

云南大学数学与统计学实验教学中心
《高级语言程序设计》实验报告

课程名称：程序设计和算法语言	学期：2016~2017 学年上学期	成绩：
指导教师：赵越	学生姓名：刘鹏	学生学号：20151910042
实验名称：循环结构程序设计		
实验编号：No.04	实验日期：2017 年 8 月 9 日	实验学时：2
学院：数学与统计学院	专业：信息与计算科学	年级：2015 级

一、实验目的

- 1. 进一步练习选择结构的程序设计。
- 2. 练习并掌握实现循环结构的三种方法。
- 3. 练习并掌握选择结构与循环结构的嵌套。
- 4. 掌握多重循环的应用。
- 5. 学会单步跟踪的操作方法。

二、实验环境

Windows10 Pro Workstation 17096;
Code::Blocks 16.01 GCC 集成开发环境;
Cygwin GCC 编译器。

三、实验内容

3.1 2 题

求

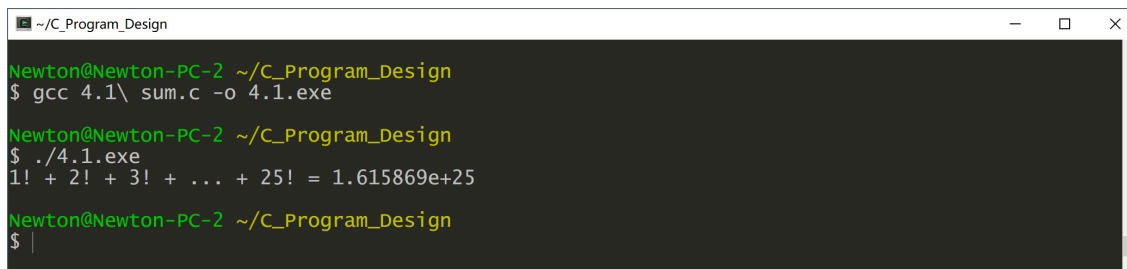
$$\sum_{n=1}^{25} n! (1! + 2! + \cdots + 25!)$$

3.1.1 程序代码：

```
1  /*
2  * filename: 4.1 sum.c
3  * property: exercise
4  */
5
6  #include <stdio.h>
7
8  int main() {
9      float n, s = 0, t = 1;
10     for(n = 1; n <= 25; n++) {
11         t *= n;
12         s += t;
```

```
13     }
14     printf("1! + 2! + 3! + ... + 25! = %e\n", s);
15     return 0;
16 }
```

上机运行，并记录下结果。然后用另外两种循环语句实现上述功能。



```
~/C_Program_Design
Newton@Newton-PC-2 ~/C_Program_Design
$ gcc 4.1\ sum.c -o 4.1.exe
Newton@Newton-PC-2 ~/C_Program_Design
$ ./4.1.exe
1! + 2! + 3! + ... + 25! = 1.615869e+25
Newton@Newton-PC-2 ~/C_Program_Design
$
```

3.2 3 题

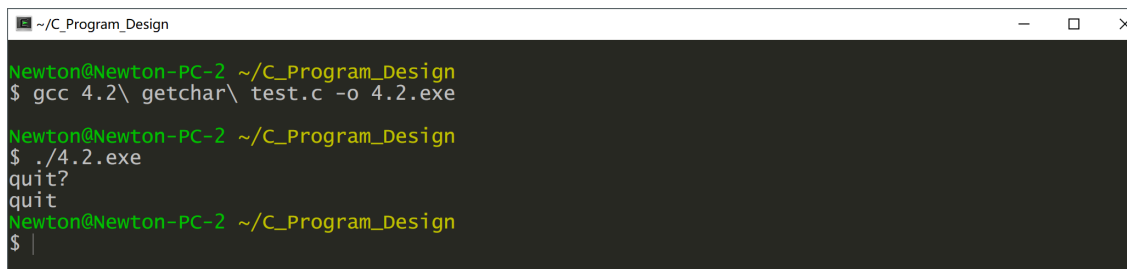
指出下面三个程序的功能，当输入“quit?”时，它们的执行结果是什么？

3.2.1 程序 1

3.2.1.1 程序代码

```
1  /*
2  * filename: 4.2 getchar test.c
3  * property: test
4  */
5
6  #include <stdio.h>
7
8  int main() {
9      char c;
10     c = getchar();
11     while(c != '?') {
12         putchar(c);
13         c = getchar();
14     }
15     return 0;
16 }
```

3.2.1.2 运行结果



```
~/C_Program_Design
Newton@Newton-PC-2 ~/C_Program_Design
$ gcc 4.2\ getchar\ test.c -o 4.2.exe
Newton@Newton-PC-2 ~/C_Program_Design
$ ./4.2.exe
quit?
quit
Newton@Newton-PC-2 ~/C_Program_Design
$
```

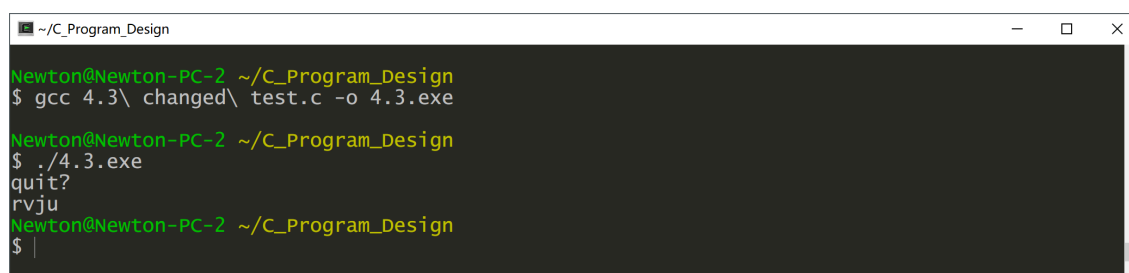
3.2.2 程序 2

3.2.2.1 程序代码

```
1  /*
```

```
2  * filename: 4.3 changed test.c
3  * property: test
4  */
5
6  #include <stdio.h>
7
8  int main() {
9      char c;
10     while((c=getchar()) != '?') {
11         putchar(++c);
12     }
13     return 0;
14 }
```

3.2.2.2 运行结果



```
~/C_Program_Design
Newton@Newton-PC-2 ~/C_Program_Design
$ gcc 4.3\ changed\ test.c -o 4.3.exe

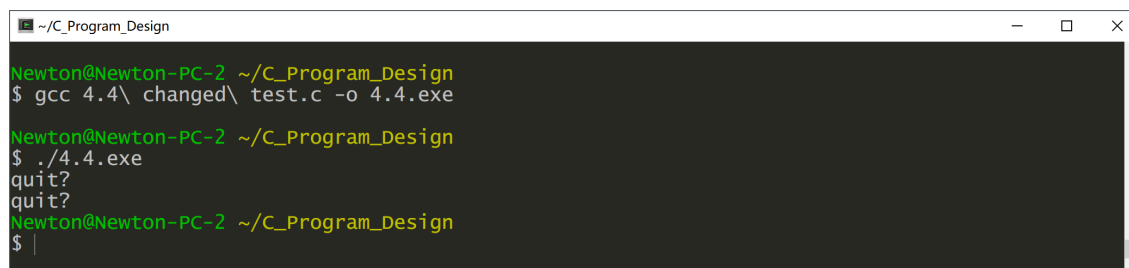
Newton@Newton-PC-2 ~/C_Program_Design
$ ./4.3.exe
quit?
rvju
Newton@Newton-PC-2 ~/C_Program_Design
$
```

3.2.3 程序 3

3.2.3.1 程序代码

```
1  /*
2  * filename: 4.4 changed test.c
3  * property: test
4  */
5
6  #include <stdio.h>
7
8  int main() {
9      char c;
10     while(putchar(getchar()) != '?') {
11         putchar(++c);
12     }
13     return 0;
14 }
```

3.2.3.2 运行结果



```
~/C_Program_Design
Newton@Newton-PC-2 ~/C_Program_Design
$ gcc 4.4\ changed\ test.c -o 4.4.exe

Newton@Newton-PC-2 ~/C_Program_Design
$ ./4.4.exe
quit?
quit?
Newton@Newton-PC-2 ~/C_Program_Design
$
```

分析输出的三种不同结果，在实验报告中写出为什么。

原因：

- (1) 输入了'?'就结束循环.
- (2) 当输入的不是'?'时候，就输出该字符的下一个字符，当输入'?'后，结束循环。
- (3) 直接输出，但是在这里，变量 `c` 没有被改动过，`++c` 都是乱码；程序的意义，就是输出从键盘得到的字符，之后判断输出的是不是'?'，之后输出`++c`。

3.3 换硬币

换零钱。把一元钱全兑换成硬币，有多少种兑换方法？有五角、两角和一分的硬币。

3.3.1 程序代码

```

1  /*
2  * filename: 4.5 coin.c
3  * property: exercise
4  */
5
6  #include <stdio.h>
7
8  int main() {
9      int i, j, k, n;
10     n = 10, k = 0;
11     for(i = 0; i <= n / 5; i++) {
12         for(j = 0; j <= (n-5*i)/2; j++) {
13             printf("5 dime = %d\t, 2 dime = %d\t,1 dime = %d\n",i,j,n-i*5-j*2);
14             k++;
15         }
16     }
17     printf("total times = %d\n", k);
18     return 0;
19 }

```

3.3.2 运行结果

```

~/C_Program_Design
Newton@Newton-PC-2 ~/C_Program_Design
$ ./4.5.exe
5 dime = 0      , 2 dime = 0      ,1 dime = 10
5 dime = 0      , 2 dime = 1      ,1 dime = 8
5 dime = 0      , 2 dime = 2      ,1 dime = 6
5 dime = 0      , 2 dime = 3      ,1 dime = 4
5 dime = 0      , 2 dime = 4      ,1 dime = 2
5 dime = 0      , 2 dime = 5      ,1 dime = 0
5 dime = 1      , 2 dime = 0      ,1 dime = 5
5 dime = 1      , 2 dime = 1      ,1 dime = 3
5 dime = 1      , 2 dime = 2      ,1 dime = 1
5 dime = 2      , 2 dime = 0      ,1 dime = 0
total times = 10
Newton@Newton-PC-2 ~/C_Program_Design
$

```

3.4 *5 题


穿越沙漠。用一辆吉普车穿越 1000 公里的沙漠。吉普车的总装油量为 500 加仑，耗油量为 1 加仑/公

里。由于沙漠中没有油库，必须先用车在沙漠中建立临时加油站，该吉普车要以最少的油耗穿越沙漠，应在什么地方建立临时油库，以及在什么地方安放多少油最好？

3.4.1 参考程序:

```
1  /*
2   * filename: 4.6 Jeep.cpp
3   * property: difficulty
4   */
5
6  #include <iostream>
7  #include <stdio.h>
8
9  using namespace std;
10
11 int main() {
12     int dis, k, oil;
13     dis = 500;
14     k = 1;
15     oil = 500;
16     while(dis<1000) {
17         printf("%d %d %d\n", k, oil, 1000 - dis);
18         k = k + 1;
19         dis = dis + 500 / (2*k + 1);
20         oil = 500 * k;
21     }
22     oil = 500 * (k - 1) + (1000 - dis) * (2*k - 1);
23     printf("%d %d %d\n", k, oil, dis);
24     return 0;
25 }
```

3.4.2 运行结果



```
~/C_Program_Design
Newton@Newton-PC-2 ~/C_Program_Design
$ g++ 4.6\ Jeep.cpp -o 4.6.exe

Newton@Newton-PC-2 ~/C_Program_Design
$ ./4.6.exe
1 500 500
2 1000 400
3 1500 329
4 2000 274
5 2500 229
6 3000 191
7 3500 158
8 4000 129
9 4500 103
10 5000 80
11 5500 59
12 6000 39
13 6500 21
14 7000 4
15 6652 1012

Newton@Newton-PC-2 ~/C_Program_Design
$
```

3.5 动态调试练习

①上机运行程序，分析运行结果。

②用单步跟踪观察 **while** 语句的执行过程：连续按三次 F8 键，再用两次 Ctrl-F7 操作分别将 **i** 和 **sum** 的值显示出来，然后不按断 F8 键，每次按 F8 后，观察绿条的变化和变量值的变化情况，以此来分析并弄清 **while** 语句的执行过程。

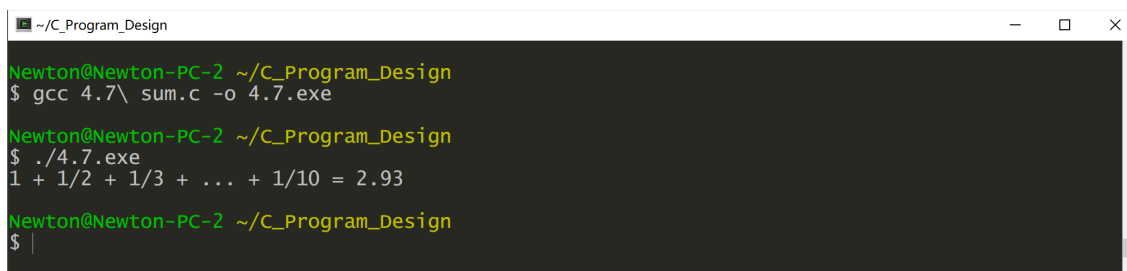
③修改程序，实现

$$s = 1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{10}$$

3.5.1 程序代码

```
1  /*
2  * filename: 4.7 sum.c
3  * property: test
4  */
5
6  #include <stdio.h>
7
8  int main() {
9      float i = 1;
10     float s = 0;
11
12     for (; i <= 10; i++) {
13         s += 1 / i;
14     }
15     printf("1 + 1/2 + 1/3 + ... + 1/10 = %3.2f\n", s);
16     return 0;
17 }
```

3.5.2 运行结果



```
~/C_Program_Design
Newton@Newton-PC-2 ~/C_Program_Design
$ gcc 4.7\ sum.c -o 4.7.exe
Newton@Newton-PC-2 ~/C_Program_Design
$ ./4.7.exe
1 + 1/2 + 1/3 + ... + 1/10 = 2.93
Newton@Newton-PC-2 ~/C_Program_Design
$ |
```

3.6 计算级数

$$s = 1 - \frac{2}{3} + \frac{3}{5} - \frac{4}{7} + \cdots + (-1)^n \frac{n+1}{2n+1}$$

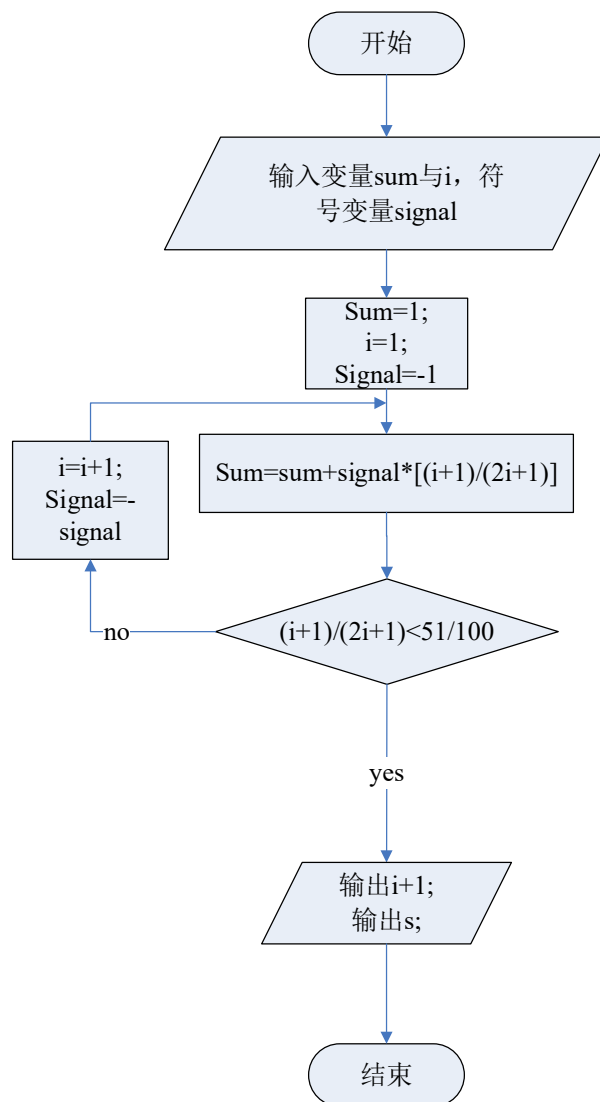
直到最后一项的绝对值小于 10^{-4} 为止（注意：是 $\frac{1}{2n+1}$ ）。

具体要求如下：

①画出流程图。

②除了要输出级数和 s 外，同时要求输出总的项数 n 。输出形式为：

n =具体值， s =具体值



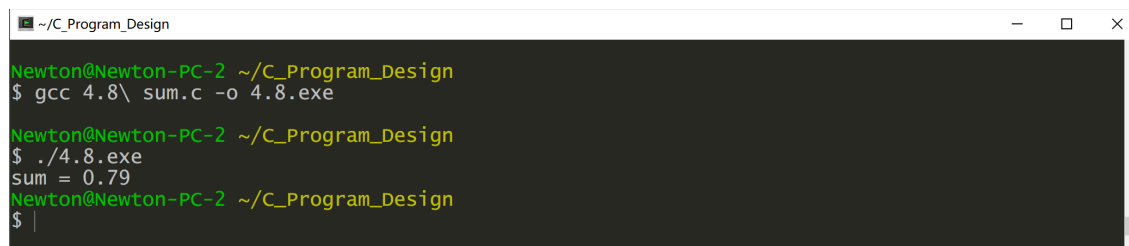
3.6.1 程序代码

```

1  /*
2  * filename: 4.8 sum.c
3  * property: exercise
4  */
5
6  #include <stdio.h>
7  #include <math.h>
8
9  int main() {
10     float i = 1;
11     float s = 1;
12
13     float test;
14
15     for (; (test = 1 / (2*i+1)) >= 1e-4; i++) {
16         s += pow(-1, i) * test;
17         // printf("test = %2.5f\n", test);
18     }
  
```

```
19
20     printf("sum = %3.2f", s);
21     return 0;
22 }
```

3.6.2 运行结果



```
~/C_Program_Design
Newton@Newton-PC-2 ~/C_Program_Design
$ gcc 4.8\ sum.c -o 4.8.exe
Newton@Newton-PC-2 ~/C_Program_Design
$ ./4.8.exe
sum = 0.79
Newton@Newton-PC-2 ~/C_Program_Design
$
```

3.7 完数之和

如果一个数恰好等于它的所有因子(包括1但不包括自身)之和,例如:6的因子为1,2,3,且 $1+2+3=6$,因此6是一个“完数”。计算并输出1000以内的所有“完数”之和。

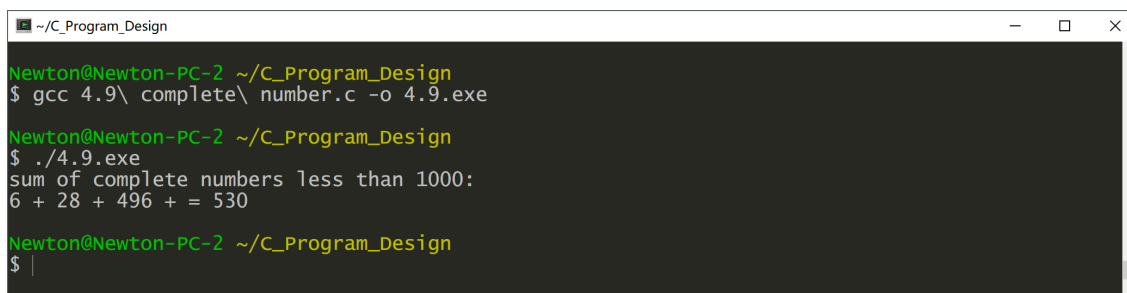
具体要求如下:

- ①所有循环均用 **for** 循环。
- ②输出要有文字说明,并同时输出各“完数”。输出形式为:完数 1 + 完数 2 + ... =和值

3.7.1 程序代码

```
1  /*
2  * filename: 4.9 complete number.c
3  * property: exercise
4  */
5
6  #include <stdio.h>
7
8  int main() {
9      int i, j, sum_1, sum_2 = 0;
10     printf("sum of complete numbers less than 1000:\n");
11     for(i = 2; i <= 1000; i++) {
12         sum_1 = 0;
13         for(j = 1; j <= i-1; j++) {
14             if(i % j == 0) {
15                 sum_1 = sum_1 + j;
16             }
17         }
18         if(sum_1 == i) {
19             printf("%d + ", i);
20             sum_2 = sum_2 + i;
21         }
22     }
23     printf("= %d \n", sum_2);
24     return 0;
25 }
```


3.7.2 运行结果



```
~/C_Program_Design
Newton@Newton-PC-2 ~/C_Program_Design
$ gcc 4.9\ complete\ number.c -o 4.9.exe
Newton@Newton-PC-2 ~/C_Program_Design
$ ./4.9.exe
sum of complete numbers less than 1000:
6 + 28 + 496 + = 530
Newton@Newton-PC-2 ~/C_Program_Design
$ |
```

3.8 1 题

分别用三种循环语句（while 语句、do-while 语句、for 语句），实现求 1~100 的累加和。编程上机调试，总结出三种循环语句哪种实现起来方便、灵活。

3.8.1 程序代码

```
1  /*
2  * filename: 4.10 loop.c
3  * property: test
4  */
5
6  #include <stdio.h>
7
8  int main() {
9      int i = 1;
10     int s = 0;
11
12     // while
13     while (i <= 100) {
14         s += i++;
15     }
16     printf("while      sum(1~100) = %d\n", s);
17
18     i = 1;
19     s = 0;
20     // do while
21     do {
22         s += i++;
23
24     } while(i <= 100);
25     printf("do while  sum(1~100) = %d\n", s);
26
27     i = 0;
28     s = 0;
29     // for
30     for(; i <= 100; s += i++);
31     printf("for      sum(1~100) = %d\n", s);
32
33     return 0;
34 }
```

3.8.2 运行结果

```
~/C_Program_Design
Newton@Newton-PC-2 ~/C_Program_Design
$ gcc 4.10\ loop.c -o 4.10.exe

Newton@Newton-PC-2 ~/C_Program_Design
$ ./4.10.exe
while      sum(1~100) = 5050
do while   sum(1~100) = 5050
for        sum(1~100) = 5050

Newton@Newton-PC-2 ~/C_Program_Design
$ |
```

3.9 百元买鸡问题

已知公鸡每只 5 元，母鸡每只 3 元，小鸡 1 元 3 只，要求 100 元钱正好买 100 只鸡，则应买公鸡、母鸡的小鸡各多少只？

3.9.1 程序代码

```
1  /*
2  * filename: 4.11 chicken.cpp
3  * property: test
4  */
5
6  #include <iostream>
7
8  using namespace std;
9
10 int main() {
11     cout << "XY\t" << "XX\t" << "L" << endl;
12     int x, y, z;
13     for(x = 0; x <= 20; x++) {
14         for(y = 0; y <= 33; y++) {
15             z = 100 - x - y;
16             if(5*x + 3*y + z/3 == 100 && z % 3 == 0)
17                 cout << x << "\t" << y << "\t" << z << endl;
18         }
19     }
20     return 0;
21 }
```

3.9.2 运行结果

```
~/C_Program_Design
Newton@Newton-PC-2 ~/C_Program_Design
$ g++ 4.11\ chicken.cpp -o 4.11.exe

Newton@Newton-PC-2 ~/C_Program_Design
$ ./4.11.exe
XY      XX      L
0        25      75
4        18      78
8        11      81
12       4       84

Newton@Newton-PC-2 ~/C_Program_Design
$ |
```

3.10 成绩排序

某班有学生 n 人，从终端读入 n 及 n 个人学生的成绩，试编程实现以下功能：

- ①印出前 3 个最好成绩及取得每个最好成绩的人数；
- ②若 90 分以上计为 A，75—89 分计为 B，60—74 分计为 C，60 分以下计为 D，试统计各档成绩所占百分率。

3.10.1 程序代码

一般而言，这种动态数据类型都需要用指针来实现，对于初学者，往往用一个很大的数组，按照限制在其一个子块上进行操作就可以了。但是这种操作被认为是有害的，因为无法估计内存的情况，所以这个问题的实现放到后面的练习性实验报告里完成。到时候会涉及数据结构与算法的一些东西。

当然，这里也可以直接使用 C++ STL 的高级操作，但是作为初级报告，不再展开。

3.11 习题

1. 下列论题哪些是错误的？

C 语言没有 goto 语句。（×）

While 表达式语句的作用是：当表达式的值为 0 时重复执行循环体语句。（×）

do（语句）while（表达式）的作用是：重复执行循环体（“语句”），直到表达式成立（其值为真）。（×）

“do…while”语句中，写在 do 后面、While 前面的若干语句，不必用花括号括起来。（×）

break 语句用于退出条件语句和循环语句的判断。（√）

contiune 语句表示将循环继续下去。

凡是 while 语句能解决的问题也能用 do…while 语句解决。（√）

凡是用 while 语句能解决的问题都可以用 for 语句实现。（√）

凡是用 for 语句能解决的问题都可以用 while 语句实现。（×）

造成“死循环”的主要原因是循环变量的值没有得到必要的修改。（√）

3.12 程序排错

3.12.1 排错 1

$$\text{sum} = \sum_{i=1}^{100} \frac{1}{n}$$

3.12.1.1 程序代码

```
1  #include <stdio.h>
2  int main() {
3      int n, sum;
4      n = 1;
5      while(n<100) {
6          sum += n;
7          n++;
8      }
9      printf("sum=%f\n", sum);
```

```

10     return 0;
11 }

```

错误原因：数据类型设置错误。sum 是浮点型。

3.12.2 排错 2

从键盘输入若干学生的成绩（输入负分结束），输出平均成绩和最高分。

```

1  #include <stdio.h>
2
3  int main() {
4      int n = 0;
5      float grade, sum, max = 0;
6      scanf("%f", &grade);
7      while(grade >= 0) {
8          if(grade > max) {
9              max = grade;
10         }
11         sum = sum + grade;
12         n = n + 1;
13         scanf("%f", &grade);
14     }
15     grade = sum / n;
16     printf("max = %3.2f, a = %3.2f\n", max, grade);
17     return 0;
18 }

```

3.12.3 排错 3

计算并输出 $\sum_{n=1} (2n+1)$ 超过 1000 的第一个 n 值。

3.12.3.1 程序代码

```

1  #include <stdio.h>
2
3  int main() {
4      int n = 1, sum = 0;
5      for(; ; n++)
6          sum = sum + (2 * n + 1);
7          if(sum > 2000)
8              break;
9      printf("n = %d, sum = %d\n", n, sum);
10     return 0;
11 }

```

for 循环的语句超过了 1，所以要添加花括号

3.12.4 排错 4

求 2~1000 之间的全部素数（每行显示 10 个数）。

3.12.4.1 程序代码

```

1  #include <stdio.h>
2  #include <math.h>

```

```

3
4  int main() {
5      int m = 3, k, i, n = 1;
6      printf("%7d", 2);
7      do {
8          if(n % 10 == 0) {
9              printf("\n");
10             k=sqrt(m);
11         }
12         for(i = 2; i <= k; i++)
13             if(m % i == 0)
14                 continue;
15         if(i>=k+1) {
16             printf("%8d", m);
17             n++;
18         }
19     }
20     while(m > 1000);
21     printf("\n");
22     return 0;
23 }

```

错误:

- (1) 没有使得循环变量更改的语句;
- (2) 循环条件设置错误。第 23 行中应该是 $m < 1000$ 。

3.13 条件求和

$S(n) = a + aa + \cdots + a \cdots a$, 其中 a 是 1~9 中的一个数字。 n 为一正整数, a 和 n 均从键盘输入。(例如输入 n 为 4, a 为 8, $S_n = 8 + 88 + 888 + 8888$)。

3.13.1 程序代码

```

1  /*
2  * filename: 4.12 sum.c
3  * property: exercise
4  */
5
6  #include <stdio.h>
7  #include <math.h>
8
9  int main() {
10     int a, n;
11     float i, j, count = 0;
12     printf("please input a and n: ");
13     scanf("%d %d",&a,&n);
14     for(i = 1; i <= n; i++) {
15         for(j = 1; j <= i; j++) {
16             printf("%d", a);
17             count = count + pow(10, j-1) * a;
18             if((j == i) && (j!=n)) {

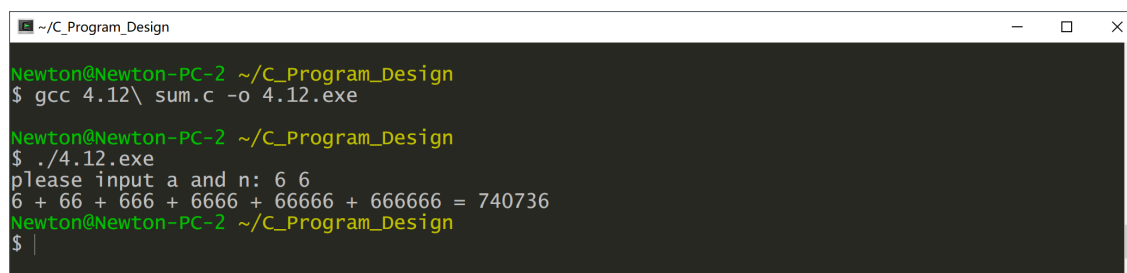
```

```

19         printf(" + ");
20     }
21 }
22 }
23 printf(" = %5.0f", count);
24 return 0;
25 }

```

3.13.2 运行结果



```

~/C_Program_Design
Newton@Newton-PC-2 ~/C_Program_Design
$ gcc 4.12\ sum.c -o 4.12.exe

Newton@Newton-PC-2 ~/C_Program_Design
$ ./4.12.exe
please input a and n: 6 6
6 + 66 + 666 + 6666 + 66666 + 666666 = 740736
Newton@Newton-PC-2 ~/C_Program_Design
$

```

3.14 打印菱形图案

```

      *
     * *
    * * *
   * * * *
  * * * * *
 * * * * *
  * * * *
   * * *
    * *
     *

```

3.14.1 程序代码

```

1  /*
2  * filename: 4.12 sum.c
3  * property: exercise
4  */
5
6  #include <stdio.h>
7  #include <math.h>
8
9  int main() {
10     int a, n;
11     float i, j, count = 0;
12     printf("please input a and n: ");
13     scanf("%d %d",&a,&n);
14     for(i = 1; i <= n; i++) {
15         for(j = 1; j <= i; j++) {
16             printf("%d", a);
17             count = count + pow(10, j-1) * a;
18             if((j == i) && (j!=n)) {
19                 printf(" + ");
20             }
21         }
22     }

```

```

23     printf(" = %5.0f", count);
24     return 0;
25 }

```

3.14.2 运行结果

```

~/C_Program_Design
Newton@Newton-PC-2 ~/C_Program_Design
$ gcc 4.13\ print\ ASCII.c -o 4.13.exe
Newton@Newton-PC-2 ~/C_Program_Design
$ ./4.13.exe
*
***
*****
*****
*****
***
*
Newton@Newton-PC-2 ~/C_Program_Design
$

```

3.15 求和

$$\sin x = \sum_{n=0}^{\infty} (-1)^n \frac{x^{2n+1}}{(2n+1)!}$$

的近似值，精确到 $\left| \frac{x^{2n+1}}{(2n+1)!} \right| < 10^{-6}$ 。

3.15.1 程序代码

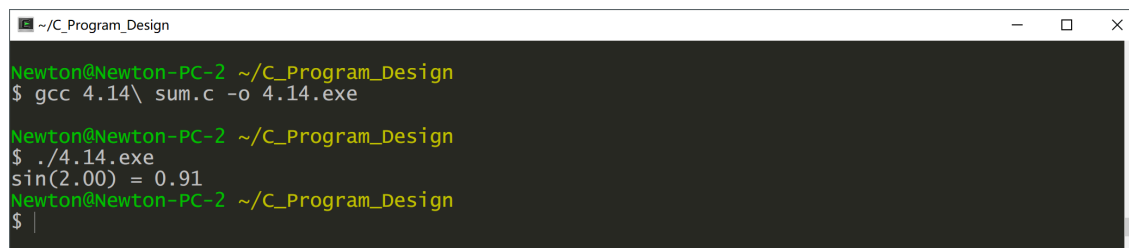
```

1  /*
2  * filename: 4.14 sum.c
3  * property: exercise
4  */
5
6  #include <stdio.h>
7  #include <math.h>
8
9  #define TYPE_ERROR -1
10
11 double factorial(double in) {
12     if(floor(in) != in) {
13         return -1;
14     }
15
16     double ans = 1;
17     while(in != 1) {
18         ans *= in;
19         in -= 1;
20     }
21
22     return ans;
23 }
24
25 int main() {
26     double i = 0;

```

```
27     double s = 0;
28     const double x = 2;    // sin(\pi) = 0
29
30     double test;
31     double low = 1;
32
33     for (; (test = pow(x, 2*i+1)/factorial(2*i+1)) >= 1e-6; i++) {
34         s += pow(-1, i) * test;
35         //printf("test = %3.4f\n", test);
36     }
37
38     printf("sin(%3.2f) = %3.2f", x, s);
39     return 0;
40 }
```

3.15.2 运行结果



```
~/C_Program_Design
Newton@Newton-PC-2 ~/C_Program_Design
$ gcc 4.14\ sum.c -o 4.14.exe

Newton@Newton-PC-2 ~/C_Program_Design
$ ./4.14.exe
sin(2.00) = 0.91
Newton@Newton-PC-2 ~/C_Program_Design
$ |
```

四、实验总结

循环问题，重在循环体的安排与设计。[1]

五、参考文献

1. Prata, S., *C++ Primer Plus*. 6th ed. 2012.

六、教师评语