C语言程序设计

计算机科学与技术学院 王志军

循环结构程序设计

- 循环嵌套
- 转向语句
- 穷举与迭代
- 字符图形

循环嵌套

- 一个循环的循环体中套有另一个完整的循环结构称为循环嵌套
- 嵌套过程可以一直重复下去
- 一个循环外面包含一层循环称为双重循环
- 一个循环外面包含多于二层循环称为多重循环
- 三种循环do-while、while、for都可以互相嵌套组成多重循环

```
for(;  ;  ){
    …
  for(;  ;  ){    … }
…
}

for(;   ;  ) {
    …
   while(…){  …  }
    …
}
```

```
while(){
    …
  for(;  ;  )  { … }
    …
}

do{
…
  for( ;   ;  )   { … }
…
} while(…);
```

一天共有多少分钟?

```
#include <stdio.h>
int main (){
    int h, m, count = 0;
    for ( h = 0; h < 24; h++) {
        for ( m = 0; m < 60; m++) {
            count++;
        }
    }
    printf("%d", count);
    return(0);
}
```

◉一天共有多少分钟?

```
#include <stdio.h>
int main (){
    int h, m, count = 0;
    for ( h = 0; h < 24; h++) {
        for ( m = 0; m < 60; m++) {
            count++;
        }
    }
    printf("%d", count);
    return(0);
}
```

外循环#	内循环#	h	m	count
-	-	-	-	0
1	1	0	0->1	1
	2	0	1->2	2
	...	...	...	...
	60	0	59->60	60
2	1	1	0->1	61
	2	1	1->2	62
	...	...	...	...
	60	1	59->60	120
.....	.....	.....	.....	.....
24	1	23	0->1	xx
	2	23	1->2	xx
	...	...	...	...
	60	23	59->60	1440

一天共有多少分钟?

```c
#include <stdio.h>
int main (){
    int h, m, count = 0;
    for ( h = 0; h < 24; h++) {
        for ( m = 0; m < 60; m++) {
            count++;
        }
    }
    printf("%d", count);
    return(0);
}
```

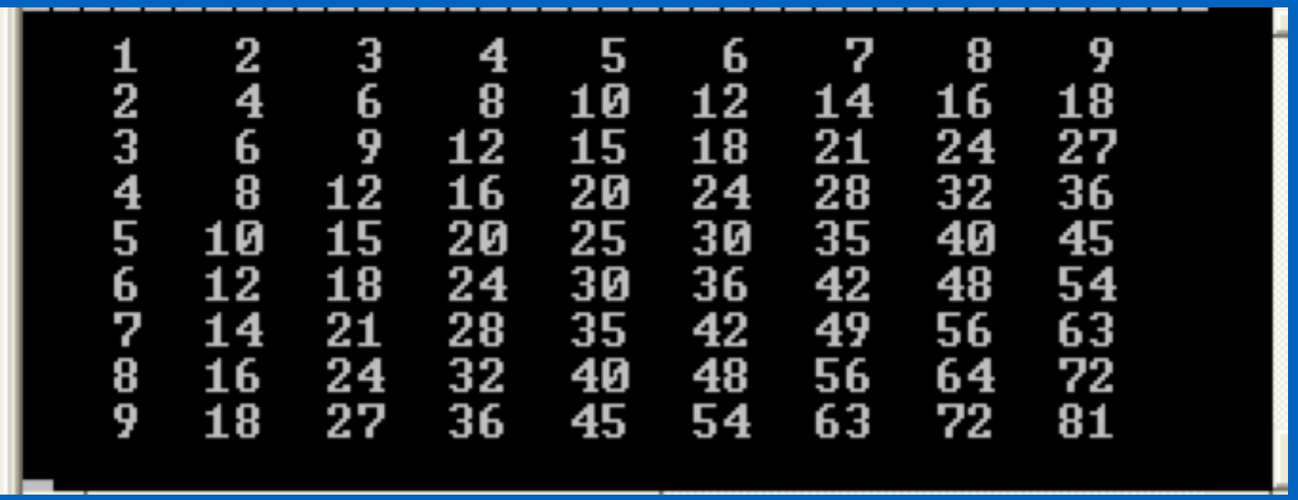
```c
#include <stdio.h>
int main (){
    int h = 0, m = 0, count = 0;
    while ( h < 24) {
        do {
            count++;
            m++;
        } while ( m < 60)
        h++;
    }
    printf("%d", count);
    return(0);
}
```

一百年（每月按30天计）共有多少天?

```
#include <stdio.h>
int main (){
    int y, m, d, count = 0;
    for ( y = 0; y < 100; y++) {
        for ( m = 0; m < 12; m++) {
            for ( d = 0; d < 30; d++) {
                count++;
            }
        }
    }
    printf(“%d”, count);
    return(0);
}
```

- 演示多重循环
- 输出：36000

输出九九表



```
 1   2   3   4   5   6   7   8   9
 2   4   6   8  10  12  14  16  18
 3   6   9  12  15  18  21  24  27
 4   8  12  16  20  24  28  32  36
 5  10  15  20  25  30  35  40  45
 6  12  18  24  30  36  42  48  54
 7  14  21  28  35  42  49  56  63
 8  16  24  32  40  48  56  64  72
 9  18  27  36  45  54  63  72  81
```

- 自顶向下，逐步求精:

```
循环 9 次 （行） {
    每次打印 1 行:
}
```

```
循环 9 次 （行） {
    循环 9 次（列）
        {每次打印 行号*列号 （数值）}
    printf("\n");
}
```

输出九九表

```
循环 9 次 （行） {
    循环 9 次（列）
        {每次打印 行号*列号 （数值）}
    printf("\n");
}
```

```
#include <stdio.h>
int main (){
    int i, j;// i:行号, j:列号
    for(i = 1; i < 10; i++){
        for(j = 1; j < 10;j++)
            printf("%4d",i * j);
        //外循环? 内循环?
        printf("\n");
    }
    return(0);
}
```

循环结构程序设计

- 循环嵌套
- 转向语句
- 穷举与迭代
- 字符图形

转向语句

- 在指令中实现跳转（JMP)
- goto：无条件跳转
- break：立即终止循环语句或switch语句
- continue：在循环语句中，中止本次循环

goto语句

- 计算s=1+2+3+ ……+100

```
#include <stdio.h>
int main (){
      int i,sum=0;
      i=1;

loop: if(i<=100){
        sum+=i;
        i++;
        goto loop;
      }

      printf("%d",sum);
      return(0);
}
```

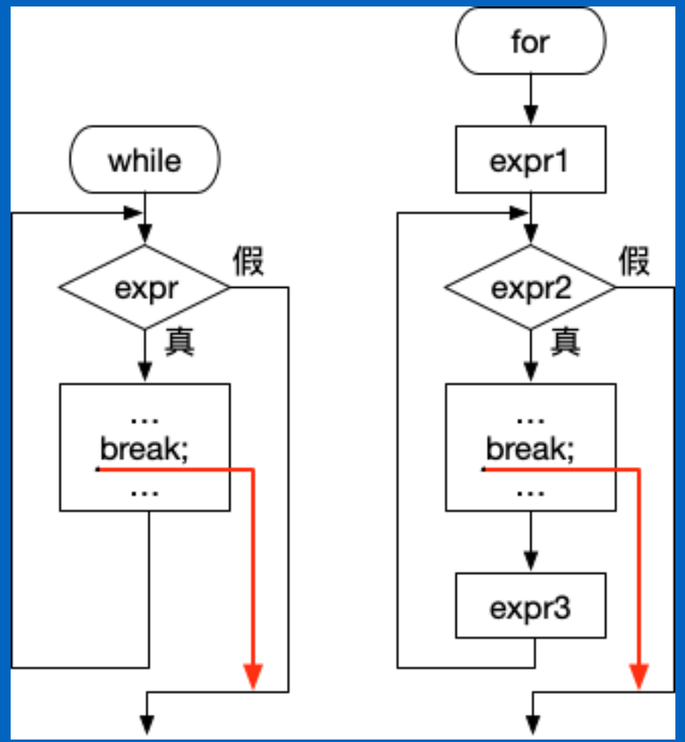
- 一般格式：

```
        ……
        goto 标号;
        ……
标号：语句;
        ……
```

- 标号
  - 不能用整数作标号
  - 只能出现在goto所在函数内，且唯一
  - 只能加在可执行语句前面
- 不建议使用goto语句
  - 实在要用，也只可从前面goto到结尾

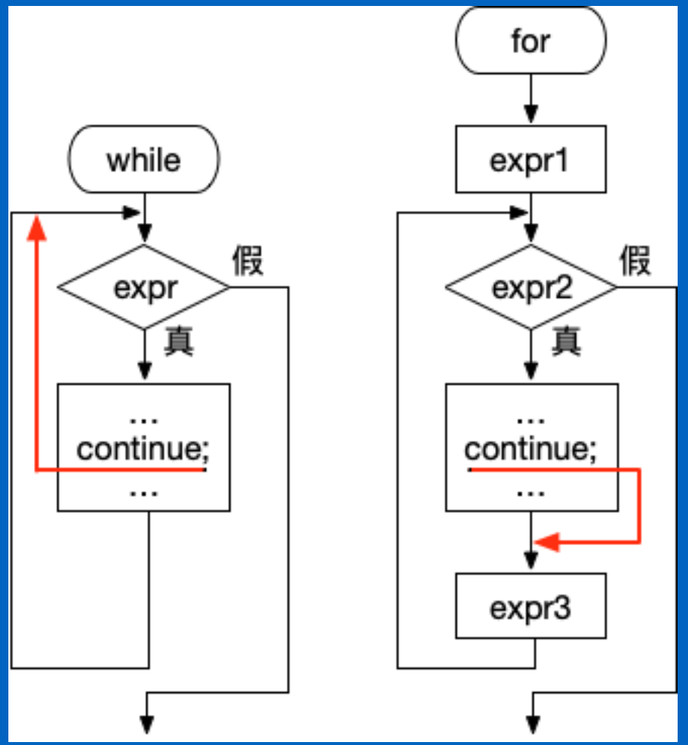
break语句

- 终止循环语句（执行其后的其他语句）
- 只能用于循环语句或switch语句之中
- 多重循环中，break语句只能跳出其所在的一层循环



continue语句

- 结束本次循环，跳过循环体中尚未执行的语句，进行下一次是否执行循环体的判断
- 只能用于循环语句



break 与 continue 区别

```c
#include <stdio.h>
int main (){
    int i;

    for(i = 1; i <= 5; i++) {
        printf("%d",i);
        break;
        //立即终止for循环语句
        //转到printf("Done.")

        printf("\n");
    }

    return(0);
}
```

输出：1

```c
#include <stdio.h>
int main (){
    int i;

    for(i = 1; i <= 5; i++) {
        printf("%d",i);
        continue;
        //中止本次循环
        //转到i<=5
        //判断下一次循环是否发生
        printf("\n");
    }

    return(0);
}
```

输出：12345

break 与 continue 区别

```c
#include <stdio.h>
int main (){

    int i, a, num=0;
    for(i = 0; i < 5; i++){
        scanf("%d",&a);
        if(a <= 0) break;
        num++;
    }
    printf("%d", num);

    return(0);
}
```

输入：1 2 3 -1
输出：3

```c
#include <stdio.h>
int main (){

    int i, a, num=0;
    for(i = 0; i < 5; i++){
        scanf("%d",&a);
        if(a <= 0) continue;
        num++;
    }
    printf(”%d”, num);

    return(0);
}
```

输入：1 2 3 -1 8
输出：4

循环结构程序设计

- 循环嵌套
- 转向语句
- 穷举与迭代
- 字符图形

穷举法

- 穷举法也称为“枚举法”，即对可能出现的每一种情况逐一进行测试，判断是否满足条件
- 常常采用循环来实现
- 算法简单，容易理解，但运算量大
- 通常可以解决“有几种组合”,“是否存在”,“求解不定方程”等类型的问题

穷举法

- 如果一个箱子的密码为2位数字，假设密码为12，编写程序进行密码破译。

```c
#include<stdio.h>
int  main( ){
    int m = 1,n = 2; //变量m和n表示密码12
    int i, j;     //i,j:循环变量,十位、个位

    for (i = 0; i <= 9; i++) {
        for (j = 0; j <= 9; j++) {
            if (i == m && j == n){
                printf("密码破解成功! 密码为%d%d\n", i,j);
                break;
            }
        }
    }
    return(0);
}
```

穷举法：完全平方数

- 求出1至500之间的完全平方数，要求每行输出8个数。
- 完全平方数是指能够表示成另一个整数的平方的整数。

```
  1    4    9   16   25   36   49   64
 81  100  121  144  169  196  225  256
289  324  361  400  441  484
```

穷举法：完全平方数

```c
#include<stdio.h>
int  main( ){

    int i, j, count = 0;

    for(i = 1; i <= 500; i++){

        for(j = 1; j <= i; j++)
            if(i == j * j){
                count++;
                break;
            }

        if(j <= i){
            printf("%5d",i);
            if(count % 8 == 0)
                printf("\n");
        }

    }
    return(0);
}
```

```c
#include<stdio.h>
int  main( ){

    int i, count = 0;

    for(i = 1;i * i <= 500;i++){
            printf("%5d",i * i);
            count++;
            if(count % 8 == 0)
                printf("\n");
    }

    return(0);
}
```

迭代法

- 迭代法也称辗转法，其基本思想是把一个复杂的计算过程转化为简单过程的多次重复
- 每次重复都从旧值的基础上递推出新值，并由新值代替旧值，这是一种不断用变量的旧值递推新值的过程

迭代法

- 判断用户输入的数字是否为素数（prime)
- 素数：大于1并且只能被自身和1整除的自然数
  - m是素数: 从 2 到 $\lfloor\sqrt{m}\rfloor$ 都不能整除 m
  - ANSI C中求平方根函数：sqrt() （定义在<math.h>中）
- 为何要判断素数?

素数用处之非对称加密算法:RSA

- 一对密钥：加密用公钥，解密有私钥
- 1977年，由Ron Rivest, Adi Shamir和Leonard Adleman发明
- 用于数字证书等
- 产生一个公钥和一个私钥：
  - 随意选择两个大的素数p和q，p不等于q，计算N=pq
  - 选择一个整数e与(p-1)(q-1)互质，并且e小于(p-1)(q-1)
  - 计算d：(d * e) ≡ 1 (mod ((p-1)(q-1)))
  - (N,e)是公钥，(N,d)是私钥
  - 特点：已知N时，难以分解出p和q
- 使用RSA加密时，选用足够长的密钥（一般至少使用1024位）

RSA-768

- 2009年12月12日被分解

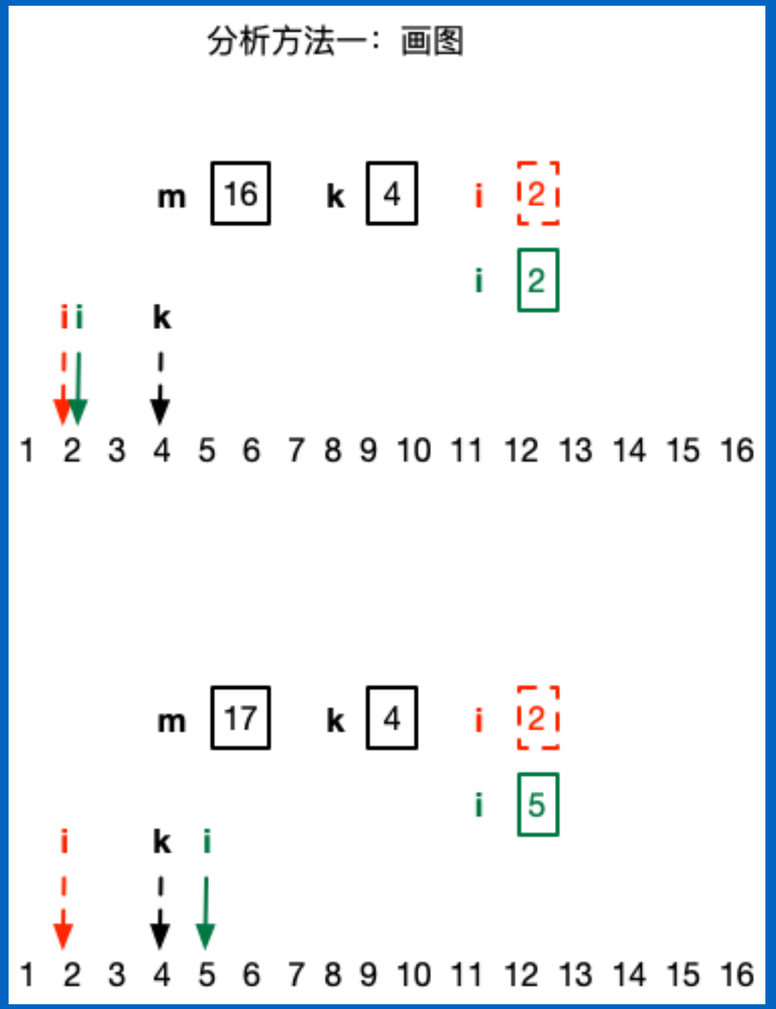
123018668453011775513049495838496272077285356959533479219732245215172640050726365751874520219978646938995647494277406384592519255732630345373154826850791702612214291346167042921431160222124047927473779408066535141959745985690214341 3
=
33478071698956898786044169848212690817704794983713768568912431388982883793878002287614711652531743087737814467999489
×
36746043666799590428244633799627952632279158164343087642676032283815739666511279233373417143396810270092798736308917

- 768位：数百cpu，2年
- 2048位：超级计算机，80年

迭代法

- 编程判断m是否为素数。

```
#include <stdio.h>
#include <math.h>
int main (){

    int m,k,i;
    scanf("%d",&m);
    k=(int)sqrt(m);

    for(i = 2; i <= k; i++)
        if(m % i == 0) break;

    //若i到达k+1，说明未发生过break;
    if(i >= k+1)
        printf("%d is a prime.",m);
    else
        printf("%d is not  a prime.",m);

    return(0);
}
```



迭代法

```c
#include <stdio.h>
int main (){
    int f0 = 0,f1 = 1,fn;//数列的第1、第2项
    int i;//循环变量i

    printf("%6d%6d", f0, f1);//输出前2项

    for(i = 2; i < 20; i++){
        //输出5个数则换行
        if(i % 5 == 0) printf("\n");
        //计算第i项
        fn = f0 + f1;
        printf("%6d", fn);
        //更新前两项的值
        f0 = f1;
        f1 = fn;
    }
    return(0);
}
```

- 编程求斐波纳契(Fibonacci)数列 0,1,1,2,3,5,8...的前20项,每输出5个数换一行。
  - 数列的生成方法为：$F_0=0$，$F_1=1$，$F_n=F_{n-1}+F_{n-2}$（n≥3），即从第3个数开始，每个数等于前2个数之和

分析方法二：列表

循环#	i	f0	f1	fn
-	-	0	1	-
1	2->3	1	1	1
2	3->4	1	2	2
3	4->5	2	3	3
.....	.....	.....	.....	.....
18	19->20	2584	4181	4181
--19--	--	--	--	--

迭代法

◉ 编程求斐波纳契(Fibonacci)数列 0,1,1,2,3,5,8...的前20项,每输出5个数换一行。

```c
#include <stdio.h>
int main (){
    int f0 = 0, f1 = 1,fn;//数列的第1、第2项
    int i;//循环变量i

    printf("%6d%6d", f0, f1);//输出前2项

    for(i = 2; i < 20; i++){
        //输出5个数则换行
        if(i % 5 == 0) printf("\n");
        //计算第i项
        fn = f0 + f1;
        printf("%6d", fn);
        //更新前两项的值
        f0 = f1;
        f1 = fn;
    }
    return(0);
}
```

```c
#include <stdio.h>
int main (){
    int f0 = 0, f1 = 1;//数列的第1、第2项
    int i;//循环变量i

    printf("%6d%6d", f0, f1);//输出前2项

    for(i = 2; i < 20; i++){
        //输出5个数则换行
        if(i % 5 == 0) printf("\n");
        //计算第i项
        f1 = f0 + f1;
        printf("%6d", f1);
        //更新前一项的值
        f0 = f1-f0;
    }
    return(0);
}
```

迭代法

- 编程把输入的整数（0不在首位和末位），按相反的顺序输出。

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

    scanf("%d",&m);

    while(m != 0) {
        n = m % 10;
        printf("%d",n);
        m /= 10;


  }



  return(0);
}
```

输入：123

输出：321

- 编程把输入的整数（0不在首位和末位），按相反的顺序组成一个新的数字并输出。

```
#include <stdio.h>
int main (){
    int m, n;
    int r = 0;
    scanf("%d",&m);

    while(m != 0) {
        n = m % 10;

        m /= 10;
        r = r * 10 + n;

  }

  printf("%d",r);

  return(0);
}
```

输入：123

输出：321

迭代法

◉ 累加/累乘：求s=a+aa+aaa+……+aaa ...a之值，其中最后一项由n个a组成，a和n由用户在程序运行时输入。

```c
#include <stdio.h>
int main (){
    int i, n, a;
    int t, s = 0;
    scanf("%d,%d",&n,&a);
    // t: 加数
    t = 0;
    for (i = 1; i <= n; i++) {
        t = t * 10 + a;
        s += t;
    }
    printf("s=%d",s);
    return(0);
}
```

编程求$1+x+x^2/2!+x^3/3!+.....x^n/n!$

- （1）求x^i

```
#include<stdio.h>
int  main( ){
    int j, x, t;
    scanf("%d",&i);
    scanf("%d",&x);

    t = 1;
    for(j = 1; j <= i; j++)
        t *= x;

    printf("%f",t);
}
```

- （2）求i!

```
#include<stdio.h>
int  main( ){
    int j, f;
    scanf("%d",&i);

    f = 1;
    for(j = 1; j <= i; j++)
        f *= j;

    printf("%f",f);
}
```

- （3）求$1+x+x^2/2!+x^3/3!+.....x^n/n!$

```
#include<stdio.h>
int  main( ){
    int i,j,x,n,t,f;
    float expr = 1;
    scanf("%d",&n);
    scanf("%d",&x);
    for(i = 1; i <= n; i++){
        t = 1;
        for(j = 1; j <= i; j++)
            t *= x;
        f = 1;
        for(j = 1; j <= i; j++)
            f *= j;
        expr += (float)t / f ;
    }
    printf("expr=%f",expr);
    return(0);
}
```

编程求1+x+x^2/2!+x^3/3!+……x^n/n!

◉ （3）求1+x+x^2/2!+x^3/3!+……x^n/n!

```
#include<stdio.h>
int  main( ){
    int i,j,x,n,t,f;
    float expr = 1;
    scanf("%d",&n);
    scanf("%d",&x);
    for(i = 1; i <= n; i++){
        t = 1;
        for(j = 1; j <= i; j++)
            t *= x;
        f = 1;
        for(j = 1; j <= i; j++)
            f *= j;
        expr += (float)t / f ;
    }
    printf("expr=%f",expr);
    return(0);
}
```

◉ （4）求1+x+x^2/2!+x^3/3!+……x^n/n!

```
#include<stdio.h>
int  main( ){
    int i,x,n,t,f;
    float expr = 1;
    scanf("%d",&n);
    scanf("%d",&x);
    t = 1;
    f = 1;
    for(i = 1; i <= n; i++){
        t *= x;
        f *= i;
        expr += (float)t / f ;
    }
    printf("expr=%f",expr);
    return(0);
}
```

循环结构程序设计

- 循环嵌套
- 转向语句
- 穷举与迭代
- 字符图形

- 以下程序执行后的输出结果是什么?

```
#include<stdio.h>
int  main( ){
    int i, j;
    for(i = 1; i <= 6; i++){

        for(j = 1; j <= 6 - i; j++)
            printf(" ");

        for(j = 1; j <= 2 * i - 1; j++)
            printf("*");

        printf("\n");

    }
    return(0);
}
```

字符图形

- 以下程序执行后的输出结果是什么?

```
#include<stdio.h>
int  main( ){
    int i, j;
    for(i = 1; i <= 6; i++){

        for(j = 1; j <= 6 - i; j++)
            printf(" ");

        for(j = 1; j <= 2 * i - 1; j++)
            printf("*");

        printf("\n");

    }
    return(0);
}
```

外循环#	i	j	输出
1	1	for1:1->6-1	␣␣␣␣␣
	1	for2:1->2*1-1	*
	1	2	\n
2	2	for1:1->6-2	␣␣␣␣
	2	for2:1->2*2-1	***
	2	4	\n
.....	.....	.....	.....
6	6	for1:1->0 (未进入循环)	
	6	for2:1->2*6-1	**********
	6	12	\n

字符图形: 三角形

- 以下程序执行后的输出结果是什么?

```
#include<stdio.h>
int  main( ){
    int i, j;
    for(i = 1; i <= 6; i++){

        for(j = 1; j <= 6 - i; j++)
            printf(" ");

        for(j = 1; j <= 2 * i - 1; j++)
            printf("*");

        printf("\n");

    }
    return(0);
}
```

外循环#	i	j	输出
1	1	for1:1->6-1	␣ ␣ ␣ ␣ ␣
	1	for2:1->2*1-1	*
	1	2	\n
2	2	for1:1->6-2	␣ ␣ ␣␣
	2	for2:1->2*2-1	***
	2	4	\n
.....	.....	.....	.....
6	6	for1:1->0 (未进入循环)	
	6	for2:1->2*6-1	**********
	6	12	\n

字符图形：字母三角形

- 以下程序执行后的输出结果是什么?

```
#include<stdio.h>
int  main( ){
    int i, j;
    char ch = "A";
    for(i = 1; i <= 6; i++){

        for(j = 1; j <= 6 - i; j++)
            printf(" ");

        for(j = 1; j <= 2 * i - 1; j++)
            printf("%c", ch + i - 1);

        printf("\n");

    }
    return(0);
}
```

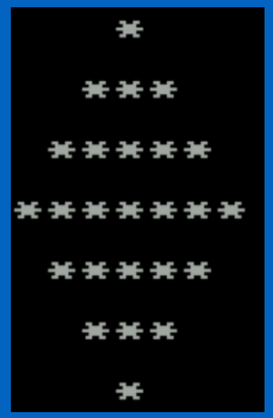
字符图形：字母倒三角形

- 以下程序执行后的输出结果是什么?

```
#include<stdio.h>
int  main( ){
    int i, j;
    char ch = "F";
    for(i = 1; i <= 6; i++){

        for(j = 1; j <= i - 1; j++)
            printf(" ");
        // j的终值: ?-2*6=1
        for(j = 1; j <= 13 - 2 * i; j++)
            printf("%c", ch  + 1 - i);

        printf("\n");

    }
    return(0);
}
```

字符图形：钻石形

- 如何打印出以下图形?
  - 提示：分上、下两部分，分别打印两个三角形



字符图形：$y=x^2$

```c
#include<stdio.h>
int  main( ){

    int i, j, y;
    // 字符方式下，屏幕默认25行80列
    // 以屏幕左上角为原点
    // 以屏幕左侧一边作为 x 轴
    for(i = 0; i <= 24; i++){
        // 缩小10倍
        y = i * i / 10;
        // 最多输出 y-1 个空格
        for(j = 0; j < y; j++){
            printf(" ");
        }
        // 输出 1 个 * 后换行
        printf("*\n");
    }

    return(0);
}
```