关于程序的控制结构

4.1 语句及程序框架

4.1.1 语句的基本分类

语句是程序的基本组成部分，一段程序或者一个函数就是由若干语句按照算法规定的逻辑关系组成的。各种表达式都要以语句的形式出现在程序中。C语言中的语句有如下类型：

(1)变量定义语句；

(2)表达式语句；

(3)复合语句；

(4)函数调用语句；

(5)控制语句；

1. 变量定义语句

语法格式如下：

〈类型标识符〉　〈变量表〉；

其含义是：告诉系统，变量表中列出的一个或多个变量的存在，同时要求系统为每个变量分配存储空间，其大小和类型与<类型标识符>所规定的一致。变量定义语句不需要与操作步骤对应。例如：

```
char  aChar;
int  total = 0;  //用于累计素数的总数
int  score;
double  x,  y;
```

2. 表达式语句

语法格式如下：

〈表达式〉;

从上述语法格式可以看出，表达式语句是由前面章节提及的各种表达式及后缀的分号构成。其作用是：一旦表达式语句被执行时，就要按照表达式的求值顺序计算出表达式的值。

例如：

```
i++;
```

3. 复合语句

语法格式如下：

{〈语句1〉〈语句2〉…〈语句n〉}

其作用是：告诉编译系统，将多个语句看成是一个整体，在语法要求上相当于一个语句。因此，在分支和循环结构中经常使用，函数的定义体也是一个复合语句。

例如：

```
{
    int  x, y = 20;
     x = y + 10;
     printf( “%d; %d”, x, y );
}
```

4. 函数调用语句

语法格式如下：

〈函数名〉(〈实际参数表〉);

函数调用语句的作用是：将执行控制流程转移到与此语句完全匹配的函数定义体的第一条语句处，开始执行；函数体执行结束后，返回到此语句的下一条语句处继续执行。执行流程如教材图4-1.

5. 控制语句

就是用于控制程序执行路径的相关语句，可以实现程序的各种控制结构。C语言中包含有九种控制语句。分成三类介绍如下：

（1）分支语句：if语句、switch语句；

（2）循环语句：do-while语句、while语句、for语句；

（3）跳转语句：break语句、continue语句、goto语句、return语句。

分支和循环语句是程序中经常使用的，灵活熟练掌握其用法是本章的主要任务，后面会详细讲解。

4.1.2 C程序框架

一个C程序由若干源文件和头文件组成，称之为一个工程。头文件包括用户自定义的和系统定义的，这两种都是由若干函数及全局量的声明和编译预处理组成，文件名一般是*.h。系统定义的如stdio.h, stdlib.h, string.h, math.h, limits.h等，其内容可以在本地安装有C编译器的机器中找到。一般在安装路径的包含文件夹内，如：\Program Files\Microsoft Visual Studio\VC98\Include。

4.2 分支结构

分支结构与顺序结构相对应，可以称之为判断结构或选择结构，也就是有选择地执行某些语句，改变了完全顺序执行的程序结构。在日常生活中的表达方式是：如果…；那么…。在计算机领域的基本表达方式是：如果条件成立，则执行操作1；否则条件不成立，则执行操作2。其中，操作1，操作2可以是一条或多条语句。

分支结构包含if和switch两个语句.

4.2.1 if语句的基本格式

语法格式如下：

```
if （<条件表达式>）
  复合语句1
else
  复合语句2
```

if 语句基本格式的语义是：如果条件表达式的值为真，则执行复合语句1，否则，即条件表达式的值不为真，则执行复合语句2。格式中“（<条件表达式>）”写法与“（<条件表达式> != 0 ）”写法等价。

【例4.1】 任意输入两个不同的整数，输出其中较大的一个。

```
#include<stdio.h>
void  main( )
  {
c4:      int   x, y;
c5:      printf( "\n Input two numbers: \n" );
c6 :     scanf( "%d%d", &x, &y );
c7 :     if  ( x > y )
c8 :       {  printf( "max = %d \n", x );  }
         else
c10:       {  printf( "max = %d \n", y );  }
   return ;
  }
```

使用if语句需要注意如下内容：
(1)从if 关键词开始到复合语句2结束，在语法上就是一条语句，虽然其中包含着两个复合语句，这并不矛盾。
(2) if语句中的两个复合语句都可以由空语句，一个语句或者多个语句构成。仅包含空语句或一个语句时，一对大括号可以省略。此处增加大括号的好处是可以减少不必要的语法错误。
(3) if语句中的条件表达式必需使用一对匹配的圆括号括起来。
(4) 条件表达式可以是逻辑表达式，关系表达式，算术表达式及赋值表达式等,单个变量构成的表达式也是正确的。只要表达式的值不是0值，条件即为成立。

【例4.2】 判断某年份是否是闰年。

满足以下两个条件之一的年份均是闰年：一是年份能够被4整除，但不能被100整除；二是能被400整除的年份。

4.2.2　if语句的第二种格式

语法格式如下：

if （<条件表达式>）

复合语句

if 语句第二种格式的语义是：如果条件表达式的值不为真，则直接执行此if语句的下一条语句，否则，条件表达式的值为真，则执行其后的复合语句，复合语句执行完成后同样要执行if语句的下一条语句。

【例4.3】 任意输入两个不同的整数，输出其中较大的一个。

```
#include   <stdio.h>
   void   main( )
   {
c4:     int   x, y, max;
c5:     printf( "\n Input  two  numbers: \n" );
c6:     scanf( "%d%d", &x, &y );
c7:     max = x;
c8:     if  ( y > x )
c9:        {  max = y;  }
c10:    printf( "max = %d \n",  max );
c11:    return ;
  }
```

4.2.3 if语句的嵌套形式

可以写出如下几种嵌套形式的if语句：

（1）在第二种格式的复合语句处嵌套第二种格式，形如：

```
if （<条件表达式1>）
  {   其它语句1
    if （<条件表达式2>）
      复合语句2
    其它语句2
   }
```

（2） 第二种格式嵌套第一种格式，形如：

```
if （<条件表达式1>）
 {
   if （<条件表达式2>）
       复合语句21
   else
         复合语句22
  }
```

（3） 第一种格式嵌套第二种格式，形如：

```
if （<条件表达式1>）
 {
   if （<条件表达式2>）
     复合语句2
 }
else
 {
   if （<条件表达式3>）
     复合语句3
 }
```

（4） 第一种格式嵌套第一种格式，形如：

```
if （<条件表达式1>）
 {
    if （<条件表达式2>）
        复合语句21
    else
         复合语句22
  }
else
 {
 if （<条件表达式3>）
         复合语句31
 else
         复合语句32
  }
```

【例4.4】 要求按任意顺序从键盘输入三个整数，编写程序完成输出最大值和最小值的算法。

基本思路：首先任取其中两个数，判断最大和最小；使用例4.3的方法。其次，用第三个数分别与刚刚得到的最大和最小数再次比较，就可以得到三个数中的最大和最小。

4.2.4 if语句的第三种格式

其语法格式如下：

```
if （<条件表达式1>）
     复合语句1
else   if （<条件表达式2>）
            复合语句2
else   if （<条件表达式3>）
            复合语句3
……
else   if （<条件表达式m>）
            复合语句m
else
       复合语句m+1
```

【例4.5】设计程序完成将百分制成绩转换成五分制表示。

```
 #include  <stdio.h>
 void  main( )
 {
    char   chscore;
int    nscore;
    printf( "\n please  input  Score ( 0 ~~ 100 ): \n" );
    scanf( "%d",  &nscore );
    if  (  nscore < 60  )
    {  chscore = 'E';    }
    else  if  ( nscore < 70 )
    {  chscore = 'D';  }
```

```
else  if  (  nscore < 80 )
      {  chscore = 'C';  }
      else  if  ( nscore < 90 )
      {  chscore = 'B'; }
      else   {  chscore = 'A';  }
      printf( "\n Your Score is  %c  !  !  ! \n", chscore );
      return ;
   }
```

【例4.6】从键盘读取一个字符数据，判断其类型是：控制类字符，数字字符，大写字符，小写字符，其他字符等类别中的哪一类？设计程序完成。

基本思路：读入字符数据，依据字符的ASCII值判断所属范围。

4.2.5 switch语句及break语句

switch语句的语法格式：

```
switch（〈表达式〉）
{
case <常值1>: 复合语句1
case <常值2> : 复合语句2
 ……
case <常值n>: 复合语句n
default:    复合语句n+1
}
```

下面使用switch语句改写例题【例4. 5】。

【例4.7】设计程序完成将百分制成绩转换成五分制表示。

```
#include  <stdio.h>
 void  main( )
 {
c4:      int    nscore;
c5:      char   chscore;
c6:      printf( "\n input  Score ( 0 ~~ 100 ): \n" );
c7:      scanf( "%d",  &nscore );
c8:      nscore  /=  10;     // 变换后，nscore 的值是0~10之间的整数
```

```
c9:      switch ( nscore  )
c10:     {
c11:      case  0:
c12:      case  1:
c13:      case  2:
c14:      case  3:
c15:      case  4:
c16:      case  5: { chscore = 'E'; }
c17:      case  6: { chscore = 'D'; }
c18:      case  7: { chscore = 'C'; }
c19:      case  8: { chscore = 'B'; }
c20:      case  9:
c21:      case  10: { chscore = 'A'; }
          }
```

```
c23:   printf( "\n Your Score is  %c  ！ ！ ！\n", chscore );
c24:    return ;
    }
```

c23句输出chscore变量的当前值。编译运行此段程序应该是正确的。但是程序运行后，不管百分制成绩是多少，输出结果总是如下：

Your Score is A ！ ！ ！

程序一定是存在问题。

- 这里说的流程控制语句，就是指跳转语句break。break语句的作用之一就是从某个case子句处跳出switch语句，继续执行switch的下一条语句。break语句还用于循环语句当中，后面介绍。
- 例4.7的正确写法需要break与switch语句配合使用，程序如下：

```
#include  <stdio.h>
    void  main( )
     {
c4:       int    nscore;
c5:       char   chscore;
c6:       printf( "\n input  Score ( 0 ~~ 100 ): \n" );
c7:       scanf( "%d",  &nscore );
c8:       nscore  /=  10;      // 变换后，nscore 的值是0~10之间的整数
c9:      switch  (  nscore   )
c10:      {
c11:       case   0:
c12:       case   1:
c13:       case   2:
c14:       case   3:
```

```
c15:      case  4:
c16:      case  5: { chscore = 'E'; }
c16_1:              break;
c17:      case  6: { chscore = 'D'; }
c17_1:              break;
c18:      case  7: { chscore = 'C'; }
c18_1:              break;
c19:      case  8: { chscore = 'B'; }
c19_1:              break;
c20:      case  9:
c21:      case  10: { chscore = 'A'; }
c21_1:              break;
     }
c23:      printf( "\n Your Score is  %c  !  !  ! \n", chscore );
    return ;
 }
```

- 【例4.8】设计能够完成单步四则运算的简易运算器，输出运算结果。