第 6 章 循环语句

课后练习题参考解答

1. 填空题

(1) 执行下面程序后，变量 result 的值是___18___，函数 function()的功能是___计算一个整数各位数字之和（保持符号不变）___。

```
int function(int n){
    int k = 0;
    do {
        k = k + n%10;
        n = n/10;
    } while(n);
    return k;
}

int main(void){
    int result = function(3456);
    return 0;
}
```

(2) 填空完成下面程序，其功能是打印 100 以内个位数为 6 且能被 3 整除的所有正整数。

```
int main(void){
    int i,j;
    for (i=0; i!=10 ;i++){
        j = i*10 + 6;
        if (  j%3   )
            continue;
        printf("%d ",j);
    }
    printf("\n");
    return 0;
}
```

说明：解答不唯一。如第一个空还可以填 `i < 10` 或 `i <= 9` 等；第二个空还可以填 `j%3 != 0` 或 `j/3*3 != j` 等。

2. 改写下列程序段，将其中的 for 循环改用 while 循环实现，要求功能不变。

(1) 程序段 1：

```
int fact(int n){
    int i;
    int result = 1;
    for (i = 2; i <= n; i++)
        result *= i;
    return result;
}
```

改写为 while 循环：

```
int fact(int n) {
    int i = 2;
    int result = 1;
    while(i <= n)
        result *= i++;
    return result;
}
```

(2) 程序段 2：

```
int main(void) {
    int sum = 0;
    int i;
    for (i = 0; i < 10; i++){
        if (i % 2)
            continue;
        sum += i;
    }
    return 0;
}
```

改为 while 循环：

方法一：用 goto 语句替代 continue 语句

```
int main(void) {
    int sum = 0;
    int i = 0;
    while (i < 10) {
        if (i % 2)
            goto update;
        sum += i;
update:
        i++;
    }
    return 0;
}
```

方法二：保持逻辑不变的情况下对 if 语句进行改写

```
int main(void) {
    int sum = 0;
    int i = 0;
    while (i < 10) {
        if (!(i % 2))
            sum += i;
        i++;
    }
    return 0;
}
```

说明：严格的改写只能按照方法一。

3. 按照下述思路，编写一个函数，用于计算一个正数的平方根。

正数 n 的平方根可以通过计算一系列近似值来获得，每个近似值都比前一个更加接近准确值。第一个近似值是 1，接下来的近似值则通过下面的公式来获得：

$$a_{i+1} = \frac{a_i + \frac{n}{a_i}}{2}$$

编写一个程序，读入一个值，计算并打印它的平方根。如果你将所有的近似值都打印出来，你会发现这种方法获得准确结果的速度有多快。原则上，这种计算可以永远进行下去，它会不断产生更加精确的结果。但在实际中，由于浮点变量的精度限制，程序无法一直计算下去。当某个近似值与前一个近似值相等时，你就可以让程序停止继续计算了。

通过与 C 标准数学库中的 sqrt()函数的计算结果对比来测试你的程序是否正确。

参考程序：

```c
#include <stdio.h>
#include <math.h>
#include <assert.h>

#define DEBUG

double my_sqrt(double n) {
    double result = 1, last_result = 0;
    while (result != last_result) {
        last_result = result;
        result = 0.5 * (last_result + n/last_result);
    }
    return result;
}

int main(void) {
#ifdef DEBUG
    assert(abs(my_sqrt(5.6)-sqrt(5.6))<0.00001);
#endif
    double num;
    printf("请输入一个正数:\n");
    scanf("%lf", &num);
    printf("%.2lf的平方根为%lf\n", num, my_sqrt(num));
    return 0;
}
```

4. 编写程序，输出如下字母塔：

参考程序：

```
#include<stdio.h>

int main(void) {
    int i,j;
    for(i=1;i<=26;i++){
        for(j=1;j<=26-i;j++)
            printf(" ");
        for(j=0;j<i;j++)
            printf("%c",j+'A');
        for(j=i-2;j>=0;j--)
            printf("%c",j+'A');
        printf("\n");
    }
    return 0;
}
```

5. 编写一个程序，提示用户从键盘输入两个正整数，在屏幕上输出这两个正整数的最大公约数和最小公倍数。

参考程序：

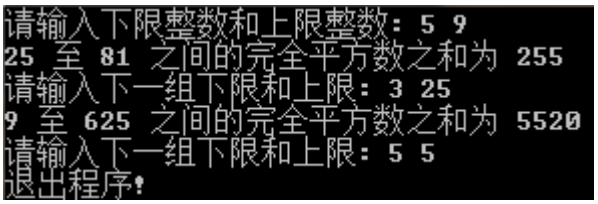
```
#include <stdio.h>

int get_greatest_common_divisor(int num1, int num2) {
    int result = num1;
    if (num2 < num1)
        result = num2;
    while ((num1%result) || (num2%result))
        result--;
    return result;
}

int get_lowest_common_multiple(int num1, int num2) {
    int result = num1;
    if (num2 > num1)
        result = num2;
    while ((result%num1) || (result%num2))
        result++;
    return result;
}

int main(void) {
    int n1, n2;
    printf("请输入两个正整数，以空格隔开：\n");
    scanf("%d%d",&n1,&n2);
    printf("%d和%d的最大公约数是%d\n",n1,n2,get_greatest_common_divisor(n1,n2));
    printf("%d和%d的最小公倍数是%d\n",n1,n2,get_lowest_common_multiple(n1,n2));
    return 0;
}
```

6. 编写一个程序，要求用户输入下限整数和上限整数，然后，依次计算从下限到上限的每一个整数的平方的加和，最后显示结果。程序将不断提示用户输入下限整数和上限整数并显示出答案，直到用户输入的上限整数等于或小于下限整数为止。程序运行的结果示例应该如下所示：



参考程序：

```c
#include <stdio.h>

int main(void){
    int lower_limit,upper_limit;
    printf("请输入下限整数和上限整数: ");
    scanf("%d%d", &lower_limit, &upper_limit);

    int i;
    int sum;
    while(lower_limit < upper_limit){
        sum = 0;
        for (i = lower_limit; i <= upper_limit; i++)
            sum += i*i;
        printf("%d 至 %d 之间的完全平方数之和为 %d\n",
            lower_limit*lower_limit, upper_limit*upper_limit, sum);

        printf("请输入下一组下限和上限: ");
        scanf("%d%d", &lower_limit, &upper_limit);
    }
    printf("退出程序!\n");

    return 0;
}
```

7. 鸡兔同笼是我国古代著名趣题之一。大约在 1500 年前，《孙子算经》中就记载了这个有趣的问题。书中是这样叙述的：“今有雉兔同笼，上有三十五头，下有九十四足，问雉兔各几何？”这四句话的意思是：有若干只鸡兔同在一个笼子里，从上面数，有 35 个头；从下面数，有 94 只脚。问笼中各有几只鸡和兔？

1）以下程序用于求解鸡兔同笼问题，请补充完整：

```
int main(void){
    int x,y;
    for (x=1;x!=35;x++){
        y = 35 - x;
        if(  2*x+4*y==94   )
            printf("笼中有%d只鸡，%d只兔。\n",x,y);
    }
    return 0;
}
```

2）以上程序的运行结果是：笼中有23只鸡，12只兔。

3）这一问题也可以采用一元一次方程来求解，请说明解题思路，并给出相应的程序。

要求：让用户分别输入笼中鸡和兔的头和足的数量，并对用户输入进行判断。若输入数据类型正确且合理，则计算结果；若输入的不是正整数，则提示输入数据类型错误，并要求用户重新输入；若输入数据类型正确但求解结果不满足条件（求解结果不是整数、鸡或兔的数量为负数），则提示输入数据不合理，并退出程序。

解题思路：

假定用户输入的头和足的数量分别为 n_1 和 n_2，并设兔有 x 只，则鸡有 n_1-x 只，依题意有：

$$4x+2(n_1-x)=n_2$$

整理得

$$2x=n_2-2n_1$$

参考程序：

```c
#include <stdio.h>

int main(void) {
    //从键盘读入数据
    int n1, n2;//n1和n2分别表示头和足的数量
    int check;
    char ch = '\0';
    printf("请分别输入头和足的数量，以空格隔开:\n");
    while( (check=scanf("%d%d", &n1, &n2))!=2
        || (check==2 && (ch=getchar())!='\n')
        || n1<=0
        || n2<=0){
        printf("输入的数据类型有误！输入量必须为两个正整数，并以空格隔开！\n");
        printf("请重新输入头和足的数量，以空格隔开:\n");
        if(ch!='\n')
            while (getchar()!='\n')//清空缓冲区
                continue;
        ch = '\0';
    }

    //计算方程系数并求解方程
    int a=2, b, x;//a、b为方程系数，x为待求量，即兔的数量
    if ((b=n2-2*n1)<0 || (x=b/a)*a!=b || n1-x<0) {
        printf("输入的数据不合理！程序终止！\n");
        return 1;
    }

    //打印结果
    printf("笼中有%d只鸡，%d只兔。\n", n1-x, x);
    return 0;
}
```

两个可能的运行结果如下：

运行结果 1：（输入数据合理的情况）

```
请分别输入头和足的数量，以空格隔开:
35 94
笼中有23只鸡，12只兔。
```

运行结果 2：（输入数据不合理的情况）

```
请分别输入头和足的数量，以空格隔开:
35 c
输入的数据类型有误！输入量必须为两个正整数，并以空格隔开！
请重新输入头和足的数量，以空格隔开:
-35 94
输入的数据类型有误！输入量必须为两个正整数，并以空格隔开！
请重新输入头和足的数量，以空格隔开:
35 93
输入的数据不合理！程序终止！
```