
    printf("请输入一个字符：");
```

```
    ch1=getchar();
    printf("你输入的字符为：");
    putchar(ch1);
    putchar('\n');/*输出换行符*/
}
```

【运行结果】

```
请输入一个字符串：abc
abc
请输入一个字符：a
a
```

3.2.5 返回语句

返回语句的作用是在函数被调用执行结束后，返回到被调用的位置。可以是返回主程序，不返回结果，也可以是返回主程序的同时返回一个结果。

返回语句由 return 完成。

格式：return;或 return 结果;

例如：return;return 0;

3.2.6 复合语句

在 C 语言中，把一对大括号{}括在一组语句前后，可以构成复合语句，{}不仅可以用作函数体的开头和结尾标志，也是复合语句的开头和结束标志。复合语句是一个语句块，其形式如下：

{ 语句 1;

语句 2;

……

语句 n }

说明

一个复合语句在语法上被视为一条语句，在一对大括号内的语句数量不限。

在复合语句中不仅可以有执行语句，也可以有定义部分，复合语句声明的变量作用域只在复合语句中，在复合语句之外就不起作用。复合语句中的变量名和主函数中的变量可以相同，但它们不是同一变量，可以认为复合语句中所声明变量是临时的变量。

例 3-2：复合语句综合使用应用实例。

【参考代码】

```
#include<stdio.h>
void main()
{
    int a=3,b=2,c=1;
    printf("[1]: %d, %d, %d\n", a, b, c);
    {
        int b=5;
        int c=12;
        printf("[2]: %d,%d,%d\n",a,b,c);
    }
```

```
        printf("[3] : %d,%d,%d\n ",a,b,c);
    }
```

【运行结果】

```
    [1]: 3, 2, 1
    [2]: 3,5,12
    [3]: 3,2,1
```

【运行结果分析】

1、3 行所得到的结果是全局变量 a,b,c 的值, 2 行所得到的结果是复合语句中局部变量 a,b,c 的值。

除以上所用到的语句外，空语句也是 C 语言程序中的语句，一个语句只有一个分号，这样的语句称作空语句。程序执行空语句时将不产生任何动作。

使用空语句使代码更为严谨对称，提高代码的可读性，延迟程序的执行。

3.3　选择结构程序设计

在 C 语言中有一些问题需要根据不同的情况执行不同的语句结构, 解决这类问题的程序结构称为选择结构。选择结构的执行是依据一定的条件选择执行路径，而不是严格按照语句出现的物理顺序。选择结构的程序设计方法的关键在于构造合适的条件和分析程序流程，应根据不同的程序流程选择适当的分支语句。选择结构适合于带有逻辑或关系比较等条件判断的计算，设计这类程序时往往都要先绘制其程序流程图，然后根据程序流程写出源程序，这样做把程序设计分析与程序书写分开，使得问题简单化，易于理解。

C 语言中的选择结构可分为单分支选择结构、双分支选择结构和多分支选择结构，其流程图如图 3-2 所示。

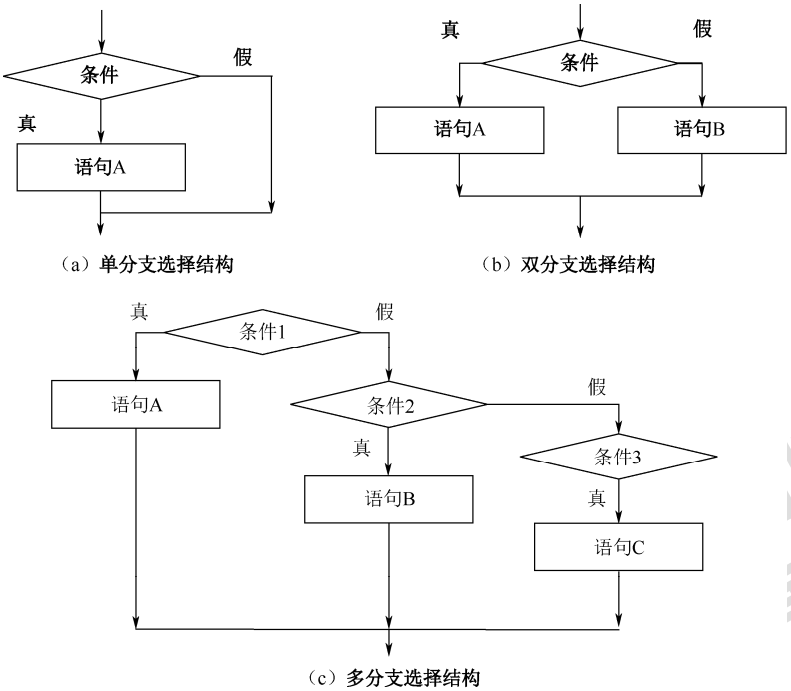


图 3-2　选择结构的流程图

3.3.1 单分支选择结构

单分支选择结构是在顺序结构的程序基础上，解决了做判断后再执行的问题，由 if 结构来实现。

（1）单分支选择结构的格式：

if(条件表达式)

{

语句组；

}

（2）执行过程：先判断条件表达式是否为真，如果是则执行其后的语句组，否则什么都不执行。

说明

如果语句组中的语句数量超过 1 条，必须用{}括起来，否则程序运行结果会不正确。

例 3-3：单分支选择结构应用实例。编写程序从键盘输入两个整数，按从小到大的顺序输出。

【问题分析】图 3-3 所示为单分支选择结构应用实例流程图。

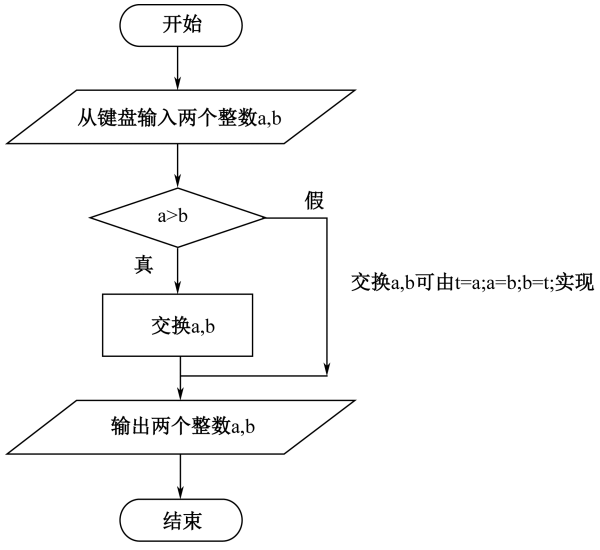


图 3-3　单分支选择结构应用实例流程图

【参考代码】

```
#include<stdio.h>
void main()
{
    int a,b,t;
    printf("请输入两个整数：");
    scanf("%d%d",&a,&b);
    printf("你输入的两个整数是：%5d%5d\n",a,b);
    if(a>b)
    {
    t=a;a=b;b=t;
```

```
        }
        printf("按从小到大的顺序输出: %5d%5d\n ",a,b);
    }
```

【运行结果】

```
请输入两个整数：100 10
你输入的两个整数是：  100   10
按从小到大的顺序输出：   10  100
```

【运行结果分析】

请输入两个整数 100 和 10，a=100，b=10，a>b 为真，则执行语句组交换了变量 a 和 b 的值。

3.3.2　双分支选择结构

双分支选择结构可根据条件判断的真假执行两种不同的操作。由 if…else 结构来实现。

（1）双分支选择结构格式：

if(条件表达式)

{

　　语句组 1；

}else

{

　　语句组 2;

}

（2）执行过程：先判断条件表达式是否为真，如果是则执行语句组 1，否则执行语句组 2。

说明

不管是 if 分支还是 else 分支，如果语句组中的语句数量超过 1 条，必须用{}括起来。否则程序运行结果会不正确。

例 3-4：双分支选择结构应用实例。编写程序从键盘输入两个整数，输出其中的较大数。

【问题分析】如图 3-4 所示为双分支选择结构应用实例流程图。

【参考代码】

```
#include<stdio.h>
void main()
{
    int a,b,max;
    printf("请输入两个整数：");
    scanf("%d%d",&a,&b);
    printf("你输入的两个整数是%5d%5d\n",a,b);
    if(a>b)
    max=a;
    else
    max=b;
    printf("两数中的较大数是: %5d \n ",max);
}
```

【运行结果】

```
请输入两个整数：100 10
你输入的两个整数是  100   10
两数中的较大数是：  100
```

【运行结果分析】

输入两个整数 100 和 10，a=100，b=10，a>b 为真，则执行语句组 1 把 a 的值赋给 max，得到结果较大数是 100。

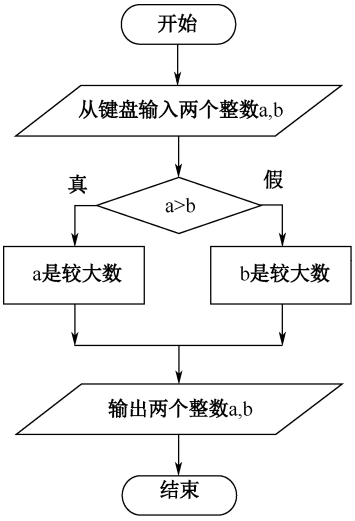


图 3-4　双分支选择结构应用实例流程图

3.3.3　多分支选择结构

1．if 与 if…else 结构

多分支选择结构可以根据多个不同的条件，执行多个不同操作。它是一种较为复杂的选择结构程序。可以使用 if 结构和 if…else…结构的嵌套来实现。

（1）多分支选择结构格式：

```
if(条件表达式 1)
{
    语句组 1;
}else if(条件表达式 2)
{
    语句组 2;
}else if (条件表达式 3)
……
{
语句组 n;
}
else {
        语句组 n+1;
}
```

（2）执行过程：先判断条件表达式 1 是否为真，如果是则执行语句组 1，否则判断条件表达式 2，如果是则执行语句组 2，依次判断到条件表达式 n，如果是则执行语句组 n，否则执行语句组 n+1。

说明

关键字 else 总是向上找离它最近的 if 关键字进行匹配，如果 else 向上没有匹配上，则程序中会出现错误。在 if…else…结构进行嵌套时，一定要注意这种匹配关系。

例 3-5：多分支选择结构应用实例。编写程序，从键盘输入学生百分制成绩，判断成绩等级并输出。（90～100 优秀，80～89 良好，70～79 中等，60～69 及格，0～59 不及格）

【问题分析】如图 3-5 所示为多分支选择结构应用实例流程图。

【参考代码】

```
#include<stdio.h>
void main()
 {
    int score;
    scanf("%d",&score);
    if(score>=90)
    printf("成绩：%d等级优秀！\n", score);
    else if(score>=80)
    printf("成绩：%d等级良好！\n", score);
    else if(score>=70)
    printf("成绩：%d等级中等！\n", score);
    else if(score>=60)
    printf("成绩：%d等级及格！\n", score);
    else
    printf("成绩：%d等级不及格！\n", score);
}
```

【运行结果】

```
请输入你的成绩：90
成绩：90等级优秀！
请输入你的成绩：80
成绩：80等级良好！
请输入你的成绩：70
成绩：70等级中等！
请输入你的成绩：60
成绩：60等级及格！
请输入你的成绩：50
成绩：50等级不及格！
```

【运行结果分析】

以上运行了 5 次程序，从键盘上输入的成绩分别满足了程序中所写的 5 个条件。所以 5 个运行结果是不相同的。

2．switch 结构

多分支选择程序也可以由 switch 结构来实现。

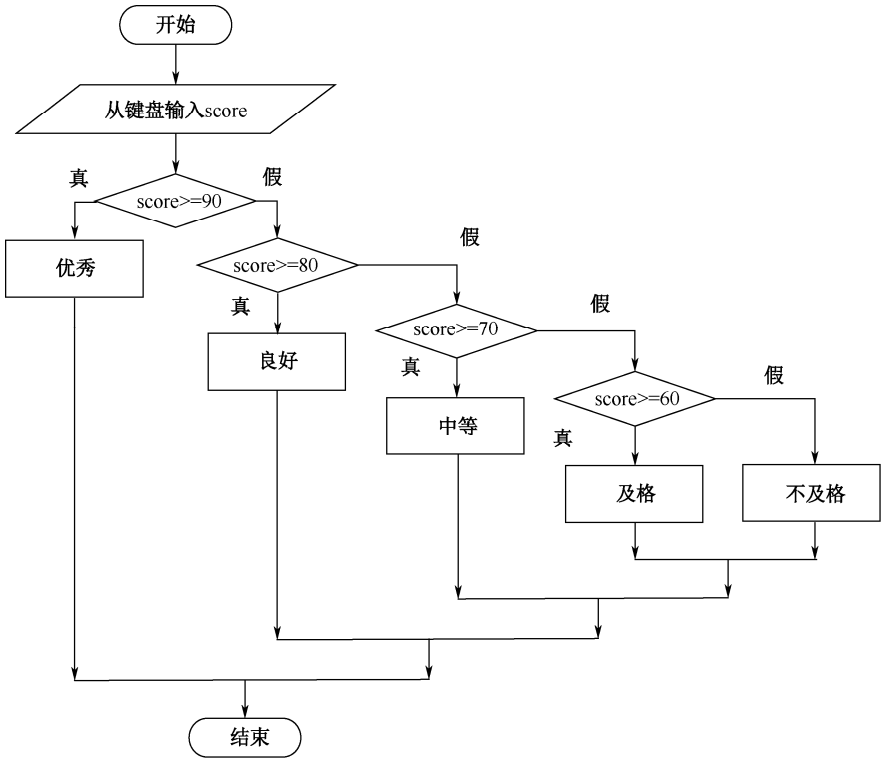


图 3-5　多分支选择结构应用实例流程图

（1）switch 结构的格式：

```
switch(常量表达式)
  {
              case  常量 1:
                    语句 1 或空语句;
              case  常量 2:
                    语句 2 或空语句;
 case  常量 n:
                    语句 n 或空语句;
              default:
                    语句 n+1 或空语句;
  }
```

（2）执行过程：在执行 switch 结构时，将常量表达式的值逐个与 case 后的常量进行比较，若与其中一个相等，则执行该常量下的语句，若没有任何一个常量和常量表达式相等，则执行 default 后面的语句。

【问题分析】图 3-6 所示为 switch 分支选择结构应用实例流程图。

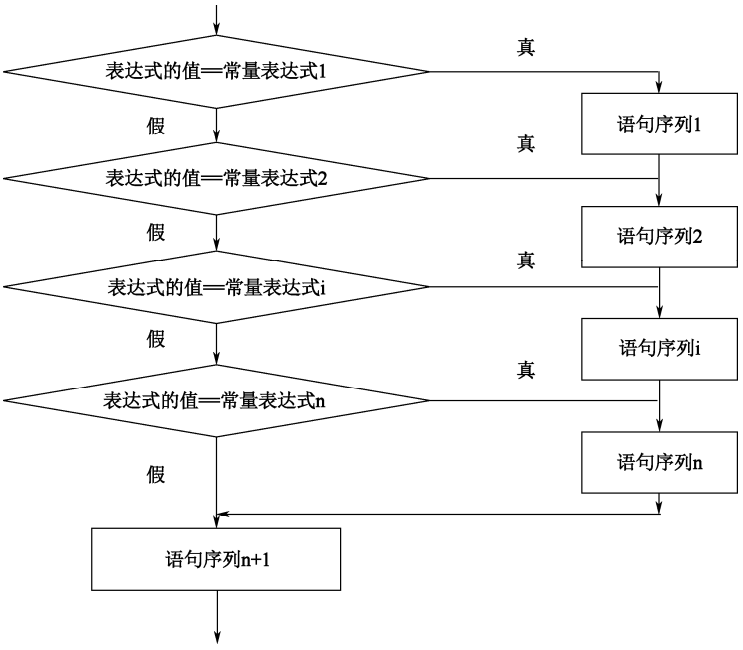


图 3-6　switch 分支选择结构应用实例流程图

说明

switch 后的表达式值可以是整型和字符型数据，不可以是其他类型数值。case 分支的数量不限。default 关键字可以不在 switch 结构中出现。每个 case 或 default 后的语句可以是语句体，但不需要使用"{"和"}"括起来。

例 3-6：根据成绩的等级输出百分制分数段。（90～100 为 A，80～89 为 B，70～79 为 C，60～69 为 D，0～59 为 E）

【参考代码】

```
#include<stdio.h>
void main()
{
    char grade;
    printf("请输入成绩等级：");
    grade=getchar();
    switch(grade)
    {
    case  'A': printf("成绩在90～100\n");
    case  'B': printf("成绩在80～89\n");
    case  'C': printf("成绩在70～79\n");
    case  'D': printf("成绩在60～69\n");
    case  'E': printf("成绩在0～59\n");
    }
}
```

【运行结果】

```
请输入成绩等级：A
成绩在90～100
```

```
成绩在80～89
成绩在70～79
成绩在60～69
成绩在0～59
```

【运行结果分析】

在程序运行时，输入“ A ”输出“ 成绩 90～100 ”后，其他 case 分支之后的语句也随着执行了。这正是 switch 结构的特点，只要一个分支执行，其下面所有分支都要执行。可见这样的结果并不正确，为了解决这个问题，可以在每个分支结束时，添加 break 关键字，在得到正确的结果后，相应分支结束，该分支之后的分支就不再执行了。switch 结构执行原理如图 3-7 所示。

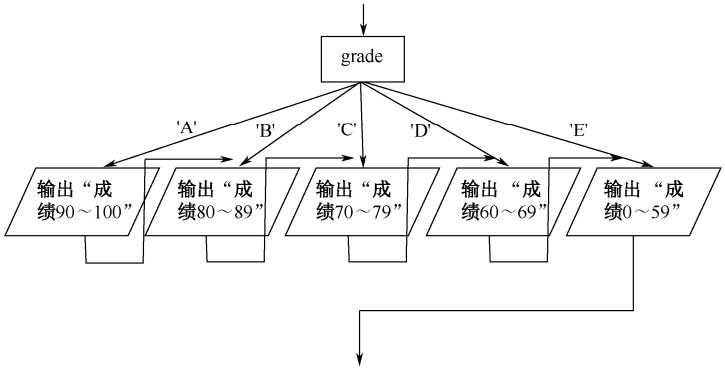


图 3-7　switch 结构执行原理

根据以上 switch 结构执行的特点，例 3-6 的代码可以改写如下，修改过后的程序在运行时就不会再有上述的错误了。

```
#include<stdio.h>
void main()
{
    char grade;
    printf("请输入成绩等级：");
    grade=getchar();
    switch(grade)
    {
    case  'A': printf("成绩在90～100\n");break;
    case  'B': printf("成绩在80～89\n"); break;
    case  'C': printf("成绩在70～79\n"); break;
    case  'D': printf("成绩在60～69\n"); break;
    case  'E': printf("成绩在0～59\n");
    }
}
```

说明

（1）并不是每个 case 后面都必须有语句。在两个或两个以上不同条件要执行相同的操作时，可以节省放在上面的分支里的语句，仅保留最后一个分支里的语句。当程序执行时，从第一次条件匹配的 case 分支开始往下执行，直到语句“break;”处结束 switch 分支结构。

（2）switch 结构像 if…else…结构一样，也可以嵌套实现一些较为复杂的程序。

例 3-7：用 switch 结构编写程序，从键盘输入学生百分制成绩，判断成绩等级并输出。（90～100 优秀，80～89 良好，70～79 中等，60～69 及格，0～59 不及格）

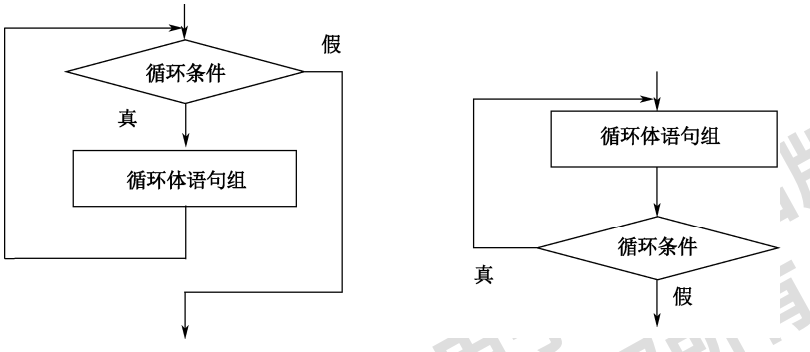
```
#include<stdiio.h>
void main()
{
    int num;
    char grade;
    printf("请输入学生成绩：");
    scanf("%d",&num);
    num/=10;
    switch(num)
    {
        case 10:
        case 9: grade='A'; break;
        case 8: grade='B'; break;
        case 7: grade='C'; break;
        case 6: grade='D'; break;
        default: grade='E';break;
     }
    printf("学生成绩等级%3c \n",grade);
}
```

3.4　循环结构程序设计

循环结构是指在程序中需要反复执行某个功能而设置的一种程序结构。它由循环体中的条件判断是否为真，再决定是否继续执行某个功能还是退出循环。根据判断条件,循环结构又可细分为两种形式：先判断后执行的循环结构和先执行后判断的循环结构。

先判断后执行的循环结构称为当型循环，先判断所给条件是否成立，若成立，则执行循环体；之后再判断条件是否成立；若成立，则又执行循环体，像这样反复，直到某一次循环条件不成立时跳出循环，执行循环结构后面的代码，如图 3-8（a）所示。

先执行后判断的循环结构称为直到型循环，先执行循环体，再判断所给条件是否成立，若成立，则再执行循环体，如此反复，直到循环条件不成立，该循环过程结束，如图 3-8（b）所示。



（a）当型循环流程图　　（b）直到型循环流程图

图 3-8　当型循环和直到型循环的流程图

不管是当型循环还是直到型循环，在解决循环相关的问题时，需要考虑的问题有四个：循环的初始化表示循环的最初状态，以及循环条件、循环体语句组和改变循环条件的语句。可以认为这是循环的四要素，只有全面地处理了这四方面问题，才能设计一个正确的循环结构，才能让循环在得到正确结果后，结束循环。

实现当型循环结构的语句有 while 结构和 for 结构，实现直到型循环结构的语句有 do…while 结构。

3.4.1 while 结构

（1）while 结构格式：

```
while(表达式)
{
        循环体；
}
```

（2）执行过程：先判断表达式是否为真，如果是则执行循环体，再次判断所给表达式是否成立，若成立，则再执行循环体，如此反复，直到循环条件不成立，该循环过程结束。

说明

关键字 while 后的表达式是一个关系表达式或逻辑表达式，需要写在一对小括号里，循环体语句组如果只有一条语句，可以省略前后两端的大括号。

例 3-8：while 结构应用实例，编写程序，计算 1+2+3+…+100 的值。

【问题分析】如图 3-9 所示为 while 结构应用实例流程图。

【参考代码】

```
#include<stdio.h>
void main()
{
    int i=1,sum=0;  // 循环的初始化
    while(i<=100)  //循环条件
    {
    sum=sum+i;  //循环体语句
    i++;    //改变循环条件
    }
    printf("1+2+3+……+100=%d\n",sum);
}
```

【运行结果】

```
1+2+3+…+100=5050
```

【运行结果分析】

在循环变量 i<=100 时，循环执行，反复将 i 的值累加到变量 sum，通过 i++得到 1～100 之间的所有整数，最后结果得出 5050。

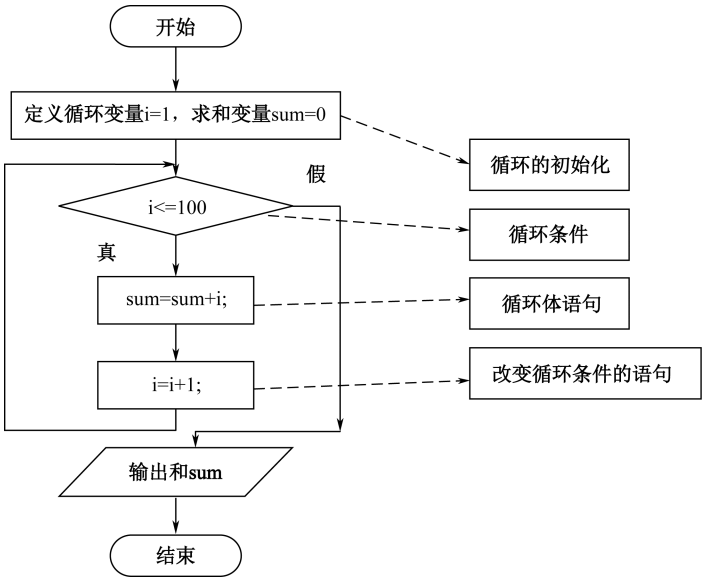


图 3-9　while 结构应用实例流程图

说明

while 循环一定要在循环体中改变循环变量，否则会因条件永远满足而无限循环下去，while(循环条件)后不能加“;”，否则会把真正的循环体与条件分开，也会造成程序错误。

3.4.2　for 结构

（1）for 结构格式：

```
for (表达式 1; 表达式 2; 表达式 3)
{
        循环体；
}
```

（2）执行过程：for 结构也是一个当型循环结构，在 for 结构执行时，可按照以下次序执行：

① 执行表达式 1；
② 判断表达式 2 的值是否为真；
③ 如果是真则执行循环体；
④ 执行表达式 3；
⑤ 再次判断表达式 2 是否成立，若成立，则在②③④反复执行；
⑥ 直到循环条件不成立，该循环过程结束。

说明

for 结构中表达式 1 所起到的作用是循环的初始化，表达式 2 所起到的作用是循环条件，表达式 3 所起到的作用是改变循环变量，使循环条件可以在适当处不成立，结束循环。

例 3-9：for 结构应用实例，编写程序，计算 1+2+3+…+100 的值。

【参考代码】

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

```
void main()
{
    int i,sum=0;
    for(i=1;i<=100; i++)
    sum=sum+i;  //循环体语句
    printf("1+2+3+……+100=%d\n",sum);
}
```

【运行结果】

```
1+2+3+…+100=5050
```

【运行结果分析】

表达式 1“ i=1 ”完成了循环变量的初始化，表达式 2“ i<=100 ”规定循环执行的范围，循环体语句“ sum=sum+i ”在反复执行过程中得到最终的结果，表达式 3“ i++ ”改变循环变量。

说明

（1）for 结构中的三个表达式一定以“;”分隔，且该行结尾不能点“;”，否则会造成循环体与 for 结构的分离，程序运行不能得到正确的结果。如果循环体语句组只有一条语句，可以省略前后两端的大括号，语句数量超过一条，前后必须添加大括号。

（2）for 结构中的表达式 1 可以缺省，该表达式可以写在 for 结构之前，后面的“;”不能省略，不会影响程序执行。表达式 3 可以缺少，可以写在循环体语句组中，也可以实现改变循环变量。

经过修改后的例 3-9 代码可写为：

```
#include<stdio.h>
void main()
{
    int i=1,sum=0;
    for(;i<=100;)
    {
    sum=sum+i;  //循环体语句
    i++;
    }
    printf("1+2+3+……+100=%d\n",sum);
}
```

3.4.3 do…while 结构

do…while 结构是直到型循环，这种循环会先执行循环体，再判断循环条件。当循环条件不成立时，循环体将被无条件执行一次。

（1）do…while 结构格式：

do{
循环体；
}while(表达式);

（2）执行过程：在 do…while 结构执行时，可按照以下次序执行：

① 执行循环体；

② 判断循环条件是否为真；

③ 如果条件为真，则在①②之间反复执行；

④ 如果条件为假，结束循环，执行循环之后的语句。

说明

do…while 结构需要两个关键字，do 关键字在上循环体包括在 do 后的一对大括号里，后面由 while 带条件表达式，最后的标点符号“;”不能省略。在 do…while 结构中也包括前面的循环四要素。

例 3-10：do…while 结构应用实例，编写程序，计算 1+2+3+…+100 的值。

【参考代码】

```
#include<stdio.h>
void main()
{
    int i=1,sum=0;
    do{
    sum=sum+i;  //循环体语句
    i++;
    }while(i<=100);
    printf("1+2+3+……+100=%d\n",sum);
}
```

【运行结果】

```
1+2+3+…+100=5050
```

【运行结果分析】

在 do…while 结构中，循环初始化，循环体及改变循环条件的语句与上面两例完全一样，只是循环的条件被放到了循环结构的最后。

例 3-11：do…while 结构应用实例，编写程序，从键盘输入若干字符，统计大写字母个数，小写字母个数，数字字符个数和其他字符个数。

【参考代码】

```
#include<stdio.h>
void main()
{
    char c;
    int n1,n2,n3,n4;
    n1=n2=n3=n4=0;
    printf("请输入字符：");
    do{
    c=getchar();
    if(c>='A'&& c<='Z')
    n1++;
    else if(c>='a'&& c<='z')
    n2++;
    else if(c>='0'&& c<='9')
    n3++;
    else
    n4++;
    }while(c!='\n');
```

```
        printf("大写字母个数：%d  ",n1);
        printf("小写字母个数：%d  ",n2);
        printf("数字字符个数：%d  ",n3);
        printf("其他字符个数：%d\n",n4-1);
    }
```

【运行结果】

```
    请输入字符：ZAQXzaqx1234+_*/
    大写字母个数：4  小写字母个数：4  数字字符个数：4  其他字符个数：4
```

【运行结果分析】

（1）例 3-11 是一个统计数字的循环，语句“ n1=n2=n3=n4=0; ”用于给四种类型的字符计数器变量置 0。

（2）在 do…while 循环的循环体中，由语句“ c=getchar(); ”接收每一次循环从键盘输入的一个字符。在程序运行时输入了一串字符“ ZAQXzaqx1234+_*/ ”以回车结束。

（3）用 if…else 的嵌套判断字符属于大写字母、小写字母、数字字符还是其他字符。相应的计数器变量计 1 个数。

（4）回车符被读入字符变量 c，循环条件“ c!='\n' ”不成立，结束循环。

（5）变量 n1,n2,n3 直接输出。输入的最后一个字符是回车符，被计入在其他字符之中，其他字符的个数“ n4-1 ”是字符串中的其他字符个数。

说明

（1）有些程序是为了求多个数字之和，称为求和类型循环。有些程序是为了统计出限制条件的数字或字符的个数，称为计数类型循环。

（2）例 3-11 是 do…while 循环实现的计数类型循环，读者可以将该例转换为用 while 循环或 for 循环来实现。比较 while 循环、for 循环和 do…while 循环分别更加适用于什么样的问题。

3.4.4 循环控制语句

认识了上述三种实现循环的基本语句结构，而关于循环结构的程序还经常出现其他的关键字。分别是 break、continue 和 goto。下面分别了解一下这些关键字在循环中的作用与应用。

（1）使用 break 语句可以退出整个循环，前面在 switch 结构中使用过 break，知道了 break 可以结束所在分支。将 break 语句写在循环体中，当执行到 break 语句时，可以结束所在的循环结构，程序继续执行循环结构之后的语句。break 语句的使用方法是点“ ;”单独成为一条语句，配合 if 语句使用。

例 3-12：break 语句应用实例，编写程序输出数字。

【参考代码】

```
    #include<stdio.h>
    void main()
    {
        int i;
        for(i=0;i<10;i++)
        {
        printf("%5d",i);
```

```
        if(i==5)break;
        }
        printf("\n");
    }
```

【运行结果】

```
        0    1    2    3    4    5
```

【运行结果分析】

从例 3-12 “ for(i=0;i<10;i++) ”中可以看出，该程序将会执行 10 次，输出从 0～9 之间的 10 个数字，因在循环体结构中有“ if(i==5)break; ”语句，在 i 的值为 5 时，break 执行，结束了循环，所以最终的结果输出的是 0～5 之间的数字。

（2）使用 continue 语句退出当前循环进行下一轮循环。当语句 continue 执行时，可以结束正在执行的这一次循环，进入下一次循环的执行，而执行 continue 语句时，并不是真正结束循环。continue 语句的使用方法也是点“ ;”单独成为一条语句，配合 if 语句使用。

例 3-13：continue 语句应用实例，编写程序输出 0～100 以内所有的偶数。

【参考代码】

```
    #include<stdio.h>
    void main()
    {
        int i;
        for(i=0;i<=100;i++)
        {
        if(i%2!=0)  continue;
        printf("%3d ",i);
        }
        printf("\n");
    }
```

【运行结果】

```
        0   2   4   6   8  10  12  14  16  18  20  22  24  26  28  30  32  34  36  38
40  42  44  46  48  50  52  54  56  58  60  62  64  66  68  70  72  74  76  78  80  82
84  86  88  90  92  94  96  98  100
```

【运行结果分析】

从例 3-13 中可以看出，在循环体结构中有“ if(i%2!=0) continue; ”的语句，使得所有的奇数经过 if 后的条件判断后，continue 语句都执行，而排列在循环体之后的“ printf("%3d ",i); ”在当次循环中不再执行，因此所有的奇数没有输出，输出的全是偶数。

（3）goto 语句称为无条件转向语句，通常用作 goto 语句跳转目标。goto 语句的作用是把程序的执行转到标号所在的位置，C 语言中可以在任何语句前加上语句标号，这个语句标号必须和 goto 语句同在一个函数内。

其一般形式为：goto 语句标号;

说明

（1）标号是合法的标识符，在标识符后面加一个冒号，该标示符就成了一个语句标号。标号可以和变量名同名。

（2）语句标号可以出现在 goto 的前面，也可以出现在 goto 的后面。

例 3-14：goto 语句应用实例。

```
#include <stdio.h>
void main()
{
    line1:printf("%d\n",1);
    goto line1;
}
```

【运行结果】

```
无限次地执行输出语句，输出1。
```

【运行结果分析】

在例 3-14 中，因为“ goto line1; ”使程序反复回到 line1 所标识的行反复执行。

3.4.5 循环的嵌套结构

循环结构是执行效率很高的一种程序基本结构，某些重复执行的算法问题，只用一个基本的循环结构是没有办法实现的，需要使用两个循环语句才能够实现，如下图形的输出。

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

循环结构的循环体里还有一个循环结构，这样的循环称为循环的嵌套。在实现循环嵌套时，也需要考虑循环结构的四个基本要素。

例 3-15：循环的嵌套结构应用实例，编写程序，输出 9×9 乘客法口诀表。

```
1×1=1
1×2=2   2×2=4
1×3=3   2×3=6   3×3=9
1×4=4   2×4=8   3×4=12  4×4=16
1×5=5   2×5=10  3×5=15  4×5=20  5×5=25
1×6=6   2×6=12  3×6=18  4×6=24  5×6=30  6×6=36
1×7=7   2×7=14  3×7=21  4×7=28  5×7=35  6×7=42  7×7=49
1×8=8   2×8=16  3×8=24  4×8=32  5×8=40  6×8=48  7×8=56  8×8=64
1×9=9   2×9=18  3×9=27  4×9=36  5×9=45  6×9=54  7×9=63  8×9=72  9×9=81
```

【问题分析】如图 3-10 所示为循环的嵌套结构流程图。

【参考代码】

```
#include <stdio.h>
void main()
{
    int row,col;
    for(row=1;row<=9;row++)
    {
    for(col=1;col<=row;col++)
    {
    printf("%d*%d=%d\t",col,row,row*col);
```

```
        }
        printf("\n");
        }
    }
```

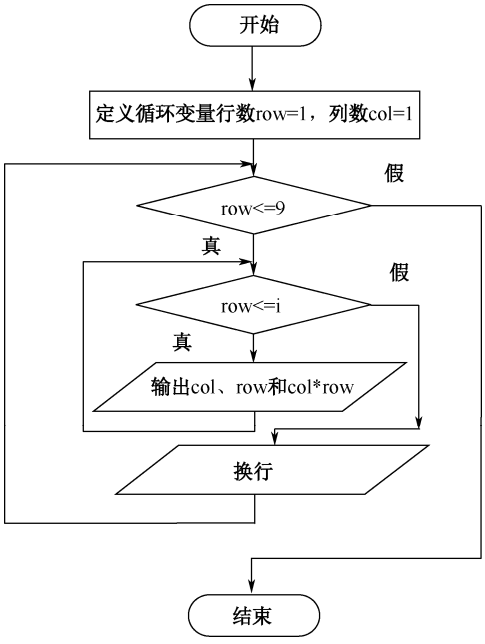


图 3-10　循环的嵌套结构流程图

【运行结果分析】

在例 3-15 中，因为有两列数字是有规律变化的，并且第二列从 1 到 9 都要和第一列数字相配合，所以在实现该算法时需要写一个循环嵌套的结构，外层的循环条件是 row<=9，决定了最终的运行结果是 9 行，内层的循环条件是 col<=row，决定了每行算式的个数等于行数。

3.5　常见错误分析与解决

（1）C 语言程序的基础是语句，在编写程序时，C 语言的语句经常会出现的问题有：

① 语句以“ ; ”结尾，一行内可写多条语句，每条语句都要写“ ; ”结尾。一条语句可以写在多行，不需要续行符。

② 复合语句的定义的变量只能在复合语句的大括号范围内使用，超出大括号，就不能再使用了。

③ 在 if、else、switch、do、for 关键字之后的语句如果超过一条，需要括在{}里。

（2）选择结构和循环结构流程需要一些关键字来实现，C 语言程序流程设计常会发生的错误有：

① 关键字要写小写形式，不要写大写形式。

② if 和 while 关键字后面的一对小括号里要写条件，条件一般是由关系表达式或逻辑表达式组成的，不要写一些连续的不等式，如“ 10<=x<=100 ”，类似的这种写法是错误的，正确的表示形式为“ x>=10&&x<=100 ”。

③ 在 if（条件表达式）和 while（条件表达式）后面写“ ; ”，VC++ 6.0 会认为，条件表达

式是真时，执行空语句“ ; ”。在 if…else 结构中，else 向上找不到 if，VC++ 6.0 会报错“ warning C4390: ';' : empty controlled statement found; is this the intent? ”和“ error C2181: illegal else without matching if ”。

④ 循环结构中条件表达式的值永远是“ 1 ”，循环就是一个死循环，程序不会有正确的结果。

⑤ for 循环的正确语法是 for（表达式 1;表达式 2;表达式 3），每个表达式之间用“ ; ”分隔，不能写其他常用的分隔符。

本 章 小 结

本章是编写 C 语言程序的应用章节，内容多为理解与实践并重的，为程序设计方法奠定了基础，是学习程序设计类课程的重要环节。

本章内容，首先介绍了 C 语言程序的特征与三种结构化程序算法的表示方法。

在顺序结构程序设计里详细地介绍了 C 语言中的语句，包括声明语句、表达式语句、输入语句、输出语句、返回语句、复合语句等。是培养书写 C 语言程序最基本的基础。

在选择结构程序设计里详细地介绍了三种结构的分支结构，以及实现三种分支结构的流程控制。这部分内容是程序设计的第一次深入与提升。

在循环结构程序设计里介绍了当型循环和直到型循环结构的四个重要元素。通过实例分析了在重复执行操作时，如何从四个方面分解问题。

通过三种结构化（顺序结构、选择结构、循环结构）程序流程的实现，完善了编程所应具备的最基本知识，实现了完成程序基本结构设计的目的。

通过本章学习重点理解和掌握的知识有:

（1）编写 C 语言程序遵守的基本原则。

（2）C 语言程序算法描述方法。

（3）分析问题的方法和步骤。

（4）顺序结构、选择结构、循环结构三种结构化程序的实现。

思考与练习

习题 3

一、填空题

1．结构化程序设计中的三种基本结构是顺序结构、分支结构和_____结构。

2．若有语句“ int a=3,b=8; ”，则“ printf("a=%%d,b=%%d\n", a, b); ”执行结果是_____。

3．若有语句“ printf("%5.3f\n",123456.12345) ;”，执行后输出窗口得到的结果为_____。

4．若有语句“ int i,j,k; ”，则表达式“ i=10,j=20,k=30,k*=i+j ”的值为_____。

5．若有语句“ char c='A'+1; ”，则“ printf("c=%c\n",c); ”执行后输出窗口得到的结果为_____。

二、选择题

1．设整型变量 x,y 和 z 的值依次为 3,2 和 1，则下列程序段的输出结果是（　　）。

```
if(x>y)x=y；if(x>z)x=z；
printf("%d,%d,%d\n",x,y,z)；
```

A．1,1,1　　B．1,2,1　　C．1,2,3　　D．3,2,1

2．设运行下列程序时，从键盘输入 ABC 和回车键，程序输出结果是（　　）。

```
#include <stdio.h>
void main()
{
    char c;
    c=getchar();
    if ( c>='a'&&c<='z'‖c>='A'&&c<='Z')
    printf("%c是英文字母\n",c)；
    else if (c>='0'&&c<='9')
    printf("%c是数字字符\n",c)；
}
```

A．A 英文字母　　B．A 数字字符　　C．ABC　　D．65

3．最适合解决选择结构“若 x>0，则 y=1；否则 y=0；”的语句是（　　）。

A．switch　　B．嵌套的 if-else　　C．if-else　　D．if

4．执行下列语句后，整型变量 x 的值是（　　）。

```
switch (x=1) {
    case 0:x=10;break;
    case 1:
    switch (x=2)
     {
    case1:x=20;break;
    case2:x=30;
    }
}
```

A．30　　B．20　　C．10　　D．1

5．两次运行下面的程序，如果从键盘上分别输入 6 和 4，则输出的结果分别是（　　）。

```
#include <stdio.h>
void main()
{
    int x;
    scanf("%d", &x)；
    if(x>)5
    printf("%d",x)；
    else
    printf("%d\n", x--)；
}
```

A．7 和 5　　B．6 和 3　　C．7 和 4　　D．6 和 4

6．以下程序段的输出结果是（　　）。

```
#include <stdio.h>
void main()
{
    int x=3;
    do{
    printf("%3d",x-=2);
    }while(！(--x));
```

```
}
```

A．1　　B．3 0　　C．1 -2　　D．死循环

7．定义变量： int n=10;则下列循环的输出结果是（　　）。

```
while(n>7)
    {n--；
    printf("%d   ", n)；
}
```

A．109　8　7　　B．9　　C．10　　D．9

8．以下程序的输出结果是（　　）。

```
#include <stdio.h>
void main()
    {int n=4；
    while(n>0)
    {
    n--;
    printf("%d ", n)；
    }
}
```

A．2 0　　B．3 1　　C．3 2 1 0　　D．2 1 0

9．在下列选项中，没有构成死循环的程序是（　　）。

A．int i =100;
while(1)
{
i=i%3;
if (i>100)
break;
}

B．for (； ；)；

C．int k=1000;
do {
k--;
} while(k>1000)；

D．int s=36;
while (s);
--s;

10．执行下列语句后，变量 k 的值是（　　）。

```
for (k=0;k<=5;k++) do k++;
while (k<5);
```

A．5　　B．6　　C．7　　D．8

11．下列程序的输出结果是（　　）。

```
#include <stdio.h>
void main()
{
   int m,n;
```

```
    for (m=11;m>10;m--)
    { for (n=m;n>9;n--) if (m%n) break;
    if (n<=m-1) printf ("%d",m);
    }
}
```

A．11　　B．9　　C．7　　D．8

12．以下说法中正确的是（　　）。

A．continue 和 break 只能用在循环体中

B．continue 只能用在循环体中

C．break 只能用在循环体中

D．continue 只能用在循环体外

三、程序填空题

1．以下程序运行时输入 100<CR>后，执行的结果是_____。

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

2．下列程序是求 $ax^2+bx+c=0$ 的实根（设 $b^2-4ac\geq 0$），程序中缺少的内容是_____。

```
#include _________
void main()
{
    float a,b,c,d,x1,x2;
    scanf("%f,%f,%f",&a,&b,&c);
    d=b*b-4.0*a*c;
    if(____________)
    printf("x1=x2=%f\n",-b/(2*a));
    else
    {
    x1=(-b+sqrt(d))/(2.0*a)；
    x2=(-b-sqrt(d))/(2.0*a);
    printf("x1=%f x2=%f\n",x1,x2);
    }
}
```

3．运行下列程序的输出结果是_____。

```
#include <stdio.h>
void main()
{
    int n=1;
    switch (n--)
    {
    case 0 : printf ("%d",n);
    case 1 : printf ("%d",n);
    case 2 : printf ("%d",n);
```

```
    }
}
```

4．执行以下程序，输出结果是_____。

```
#include <stdio.h>
void main()
{
    int num= 0;
    while(num<=2)
    {
    num++;
    printf("%d\n",num);
    }
}
```

四、编程题

1．编写程序，定义整型变量 a=5，定义实型变量 b=1.2345，定义字符型变量 c='A'。在输出窗口显示内容：

a=□□□□□5，b= 1.235

a+b=6.2

c='A',ASCII 值为 65

2．输入一个整数，判断这个整数能否被 3 整除，输出提示。

3．编写程序，从键盘输入三个数字，按从小到大的顺序输出这三个数字。

4．编写程序，计算个人所得税。

① 月工资 s 在 3500 元以下(s<=3500)，不用纳税；

② 月工资 s 在 3500 元到 5000 元之间(s>=3500&& s<5000)，纳税比率 3%；

③ 月工资 s 在 5000 元到 9000 元之间(s>=5000&& s<9000)，纳税比率 10%；

④ 月工资 s 在 9000 元以上(s>=9000)，纳税比率 20%。

5．编写程序，求 1～20 之间所有 3 的倍数之积。

6．编写程序，sum=1+11+111+1111+11111。

7．已知全班 40 个学生的计算机课的考试成绩，求全班的平均成绩。

8．编写程序，输出水仙花数。（水仙花数如：371=33+73+13）

9．编写程序，解决百钱百鸡问题。

实训 3：C 语言结构化程序设计练习

实验学时：4 学时

实验目的：

1．掌握 C 语言中各种语句的写法，顺序结构程序的分析与设计方法。

2．掌握关系表达式与逻辑表达式的表示方法，选择结构语句及程序的分析与设计。

3．掌握循环结构四个要素的设计方法、循环语句、循环流程控制，循环和循环嵌套程序的分析与设计。

操作内容：

1．编写程序，从键盘输入圆锥体的半径 r 和高度 h，计算其体积。

2．编写程序，从键盘输入一个大写字母，转换成相应的小写字母输出。

3．编写程序，从键盘输入矩形长和宽，计算矩形面积。

4．编写程序，从键盘输入一个 double 类型的数字，对小数点后第三位做四舍五入处理，输出处理后的结果。

5．编写程序，从键盘输入一个三位整数，把三个数字逆序组成一个新数，再次输出。

6．编写程序，从键盘输入两个数字，输出两个数字中的大数。

7．编写程序，从键盘输入三角形的三条边，如果能构成三角形，则求三角形面积，否则输出提示不能构成三角形。

已知三角形的三条边 a，b，c

s=(a+b+c)/2，area=sqrt(s*(s-a)*(s-b)*(s-c))

sqrt()在 VC++6.0 中的“ math.h ”头文件，函数原型为“ double sqrt(double); ”。

8．编写程序，输入年份，判断是否是闰年。非整百的年份能被 4 整除的为闰年（如 2004 年就是闰年，2100 年不是闰年）。整百的年份能被 400 整除的是闰年（如 2000 年是闰年，1900 年不是闰年）。

9．编写程序，从键盘输入当前月利润 I，求应发奖金总数。企业发放的奖金根据利润提成。利润 I 低于或等于 10 万元的，奖金可提 10%；利润高于 10 万元，低于 20 万元（100000< I<=200000）时，低于 10 万元的部分按照 10%提成，高于 10 万元的部分，可提成 7.5%；当 200000<I<=400000 时，低于 20 万元的部分仍按上述办法提成（下同），高于 20 万元的部分按 5%提成；当 400000<I<600000 时，高于 40 万元的部分按 3%提成；当 600000<I<=1000000 时，高于 60 万的部分按 1.5%提成；当 I>1000000 时，超过 100 万元的部分按 1%提成。

10．编写程序，做一个猜数游戏。（练习选择结构程序设计）

```
#include<stdio.h>
#include<stdlib.h>
void main()
{
    int guess,magic;
    magic=rand();
    printf("请输入你猜的数：");
    scanf("%d",&guess);
    if(guess==magic)
    printf("猜中了！\n");
    else
    printf("没有猜中！\n");
}
```

11．编写程序，从键盘输入整数 n，求 n 的阶乘 n!(n!=1*2*…*n)。

12．编写程序，从键盘上输入一组数（以输入 0 为结束），求累加和，并找出最大/小值。

13．编写程序，输出从 100 和 200 之间不能被 3 整除的数，要求每行输出 8 个数。

14．编写程序，从键盘上输入数字，判断 m 是否为素数。

15．编写程序，统计 100 以内的全部素数个数并逐个输出。

16．编写程序，实现钻石形状图形的输出。

实验总结

1. 问题总结

2. 收获总结