第4章 流程控制

华中科技大学计算机学院
甘早斌

内容提纲

4.1 C语句分类

C语句
- 声明语句
  - 变量声明
  - 函数原型声明
  - 常量定义
  - 类型定义
- 可执行语句
  - 表达式语句
  - 复合语句
  - 选择语句（if 和 switch）
  - 循环语句（while, do-while和for）
  - 转移语句（break, continue, goto和return）
  - 标号语句

4.2 表达式语句

- 在任何C表达式的末尾加一个分号都可以构成一个语句，即表达式语句。表达式语句的一般形式：

  表达式;
- 其中，“;”是C语句不可缺少的组成部分，它表示一个语句的结束。
- 在C语言中，赋值、输入和输出都由表达式语句实现。例：

  x = y + 1 ➡ x = y + 1;

  x += y ➡ x += y;

  i = j = k ➡ i = j = k ;

  printf(”hello”) ➡ printf(”hello”) ;

  scanf(”%d”, &x) ➡ scanf(”%d”, &x) ;

4.2 表达式语句(续)

- 再如：

  a – b;
- 也是一个表达式语句，但是由于在表达式求值的过程中并没有改变任何变量的值，这样的表达式语句并没有实际意义。
- 仅由一个分号构成的语句称为空语句，即：

  ;
- 它不执行任何操作。在程序设计中，如果某处在语法上需要一条语句，而在实际功能上不需要执行任何操作时，可以使用空语句。

4.3 复合语句

- 1．复合语句的一般形式
  - 用花括号“{}”括起来的一组语句，语法上等价于单个语句，语法格式为：

```
1. {
2.     说明部分
3.     语句部分
4. }
```

  - 说明部分可包含0至多个说明语句；语句部分可包含0至多个执行语句。
  - 复合语句又称块。函数体是一个块。

- 例4.3：不含说明语句的复合语句

```
{
    t = a;
    a = b;
    b = t;
}
```

- 例4.4：包含说明了语句的复合语句。

```
{
    int  t ;
    t = a; a = b; b = t;
}
```


4.3 复合语句（续）

- 2．嵌套的复合语句
  - 复合语句中包含复合语句，从而形成嵌套的复合语句。例如：

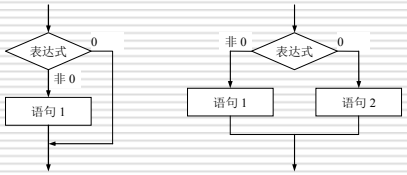
```
{
    int a = 0, b = 1;
    {
        int a=1;
        printf("a=%d\n", a);
        printf("b=%d\n", b += 1);
    }
    printf("a=%d\n", a);
    printf("b=%d\n", b);
}
```


4.3 复合语句（续）

- 3．复合语句的用途
  - 复合语句在程序设计中主要有以下两种用途：
    - (1)用于语法上只允许出现单个语句而处理上需要执行多个语句的地方，例如作为if语句的子句及循环语句的循环体。
    - (2)用于改变嵌套if-else语句的配对规则。
  - 此外，当需要说明临时使用的局部变量时，也可使用复合语句。

4.4 if语句

- 1．if 语句的一般形式
  - if语句有两种形式：
    - (1) if格式：if （表达式）语句1;
    - (2) if-else格式：if （表达式）语句1；else语句2;



(a) if 格式　　(b) if - else 格式

4.4 if语句（续）

- 2．嵌套的if 语句
  - 1) 嵌套if语句的形式
  - 当if子句或else子句中又包含if 语句时，则形成嵌套的if语句。例如，可用下面一个嵌套的if语句求a，b，c三个数中最大值：

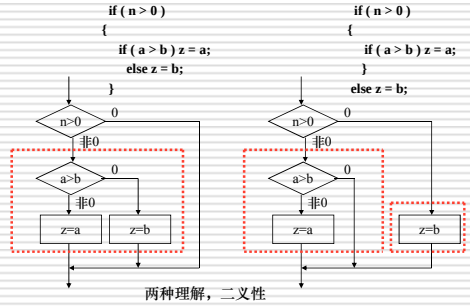
```
if ( a > b )
        if ( a > c ) max = a;
        else  max = c;
else
        if ( b > c ) max = b;
        else max = c;
```


4.4 if语句（续）

- 2)嵌套if 语句中else的配对规则
- 对嵌套if语句中else与if的配对必须制定一个规则，否则会造成理解上的二义性。例如：

```
if ( n > 0 )
if ( a > b ) z = a;
else z = b;
```

- 编译程序约定：else与其前面最靠近的还未配对的if配对，即内层优先配对原则。



两种理解，二义性

嵌套的if语句的配对规则：else与前面最靠近的if配对。

```
if (n>0)
if (a>b) z=a;
else z=b;
```

等价于

```
if (n>0)
{if (a>b) z=a;}
else z=b;}
```

实现

$$z=\begin{cases} a, & n>0\text{且}a>b \\ b, & n>0\text{且}a\le b \end{cases}$$

如果程序员需要表达下式的语义，则必须通过加{}来改变上述约定，这是复合语句的用途之一。

$$z=\begin{cases} a, & n>0\text{且}a>b \\ b, & n\le 0 \end{cases}$$

```
if (n>0)
{ if (a>b) z=a; }
else z=b;
```


3．程序设计举例

- 例4.4：学生成绩按下列标准分等(x为学生的平均成绩)：

分数范围	等级英文名
90≤x ≤100	excellent(优)
80≤x < 90	good(良)
60≤x < 80	middle(中)
x<60	bad(差)

输入某学生考试分数，输出该学生分数的英文等级。

- 分析：这是一个多分支结构的程序设计问题，可以使用多重嵌套的if语句来实现。首先需要输入分数值x，如果分数值x不在合理范围（0≤x≤100），则提示输入错误，程序结束；否则，依次判断分数值x的范围，输出成绩的相应英文等级。


```
1.  #include<stdio.h>
2.  int main(void){
3.  float  x;
4.    printf("input  the  score");   scanf("%f",&x);
5.    if  (x>100 || x<0)
6.        printf("input error");
7.    else  if  (x<=100 && x>=90)
8.            printf("excellent");
9.        else  if  (x<90 && x>=80)
10.               printf("good");
11.           else  if  (x<80 && x>=60)
12.               printf("middle");
13.           else  printf("bad");
14. return 0;   }
```


4.5 switch语句

- 1. switch语句的形式
  - switch语句的一般形式为：

```
1.  switch(表达式){
2.          case 常量表达式1：语句序列1;
3.          case 常量表达式2：语句序列2;
4.                  …
5.          case 常量表达式n：语句序列n;
6.          default:        语句序列n+1;
7.  }
```


4.5 switch语句（续）

- 2. switch语句的使用要点
  - 使用switch语句时，第一要注意列出的case应能包括选择表达式所有的取值情况，如果不能全部包括，则应使用default子句处理余下的情况。
  - 第二应特别注意break在switch中的作用，如果希望执行完某一case下的语句之后便跳出switch语句，则必须使用break或return转移语句。
    - break跳出switch语句之后继续执行switch语句后面的一个语句(如果有)
    - return语句则立即结束函数并返回到调用处(如果是主函数，则结束程序)。

4.5 switch语句（续）

- 例如：下面是一个不含转移语句的switch语句，注意观察该语句执行时的输出。

```
1.  i = 1;
2.  switch (i) {
3.          case 0: printf("%d\t", i);
4.          case 1: printf("%d\t", i++);
5.          case 2: printf("%d\t", i++);
6.          case 3: printf("%d", i++);
7.          default: printf("\n");
8.  }
9.  printf("%d\n", i);
```


3．程序设计举例

- 例1：学生考试成绩按以下标准分等（x为学生考试分数）：

分数范围	等级英文名
90≤x≤100	excellent（优）
80≤x＜90	good（良）
60≤x＜80	middle（中）
x＜60	bad（差）

- 输入某学生的考试分数，输出该学生的考试成绩的英文等级。
- 要求用switch语句来实现。

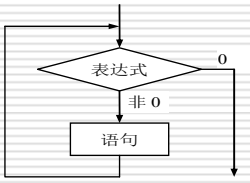
3．程序设计举例（续）

- 分析：
  - switch语句可用来解决多分支问题，但每个case后面的常量都是一个离散的值，不能表示一个数值范围。
  - 为此，将分数范围[0，100]每10分划为一段，则可划分为[0，10)，[10，20)，[20，30)，[30，40)，[40，50)，[50，60)，[60，70)，[70，80)，[80，90)，[90，100)十个分数段。
  - 进一步对分数值进行除以10然后取整的处理，可以发现上面同一分数段内的分数经处理后得到相同的一个整数。
  - 比如，[0，10）范围内的数除以10然后取整，结果都是0；[10，20）范围内的数除以10然后取整，结果都是1；……这样一来，就可以用一个离散值代表一个分数段内的所有分数值。


```
1.  #include<stdio.h>
2.  int  main(void)
3.  {
4.      float  x;    int  i;
5.      printf("input  score");  scanf("%f", &x);
6.      i = x / 10;          /*i的正确值应为：0至10*/
7.      if (i > 10 || i < 0)  printf("input error");
8.      else
9.          switch  ( i ) {
10.             case  10:
11.             case  9:     printf("excellent");  break;
12.             case  8:     printf("good");  break;
13.             case  7:
14.             case  6:     printf("middle");  break;
15.             default:     printf("bad");
16.             }
17. Return 0;
18. }
```


4.6 while语句

- 1. while语句的形式
  - while语句的一般形式为：
    while (表达式) 语句;
  - while语句流程图



- 例4.6 while循环语句示例

```
1.  i = 0;
2.  while (i < 5){
3.          printf("i=%d\t", i);
4.          i++;
5.  }
6.  printf("i=%d\n", i);
```

- 注意：如果循环体中没有改变循环变量的值，或循环变量值的变化不能使循环控制表达式的结果为0，则循环将永不终止(通常称之为"死循环")。如果循环控制表达式的值一开始就为0(条件为假)，则循环体一次都不执行。

2．程序设计举例

- 例4.10 将来自标准输入文件的正文复制到标准输出文件，每次输入和复制一个字符。
- 分析：
  - 以EOF（系统常量，值为-1）为结束标志的字符流称为一个正文，可以包含空白字符，例如空格符、制表符和换行符。根据题目要求，输入函数应使用getchar(每次输入一个字符)，输出函数应使用putchar(每次输出一个字符)；复制过程是一个重复地调用getchar读和调用putchar写的过程
  - 因此程序的流程结构是一个循环语句；读入的字符是否为EOF则是循环控制条件。

2．程序设计举例（续）

- 例4.10：算法步骤：
  - (1) 调用getchar读入一个字符并赋给字符变量c。
  - (2) 如果c不是EOF，则执行(3)；否则结束执行。
  - (3) 输出c；
  - (4) 读下一字符并赋给字符变量c；
  - (5) 转步骤(2)。


```
/*程序4.10：*/
#include<stdio.h>
void main(void)
{
    char c;
    printf("input a text end of ctrl + z:\n");
    c = getchar();
    while (c != EOF){
      putchar(c);
      c = getchar();
    }
}
```

```
/*#include <stdio.h>
void main(void)
{
      char c;
      printf("input a text end of ctrl+z:\n");
      while (( c = getchar()) != EOF )
                putchar(c);
}*/
```


2．程序设计举例（续）

- 例4.11：输入一个C程序(一段正文)，按原来格式复制输出，复制过程中删去输入程序中所有的注释。
- 分析：
  - 为了删去C程序中所有的注释，关键在于如何区分注释部分和需要复制的部分。为此，可将复制过程划分为4种状态：
    - 复制状态(COPY)
    - 开始注释状态(START)
    - 注释状态(COMMENT)
    - 结束复制状态(END)
  - 初始状态为COPY。

2．程序设计举例（续）

- 每种状态下的处理方法如下：
  - (1) 在COPY状态下，若读入字符为’/’(可能为注释开始符号)，则将状态改为START；否则将读入的字符复制到输出。
  - (2) 在START状态下，若读入字符为’*’(确定注释开始)，则将状态改为COMMENT；否则(不是注释)，将上一次读入的字符’/’复制到输出；然后检查本次读入的字符是否为’/’，若是，则状态保持START不变，否则将本次读入的字符复制到输出并将状态改为COPY。
  - (3) 在COMMENT状态下，若读入字符为’*’(可能为注释结束符号)，则将状态改为END。
  - (4) 在END状态下，若读入字符为’/’(确定注释结束)，则将状态恢复成COPY；否则(不是注释结束)，如果读入字符是’*’，则状态保持END不变，否则将状态改为COMMENT。

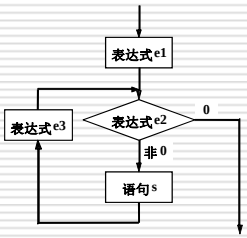

```
/*程序4.11：*/
#include<stdio.h>
#define COPY 0
#define START      1
#define COMMENT    2
#define END  3

void main(void)
{
   char c;
   int state;
   state = COPY;
   printf("input C program end with ctrl+z:\n");
   while ((c = getchar()) != EOF)
      switch (state){
case COPY:
         if (c == '/') state = START;
         else putchar(c);
         break;
case START:
         if (c == '*') state = COMMENT;
         else {  putchar('/');
            state = (c == '/') ? START : (putchar(c), COPY);     }
         break;
      case COMMENT:
         if (c == '*') state = END;
         break;
      case END:
         state = (c == '/') ? COPY : ((c == '*') ? END : COMMENT);
      } /* end of switch，end of while */
}
```

4.7 for语句

- 1. for语句的形式
  - for语句的一般语法形式表示为：
    for(e1；e2；e3) s;
  - for语句流程图：
  - 等价于：

```
1. e1;
2. while(e2) {
3.         s;
4.         e3;
5. }
```



4.7 for语句（续）

- 使用for语句时须注意表达式e1，e2，e3的用法：
  - (1)三个表达式可以全部或部分缺省，但无论缺省e1，e2或e3，它们之间的分号不能省。
  - (2)缺省e1和e3时的for语句形如for(;e2;) s，等价于一个形如while (e2) s的while语句。
  - (3)缺省e2时的for语句for(e1 ; ; e3) s和三个表达式都缺省的for语句for(;)s都是无限循环语句。被省略的e2缺省值恒为非0(e1和e3没有缺省值)。

4.7 for语句（续）

- 例4.12：for循环语句示例。

```
1. int i;
2. for(i = 1; i < 4; i++)
3.     printf("i=%d s=%d\n", i, 2 * i);
```

- 几种等价的形式：

```
1. i = 1;
2. for(; i < 4; i++)
3.     printf("i=%d s=%d\n", i, 2 * i);
```

- 或

```
1. for( i = 1; i < 4;){
2.     printf("i=%d s=%d\n", i, 2 * i);
3.     i++;
4. }
```


4.7 for语句（续）

- 或

```
1. i = 1;
2. for(; i < 4;){
3.     printf("i=%d s=%d\n", i, 2 * i);
4.     i++;
5. }
```

- 或

```
1. i = 1;
2. for(; ;){
3.     printf("i=%d s=%d\n", i, 2 * i);
4.     i++;
5.     if (!(i < 4)) break;
6. }
```


2．程序设计举例

- 例4.15：输入一批整数，以0为结束。输出其中最大的一个值。
- 分析：
  - 从若干个数中找出最大的一个数可用“打擂台”的方法，即两两相比，大者留下；当所有的数比完时留下的那个数为最大。
  - 比较过程采用每次输入一个数立即与上一次留下的那个较大的数比较的方法。
  - 因此程序中只需用两个变量：一个变量x用于保存每次输入的一个数据，另一个变量max用于保存每两个数相比中较大的一个数。
  - 这种方法可以用于从任意多个输入数据中找出其中的最大值。

2．程序设计举例（续）

- 例4.15：算法步骤：
  - (1) 输入第一个数(x)。
  - (2) 置最大数max初值为x。
  - (3) 如果x不等于0，则执行(4)；否则，执行(7)。
  - (4) 输入下一个数(x)。
  - (5) 如果max < x，则将max修改为x(赋值)。
  - (6) 转步骤(3)。
  - (7) 输出max，结束。

```
#include<stdio.h>
int main(void)
{
        int x, max;
       printf("input a group integer end of 0：\n");
        scanf("%d", &x);
        max = x;
        for( ; x != 0; ){
           scanf("%d", &x);
           if (max < x) max = x;
        }
        printf("max=%d\n", max);
       return 0;
}
```


2．程序设计举例（续）

- 例4.16：求n!，n从终端输入。
- 分析：
  - 根据阶乘的定义，n为≥0的整数，n的阶乘等于1至n连乘，即n!=1*2*3*…*n。这种反复进行的相同或类似的操作可以通过循环语句来实现。循环体实现的操作是，每循环一次时将第i项(i分别为1，2，3，…，n)与前面各项相乘的结果(即 i-1的阶乘)相乘(称为累乘)；循环结束条件为i等于n。i起着双重作用，既是阶乘因子又是循环变量。n!=(n-1)!*n
  - 考虑到n较大时n!是一个相当大的数，为避免溢出，应将结果变量说明为long，unsigned long，或double。此外，累乘过程开始之前一定要将存放累乘积的变量置初值1，而不能置为0。

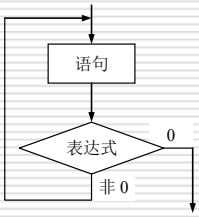
2．程序设计举例（续）

- 例4.16：算法步骤：
  - (1) 输入n
  - (2) 将累乘积变量fac及循环变量i置初值1
  - (3) 如果i≤n，则执行累乘(fac=fac*i)；否则(i>n)，转步骤(5)
  - (4) 将i值增加1，转步骤(3)
  - (5) 输出累乘结果fac，结束


```
/*程序416：*/
#include<stdio.h>
int main(void)
{
    int n, i;
    unsigned long fac;
    printf("input n:\n");
    scanf("%d", &n);
    for (fac = 1, i = 1; i <= n; i++)
            fac *= i;
    printf("%d!=%lu\n", n, fac);
    return 0;
}
```


4.8 do-while语句

- 1. do-while语句的形式
  - do-while语句的一般形式为：
    do 语句 while (表达式);
  - do-while语句流程图：



4.8 do-while语句（续）

- do-while语句：
  do 语句 while (表达式);
- 可以用以下等价的while循环语句来代替。

```
1. 语句;
2. while (表达式)
3. {
4.         语句;
5. }
```

2．程序设计举例

- 例4.17：把输入的整数按反方向输出。例如，输入的数是12345，要求输出结果是54321。
- 分析：
  - 在输入一个整数时，是从高位到低位（或者说从左到右）依次输入各位上的数字。
  - 要按反方向输出，就是从低位到高位（或者说从右到左）连续地输出该数的各位数字。具体来说，就是先输出个位数字，再输出十位数字，直到最高位数字。
  - 获取一个整数的个位数字的算法是将该整数除以10取余(%10)。去掉一个整数的个位数字(使十位数字变个位数字，百位数字变十位数字，直到最高位数字变次高位数字)的算法是将该整数除以10(整数除)
  - 这样，可以用循环语句从低位到高位依次输出原整数的数字。


```
1.  /*程序4.21：*/
2.  #include<stdio.h>
3.  void main(void)
4.  {
5.    int x, digit;
6.    printf("input an integer:\n");
7.    scanf("%d", &x);
8.    do
9.    {
10.     digit = x % 10;
11.     printf("%d", digit);
12.     x /= 10;
13.   }while (x != 0);
14.   printf("\n");
15. }
```


2．程序设计举例（续）

- 例4.19：输入任意一个大于或等于2的整数n，判断该数是否为素数并输出相应结果。
- 分析：
  - 根据数学定义，一个大于2的整数n，如果除1和n外不能被任何数整除(即n不含1和n以外的任何因子)，则n是素数；此外，规定2是最小素数。为了确定n是否含有1和n以外的因子，只需用2至$\sqrt{n}$（也可以用2至n-1或2至n/2)作除数除n。？
  - 如果均不能整除n，则n是素数，否则(即只要发现一个因子)n不是素数。显然，用2至$\sqrt{n}$作除数时所做的除法次数比用2至n-1或2至n/2作除数时少得多。

2．程序设计举例（续）

- 例4.19：算法步骤：
  - (1) 输入n，直到n符合要求为止(循环语句)；
  - (2) 确定除数i的初值(i=2)及终值j(j=sqrt(n))；
  - (3) 检查2 ～ sqrt(n)的每一个数是否都不是n的因子(循环语句)，方法是，i从2开始，用i除n，若余数非0且i <=j，则使i值增加1再重复该过程；若余数为0(找到一个因子)或i <= j不成立，则结束循环。
  - (4) 如果循环结束后余数为非0，则说明2 ～ sqrt(n)范围内的整数都不是n的因子，因此可以判定n是素数；否则(发现一个因子)，n不是素数。


```
1.  /*程序4.19：*/
2.  #include <stdio.h>
3.  #include <math.h>
4.  void main(void)
5.  {
6.    int n, i, k, r;
7.    printf("input n:\n");
8.    scanf("%d", &n);
9.    if (n == 2)      printf("2 is a prime\n");
10.   else if (n > 2){
11.         i = 1;          k = sqrt(n);
12.         do {            ++i; r = n % i;
13.         } while (r && i <= k);
14.         if(r)     printf("%d is a prime\n", n);
15.         else      printf("%d isn't a prime\n", n);
16.   }   /* end of else-if */
17. }
```


3．循环语句小结

- 三种循环语句的区别及使用要点归纳如下(s是循环体；e,e1,e2,e3是表达式)：
  - (1)while(e) s; 和for(e1；e2；e3) s;
    - 先测试e或e2，后执行s，若第一次测试时e或e2结果为0，则s一次也不执行；
  - (2) do s while(e);
    - 先执行s，后测试e，所以s总是至少被执行一次。
  - 使用时应根据具体情况选用，一般说来，必定要执行的循环可以用三种循环语句中任何一种；可能不被执行的循环则不能用do-while。

3．循环语句小结（续）

- (3) 对于while(e) s; 和for(e1；e2；e3) s; 和do s while(e);
  - 第一次测试循环条件(e或e2)之前，循环变量必须赋初值，初值只赋一次；
  - 在循环体(s)或e3(对于for语句)中必须有能够改变循环变量值的语句或表达式。
  - 写循环条件时，应注意避免无限循环、永不执行的循环或执行次数不正确的循环等情况。
- (4) 对于for(e1；e2；e3) s;
  - 控制部分的e1可以包含给循环变量赋初值以及其他与循环有关的运算，即在循环开始之前仅执行一次的运算；
  - e2不要求一定是关系表达式或逻辑表达式，只要能正确控制循环体的执行（非0值执行循环体，0值结束循环），任何表达式都可以；

3．循环语句小结（续）

- (4) 对于for(e1；e2；e3) s;
  - e3是每次执行循环体后紧接着要执行的表达式，通常用于改变循环变量的值，如i++之类，e3也可以包括某些属于循环体部分的内容，也可将e3放到循环体最后。
  - 可见，for语句使用非常灵活，其控制部分的三个表达式可以容纳除循环变量赋初值、测试循环条件和修改循环变量值的运算以外的其他与循环有关的运算。
- (5)任何循环语句当循环体含有一个以上语句时，必须写成复合语句(用{ }括起来)；当循环体为空语句时不要掉了分号(;)。

4.9 goto语句和标号语句

- goto语句又称为无条件转移语句，它的一般形式为：

  goto 标号;
- 任何可执行C语句都可以加标号前缀成为标号语句。标号语句的形式为：

  标号：语句;
- goto语句中的标号是对标号的引用，标号语句中的标号是对标号的定义。
- 被goto语句引用的标号必须有且仅有一个对应的标号语句，对应的标号语句称为该goto语句的目标语句；而允许标号语句没有对应的goto语句。

4.9 goto语句和标号语句（续）

- 有标号的引用必须有惟一的标号定义，而有标号的定义不必有标号的引用。
- goto语句的目标语句允许出现的范围称为标号的作用域。C语言中标号的作用域是goto语句所在的函数，即goto语句不能从一个函数转移到另一个函数中，但可以在一个函数内从嵌套结构的内层直接转到最外层。
- 使用标号语句时，同一函数内的标号不能同名(Why?) 。goto语句和标号语句在函数中出现的先后位置没有约束，即对标号的定义和对标号的引用没有先后次序的规定。

goto语句和标号语句的用法

- 例4.20：输入一个算式，模拟袖珍计算器的加、减、乘、除四则运算。假定计算时不考虑运算符的优先级，也不允许输入圆括号()，而是按照运算符出现的先后顺序执行运算。例如，

  输入10.8+0.13*100

  计算结果为1093.000000
- 例4.20源程序代码ex4.20.c。


```
/*程序4.20：*/
#include <stdio.h>
void main(void)
{
  double x, y;
  char op;
inx:
  printf("input arithmetic expression:\n");
  scanf("%lf", &x);
  while((op = getchar()) !='\n'){
iny:
    scanf("%lf", &y);
    switch(op){
      case '+': x += y; break;
      case '-': x -= y; break;
      case '*': x *= y; break;
      case '/':
        if (y) x /= y;
        else {/* 除数为0，重新输入除数 */
          printf("divisor is zero , input divisor again!\n");
          goto iny;
        }
        break;
      default:/* 运算符非法，重输入算式 */
        printf("illegal operator , input arithmetic expression again!\n");
        goto inx;
    }/* end of switch */
  }/* end of while */
  printf("%lf\n", x);
}
```

goto语句和标号语句的用法

- 注意：
  - goto语句不是必需的语言成分。 (Why?)
  - goto语句的惟一好处是可以从嵌套结构的最内层(switch语句或循环语句)直接转到最外层(隔层转移)，用起来较方便；
  - 但如果随意地使用goto语句则会破坏程序的结构化特性，使程序的逻辑结构不清;
  - 因此应尽量少用或不用goto语句。

4.10 break、continue和return

- 1. break语句
  - break语句的形式为：
    break;
  - break语句有以下两种用途：
    - (1)用于switch语句中，从中途退出switch语句；
    - (2)用于循环语句中，从循环体内直接退出当前循环。
  - 注意：
    - 对于嵌套的循环语句和switch语句，break语句的执行只能退出直接包含break的那一层结构。

4.10 break、continue和return(续)

- 例4.22：打印2～100之间的所有素数，每行输出10个数。
- 分析：
  - 如4.8节例4.24的程序所示，判断一个数是否为素数(找因子)要用循环语句实现，因此判断2～100之间的每一个数是否为素数要用二重循环。
  - 例4.22源程序代码ex4.22.c。
  - if(m) s1; else s2; ?
  - if(!m) s1; else s2; ?


```
1.  /**【例4.22】 打印2～100之间的所有素数，每行输出10个数。 */
2.  #include<stdio.h>
3.  #include<math.h>
4.  int main(void)
5.  {
6.     int i, n, k, m=1;  ?
7.     for (k=0,n=2; n<100; ++n){
8.        for (i=2; i<=sqrt(n); ++i)
9.           if(!(m = n % i)) break;
10.       if (m){
11.          printf("%6d", n);
12.          if (!(++k % 10))  printf("\n");
13.       }
14.    }
15.    printf("\n");
16.    return 0;
17. }
```


2． continue语句

- continue语句的形式为：
  continue；(continue是关键字)
- continue语句只能出现在循环语句中，用于终止循环体的本次执行(并非退出循环语句)；
- 即在循环体的本次执行中，跳过从continue语句之后直到循环体结束的所有语句，控制转移到循环体的末尾。
- 对于while (e) s;和do s while(e);
  - 在执行continue语句之后马上执行对循环控制表达式(e)的计算和测试；

2． continue语句（续）

- 对于for (e1;e2;e3) s;
  - 则马上执行表达式e3，然后执行对表达式e2的计算和测试。
- 例4.23：输入10个整数，输出其中正数的个数及平均值。
- 例4.23源程序代码ex4.23.c。
- 本例不用continue语句也能实现同样的功能，改写工作作为练习请读者自己完成。

```
/*【例4.23】输入10个整数，输出其中正数的个数及平均值。 */
#include <stdio.h>
int main(void)
{
    int a, i, k, x;
    printf("input 10 numbers:\n");
    for (a=i=k=0; i<10; ++i) {
        scanf("%d", &x);
        if (x <= 0)  continue;     /* 对负数不作处理 */
        a += x;                    /* 计算正数的和，存入a */
        ++k;                       /* 正数的个数 */
    }
    if (k)       printf("numbers=%d，average=%f\n", k, 1.0 * a / k);
    return 0;
}
```


3． return语句

- return语句的功能是从被调用函数返回到调用函数。
- return语句有下面两种形式：
  - (1)不带表达式的return语句：
    return;
    - 不带表达式的return语句只能返回控制、不能返回值。
    - 只能用于从无返回值的函数中返回。
  - (2)带表达式的return语句：
    return 表达式;
    - 带表达式的return语句(表达式可以用()括起来)在返回控制的同时，将表达式的值返回到调用处。
    - 函数调用表达式的值就是这个返回值。

3． return语句（续）

- 例4.24：写一个函数sign，返回浮点数x的符号。如果x小于0，则返回-1；如果x等于0，则返回0；如果x大于0，则返回1。

```
/* sign：返回浮点数符号的函数，包含多个带表达式的return语句 */
int sign(double x)
{
    if (x < 0)
        return -1;
    else
        return ((!x) ? 0 : 1);
}
```

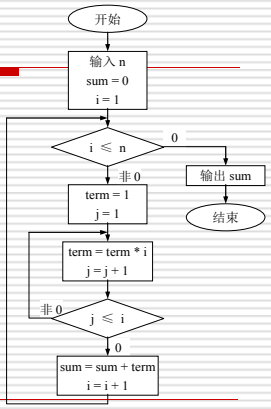

4.11 嵌套循环程序设计

- 嵌套循环指循环体是一个循环语句，或循环体包含循环语句。
- 嵌套循环又称为多重循环，三种循环语句可以相互任意嵌套。具有两层嵌套的循环称为二重循环，具有n重嵌套的循环称为n重循环；二重及以上的循环统称为多重循环。
- C语言对循环的嵌套层数没有限制，其中二重循环应用最为普遍，其次是三重循环。

4.11.1 嵌套循环

- 例4.25：计算$s=1^1+2^2+3^3+...+n^n$，n由终端输入。
- 分析：
  - 设每一项的底用整型变量i表示，i从1开始每次增1直至n。考虑到溢出，i^i及各项之和分别用长整型变量term和s表示。计算term是用循环对同一个i累乘i次；
  - 计算s也是用循环对每个term累加n次，而且计算term的循环是嵌套在计算s的循环体内的，所以计算s的算法是一个二重循环语句。
  - 外层循环(简称外循环)控制项数，内层循环(简称内循环)控制每项i的累乘次数。

- 例4.25：程序4.25的流程图。
- 例4.25源程序代码ex4.25.c。



```
/**【例4.25】计算s=1¹+2²+3³+...+nⁿ，n由终端输入。*/
#include <stdio.h>
int main(void) {
    int i, j, n;
    long s, term;
    printf("input n:\n");
    scanf("%d", &n);
    for (s=0,i=1; i <= n; ++i) {
        term = 1;
        j = 1;
        do  term *= i;
        while (++j <= i);
        s += term;
    }                   /*  end of for  */
    printf("s=%ld\n", s);
    return 0;
}
```


4.11.1 嵌套循环（续）

- 例4.26：输入一个字母，在屏幕正中输出由这个字母决定其高度的字符"金字塔"。例如输入小写字母d，则输出下列左边图形，如果输入大写字母D，则输出右边图形。

```
   a                A
  a b a            A B A
 a b c b a        A B C B A
a b c d c b a   A B C D C B A
```


- 分析
  - 图形的高度：c-'a'+1或c-'A'+1
  - 图形的每一行的字符数：2*n-1（n为行号）
  - 如果c是小写字母则输出图形由小写字母组成；如果c是大写字母则输出图形由大写字母组成；如果c为非字母则无输出
  - 屏幕的总宽度为80个字符，正中位置为第40个字符处，则左边的空格数目可由表达式40-2*(c-'a')或40-2*(c-'A')的值确定

- 例4.26：算法步骤
  - (1) 输入字符c；
  - (2) 如果c是小写字母则置塔顶top为'a'，如果c是大写字母则置top为'A'，如果c是非字母字符，则置top为'\0'；
  - (3) 如果top非0，则输出图形
    - (3)-1 置c1为top(外循环变量初值)；
    - (3)-2 如果c1≤c则输出一行：
    - (3)-2-1 输出一行左边的所有空格，空格数目为40-2*(c1-top)；
    - (3)-2-2 输出一行的前半段(包括正中的一个字符)；
    - (3)-2-3 输出一行的后半段；
    - (3)-2-4 输出换行；
    - (3)-2-5 c1=c1+1，转(3)-2(循环，输出下一行)；
  - 如果c1＞c则结束循环。（例4.32源程序代码ex4.32.c。）


```
/**【例4.26】 输入一个字母，在屏幕正中输出由这个字母决定其高度的字符"金字塔"。例如输入小写字母d，则输出下列左边图形，如果输入大写字母D，则输出右边图形。 */
#include <stdio.h>
#include <ctype.h>
int main(void)
{
  char c, c1, c2, top;      int i;
  printf("input a character:\n");
  top = isupper(c = getchar()) ? 'A' : (islower(c) ? 'a' : '\0');
  if (top) {
    for (c1=top; c1<=c; ++c1){
      for (i=1; i<=40-2*(c1-top); ++i)     putchar(' ');
      for (c2=top; c2<=c1; ++c2)           printf("%2c", c2);
      for (c2=c1-1; c2>=top; --c2)         printf("%2c", c2);
      printf("\n");
    }                   /*  end of external for  */
  }   /*  end of if  */
  return 0;
}
```


多重循环语句的使用要点

- (1) 对于多重循环，特别要注意给与循环有关的变量赋初值的位置。
  - 只需赋一次初值的操作应放在最外层循环开始执行之前；
  - 给内循环的有关变量赋初值应放在外循环体内、内循环开始执行之前。
- (2) 内、外循环变量不应同名！(Why?)
- (3)应正确书写内、外循环的循环体：
  - 需要在内循环中执行的所有语句必须用｛｝括起来组成复合语句作为内循环体；
  - 属于外循环的语句应放在内循环体之外、外循环体之中。

多重循环语句的使用要点（续）

- 例4.25的赋值语句term=1；j=1;和sum+=term;都是组成外循环体的语句，其中term=1;和j=1;位于内循环do-while语句之前，
- sum+ =term；语句位于do-while语句之后，它们均位于内循环体之外。

- (4)不应在循环中执行的操作，应放在进入最外层循环之前或最外层循环结束之后。
  - 例如，程序4.25中对输入提示(input n:)的输出及读入项数(n)是在程序运行过程中仅需执行一次的操作，且需在循环开始之前执行；
  - 最后输出整个计算结果(sum)只需执行一次，且应在循环结束之后执行。

4.11.2 枚举

- 许多问题不能利用数学公式或模拟算法求解，它们往往有一个庞大的问题状态空间，并且给出了一些约束条件，要求寻找问题解空间中的一个或多个解，求解此类问题需要使用搜索技术。
- 常用的搜索算法：
  - 枚举、深度优先搜索、广度优先搜索等。这里重点介绍枚举算法，而深度优先搜索和广度优先搜索在后续章节介绍。
- 枚举算法是最简单、最基本的搜索算法，其核心思想：
  - 通过对问题状态空间的每一种可能状态进行求解判断，从而求得满足条件的解。
  - 枚举算法通常可用循环语句和条件语句来实现，编程相对简单。

枚举算法举例

- 例4.27 下列乘法算式中，每个汉字代表1个0～9的数字，不同的汉字代表不同的数字。试编程确定使得整个算式成立的数字组合，如有多种情况，请给出所有可能的答案。

  赛软件 * 比赛 = 软件比拼

- 分析：
  - 这个算式中有5个不同的汉字，分别用变量a、b、c、d、e来代表汉字：赛、软、件、比、拼，此算式可表示为：

    (a*100+b*10+c) * (d*10+a) == b*1000+c*100+d*10+e
  - 每个变量的取值范围为0～9，由于取值不得重复，所以这5个变量所有可能的取值情况是一个10取5的排列，一共有10*9*8*7*6种。本题可采用枚举算法，依次测试每一种取值情况，如果满足上式，则得到一个解，直到测试完毕，得到所有解。


```
1.  #include <stdio.h>
2.  int main()
3.  {
4.    int a, b, c, d, e;
5.    for (a=0; a<10; a++)      /* 第一层循环执行10次 */
6.      for (b=0; b<10; b++) {
7.        if (b-a==0)  continue;   // 第二层循环执行9次，里层循环执行次数依次递减
8.        for (c=0; c<10; c++) {
9.          if ((c-b)*(c-a) == 0)
10.           continue;     /* (c-b)*(c-a) == 0等价于(c-b)==0 || (c-a)==0，下同 */
11.         for (d=0; d<10; d++) {
12.           if ((d-c)*(d-b)*(d-a) == 0)  continue;
13.           for (e=0; e<10; e++) {
14.             if ((e-d)*(e-c)*(e-b)*(e-a) == 0)  continue;
15.             if ((a*100+b*10+c) * (d*10+a) == b*1000+c*100+d*10+e)
16.               printf("%d%d%d * %d%d = %d%d%d%d\n", a, b, c, d, a, b, c, d,e);
17.           }
18.         }
19.       }
20.     }
21.   return 0;
22. }
```


枚举法的特点

- 枚举算法要测试问题状态空间的所有可能的状态来求得满足题目要求的解
- 当问题规模变大时，其效率是比较低的
- 优点：
  - 简单、直接，容易实现
- 在满足时间和空间约束的前提下，枚举法是一种可供选择的有效算法

4.11.3 筛法

- 筛法是一种构造素数表的有效方法。具体做法是：
  - 将1～n之间的数按升序排列，按照定义，1既非素数又非合数，将1划掉；
  - 2是素数，将3～n之间所有2的倍数都划掉；
  - 从2往后搜索，重复以下操作：碰到未被划掉的第一个数是素数，从该数后面的序列中划掉该数的整数倍；
  - 如此下去，直至搜索到n为止，最后序列中未被划掉的数都是素数。

- 例4.28 用筛法构造2～100之间的素数表，并输出该素数中的素数。

```c
#include <stdio.h>
int main()
{
    int i, j, a[100];
    for (a[0]=a[1]=0,i=2; i<100; i++)  /* 数组初始化，先假定2~99都是素数 */
        a[i] = 1;                      /* 数组下标值是被测数，用1标记素数 */
    for (i=2; i<50; i++)               /* 从2开始，“筛掉”每个素数的倍数 */
        if (a[i]) {                    /* 如果a[i]值为1，则i为素数 */
            for (j=i*i; j<100; j+=i)   /* 从i倍的i开始“筛”,因为之前的倍数已被“筛掉”
*/
                a[j] = 0;  /* 元素值为0表示该元素下标值不是素数 */
        }
    for (i=2,j=0; i<100; i++)  /* 输出2~100间的素数表 */
        if (a[i]) {  /* “筛”过之后，值为1的元素对应下标为素数 */
            printf("%4d", i);
            if (++j == 8) {  /* 为了输出整齐，每行输出8个素数 */
                j = 0;
                printf("\n");
            }
        }
    return 0;
}
```


4.11.4 递推

- 当一个问题具备递推关系时，通常可以采用递推法求解。
- 例4.29 盒子里有n(n<10000)个球，A、B两人轮流从盒中取球，我们约定游戏规则为：
  - (1)每次从盒子中取出的球的数目必须是1，3，7或者8个；
  - (2)A先取球，B接着取球，如此双方交替取球，直到盒里的球被取完；
  - (3) 取走盒中最后一个球的一方为输方。
  - 假设取球过程中A和B都不存在失误，即不会将赢的局面变为输。编程实现，输入小球的数目n，按规则取球，如果A为赢方则输出1，A为输方则输出0。


```c
#include <stdio.h>
int main()
{
    char tab[10000];
    int ins[] = {1, 3, 7, 8};
    int i, j, n, num;
    for (tab[0]=1,i=1; i<10000; i++)
        tab[i] = 0;
    for (i=1; i<10000; i++) {
        if (tab[i] == 0)
            for (j=0; j<4; j++)
                if ((num=i+ins[j]) < 10000)
                    tab[num] = 1;
    }
    scanf("%d", &n);
    printf("%d\n", tab[n]);
    return 0;
}
```


本章小结

- 流程控制是程序设计必须掌握的最基本内容。算法只有通过流程控制语句描述才能在计算机上实现。
- 程序的流程控制表现为顺序、分支、循环和转移几种结构。
- 结构化程序由顺序、分支和循环三种结构的语句组成，不包含转移语句。任何算法都能够用顺序、分支和循环三种结构的语句来实现，即都可以写成结构化程序。然而，适当使用转移语句可以起到简化程序的作用，但滥用转移语句将降低程序的可读性，使程序难以维护。
- 本章重点要求掌握if语句嵌套规则，switch语句执行流程，while语句、for语句和do-while语句循环控制条件与循环体执行的次数，几种转移语句与分支和循环语句的配合来实现灵活的流程控制等。
- 要求掌握一些基本的算法，比如计算日期、解方程、处理正文、求阶乘、判断素数、输出二维字符图形、枚举算法、筛法、递推算法等。

Assignment

- 必做题：
  - 4.1，4.2，4.3，4.4，4.5，4.6，4.7，4.8，4.9，4.10，
  - 4.11，4.12，4.13，4.15，4.16
- 我建议：
  - 书上的所有例题自己独立地编写一遍，在机器上调试、执行，确保准确理解和灵活掌握！
  - 后面习题，每题都做，并且搞懂！