第 3 章　流程控制

任何一个计算机程序在逻辑上即程序执行时的控制流程都可以用 3 种基本结构表示：顺序结构、分支结构和循环结构。顺序结构按语句在源程序中出现的顺序依次执行；分支结构根据一定的条件有选择地执行或不执行某些语句；循环结构在一定条件下重复执行相同的语句。所有的流程控制都是由语句实现的，能够支持结构程序设计的语言必须有好的流程控制语句。

C 语言通过 6 种执行语句实现上述 3 种基本结构，这 6 种语句是：①表达式语句；②复合语句；③标号语句；④选择语句(if 和 switch)；⑤循环语句(while、for 和 do-while)；⑥转移语句(break、continue、goto 和 return)。

3.1　内容提要

3.1.1　复合语句

复合语句的一般形式如下：

```
{
    声明语句部分
    可执行语句部分
}
```

一个复合语句在语法上等价于单个语句，这包含两层含义：一是凡一个语句能够出现的地方都能够出现复合语句；二是花括号中的所有语句是一个整体，要么全部执行，要么一句也不执行。复合语句可以嵌套，即复合语句中还可以有复合语句。

3.1.2　选择语句

1. if 语句

if 语句有以下两种形式。

(1) if 格式如下：

```
if(表达式)语句
```

(2) if-else 格式如下：

```
if(表达式)语句 1  else 语句 2
```

if 语句允许嵌套，嵌套时 else 与其前面最靠近的未配对的 if 配对，此即内层优先配对原则。

2. switch 语句

switch 语句的一般形式如下：

```
switch (表达式) {
    case 常量表达式 1: 语句序列;
    case 常量表达式 2: 语句序列;
    ⋮
    case 常量表达式 n: 语句序列;
    default: 语句序列;
}
```

case 和 default 只能在 switch 语句中使用。表达式是选择条件，其值必须为整型（包括字符型和枚举型）。switch 语句的语句体由多个 case 子句和至多一个（可以没有）default 子句组成；case 后面的常量表达式是值为常数的表达式，通常为字面常量或符号常量，其值在类型上必须和选择条件的类型相一致；同一个 switch 语句中的所有 case 常量值必须互不相同；每个 case（称为一种情况）下可以有零个或多个语句，有多个语句时可以不用加{ }。

switch 语句执行时，先计算作为选择条件的表达式，并将表达式的值依次与 case 后面的每个常量比较，当与某个 case 的常量值相等时，则执行该 case 后面的语句。若表达式的值与各 case 的常量值都不相等，在有 default 的情况下则执行 default 后面的语句；否则不执行 switch 中的任何语句，此时 switch 等价于一个空语句。switch 语句体在执行时，一旦遇到 break 语句，则终止剩余语句执行，而执行该 switch 语句的下一条语句。

3.1.3　循环语句

C 语言提供 while、for 和 do-while 3 种循环语句。

1. while 语句、for 语句和 do-while 语句

while 语句一般形式如下：

```
while (e)  s
```

for 语句一般形式如下：

```
for (e1; e2; e3)  s
```

do-while 语句一般形式如下：

```
do  s  while (e);
```

其中 e 代表表达式，s 代表语句（即循环体）。while 语句和 for 语句都是先计算并测试表达式（e 或 e2）的值，后执行循环体，若第一次测试时表达式的值就为 0 值，则循环体一次也不执行。do-while 语句则先执行循环体，然后计算并测试表达式的值，所以循环体至少被执行一次。

循环语句的使用要点如下：

（1）第一次测试循环条件（e 或 e2）之前，循环变量必须赋初值；在循环体或循环控制部分的表达式 e、e2 或 e3 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。

(2) for 语句控制部分的 e1 可以包含除给循环变量赋初值之外的其他与循环有关的运算(在循环开始之前仅执行一次的运算)。e2 只须具有确定循环是否继续的测试值,而不要求一定是 i<n 之类的关系表达式。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 可以包括某些属于循环体部分的内容,也可将 e3 放到循环体中。可见,for 语句的控制部分很灵活,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。写 for 语句时应兼顾算法的简洁性和可读性。此外,对于嵌套的循环语句,应写成缩进对齐格式,以增加程序结构的清晰感和美感。

(3) 任何循环语句,当循环体含有一个以上语句时。必须加大括号组成复合语句;当循环体为空语句时不要掉了分号。

2. 多重循环

当循环语句的循环体是一个循环语句或者包含有循环语句时,即为多重循环(或嵌套的循环)语句。

多重循环语句的使用要点如下:

(1) 对于多重循环语句,特别要注意给出与循环有关的变量赋初值的位置,只须执行一次的赋初值操作应放在最外层循环开始执行之前。

(2) 内、外循环变量不应同名,否则将造成循环控制混乱,导致死循环或计算结果错误。

(3) 应正确书写内、外循环的循环体。需要在内循环语句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;属于外循环体的语句如果有多个,则应放在内循环体的{}之外,并用外层{}括起来组成复合语句作为外循环体。

(4) 不应在循环中执行的操作应该放在进入最外层循环之前或最外层循环结束之后。

3.1.4 转移语句和标号语句

转移语句包括 break、continue、goto 和 return 语句。其中,break、continue 和 goto 用于改变由 3 种基本结构(顺序、分支和循环)的语句预定的程序流程,return 用于从函数返回到函数的被调用点。

1. break 语句

break 语句的形式如下:

```
break;
```

break 语句只能用于下面两种情况:

(1) 用于循环语句的循环体中,当循环条件还未变为假时提前结束循环语句的执行(强行退出循环)。

(2) 用于 switch 语句中,从中途退出 switch 语句,即跳过 break 语句之后直到 switch 语句结束的所有语句。

对于嵌套的循环语句或 switch 语句,break 语句的执行只能退出包含 break 语句的那一层结构,不能隔层退出。

2. continue 语句

continue 语句的形式如下：

```
continue;
```

continue 话句只能用于循环语句的循环体中，终止循环体的本次执行，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(未退出循环语句)，控制转移到循环体的末尾。对于 while (e) s 和 do s while (e)；语句，执行 continue 语句之后马上执行表达式 e。而对于 for (el; e2; e3)　s 语句，则马上执行表达式 e3。

3. goto 语句和标号语句

goto 语句的形式如下：

```
goto 标号;
```

标号语句的形式如下：

```
标号:语句;
```

goto 语句的用途是将控制转移到由标号指定的语句(标号语句)开始执行。标号语句是 goto 语句转向的目标。

标号是一个标识符，任何可执行语句都可以加标号而成为标号语句。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 语句不能将控制转出它所在的函数。goto 语句和标号语句在函数中的位置没有先后关系约束。

4. return 语句

retum 语句有以下两种形式。

(1) 不带表达式的 return 语句，其形式如下：

```
return;
```

(2) 带表达式的 return 语句，其形式如下：

```
return 表达式;
```

return 语句的用途是将控制返回到函数的被调用点。不带表达式的 return 语句只返回控制，但不返回值。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被调用处(该值作为函数调用表达式的值使用)。对于无 return 语句的函数，当执行完函数体中最后一个语句后控制自动返回到调用点。

3.2　典型题解析

3.2.1　判断题

【例 3.1】 下列各小题程序均是为测试所使用机器的机器字长而编制的。请指出哪些

程序是正确的？哪些程序不正确？为什么？

(1)

```
#include<stdio.h>
int main(void)
{
    unsigned k =~0;
    int bits =0;
    while (k !=0)     {
        k<<=1;
        ++bits;
    }
    printf("bits =%d\n", bits);
    return 0;
}
```

(2)

```
#include<stdio.h>
int main(void)
{
    unsigned k;
    int bits;

    for (k =~0, bits =0; k; k >>=1)
          ++bits;
    printf("bits =%d\n", bits);
    return 0;
}
```

(3)

```
#include<stdio.h>
int main(void)
{
    int k =~0, bits =0;

    do  {
        k<<=1:
        ++bits;
    }  while (k);
    printf("bits=%d\n", bits);
    return 0;
}
```

(4)

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

    for (k=~0, bits=0; k; k>>=1)
        ++bits;
    printf("bits=%d\n", bits);
    return 0;
}
```

【解析与答案】 因为整型(int 或 unsigned)的长度与机器字长相同,所以测试所使用机器的机器字长即确定一个整型变量存储单元的位数。实现方法上可以先将一个整型变量置为全 1,然后通过移位来统计最初的 1 的个数。移位时,无论左移或右移都必须保证填充位为 0;否则,程序将陷于死循环,永远不可能得到正确的统计结果。

题目所给的 4 个程序中,(1)、(2)和(3)是正确的,(4)是不正确的。

说明:4 个程序在解题思路方面是一样的,都是首先将一个字(int 型或 unsigned 型变量)的所有位置全为 1;然后用一个循环语句将这个字每次移 1 位,并计算循环的次数(即最初的 1 的个数),直到没有为 1 的位为止。循环结束时的计数结果则为所求。

程序(1)和(2)是通过将一个值为全 1 的 unsigned 变量 k 向左或向右移位来达到目的的。unsigned 整数无论是左移还是右移,都是逻辑移位,空出的位用 0 来填充;当移位的次数等于字的位数时,此 unsigned 整数为 0,从而统计出所使用机器的字长。

程序(3)使用的是 int 整数左移的方法。虽然 int 整数的移位运算是算术移位,但左移时低位仍然是用 0 来填充,所以同样可以达到目的。

程序(4)使用的是 int 整数 k 右移的方法。由于各位均为 1 的 int 整数即－1,右移时是用符号位(1)来填充高位,k 右移的结果永远是－1(非 0)。所以,该程序是一个死循环程序,永远无法达到目的。

此外,作为循环条件的表达式 k!＝0 可简化为 k,使程序的代码更优。

【例 3.2】 输入任意一个大于或等于 2 的整数 n,判断该数是否是素数并输出相应的结果。请阅读下列程序,指出哪些程序是正确的?哪些程序不正确?为什么?

(1)

```
#include<stdio.h>
int main(void)
{
    int i, n;
    printf("input n (n >=2) :");
    scanf("%d", &n);
    if (n<2) {
        printf("input error\n");
```

```
        return -1;
    }
    if (n==2)
        printf("2 is a prime\n");
    else {
        for (i=2; i<n; ++i)
            if (!(n %i)) {
                printf("%d isn't a prime\n", n);
                return 0;
            }
        printf("%d is a prime\n", n);
    }
    return 0;
}
```

(2)

```
#include<stdio.h>
int main(void)
{
    int i, j, n;
    printf("input n (n >=2) : ");
    scanf("%d", &n)
    if (n <2) {
        printf("input error\n");
        return -1;
    }
    for (i=2, j=n>>1; i<j; ++i)
        if (!(n %i)) {
            printf("%d is not a prime\n",n);
            return 0;
        }
    printf("%d is a prime\n", n);
    return 0;
}
```

(3)

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

    printf("input n (n >=2) : ");
    scanf("%d", &n)
    if (n <2) {
```

```
        printf("input error\n");
        return -1;
    }
    if (!(n & 1) && n!=2) {
        printf("%d is not a prime\n", n);
        return 0;
    }
    for (i=3,j=sqrt(n); i<j; i+=2)
        if (!(n%i)) {
            printf("%d is not a prime\n", n);
            return 0;
        )
    printf("%d is a prime\n", n);
    return 0;
}
```

【解析与答案】 按照素数的定义，除 1 和自身以外不含任何其他因子的整数是素数，2 是最小素数。因此，判断一个整数 n 是否是一个素数就是要找出 n 是否包含 1 和自身以外的其他因子。根据数学知识可知，用 2～n－1(或 n/2、$\sqrt{n}$)作为除数 i，如果 i 均不能整除 n(表达式 n%i 结果非 0)，则 n 是一个素数；否则，n 不是素数。显然，用 2～$\sqrt{n}$作为除数时，所做的除法次数比用 2～n－1 或 2～n/2 作除数时少得多。

此外，按照素数的定义，除 2 以外的偶数一定不是素数。因此，可以首先检查输入的整数 n，如果 n 是一个偶数且不等于 2(表达式!(n%2) && n!=2 非 0，或者!(n&1) && n!=2 非 0)，则可直接输出 n 不是素数的结论。如果 n 已经确定为奇数，还可以使除数 i 从 3 开始，且 i 每次增加 2(奇数的因子不可能为偶数)。这样处理减少了运算次数，从而提高了程序的运行速度。

$\sqrt{n}$在程序中应表示为标准函数 sqrt(n)，且应包含所需头文件<math.h>，sqrt(n)的结果是浮点数，应通过赋值或类型强制符强制为整型。sqrt(n)要执行函数调用，表达式 n/2 要做除运算，而 sqrt(n)或 n/2 的结果在找因子的过程中(循环)是不变的。因此，应在循环开始之前执行 sqrt(n)或 n/2，以免在循环过程中重复计算相同的结果。n/2 的结果是整数，如果用 n>>1 代替 n/2 可达到相同的目的，而且移位运算比除运算快得多。

可见，一个好的程序不仅要求正确，而且应考虑影响程序运行速度的各种因素。

题目所给的 3 个程序中，(1)是正确的，(2)和(3)是不正确的。

程序(2)和(3)中 j 的值是最后一个除数，因此循环条件应为 i≤j 而不是 i<j；否则，会因漏掉一个除数而使不符合素数定义的数被作为素数输出。例如，程序(2)中的 4 和 9，本不是素数但会被作为素数输出。此外，程序(3)中用了标准函数 sqrt 但未包含头文件<math.h>。

本例题给读者的启示：编写程序，特别是编写循环条件，一定要仔细、严密、逻辑简单且结构清晰。

3.2.2 写运行结果题

对于此类型题目，应按照程序运行时的实际输出顺序及输出形式写出结果。

【例 3.3】 读下列程序，假定每次运行程序时输入依次为：

1 1.5 1.9 2 2.5 2.9 3 3.5 3.9 4 4.5 4.9 0.9 5

请写出程序各次运行时的输出结果。

```
#include<stdio.h>
int main(void)
{
    double x, y;
    printf("please input a real number x,(x>=1 and x<5): ");
    scanf("%lf", &x);
    switch ((int)x) {
        case 1:
            printf("x =%.1f\tY=%.2f\n", x, y=3*x+5);
            break;
        case 2:
            printf("x =%.1f\tY=%.2f\n", x, y=(2+x)+(2+x));
            break;
        case 3:
            printf("x=%.1f\tY=%.2f\n", x, y=1+x+x);
            break;
        case 4:
            printf("x=%.1f\tY=%.2f\n", x, y=x*x-2*x+5);
            break;
        default:
            printf("x=%.1f\terror in input data\n", x);
        }
    return 0;
}
```

【解析与答案】 此程序中，double 变量 x 是分支的选择条件，由于 switch 语句中选择条件表达式的值要求为整型，因而用强制类型符(int)将 x 的值强制转换为整数。(int)x 的结果为 x 截去小数部分后的整数值。当 x≥1 且 x<5 成立时，(int)x 的值分别为 1、2、3、4，正好与 case 后面常量表达式的值匹配；否则，对 x 不进行处理，而是输出 error in inputdata 信息。

本程序实现的功能是求分支函数的函数值，程序各次运行时的输出结果如下：

```
x =1.0,  Y=8.00
x =1.5,  Y=9.50
x =1.9,  Y=10.70
X =2.0,  Y~16.00
x =2.5,  Y=20.25
x =2.9,  Y~24.01
x =3.0,  Y=10.00
X =3.5,  Y=13.25
```

```
x =3.9,  Y=16.21
x =4.0,  Y=13.00
x =4.5,  Y=16.25
x =4.9,  Y=19.21
x =0.9,  error in input data
x =5.0,  error in input data
```

【例 3.4】 阅读下列程序，假设输入为：HeHeHeHeHeHe↙，写出程序的输出结果。

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

int main()
{
    int i, j, len;
    char word[100];

    scanf("%s", word);
    len =strlen(word);
    for (i=1; i<=len; i++)   /*从小到大找周期*/
        if (len %i ==0) {     /*周期必须能整除长度*/
            int ok =1;
            for (j=i; j<len; j++)
                if (word[j] !=word[j %i]) {  /*不满足相隔周期步长相等,做标记并终止
                                                循环*/
                    ok =0;
                    break;
                }
            if (ok) {          /*一旦找到满足条件的周期,则输出周期并终止*/
                printf("%d\n", i);
                break;
            }
        }
    return 0;
}
```

【解析与答案】 此程序的主体是一个嵌套的循环语句。外循环是 for 语句，外循环的循环体由一条 if 语句组成，该 if 语句的 if 子句中，定义了标记变量 ok 并初始化为 1；其后是一条 for 循环语句，这个内层循环的循环体又是一条 if 语句，这条 if 语句在条件 word[j]!=word[j%i]成立时，将标记变量 ok 置为 0 并终止内层循环，word[j]和 word[j % i]是字符串 word 中相隔若干个间隔为 i 的两个字符，因此内层循环是在判断串中字符是否按周期 i 重复出现，由此可知，程序用于分析所输入字符串的周期。如果一个字符串可由长度为 k 的字符串重复多次得到，则该串以 k 为周期。本程序从小到大枚举各个周期，一旦符合条件就立即输出并终止循环。可见，如果一个周期串存在多个周期，那么输出的将是最小周期。

程序的输出结果是：

```
2
```

【例 3.5】 阅读下列程序，假定输入为：cast=3，house=5，sum is 8↙，请写出输出结果。

```
#include<stdio.h>
int main(void)
{
    char c;
    int alpha, digit, other;

    alpha =digit =other =0;
    printf("input characters end of newline:\n");
    while ((c=getchar()) !='\n') {
        if (c>='a' && c<='z' || c>='A' && c<='Z')
            alpha++;
        else if (c>='0' && c<='9')
            digit++;
        else
            other++;
    }
    printf("alphas=%d\ndigits=%d\nother=%d\n", alpha, digit, other);
    return 0;
}
```

【解析与答案】 该程序通过一个 while 语句对输入的一行字符(以'\n'结束)进行分类计数。分类方法是字母字符为一类，数字字符为一类，其余字符为一类。表达式 c>='0'&&c<='9'为 c 是否数字的条件；c>='a'&&c<='z' || c>='A'&&c<='Z'为 c 是否字母的条件，该条件表示对大小写字母统一计数。

此外，函数 getchar 每次读入一个字符存入变量 c，包括输入行上的空格字符在内。该程序的输出是：

```
alphas=14
digits=3
other=6
```

【例 3.6】 阅读下列程序，假定运行程序时输入如下，请写出运行程序时的输出结果。

```
123  456↙
555  555↙
123  594↙
0    0↙
#include<stdio.h>
int main()
{
```

```
    int a, b;
    int c, ans;
    int i;

    while (scanf("%d%d", &a, &b) ==2) {
        if (!a && !b)
            return 0;
        c =0, ans =0;
        for (i=9; i>=0; i--) {
            c =(a%10 +b%10 +c) >9 ?1 : 0;
            ans +=c;
            a /=10;
            b /=10;
        }
        printf("%d\n", ans);
    }
    return 0;
}
```

【解析与答案】 外层 while 循环的循环条件：scanf("%d%d", &a, &b)==2，库函数 scanf 的返回值表示按格式从标准输入设备成功转化的数据的个数，所以循环条件表达式从输入设备读取两个十进制整数，并分别存入变量 a 和 b，如果这两个变量输入成功，则循环继续下去，否则退出循环。循环体中，首先判断 a 和 b 是否同时为 0，若同时为 0，则终止程序；接下来，在进入内层 for 循环前，将变量 c 和 ans 赋值为 0，标识符 ans 通常用来存放问题的解；for 循环体内是 4 条赋值语句，第一条根据变量 a 和 b 的个位数字及 c 的和值是否大于 9，对 c 赋值 1 或 0。可见，c 存放的是进位值：产生进位(本位数字和加低位进位值大于 9)，c 为 1；否则，c 为 0。语句"ans+=c;"是对进位情况进行累计，内层循环的后两条语句"a/=10; b /=10;"将变量 a 和 b 的个位切除，使十位变个位，百位变十位。因此，程序用来计算两个十进制整数在做加法运算时，所产生进位的次数。

运行程序时，输出结果为：

```
0
3
1
```

3.2.3　完善程序题

【例 3.7】 下面的程序是计算级数 $S=1+\sqrt{2}+\sqrt{3}+\cdots+\sqrt{10}$，将程序中下画线处的内容补充完整。

级数中自然数 A 的平方根是通过牛顿迭代法求出的，牛顿迭代公式为：

$$X_1=1,\qquad X_{n+1}=1/2*(X_n+A/X_n),\qquad n=1,2,3\cdots$$

迭代过程直到$|X_{n+1}-X_n|\leqslant 10^{-7}$时为止，$X_{n+1}$即为所求自然数的平方根。

下面的程序中，变量 x 为自然数，它在 1～10 变化；变量 xl、x2 和 limit 分别为公式中的

X_n、X_{n+1}和$|X_{n+1}-X_n|$；变量 s 为求得的结果。

```
/*例 3.7 程序:牛顿迭代法求级数*/
#include<stdio.h>
#include<math.h>
#define EPS 1.0E-7

int main(void)
{
    int x;
    double s =0, x1 =1, x2, limit;

    for (x=1; x<=10; x++) {
        do {
            x2 = ____(1)____;
            limit =fabs(x1 -x2);
            ____(2)____;
        } while (____(3)____);
        ____(4)____;
    }
    printf("s =%.8f\n", s);
    return 0;
}
```

【解析与答案】 程序有两重循环，外循环控制代表自然数的变量 x 由 1～n 变化，内循环是通过牛顿迭代法求 $\sqrt{X}$的迭代过程。

迭代时，X_{n+1}(即 x2)可通过迭代公式 1/2 *（X_n＋A/Xn)求出，写为表达式的形式即 x2＝(x1＋x/x1)/2，当求出 X_{n+1}后，调用库函数 abs 求出$|X_{n+1}-Xn|$，并将它存入变量 limit 中。为了能正确地执行下一轮迭代，应更新 X_n(即 x1)。当$|X_{n+1}-X_n|$(即 limit)＞EPS 时，继续迭代，否则停止迭代。

在结束一个自然数平方根的求值之后，x2 为该自然数的开平方结果。在求下一个自然数的平方根时，x1 的值为迭代初值。

说明：除$\sqrt{2}$的迭代初值与迭代结果相差较大外(也只有 0.4)，以后各 $\sqrt{X}$的迭代初值愈来愈接近迭代结果。也就是说，直接使用上一自然数的开平方结果作为下一个自然数开平方的迭代初值是合适的，不需要重新赋迭代初值。

综上所述，填入各下画线处的语句如下：

(1) (x1＋x / x1) / 2　　(2) x1＝x2

(3) limit ＞ EPS　　(4) s ＋＝x2 或 s＝s＋x2

【例 3.8】 下面的程序将输入正文中的横向制表符('\t')换成空格符(' ')输出。将程序中下画线处的内容补充完整。

当输出横向制表符时，光标定位在某数(即下列程序中的 TAB，通常为 8)倍数的位置上。将横向制表符换成空格符输出时，也应保持这种特性。为此，每次输出字符时都需记录

当前光标的位置，以便在将横向制表符换成空格符时计算出应输出的空格数目。

设光标当前位置记录在整型变量 pos 中，换掉横向制表符时应输出的空格数目 nb 可由公式 TAB－pos %TAB 计算出来。

```
/* 例 3.8 程序:将输入正文中的横向制表符换成空格符输出 */
#include<stdio.h>
#define TAB 8

int main(void)
{
    int nb;                         /* 应输出的空格数目 */
    int pos;                        /* 记录光标当前位置 */
    char ch;

    for (    (1)    ; (ch=getchar0)!=EOF; )
        switch (ch) {
        case '\t':
            nb =TAB -pos %TAB;
            (2)    ;
            while (nb --)
                putchar(' ');
            break;
        caset '\n':
            putchar(ch);
            (3)    ;
            break;
        default:
            putchar(ch);
            (4)    ;
        }
    return 0;
}
```

【解析与答案】 程序从正文中读取字符，当读取的字符是横向制表符时，由公式 TAB－pos% TAB 算出应输出的空格字符数 nb，并换成 nb 个空格输出；当读取的字符是其他字符时，则原样输出。重复此过程直至遇到文件尾为止。

程序在计算空格字符数 nb 时使用了变量 pos，而变量 pos 在题目给出的程序中从未赋过值，显然是在预留的空处给变量 pos 赋值的。从题目说明可知，变量 pos 用于记录光标的当前位置。开始读入字符时，pos 应置初值 0；输出行结束符('\n')后，光标回到行首，pos 应重新置 0；在遇到横向制表符输出 nb 个空格后，光标向后移动了 nb 个字符位置，pos 也应加 nb；至于其他字符，每输出一个字符，pos 都应增 1。

综上所述，填入各下画线处的语句如下：

(1) pos＝0　　(2) pos ＋＝nb　　(3) pos＝0　　(4) ＋＋pos

3.2.4 程序设计题

【例 3.9】 猴子吃桃问题。猴子第一天吃掉桃子总数的一半多一个，第二天又将剩下的桃子吃掉一半多一个，以后每天吃掉前一天剩下的桃子一半多一个，到第十天准备吃的时候见只剩下一个桃子。猴子第一天开始吃的时候桃子的总数是多少？

【解析与答案】 第 n 天吃之前余下的桃子数 X_n(n=10,9,…,1)如下：

$$X_{10}=1,\quad X_9=2(X_{10}+1),\quad X_8=2(X_9+1),\quad \cdots,\quad X_1=2(X_2+1)$$

从上面的分析可以得到求 X_n 的递推公式：$X_n=2\times(X_{n-1}+1)$，可根据递推公式采用循环方法实现该解法。

```
/* 例 3.9 程序:猴子吃桃问题 */
#include<stdio.h>
int main(void)
{
    int d, x;

    for (d=10,x=1; d>1; --d)   /* 从第十天倒推到第一天 */
        x = (x +1) <<1;
    printf("x=%d\n", x);
    return 0;
}
```

【例 3.10】 微生物分裂问题。假设有两种微生物 X 和 Y，X 出生后每隔 3 分钟分裂一次(数目加倍)，Y 出生后每隔 2 分钟分裂一次(数目加倍)。一个新出生的 X，半分钟之后吃掉一个 Y，并且从此开始，每隔 1 分钟吃一个 Y。现在已知有新出生的 X=10，Y=89，求 60 分钟后 Y 的数目。如果 X=10，Y=90，那么 60 分钟后 Y 的数目又是多少呢？

【解析与答案】 用 X 和 Y 表示两种微生物的数量。从零时刻开始，每过 1 分钟，Y 的数量减少 X，即 Y=Y−X>=0? Y−X:0；每过 2 分钟，Y 的数量翻倍，即 Y=Y<<1；每过 3 分钟，X 的数量翻倍，即 X=X<<1。我们用 t 表示从零时刻起，时间所过的分钟数，那么用 t 作循环变量，使其从 0 变为 60，每次循环 t 增加 1，在循环体内按上述规律改变 X 和 Y 的值，直到循环结束，输出 Y 的值即可。值得注意的是，循环在边界条件时的处理：t=0 时，X 和 Y 均不翻倍；t=60 时，X 和 Y 均翻倍。

题目的结果令你震惊吗？这不是简单的数字游戏。真实的生物圈有着同样脆弱的性质，也许因为你消灭的那只 Y 就是最终导致 Y 种群灭绝的最后一根稻草。

```
/* 例 3.10 程序:微生物分裂问题 */
#include<stdio.h>

int main(void)
{
    int X, Y, t;
```

```
    printf("In the beginning, X =10, Y =89\n");
    t =0, X =10, Y =89;          /* 0 时刻 X 和 Y 的数量 */
    Y =Y -X;                     /* 半分钟后,每个 X 吃掉一个 Y */
    for (t=1; t<60; t++) {
        if (!(t %2)) {           /* 每过 2 分钟,Y 数量翻倍 */
            Y <<=1;
        }
        if (!(t %3)) {           /* 每过 3 分钟,X 数量翻倍 */
            X <<=1;
        }
        Y =Y -X;                 /* 每过 1 分钟,每个 X 吃掉一个 Y */
        if (Y <0) {              /* Y 不可能为负数 */
            Y =0;
        }
    }
    Y <<=1, X <<=1;
    printf("After 60 minutes, Y =%d\n", Y);

}
```

【例 3.11】 奇怪的算式。福尔摩斯到某古堡探险,看到门上写着一个奇怪的算式:

ABCDE * ? = EDCBA

A、B、C、D、E 代表不同的数字,问号也代表某个数字,福尔摩斯想了好久没有算出合适的结果来。请你利用计算机的优势,找到破解的答案。把 A、B、C、D、E 所代表的数字写出来。

【解析与答案】 可用枚举的方法来求解,涉及 6 个变量 A、B、C、D、E 和 X(X 代表?),构造 6 重循环,分别让这 6 个变量从 0 变到 9,判断下式是否成立:

(A×10000+B×1000+C×100+D×10+E)×X==E×10000+D×1000+C×100+B×10+A

如果成立,则得到一组解,输出 A、B、C、D、E 和 X 的值。

根据题意,A、B、C、D、E 互不相等,且 A 和 E 不得为 0。

```
/* 例 3.11 程序:奇怪的算式 ABCDE * ? =EDCBA */
#include<stdio.h>
int main(void)
{
  int A, B, C, D, E, X;

  for (A=1; A<10; A++) {
    for (B=0; B<10; B++) {
      if (B-A ==0) continue;
      for (C=0; C<10; C++) {
        if ((C-A) * (C-B) ==0) continue;
        for (D=0; D<10; D++) {
          if ((D-A) * (D-B) * (D-C) ==0) continue;
```

```
            for (E=1; E<10; E++) {
              if ((E-A) * (E-B) * (E-C) * (E-D) ==0) continue;
              for (X=2; X<10; X++) {
              if(A * 10000+B * 1000+C * 100+D * 10+E) * X==E * 10000+D * 1000+C * 100+B * 10+A){
                  printf("A=%d, B=%d, C=%d, D=%d, E=%d, X=%d\n", A, B, C, D, E, X);
                }
              }
            }
          }
        }
      }
      return 0;
    }
```

【例 3.12】 比拼酒量。有一群海盗(不多于 20 人)在船上比拼酒量。过程如下：打开一瓶酒，所有在场的人平分喝下，有几个人倒下了。再打开一瓶酒平分，又有人倒下了……直到打开第四瓶酒，坐着的人已经所剩无几，海盗船长也在其中。当第四瓶酒平分喝下后，大家都倒下了。等船长醒来，发现海盗船搁浅了。他在航海日志中写到："昨天，我正好喝了一瓶……奉劝大家，开船不喝酒，喝酒别开船。"

请你根据这些信息，推断开始有多少人，每一轮喝下来还剩多少人。如果有多个可能的答案，请列出所有的答案，每个答案占一行。格式是：人数，人数，……。

例如，有一种可能是：20，5，4，2，0。

【解析与答案】 根据题目意思，一共喝了 4 轮，船长最后一轮醉倒，醉倒前正好喝了一整瓶。本题仍然可用枚举的方法求解。假设开始人数为 x0，则 x0 不多于 20 人，第一轮到第三轮喝下来，分别还剩 x1、x2、x3，第四轮喝过后还剩 0 人。设计一个 4 重循环，x0 从 20 递减到 1，x1 从 x0 递减到 1，依次，x3 从 x2 递减到 1，4 重循环的循环变量每次减少 1，那么当 1.0/x0＋1.0/x1＋ 1.0/x2＋ 1.0/x3＝＝1 时，得到一组解，将解输出。

```
/* 例 3.12 程序:比拼酒量 */
#include<stdio.h>

int main(void)
{
    int x0, x1, x2, x3;

    for (x0=20; x0>0; x0--) {
        for (x1=x0; x1>0; x1--) {
            for (x2=x1; x2>0; x2--) {
                for (x3=x2; x3>0; x3--) {
                    if (1.0/x0 +1.0/x1 +1.0/x2 +1.0/x3 ==1) {
                        printf("%d, %d, %d, %d, 0\n", x0, x1, x2, x3);
                    }
```

```
                }
            }
        }
    }
    return 0;
}
```

【例 3.13】 输入的正文复制到输出，复制过程中删去每个输入行的前置空格。

【解析与答案】 为了删去每个输入行的前置空格，可以用一个整型变量 flag 使其置为非 0 或 0 来标记当前的空格字符是处于行外或行内。如果 flag 为非 0，表明当前空格字符处于行外，则不复制，继续读入下一个字符；否则（flag 为 0），将当前字符（无论是否空格字符）复制到输出。

```
/* 例 3.13 程序:输入的正文复制到输出,前置空格不输出 */
#include<stdio.h>
int main(void)
{
    char ch;
    int flag=1;
    printf("input text:\n");
    while((ch=getchar())!=EOF)    {
        if (flag && (ch==' '||ch=='\t'))
            continue;
        putchar(ch);
        if (ch=='\n')
            flag=1;
        else
            flag=0;
    }
    return 0;
}
```

【例 3.14】 大奖赛计分。某电视台举办了低碳生活大奖赛，题目的计分规则相当奇怪：每位选手需要回答 10 个问题（其编号为 1～10），越到后面越有难度。答对的，当前分数翻倍；答错了则扣掉与题号相同的分数（选手必须回答问题，不回答按错误处理）。

每位选手都有一个起步的分数 10 分。某获胜选手最终得分刚好是 100 分，如果不让你看比赛过程，你能推断出他（她）哪个题目答对了，哪个题目答错了吗。

如果把答对的记为 1，答错的记为 0，则 10 个题目的回答情况可以用仅含有 1 和 0 的串来表示。例如，0010110011 就是可能的情况。你的任务是算出所有可能情况，每个答案占一行。

【解析与答案】 每个题目的回答情况有两种，则 10 个题目的回答情况有 $2^{10}=1024$ 种，按照计分规则依次判断每种情况下的最终得分，如果是 100 分，则得到一个答案。

```
/* 例 3.14 程序:大奖赛计分 */
#include<stdio.h>
```

```
#define MSK 0x200
int main(void)
{
    int i, x, j;

    for (i=0; i<1024; i++) {
        x =10;
        for (j=0; j<10; j++) {
            if ((i<<j) & MSK)
                x <<=1;
            else
                x -=j +1;
        }
        if (x ==100) {
            for (j=9; j>=0; j--)
                putchar((i>>j&1)+'0');
            putchar('\n');
        }
    }
    return 0;
}
```

3.3 实验三　流程控制实验

3.3.1 实验目的

(1) 掌握复合语句和 if 语句和 switch 语句的使用,熟练掌握 for、while、do-while 3 种基本的循环控制语句的使用,掌握重复循环技术,了解转移语句与标号语句。

(2) 熟练运用 for、while、do-while 语句来编写程序。

(3) 练习转移语句和标号语句的使用。

(4) 使用集成开发环境中的调试功能:单步执行,设置断点,观察变量值。

3.3.2 实验内容及要求

1. 程序改错

下面的实验 3-1 程序是合数判断器(合数指自然数中除了能被 1 和本身整除外,还能被其他数整除的数),在该源程序中存在若干语法和逻辑错误。要求对该程序进行调试修改,使之能够正确完成指定任务。

```
/*实验 3-1 改错题程序:合数判断器*/
#include<stdio.h>
int main( )
{
```

```
    int i,x,k,flag=0;
    printf("本程序判断合数,请输入大于 1 的整数,以 Ctrl+Z 结束\n");
    while(scanf("%d",&x)!=EOF) {
    for(i=2,k=x>>1;i<=k;i++)
           if(!x%i) {
               flag=1;
               break;
       }
    if(flag=1)  printf("%d 是合数\n",x);
    else  printf("%d 不是合数\n",x);
    }
    return 0;
}
```

2. 程序修改替换

(1) 修改实验 3-1 程序,将内层两出口的 for 循环结构改用单出口结构,即不允许使用 break、goto 等非结构化语句。

(2) 修改实验 3-1 程序,将 for 循环改用 do-while 循环。

(3) 修改实验 3-1 程序,将其改为纯粹合数求解器,求出所有的 3 位纯粹合数。一个合数去掉最低位,剩下的数仍是合数;再去掉剩下的数的最低位,余留下来的数还是合数,这样反复,一直到最后剩下的一位数仍是合数,这样的数被称为纯粹合数。

3. 程序设计

(1) 假设工资税金按以下方法计算: x＜1000 元,不收取税金;1000≤x＜2000,收取 5%的税金;2000≤x＜3000,收取 10%的税金;3000≤x＜4000,收取 15%的税金;4000≤x＜5000,收取 20%的税金;x＞5000,收取 25%的税金。输入工资金额,输出应收取的税金额度,要求分别用 if 语句和 switch 语句实现。

(2) 625 这个数很特别,625 的平方等于 390625,其末 3 位也是 625。请编程输出所有这样的 3 位数: 它的平方的末 3 位是这个数本身。

(3) 输入一只股票连续 n 天的收盘价格,输出该股票这 n 天中的最大波动值,波动值是指某天收盘价格与前一天收盘价格之差的绝对值。

(4) 将输入的正文复制到输出,复制过程中将每行一个以上的空格字符用一个空格代替。

(5) 打印如下杨辉三角形。

```
                1          /*第 0 行*/
              1   1        /*第 1 行*/
            1   2   1      /*第 2 行*/
          1   3   3   1
        1   4   6   4   1
      1   5   10  10  5   1
    1   6   15  20  15  6   1
  1   7   21  35  35  21  7   1
1   8   28  56  70  56  28  8   1
1   9   36  84  126 126 84  36  9   1
```