

C语言经典算法100例

参考:

<https://blog.csdn.net/u012027907/article/details/12624829>

(1) 输出9*9乘法口诀。

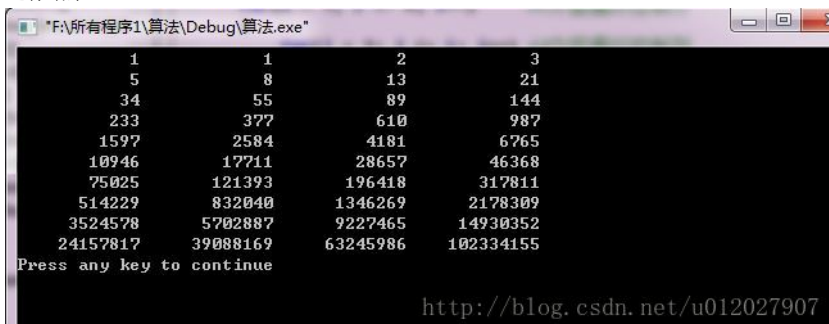
```
1. //9*9乘法口诀表
2. void Table99()
3. {
4.     int i,j;
5.     for(i = 1; i <= 9; i++)        //外层循环控制行
6.     {
7.         for(j = 1; j <= i; j++) //内层循环控制列
8.         {
9.             printf("%d*%d=%-4d",i,j,i*j);
10.        }
11.        printf("\n");
12.    }
13. }
```

运行结果:

(2) 古典问题：有一对兔子，从出生后第3个月起每个月都生一对兔子，小兔子长到第三个月后每个月又生一对兔子，假如兔子都不死，问每个月的兔子总数为多少？（兔子的规律为数列1, 1, 2, 3, 5, 8, 13, 21....）这也是著名的斐波那契数列。

```
1. //斐波那契数列
2. void Fabocci()
3. {
4.     long int f1,f2;
5.     f1 = f2 = 1;
6.     int i;
7.     for(i = 1; i <= 20; i++)
8.     {
9.         printf("%12ld %12ld ",f1,f2);
10.        if(i % 2 == 0)        //控制输出，每行输出4个
11.            printf("\n");
12.        f1 = f1+f2;            //后一个数是前两个数的和
13.        f2 = f1+f2;            //后一个数是前两个数的和
14.    }
15.
16. }
```

运行结果:



```
1      1      2      3
5      8      13     21
34     55     89     144
233    377    610    987
1597   2584   4181   6765
10946  17711  28657  46368
75025  121393 196418 317811
514229 832040 1346269 2178309
3524578 5702887 9227465 14930352
24157817 39088169 63245986 102334155
Press any key to continue
http://blog.csdn.net/u012027907
```

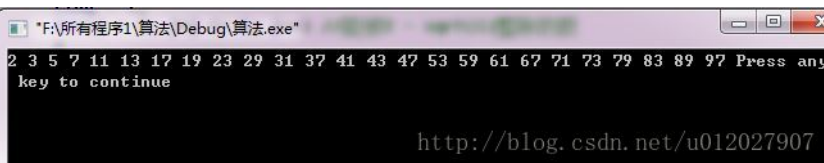
(3) 1-100之间有多少个素数，并输出所有素数及素数的个数。

程序分析：判断素数的方法：用一个数分别去除2到sqrt(这个数)，如果能被整除，则表明此数不是素数，反之是素数。

```
1. //输出1-100的所有素数
2. void Prime()
3. {
4.     int i,j,flag,n;
5.     n = 100;    //100以内的素数
6.     flag = 1;   //标识变量，是素数则为1
7.
8.     for(i = 2; i <= 100; i++)    //从2开始，遍历到100
9.     {
10.         flag = 1;
11.         for(j = 2; j*j <= i; j++) //能被2 - sqrt(i) 整除的数
12.         {
13.             if(i % j == 0)
14.             {
15.                 flag = 0;
16.                 break;
17.             }
18.         }
19.         if(flag == 1)
20.             printf("%d ",i);    //输出素数
21.     }
22. }
```

关于一个数是否是素数，还有更高效的算法，大家可以先考虑一下，以后我会给出算法。

运行结果：



(4) 一个数如果恰好等于它的因子之和，这个数就称为“完数”。例如 $6 = 1+2+3$

找出10000以内的所有完数。

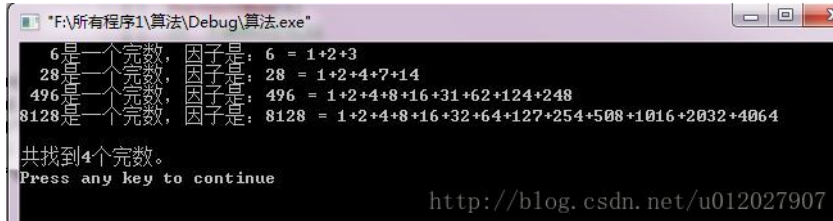
```
1. //找出10000以内的所有完数(一个数等于其因子之和)
2. void PerfectNumber()
3. {
4.     int p[80];    //保存分解的因子
5.     int i,num,count,s,c = 0;
6.     int MaxNum = 10000;
7.
8.     for(num = 2; num < MaxNum; num++)
9.     {
10.         count = 0;
11.         s = num;
12.         for(i = 1; i < num/2+1; i++)    //循环处理每个数
13.         {
14.             if(num % i == 0)    //能被i整除
15.             {
16.                 p[count++] = i;    //保存因子，让计数器count增加1
17.                 s -= i;    //减去一个因子
18.             }
19.         }
20.         if( 0 == s)
21.         {
```

```

22.     printf("%4d是一个完数, 因子是: ", num);
23.     printf("%d = %d", num, p[0]);    //输出完数
24.     for(i = 1; i < count; i++)
25.         printf("+%d", p[i]);
26.     printf("\n");
27.     c++;
28. }
29. }
30. printf("\n共找到%d个完数。 \n", c);
31. }

```

运行结果:



```

6是一个完数, 因子是: 6 = 1+2+3
28是一个完数, 因子是: 28 = 1+2+4+7+14
496是一个完数, 因子是: 496 = 1+2+4+8+16+31+62+124+248
8128是一个完数, 因子是: 8128 = 1+2+4+8+16+32+64+127+254+508+1016+2032+4064
共找到4个完数。
Press any key to continue

```

(5) 下面程序的功能是将一个 4×4 的数组进行逆时针旋转90度后输出, 要求原始数组的数据随机输入, 新数组以4行4列的方式输出。

```

1. void Array4_4()
2. {
3.     int A[4][4], B[4][4], i, j;
4.
5.     printf("Please Input 16 numbers:");
6.     for(i = 0; i < 4; i++)
7.         for(j = 0; j < 4; j++)
8.         {
9.             scanf("%d", &A[i][j]);    //输入16个数
10.            B[3-j][i] = A[i][j];    //旋转90度赋值
11.        }
12.     printf("Array A:\n");            //输出矩阵A
13.     for( i = 0; i < 4; i++)
14.     {
15.         for(j = 0 ; j < 4; j++)
16.         {
17.             printf("%4d", A[i][j]);
18.         }
19.         printf("\n");
20.     }
21.     printf("Array B:\n");            //输出矩阵B
22.     for( i = 0; i < 4; i++)
23.     {
24.         for(j = 0 ; j < 4; j++)
25.         {
26.             printf("%4d", B[i][j]);
27.         }
28.         printf("\n");
29.     }
30. }

```

运行结果:

```
"F:\所有程序1\算法\Debug\算法.exe"
Please Input 16 numbers:1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
Array A:
1 2 3 4
5 6 7 8
9 10 11 12
13 14 15 16
Array B:
4 8 12 16
3 7 11 15
2 6 10 14
1 5 9 13
Press any key to continue

http://blog.csdn.net/u012027907
```

(6) 编程打印杨辉三角。

```
1. //打印杨辉三角
2. void YangHuiTriangle()
3. {
4.     int i,j,triangle[8][8];
5.
6.     for(i = 0; i < 8; i++)
7.         for(j = 0; j < 8; j++)
8.             triangle[i][j] = 1;
9.
10.    for(i = 2; i < 8; i++)
11.    {
12.        for(j = 1; j < i; j++)
13.        {
14.            triangle[i][j] = triangle[i-1][j]+triangle[i-1][j-1];
15.        }
16.    }
17.    for(i = 0; i < 8; i++)
18.    {
19.        for(j = 0; j <= i; j++)
20.            printf("%-4d",triangle[i][j]);
21.        printf("\n");
22.    }
23.
24. }
```

运行结果:

```
"F:\所有程序1\算法\Debug\算法.exe"
1
1 1
1 2 1
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
Press any key to continue

http://blog.csdn.net/u012027907
```

(7) 实现将输入的字符串反序输出。

```
1. /*实现字符串翻转*/
2. char* reverse_str(char* str)
3. {
4.     if(NULL == str) //字符串为空直接返回
5.     {
6.         return str;
7.     }
8.     char *begin;
9.     char *end;
10.    begin = end = str;
11. }
```

```

12. while(*end != '\0') //end指向字符串的末尾
13. {
14.     end++;
15. }
16. --end;
17.
18. char temp;
19. while(begin < end) //交换两个字符
20. {
21.     temp = *begin;
22.     *begin = *end;
23.     *end = temp;
24.     begin++;
25.     end--;
26. }
27.
28. return str; //返回结果
29. }

```

运行结果:



(8) 实现字符串拷贝函数strcpy(char*src, char* dest)

```

1. void strcpy(char *str, char *dest)
2. {
3.     while(*str != '\0')
4.     {
5.         *dest++ = *str++;
6.     }
7.     *dest = '\0';
8. }

```

(9) 求近似Pi值。可以用公式 (如: $\pi/2 = 1 + 1/3 + 1/3 \cdot 2/5 + 1/3 \cdot 2/5 \cdot 3/7 + 1/3 \cdot 2/5 \cdot 3/7 \cdot 4/9 + \dots$)

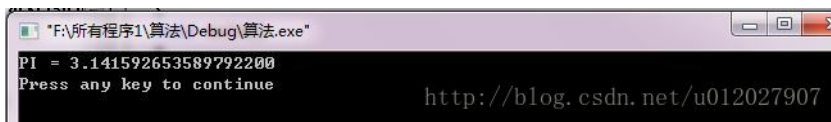
```

1. void Pi()
2. {
3.     double pi = 2, temp = 2;           //初始化pi值和临时值
4.     int numerator = 1, denominator = 3; //初始化分子和分母
5.
6.     while(temp > 1e-16)                 //数列大于指定精度
7.     {
8.         temp = temp * numerator / denominator; //计算一个数列的值
9.         pi += temp;
10.        numerator++;
11.        denominator += 2;
12.    }
13.    printf("PI = %.18f\n", pi);
14. }

```

这里求得的只是近似的值，精度不高，对于求任意位的pi值就无能为力了，大家可以考虑如何求任意位数的pi值，关于任意位数的pi值求法，可以参见我的博客：[《计算任意位数的Pi》](http://blog.csdn.net/u012027907)

运行结果:



(10) 输入一个字符串，判断其是否为回文。回文字符串是指从左到右读和从右到左读完全相同的字符串。

```
1. //判断一个字符串是否是回文
2. void IsHuiWen()
3. {
4.     char str[100];
5.     int i,j,n;
6.     printf("请输入一段字符串: ");
7.     gets(str);
8.     n = strlen(str);
9.     for(i = 0,j = n-1; i < j; i++,j--)
10.        if(str[i] != str[j])
11.            break;
12.     if(i >= j)
13.         printf("是回文!\n");
14.     else
15.         printf("不是回文!\n");
16.
17. }
```

运行结果:

