



# 第5讲 穷举和递推

## 教材6.5节

### MOOC第5周

哈尔滨工业大学

苏小红

sxh@hit.edu.cn

# 何为算法？

## ■ 平方根的数学定义

- \* 对于一个数 $x$ ，如果有另外一个数 $r$ ， $r \geq 0$ ，并且 $r^2 = x$ ，则 $r$ 就是 $x$ 的平方根
- \* 只是描述了平方根**是什么**，并未告诉具体的**计算过程**

```
double square_root(double x)
{
    给朕找一个r，必须确保r的平方等于x
    return r;
}
```



- \* 在当前的计算机体系下，这是不可能完成的任务
- \* 程序员必须告诉计算机**怎么做**，而不是**做什么**

# 何为算法？



图片来源：视觉中国 [www.vcg.com](http://www.vcg.com)

- **是什么**和**怎么做**之间存在的巨大鸿沟，需要程序员用大脑去填补

# 何为算法？

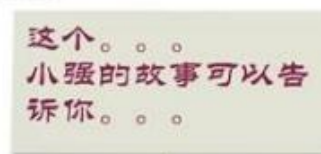
## ■ 求平方根的算法—描述了计算机求平方根应该怎么做

- \* 先猜测 $r=1$ ，判断 $r$ 的平方和 $x$ 是不是足够接近（例如相差 $0.0001$ ）
- \* 若不接近，则让 $r=(r+x/r)/2$ ，继续判断 $r$ 的平方与 $x$ 是否足够接近

```
double square_root(double x)
{
    r = 1;    //确定迭代变量
    while (fabs(r*r-x) > 0.0001) //对迭代过程进行控制
    {
        r = (r + x / r) / 2;    //建立迭代关系式
    }
    return r;
}
```

# 黑客破解密码的常用方法是什么？

- 不建议用纯数字设置密码
  - \*  $n$ 位数的可能性：最多尝试 $10^n$ 次
  - \* 理论上可破解任何密码
- 问题在于如何缩短试误时间
  - \* 设置可容许的试误次数





# 穷举法求解问题的两个基本要素

- 某大一狗想登录QQ，可他忘了密码，他只记得密码是一个4位数即96xx，且该数是8和6的倍数。现在，请你帮捉急的他找回密码。
  - \* 搜索范围： 9600~9699
  - \* 判定条件： 能被6、8整除
- 生活中还有哪些用穷举法的例子？
  - \* 在一堆钥匙中寻找唯一可以开门的



# 穷举法求解问题的两个基本要素

确定穷举对象  
和穷举范围



- 影响算法的时间复杂度
- 循环结构实现

确定  
判定条件

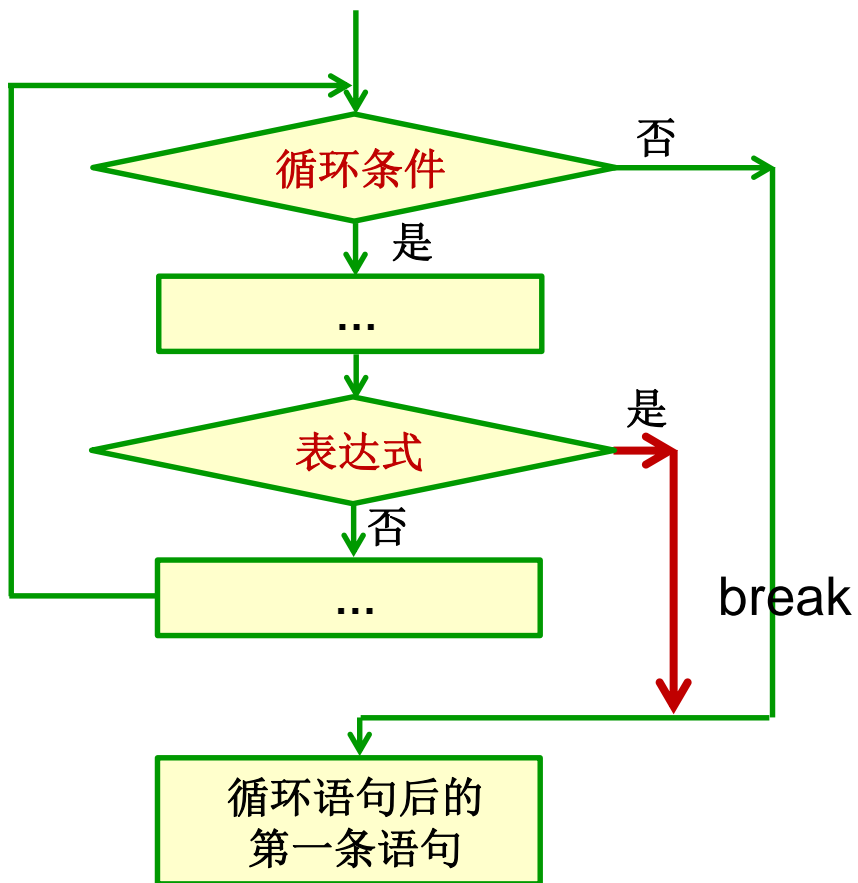


- 符合答案的条件
- 分支结构实现

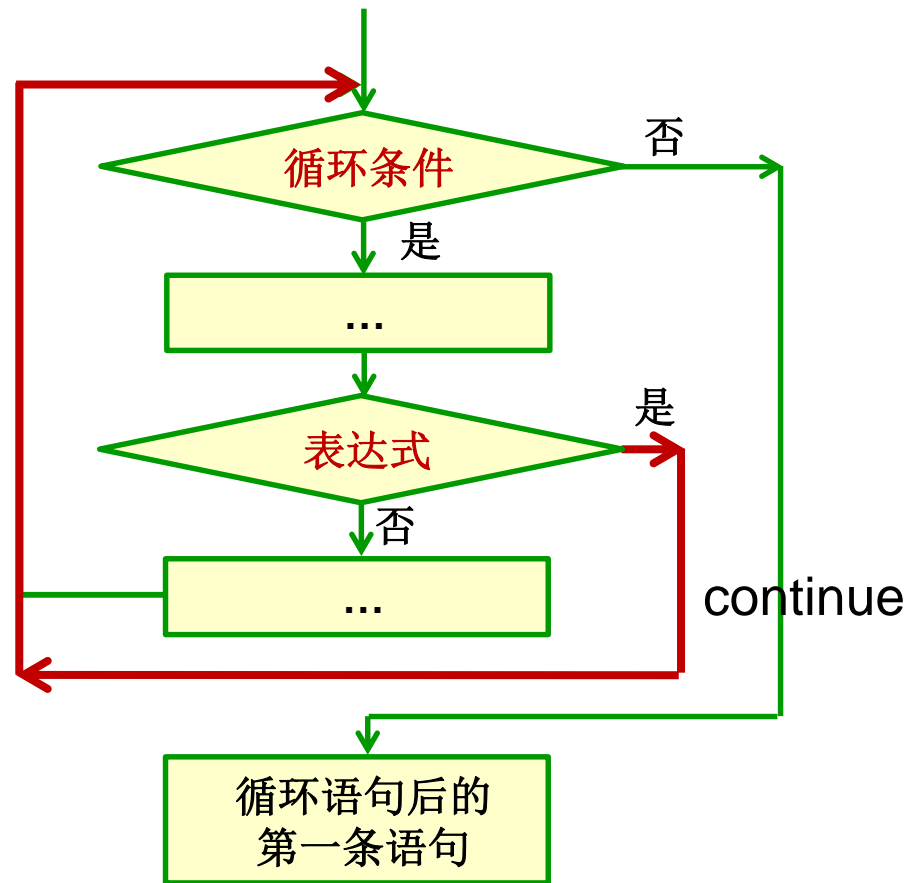
```
for (穷举范围)
{
    .....
    if (符合答案的条件)
    {
        .....
    }
}
```

找到符合答案的解  
如何退出循环?

# continue语句与break语句



退出一层循环或switch，  
转到闭合循环之后的那一点



中断此次循环，  
开始下一次循环

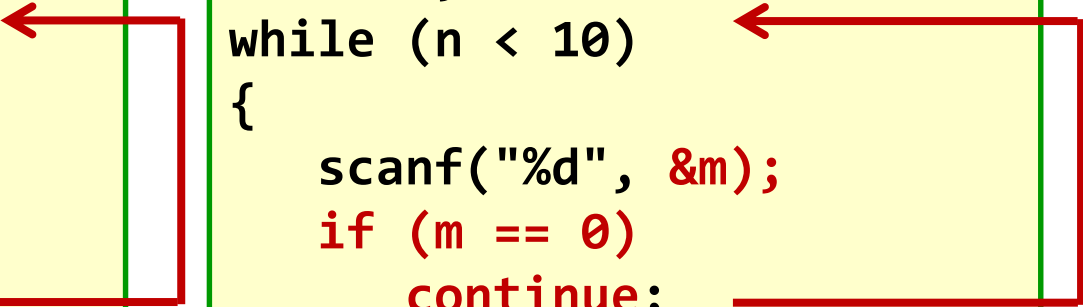


# continue对while和for循环的影响？

- 大多数for循环可以转换为while循环
- 但并非全部，例如当循环体中有continue时

```
sum = 0;
for (n=0; n<10; n++)
{
    scanf("%d", &m);
    if (m == 0)
        continue;
    sum = sum + m;
}
```

```
n = 0;
sum = 0;
while (n < 10)
{
    scanf("%d", &m);
    if (m == 0)
        continue;
    sum = sum + m;
    n++;
}
```



# goto语句的是与非

```
START_LOOP:
if (fStatusOk)
{
    if (fDataAvaible)
    {
        i = 10;
        goto MID_LOOP;
    }
    else
    {
        goto END_LOOP;
    }
}
else
{
    for (i = 0; i < 100; i++)
    {
        MID_LOOP:
        // lots of code here
        goto START_LOOP;
    }
}
END_LOOP:
```

\* 跳出多重循环的一条捷径

```
* {...
    {...
        goto END;
    }
}
END:
```

## 要好结构

goto语句破坏了程序的清晰结构和可读性

破坏了“单入口、单出口”的原则

使程序正确性证明复杂化  
编译器很难优化程序

正方 VS 反方



## 要高效率

goto语句使用灵活，有时可以提高程序的执行效率（例如跳出深层的循环）

在单入口、单出口的模块内使用前跳的goto，可减少重复代码，增强可读性

混乱的根源并非goto本身，而在于使用了较多的goto语句标号  
有无goto语句，并不是程序结构好坏的标志  
有害，但保留



D. E. Knuth（图灵奖1974）

# 怎样设计“好”结构的程序？

- 结构化程序设计
- Structured Programming



D. E. Knuth (图灵奖1974)

限制和避免使用goto语句  
采用“自顶向下逐步求精”方法进行程序设计

采用顺序、选择和循环三种基本控制结构作为程序设计的基本单元

原则  
方法  
特点

严格遵循“单入口单出口”的原则，无死循环，无死语句

结构清晰  
容易阅读  
容易修改  
容易验证

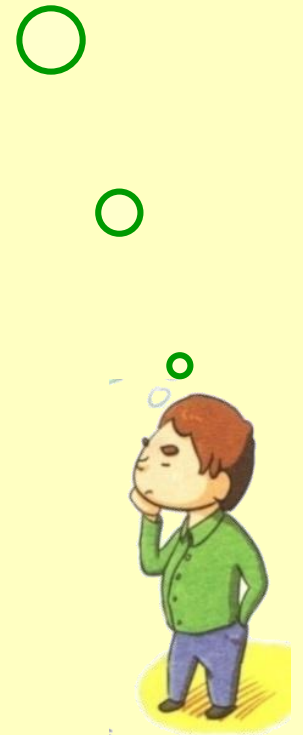
# 动物百米赛跑

- 小狗、小兔、小猫、小猴和小鹿参加百米赛跑，比赛结束后，
- 小猴说：“我比小猫跑得快”  $\text{monkey} < \text{cat}$
- 小狗说：“小鹿在我的前面冲过了终点线”  $\text{deer} < \text{dog}$
- 小兔说：“我的名次排在小猴的前面，小狗的后面”
- 请根据他们的回答编写程序排出名次。  $\text{rabbit} < \text{monkey} \ \&\& \ \text{rabbit} > \text{dog}$ 
  - \* 确定问题的输入输出：输入：无；输出：  $\text{dog}, \text{rabbit}, \text{cat}, \text{monkey}, \text{deer}$
  - \* 确定穷举对象：每个动物的名次：  $\text{dog}, \text{rabbit}, \text{cat}, \text{monkey}, \text{deer}$
  - 确定搜索范围：1, 2, 3, 4, 5
  - \* 如何确定判定条件？



```
#include <stdio.h>
int main()
{
    int dog, rabbit, cat, monkey, deer;
    for (dog=1; dog<=5; dog++)
    {
        for (rabbit=1; rabbit<=5; rabbit++)
        {
            for (cat=1; cat<=5; cat++)
            {
                for (monkey=1; monkey<=5; monkey++)
                {
                    deer = 15 - dog - rabbit - cat - monkey;
                    if (monkey<cat && deer<dog
                        && (rabbit<monkey&&dog<rabbit))
                    {
                        printf("%d,%d,%d,%d,%d\n",
                            dog,rabbit,cat,monkey,deer);
                    }
                }
            }
        }
    }
    return 0;
}
```

程序循环了多少次  
？效率如何？  
如何对循环加速？



```

#include <stdio.h>
int main()
{
    int dog, rabbit, cat, monkey, deer;
    for (dog=1; dog<=5; dog++)
    {
        for (rabbit=1; rabbit<=5; rabbit++)
        {
            for (cat=1; cat<=5; cat++)
            {
                for (monkey=1; monkey<=5; monkey++)
                {
                    deer = 15 - dog - rabbit - cat - monkey;
                    if (monkey<cat && deer<dog
                        && (rabbit<monkey&&dog<rabbit))
                    {
                        printf("%d,%d,%d,%d,%d\n",
                               dog,rabbit,cat,monkey,deer);
                        break; //加上管用吗? ? ? ? ? ? ?
                    }
                }
            }
        }
    }
    return 0;
}

```





```

#include <stdio.h>
int main()
{
    int dog, rabbit, cat, monkey, deer;

    for (dog=1; dog<=5; dog++)
    {
        for (rabbit=1; rabbit<=5; rabbit++)
        {
            for (cat=1; cat<=5; cat++)
            {
                for (monkey=1; monkey<=5; monkey++)
                {
                    deer = 15 - dog - rabbit - cat - monkey;
                    if (monkey<cat && deer<dog
                        && (rabbit<monkey&&dog<rabbit))
                    {
                        printf("%d,%d,%d,%d,%d\n",
                               dog,rabbit,cat,monkey,deer);
                        goto END;
                    }
                }
            }
        }
    }
    END: return 0;
}

```



```

#include <stdio.h>
int main()
{
    int dog, rabbit, cat, monkey, deer;
    int flag = 0;
    for (dog=1; dog<=5&&!flag; dog++)
    {
        for (rabbit=1; rabbit<=5&&!flag; rabbit++)
        {
            for (cat=1; cat<=5&&!flag; cat++)
            {
                for (monkey=1; monkey<=5&&!flag; monkey++)
                {
                    deer = 15 - dog - rabbit - cat - monkey;
                    if (monkey<cat && deer<dog
                        && (rabbit<monkey&&dog<rabbit))
                    {
                        printf("%d,%d,%d,%d,%d\n",
                               dog,rabbit,cat,monkey,deer);
                        flag = 1; //标记找到了
                    }
                }
            }
        }
    }
    return 0;
}

```



```

int dog, rabbit, cat, monkey, deer;
int flag = 0;
for (dog=1; dog<=5&&!flag; dog++)
{
    for (rabbit=1; rabbit<=5&&!flag; rabbit++)
    {
        if (dog != rabbit)
        for (cat=1; cat<=5&&!flag; cat++)
        {
            if (dog != cat && rabbit != cat)
            for (monkey=1; monkey<=5&&!flag; monkey++)
            {
                if (dog!=monkey&&rabbit!=monkey&&cat!=monkey)
                {
                    deer = 15 - dog - rabbit - cat - monkey;
                    if (monkey<cat && deer<dog
                        && (rabbit<monkey&&dog<rabbit))
                    {
                        printf("%d,%d,%d,%d,%d\n",
                            dog,rabbit,cat,monkey,deer);
                        flag = 1;
                    }
                }
            }
        }
    }
}

```



```

int dog, rabbit, cat, monkey, deer;
int flag = 0;
for (dog=1; dog<=5&&!flag; dog++)
{
    for (rabbit=1; rabbit<=5&&!flag; rabbit++)
    {
        if (dog == rabbit) continue;
        for (cat=1; cat<=5&&!flag; cat++)
        {
            if (dog==cat || rabbit==cat) continue;
            for (monkey=1; monkey<=5&&!flag; monkey++)
            {
                deer = 15 - dog - rabbit - cat - monkey;
                if (dog==monkey || rabbit==monkey || cat==monkey) continue;
                if (monkey<cat && deer<dog
                    && (rabbit<monkey&&dog<rabbit))
                {
                    printf("%d,%d,%d,%d,%d\n",
                        dog,rabbit,cat,monkey,deer);
                    flag = 1;
                }
            }
        }
    }
}

```



```
int main()
{
    int dog = 1, rabbit = 2, cat = 3, monkey = 4, deer = 5, flag;
    do{
        flag = 0;
        if (monkey > cat)
        {
            交换monkey和cat;
            flag++;
        }
        if (dog < deer)
        {
            交换dog和deer;
            flag++;
        }
        if (rabbit > monkey)
        {
            交换rabbit和monkey;
            flag++;
        }
        if (rabbit < dog)
        {
            交换rabbit和dog;
            flag++;
        }
    } while (flag);
    printf("%d,%d,%d,%d,%d\n", dog, rabbit, cat, monkey, deer);
    return 0;
}
```

## 动物百米赛跑

monkey < cat

deer < dog

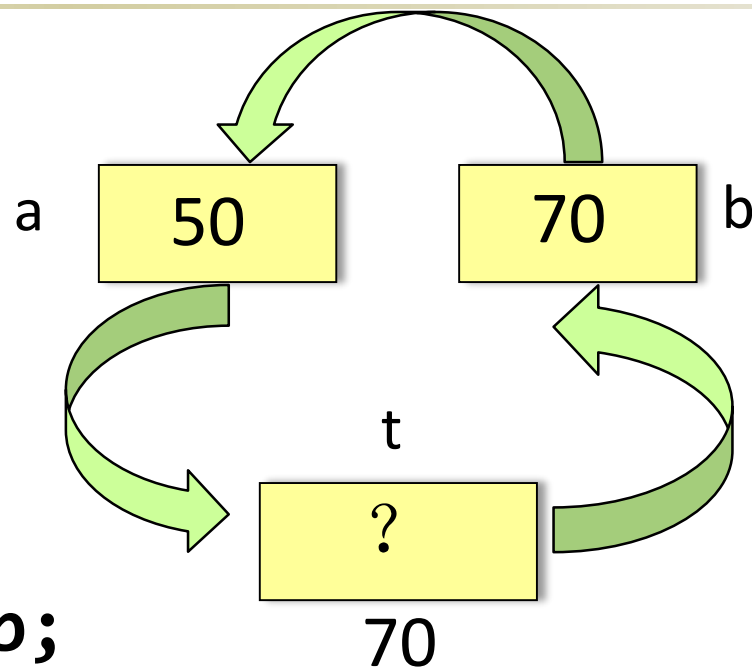
rabbit < monkey && rabbit > dog

# 实现两数交换的常用方法

`t = a;`

`b = a;`

`a = t;`



`a = a + b;`      `b = a * b;`

`b = a - b;`      `a = b / a;`

`a = a - b;`      `b = b / a;`

- 你觉得哪个方法更好？请说出你的理由。





# 还原算术表达式v1.0

- 已知在下列算式中，**不同的字母代表不同的数字**，加法算式的结果是一个三位数  $n$  ( $n < 1000$ )。从键盘输入  $n$  的值，然后编程求解以下算式中各字母所代表的数字的值，输出所有满足条件的解。

$$\begin{array}{r} XYZ \\ + YZZ \\ \hline \end{array}$$

一个三位数  $n$

F:\c\ss\bin\Debug\ss.exe

Input  $n$  ( $n < 1000$ ): 872  
 $X=2, Y=6, Z=1$



- \* 确定问题的输入:  $n$ ; 输出:  $x, y, z$
- \* **算法1**: 确定**穷举对象**:  $x, y, z$
- \* 确定 $x, y, z$ 的**搜索范围**: 0 ( ? ) , 1, 2, 3, 4, 5, 6, 7, 8, 9
- \* 确定**判定条件**
- \*  $x * 100 + y * 10 + z + (y * 100 + z * 10 + z) == n \ \&\& \ x \neq y \ \&\& \ y \neq z \ \&\& \ x \neq z$

# 还原算术表达式v1.0

- 已知在下列算式中，**不同的字母代表不同的数字**，加法算式的结果是一个三位数  $n$  ( $n < 1000$ )。从键盘输入  $n$  的值，然后编程求解以下算式中各字母所代表的数字的值，输出所有满足条件的解。

$$\begin{array}{r} XYZ \\ + YZZ \\ \hline \end{array}$$

一个三位数  $n$

F:\c\ss\bin\Debug\ss.exe

```
Input n(n<1000):872  
X=2, Y=6, Z=1
```



\* 问题的输入:  $n$ ; 输出:  $x, y, z$

\* **算法2: 确定穷举对象:**  $n$

\* **确定 $n$ 的搜索范围:** 100, 101, ... $n-1$

\* **确定判定条件**

\* **需要先将 $n$ 分解为 $x, y, z$**

\*  **$x + y*100 + z*10 + z == n \ \&\& \ x \neq y \ \&\& \ y \neq z \ \&\& \ x \neq z \ \&\& \ y \neq 0$**

- 找到一个解能否马上退出?**

# 课后作业：还原算术表达式v2.0

- 编写程序求解下式中个字母所代表的数字，不同的字母代表不同的数字。

$$\begin{array}{r} P \ E \ A \ R \\ - \quad \quad A \ R \ A \\ \hline P \ E \ A \end{array}$$

- \* 确定问题的输入：无；输出：p,e,a,r
- \* 确定穷举对象：p,e,a,r
- \* 确定搜索范围：0 ( ? ) , 1, 2, 3, 4, 5, 6, 7, 8, 9
- \* 确定判定条件
- \* 
$$p*1000+e*100+a*10+r-(a*100+r*10+a)==p*100+e*10+a$$
$$\&\& \ r!=p \ \&\& \ r!=e \ \&\& \ r!=a \ \&\& \ p!=e \ \&\& \ p!=a \ \&\& \ e!=a \ \&\& \ e!=r$$
$$\&\& \ a!=r$$

## 多选题

下列说法中错误的是

- ☒ A 循环语句中的break语句的作用就是跳出循环体，无论循环有多少层
- ☒ B continue语句的作用就是继续执行循环体
- ☒ C for和while都是当型循环，因此二者在任何情况下都可以相互等价转换
- ☐ D 穷举法求解问题的关键就是确定穷举对象、穷举范围以及符合答案的判定条件

提交

# 正向顺推的实例

- 兔子的理想化繁衍问题（13世纪数学家Fibonacci）
- 每个月兔子对数的变化规律

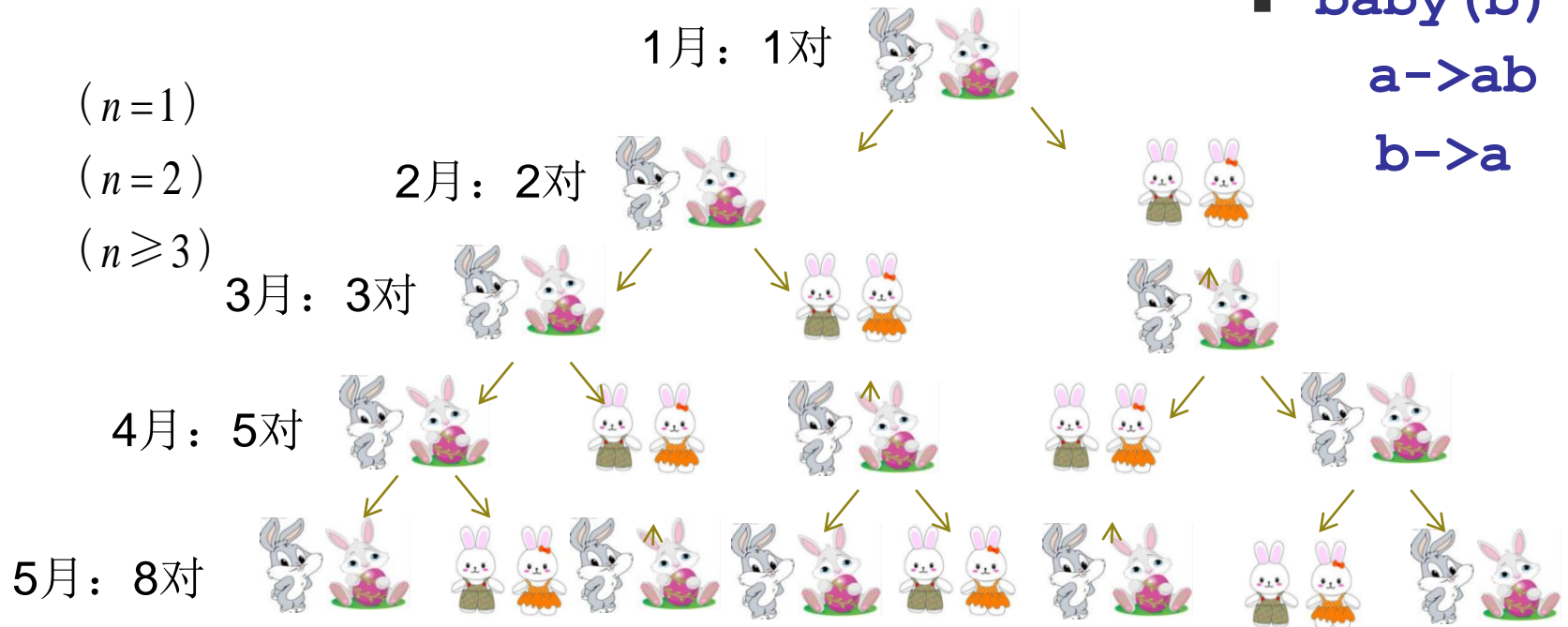
■ **adult(a)**

■ **baby(b)**

**a→ab**

**b→a**

$$\begin{cases} f_1 = 1 & (n=1) \\ f_2 = 1 & (n=2) \\ f_n = f_{n-1} + f_{n-2} & (n \geq 3) \end{cases}$$



	a	ab	aba	abaab	abaababa	abaababaabaab.....		
b:	0	1	1	2	3	5	8	13.....
a:	1	1	2	3	5	8	13	21.....
a+b:	1	2	3	5	8	13	21	34.....

**Fibonacci数列**

# 正向顺推的实例

- **走台阶：**假设上楼可一步上一级，也可1步上两级， $n$ 级台阶总共有多少走法？

- \* 一级台阶只有一种走法
- \* 两级台阶时既可一步走一级，也可1步走两级，共两种走法
- \* 三级，可每次都走一级，或第一次走一级，第二次走两级，也可第一次走两级，第二次走一级，总计一共3种走法
- \* 依此类推，4级台阶共有5种走法，.....，直到计算出 $n$ 级台阶的走法

- \* **边界条件：**一级台阶和两级台阶的走法

$$\begin{cases} f_1 = 1 & (n=1) \\ f_2 = 1 & (n=2) \\ f_n = f_{n-1} + f_{n-2} & (n \geq 3) \end{cases}$$



# 计算Fibonacci数列

## 递推法计算Fibonacci数列的前n项

$$fib(n) = \begin{cases} 0 & n=0 \\ 1 & n=1 \\ fib(n-1) + fib(n-2) & n>1 \end{cases}$$

0   1   1   2   3   5   8

.....  
*f1*   *f2*   *f3*

*f1*   *f2*   *f3*

*f1*   *f2*   *f3*

*f1*   *f2*   *f3*

```
int Fib(int n)
{
    int i, f1 = 0, f2 = 1, f3;
    if (n == 0)
    {
        return 0;
    }
    else if (n == 1)
    {
        return 1;
    }
    else
    {
        for (i=2; i<=n; i++)
        {
            f3 = f1 + f2;
            f1 = f2;
            f2 = f3;
        }
        return f3;
    }
}
```

# 计算Fibonacci数列

## 递推法计算Fibonacci数列的前n项

$$fib(n) = \begin{cases} 0 & n=0 \\ 1 & n=1 \\ fib(n-1) + fib(n-2) & n>1 \end{cases}$$

0 1 1 2 3 5 8

$f1$   $f2$

$f1$   $f2$

$f1$   $f2$

$f1$

```
int Fib(int n)
{
    int i, f1 = 0, f2 = 1;
    if (n == 0)
    {
        return 0;
    }
    else if (n == 1)
    {
        return 1;
    }
    else
    {
        for (i=0; i<n/2; i++)
        {
            f1 = f1 + f2;
            f2 = f2 + f1;
        }
        return n%2==0 ? f1 : f2;
    }
}
```

# 反向逆推的实例



## ■ 猴子吃桃问题

- \* 猴子第一天摘下若干个桃子，吃了一半，还不过瘾，又多吃了一个。第二天早上又将剩下的桃子吃掉一半，并且又多吃了一个。以后每天早上都吃掉前一天剩下的一半零一个。到第10天早上再想吃时，发现只剩下一个桃子。问第一天共摘了多少桃子。

- \* 每天剩下的桃子数比前一天的一半少一个

- \* 每天剩下的桃子数加1之后，刚好是前一天的一半

$$x_{n+1} = x_n / 2 - 1$$

天数：    1        2        3        4        5        6        7        8        9        10

 1534 ← 766 ← 382 ← 190 ← 94 ← 46 ← 22 ← 10 ← 4 ← 

$$x_n = 1$$

$$n = 10$$

递推9次

$$x_n = 2 \times (x_{n+1} + 1)$$

$$1 \leq n < 10$$



```
#include <stdio.h>
int main()
{
    int x = 1, days = 10;
    while (days > 1)
    {
        //由第days天推出第days-1天
        x = 2 * (x + 1);
        days--;
    }
    printf("x = %d\n", x);
    return 0;
}
```

天数:     1            2            3            4

 1534 ← 766 ← 382 ← 190 ←

$$x_n = 1$$

$$n = 10$$

$$x_n = 2 \times (x_{n+1} + 1)$$

$$1 \leq n < 10$$

```
#include <stdio.h>
int MonkeyEatPeach(int days);

int main()
{
    int x, days;
    printf("Input days:");
    scanf("%d", &days);
    x = MonkeyEatPeach(days);
    printf("x = %d\n", x);
    return 0;
}

int MonkeyEatPeach(int days)
{
    int x = 1;
    while (days > 1)
    {
        x = 2 * (x + 1);
        days--;
    }
    return x;
}
```



# 新版猴子吃桃

■ 现有 $n$  ( $1 \leq n \leq 10^{18}$ ) 个桃子，无限可列个猴子去领桃子吃。在桃子足够的情况下，排在第 $i$ 位的猴子领 $\text{Fib}(i)$ 个桃子， $\text{Fib}$ 是Fibonacci数列(1,1,2,3,5,8.....)。若轮到第 $i$ 个猴子时，剩余的桃子不到 $\text{Fib}(i)$ 个，它就获得剩余的桃子，第 $i+1$ 个及以后的猴子就要挨饿了。请编程计算输入 $n$ 的情况下一共有多少只猴子可以分到桃子，以及最后一个猴子分的多少桃子。

```
int main()
{
    int n, i;
    long sum = 1, next;
    printf("Input n:");
    scanf("%d", &n);
    if (n == 1 || n == 2)
    {
        printf("count=%d\npeach=%ld\n", n, sum);
    }
    else
    {
        for (i=3, sum=2; n-sum>0; i++)
        {
            next = Fib(i);
            if (n - sum > next)
                sum += next;
            else break;
        }
        printf("count=%d\npeach=%ld\n", i, n-sum);
    }
    return 0;
}
```

# 新版猴子吃桃

■ 请编程计算输入n的情况下一共有多少只猴子可以分的桃子，以及最后一个猴子分的多少桃子。

■ 某只小猴子能拿到最多的桃子，那么这只猴子应该排在第几个位置，又能吃到几个桃子呢？



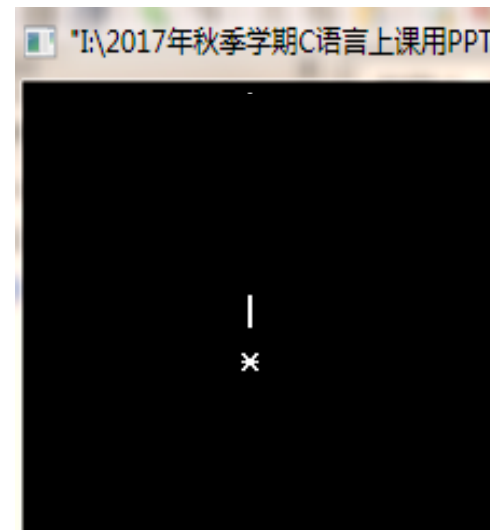
```
int main()
{
    int n, i;
    long sum = 1, next;
    printf("Input n:");
    scanf("%d", &n);
    if (n == 1 || n == 2)
    {
        printf("count=%d\npeach=%ld\n", n, sum);
    }
    else
    {
        for (i=3,sum=2; n-sum>0; i++)
        {
            next = Fib(i);
            if (n - sum > next)
                sum += next;
            else break;
        }
        printf("count=%d\npeach=%ld\n", i, n-sum);
    }
    return 0;
}
```



# 飞机游戏V1.0版

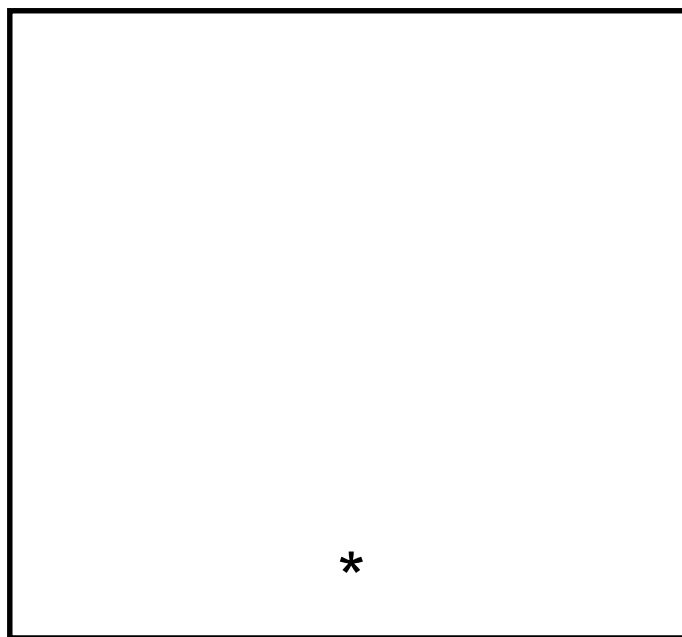
```
int high = 20, width = 30; //游戏画面尺寸
int planeX = 10, planeY = 15; //飞机位置
int bulletX = 0, bulletY = 15; //子弹位置
while (1)
{
    system("cls"); //清屏
    //循环输出空行空格、飞机和子弹
    for (i=0; i<high; i++)
    {
        for (j=0; j<width; j++)
        {
            if (i == planeX && j == planeY)
                printf("*"); //输出飞机
            else if (i == bulletX && j == bulletY)
                printf("|"); //输出子弹
            else
                printf(" "); //输出空格
        }
        printf("\n");
    }
}
```

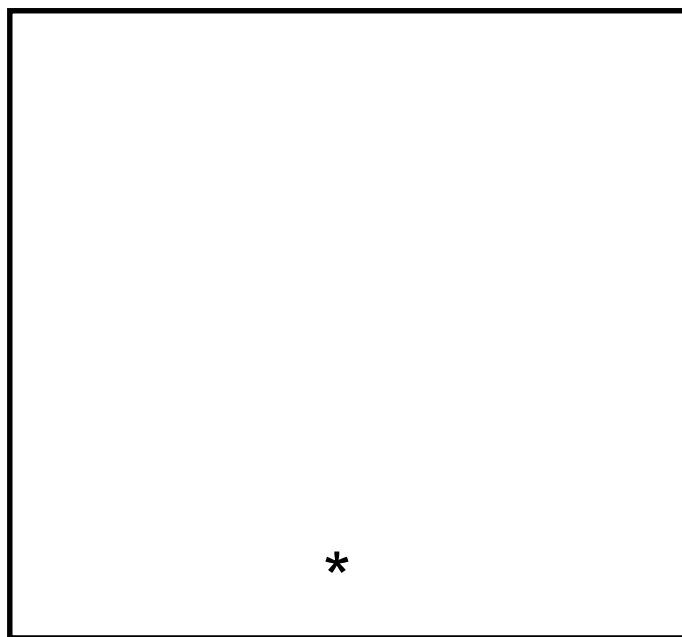
x

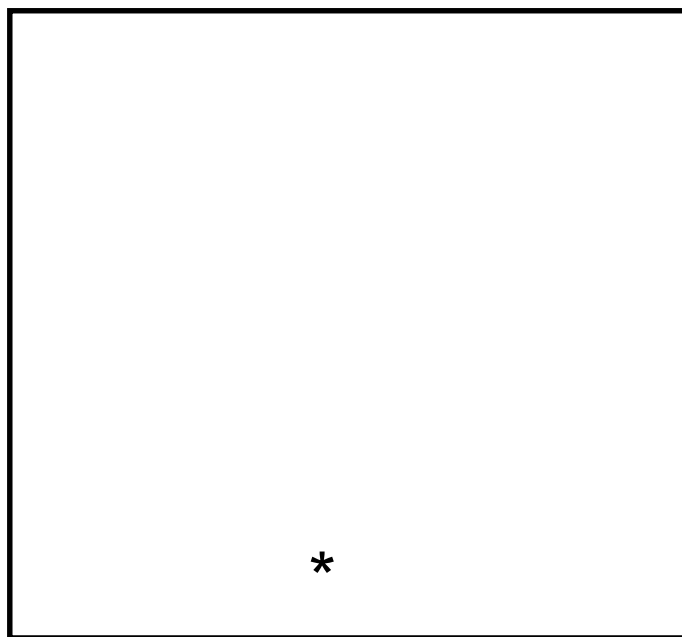


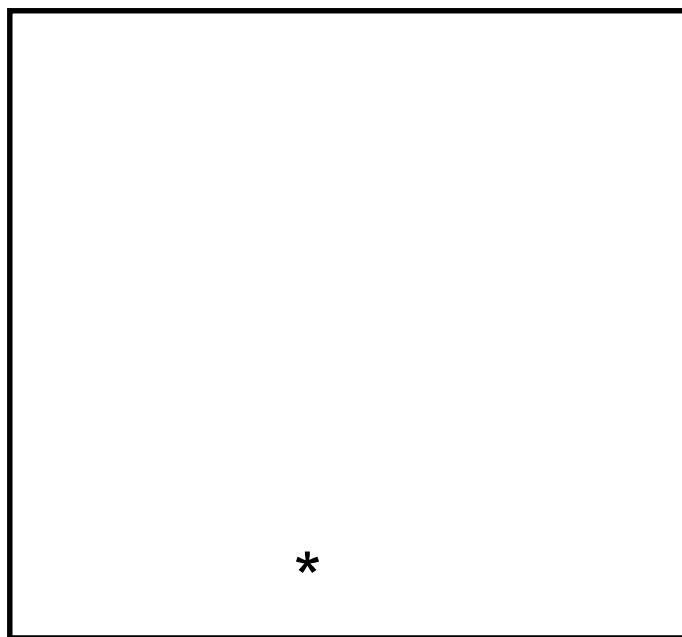
y









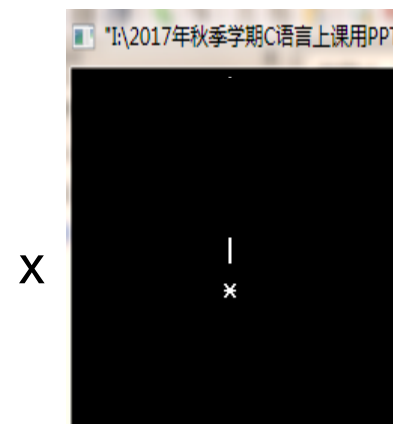


# 飞机游戏V1.0版

- 控制飞机移动方式

- \* 用scanf()输入a,s,d,w改变,y坐标

```
while (1)
{
    system("cls"); //清屏
    循环输出空行空格和飞机和子弹
    scanf("%c", &input); //等待键盘输入
    if (input == 'a') y--; //左
    if (input == 'd') y++; //右
    if (input == 'w') y--; //上
    if (input == 's') y++; //下
}
```

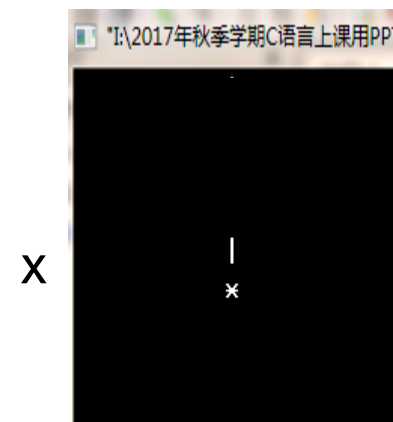


# 飞机游戏V1.0版

## ■ 控制飞机移动方式

- \* 用`getch()`输入a,s,d,w改变,y坐标
- \* `#include <conio.h>`

```
while (1)
{
    system("cls"); //清屏
    循环输出空行空格和飞机和子弹
    input = getch(); //试试看你发现了什么?
    if (input == 'a') y--;
    if (input == 'd') y++;
    if (input == 'w') x--;
    if (input == 's') x++;
}
```

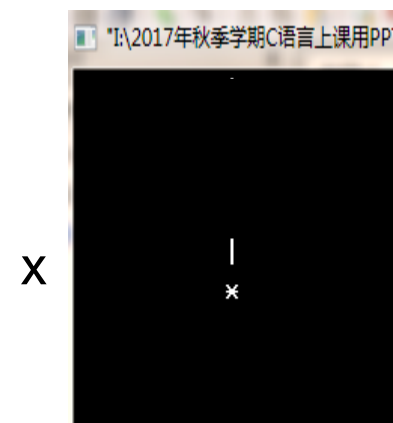


# 飞机游戏V1.0版

## ■ 控制飞机移动方式

- \* 用**kbhit()**检测是否有键盘输入
- \* **#include <conio.h>**

```
while (1)
{
    system("cls"); //清屏
    没有键盘输入就循环输出空行空格和飞机和子弹
    if (kbhit()) //有键盘输入
    {
        input = getch();
        if (input == 'a') y--;
        if (input == 'd') y++;
        if (input == 'w') x--;
        if (input == 's') x++;
    }
}
```



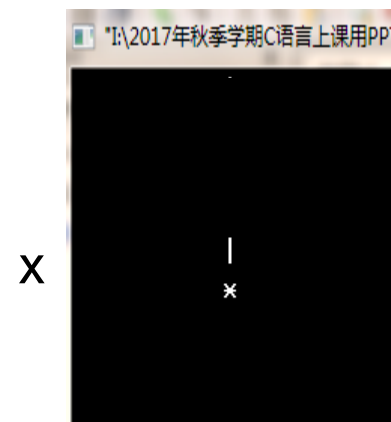


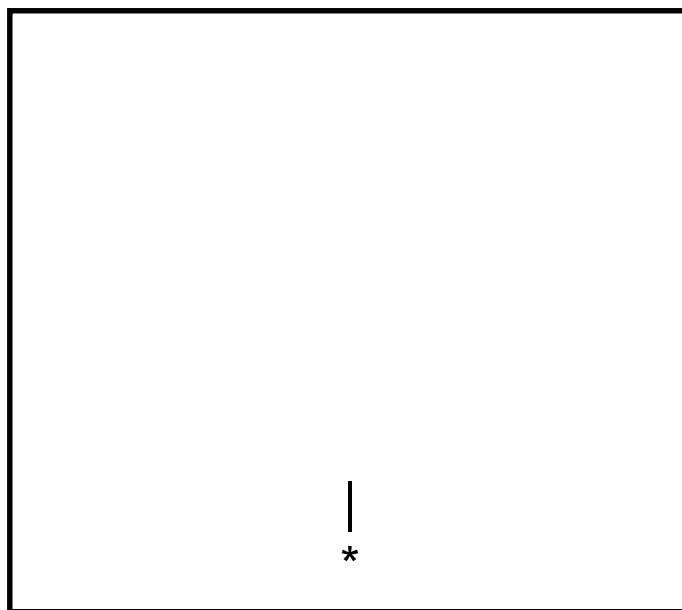
# 飞机游戏V1.0版

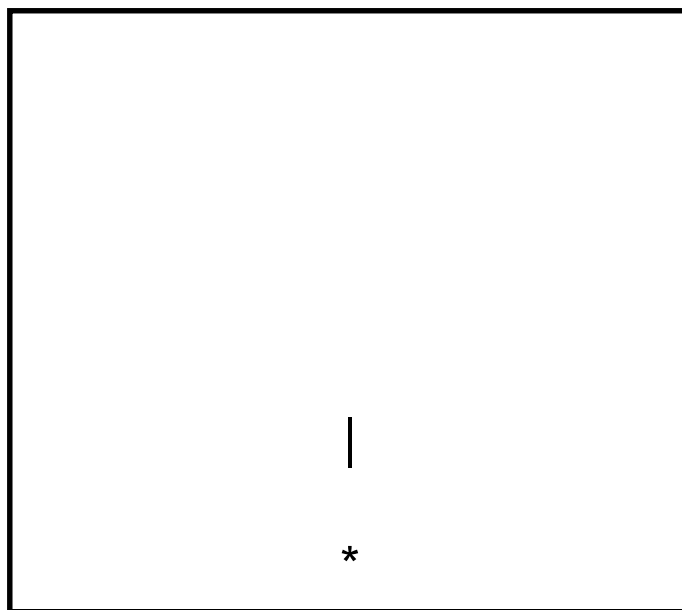
## ■ 控制飞机移动方式

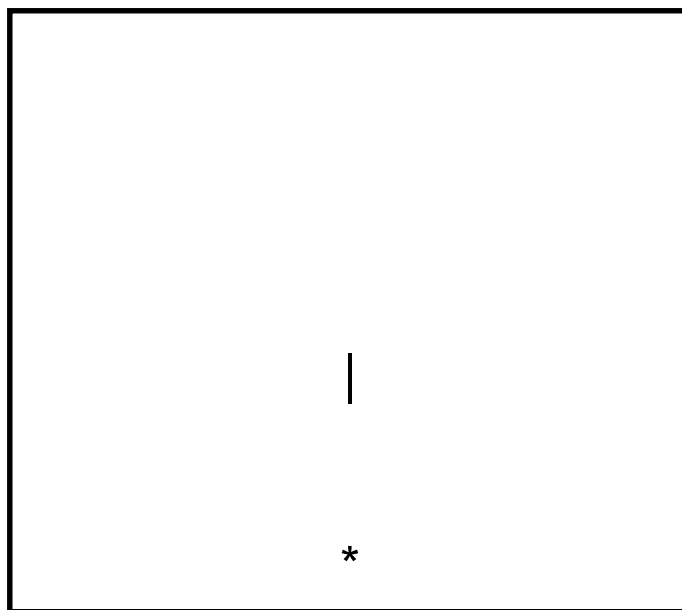
- \* 用上下左右键控制飞机移动
- \* 键盘上的每一个键都有两个唯一的数值（键盘扫描码），标志被按下还是被释放

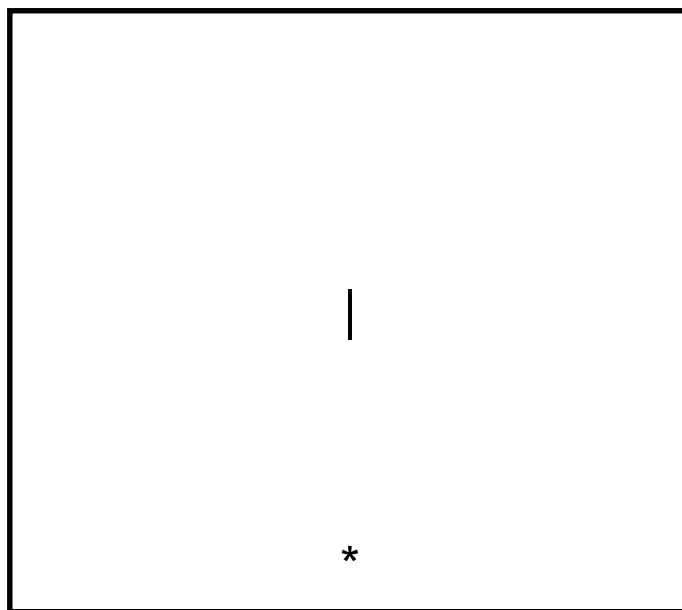
```
while (1)
{
    system("cls"); //清屏
    循环输出空行空格和飞机和子弹
    if (kbhit()) //没有键盘输入就循环显示 '*'
    {
        input = getch();
        if (input == 75) y--; //左
        if (input == 77) y++; //右
        if (input == 72) x--; //上
        if (input == 80) x++; //下
    }
}
```











```

while (1)
{
    system("cls"); //清屏
    循环输出空行空格和飞机和子弹

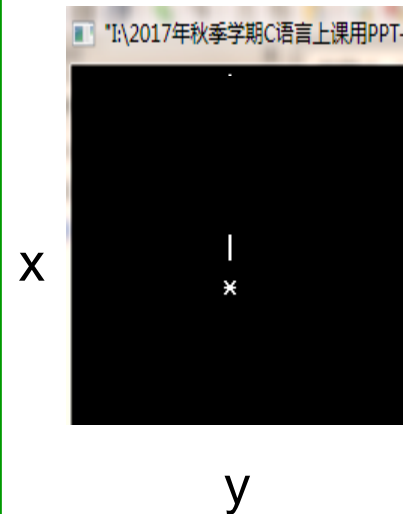
    if (bulletX >= 0)
    {
        bulletX--;    //子弹上移，超边界不显示
    }

    if (kbhit())
    {
        input = getch();
        根据用户的输入确定上下左右移动飞机的位置
        if (input == ' ') //发射子弹
        {
            bulletX = planeX - 1; //子弹初始位置在飞机正上方
            bulletY = planeY;
        }
    }
}
}

```

# 飞机游戏 V2.0版

按空格键时，  
让飞机发射移  
动的激光子弹  
('|')



```

.....
int enemyX = 0, enemyY = 15;    //敌机位置
while (1)
{
    system("cls"); //清屏
    for (i=0; i<high; i++)
    {
        for (j=0; j<width; j++)
        {
            if (i == planeX && j == planeY)
                printf("*"); //输出飞机
            else if (i == enemyX && j == enemyY)
                printf("@"); //输出敌机
            else if (i == bulletX && j == bulletY)
                printf("|"); //输出子弹
            else
                printf(" "); //输出空格
        }
        printf("\n");
    }
    if (bulletX >= 0)    bulletX--; //子弹上移, 超边界不显示
    if (bulletX == enemyX && bulletY == enemyY)
    {
        enemyX = -1;    //击中敌机, 敌机消失
    }
    if (kbhit())
    .....
}

```

# 飞机游戏 V3.0版

移动的子弹  
打静止的敌  
机 (@)



# 飞机游戏V4.0

## ■ 用模块化编程方法

- \* `void Initialize()`           //数据初始化
- \* `void Show()`                //显示游戏画面
- \* `void UpdateWithoutInput()`   //与用户输入无关的更新
- \* `void UpdateWithInput()`      //与用户输入有关的更新

```
int main()
{
    Initialize();//数据初始化
    while (1)
    {
        Show(); //显示游戏画面
        UpdateWithoutInput(); //与用户输入无关的更新
        UpdateWithInput();    //与用户输入有关的更新
    }
    return 0;
}
```



# 飞机游戏 V4.0版

- 移动的子弹  
打静止的敌  
机 (@)

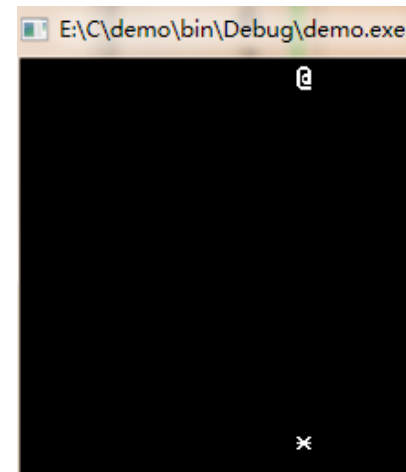
```
//全局变量
int high, width;      //游戏画面尺寸
int planeX, planeY;   //飞机位置
int bulletX, bulletY; //子弹位置
int enemyX, enemyY;   //敌机位置

void Initialize()      //数据的初始化
{
    high = 20;
    width = 30;

    planeX = high / 2;
    planeY = width / 2;

    bulletX = 0;
    bulletY = planeY;

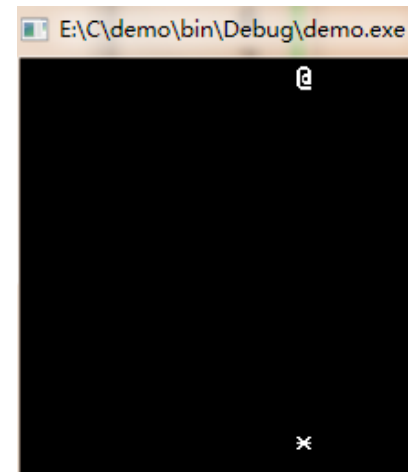
    enemyX = 0;
    enemyY = 15;
}
```



# 飞机游戏 V4.0版

- 移动的子弹打  
静止的敌机（  
用@显示）

```
void Show()                //显示游戏画面
{
    system("cls");
    int i, j;
    for (i=0; i<high; i++)
    {
        for (j=0; j<width; j++)
        {
            if (i == planeX && j == planeY)
                printf("*"); //输出飞机
            else if (i == enemyX && j == enemyY)
                printf("@"); //输出敌机
            else if (i == bulletX && j == bulletY)
                printf("|"); //输出子弹
            else
                printf(" "); //输出空格
        }
        printf("\n");
    }
}
```

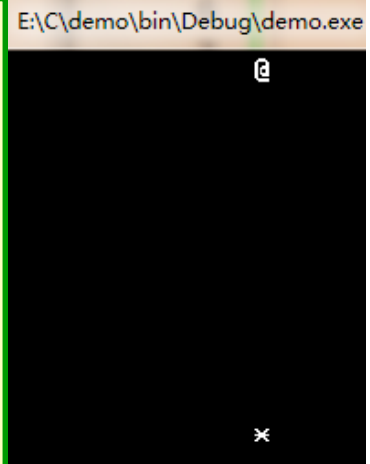


# 飞机游戏 V4.0版

- 移动的子弹打  
静止的敌机（  
用@显示）

```
void UpdateWithoutInput() //与用户输入无关的更新
{
    if (bulletX > -1)
    {
        bulletX--; //子弹上移，超出屏幕上边界不显示
    }
    if (bulletX == enemyX && bulletY == enemyY)
    {
        enemyX = -1; //击中敌机后敌机消失
    }
}
```

```
void UpdateWithInput() //与用户输入有关的更新
{
    char input;
    if (kbhit()) //判断是否有输入
    {
        input = getch(); //根据用户的不同输入移动飞机
        if (input == 'a') planeY--; //左移
        if (input == 'd') planeY++; //右移
        if (input == 'w') planeX--; //上移
        if (input == 's') planeX++; //下移
        if (input == ' ') //发射子弹
        {
            bulletX = planeX - 1; //子弹初始位置在飞机正上方
            bulletY = planeY;
        }
    }
}
```



# 挑战速度：计算 $n$ ( $<10000000$ ) 以内的所有完数

```
int IsPerfect(int x)
{
    int i, sum = 0;
    for (i=1; i<=x/2; i++)
    {
        if (x%i == 0)
        {
            sum = sum + i;
        }
    }
    return sum==x ? 1 : 0;
}
```

```
int IsPerfect(int x)
{
    int i;
    int sum = 1;
    int k = (int)sqrt(x);
    for (i=2; i<=k; i++)
    {
        if (x%i == 0)
        {
            sum += i;
            sum += x/i;
        }
    }
    return sum==x ? 1 : 0;
}
```

## ■ 求解算法

- \*  $i$ 从1试到 $x/2$ ，看 $i$ 是否是 $x$ 的真因子
- \* 若 $x$ 能被 $i$ 整除，则累加到 $sum$
- \* 累加结束判断 $x$ 是否等于 $sum$ ，返回判断结果

## ■ 为什么结果会这样？



```
#include <stdio.h>
double Fact(unsigned int n);
int main()
{
    int i;
    for (i=1; i<=40; i++)
    {
        printf("%d! = %.0f\n",i,Fact(i));
    }
    return 0;
}
double Fact(unsigned int n)
{
    unsigned int i;
    double result = 1;
    for (i=1; i<=n; i++)
    {
        result = result * i;
    }
    return result;
}
```

```
1! = 1
2! = 2
3! = 6
4! = 24
5! = 120
6! = 720
7! = 5040
8! = 40320
9! = 362880
10! = 3628800
11! = 39916800
12! = 479001600
13! = 6227020800
14! = 87178291200
15! = 1307674368000
16! = 20922789888000
17! = 355687428096000
18! = 6402373705728000
19! = 121645100408832000
20! = 2432902008176640000
21! = 51090942171709440000
22! = 112400072777607700000
23! = 25852016738884978000000
24! = 6204484017332394100000000
25! = 155112100433309860000000000
26! = 4032914611266056500000000000
27! = 108888694504183520000000000000
28! = 3048883446117138400000000000000
29! = 88417619937397008000000000000000
30! = 265252859812191030000000000000000
31! = 8222838654177922400000000000000000
32! = 263130836933693520000000000000000000
33! = 8683317618811885900000000000000000000
34! = 295232799039604120000000000000000000000
35! = 1033314796638614400000000000000000000000
36! = 3719933267899011800000000000000000000000
37! = 137637530912263430000000000000000000000000
38! = 5230226174666010400000000000000000000000000
39! = 203978820811974420000000000000000000000000000
40! = 8159152832478976800000000000000000000000000000
```

```
40! = 8159152832478976800000000000000000000000000000000000000
```

## 课后思考题：挑战类型的极限

- 从键盘任意输入一个数 $n$  ( $0 < n \leq 1000000$ )，编程计算并输出 $S = 1! + 2! + \dots + n!$ 的**末6位** (不含前导0)。
  - \* 若 $S$ 不足6位，则直接输出 $S$ 。
  - \* 不含前导0的意思是，如果末6位为001234，则只输出1234即可。
  - \* 如果输入的 $n$ 不在1到1000000之间，则输出 “Input error!”。

Enter a number to be calculated:

```
40
1! = 1
2! = 2
3! = 6
4! = 24
5! = 120
6! = 720
7! = 5040
8! = 40320
9! = 362880
10! = 3628800
11! = 39916800
12! = 479001600
13! = 6227020800
14! = 87178291200
15! = 1307674368000
16! = 20922789888000
17! = 355687428096000
18! = 6402373705728000
19! = 121645100408832000
20! = 2432902008176640000
21! = 51090942171709440000
22! = 112400072777607680000
23! = 25852016738884976640000
24! = 620448401733239439360000
25! = 15511210043330985984000000
26! = 4032914611266056355840000000
27! = 108888694504183521607680000000
28! = 3048883446117138605015040000000
29! = 88417619937397019545436160000000
30! = 2652528598121910586363084800000000
31! = 82228386541779228177255628800000000
32! = 2631308369336935301672180121600000000
33! = 86833176188118864955181944012800000000
34! = 2952327990396041408476186096435200000000
35! = 103331479663861449296666513375232000000000
36! = 3719933267899012174679994481508352000000000
37! = 137637530912263450463159795815809024000000000
38! = 5230226174666011117600072241000742912000000000
39! = 203978820811974433586402817399028973568000000000
40! = 8159152832478977343456112695961158942720000000000
```

```

#include <stdio.h>
#define MOD 1000000
long Func(int n);
int main()
{
    int n;
    long s;
    printf("Input n:");
    scanf("%d", &n);
    if (n>0 && n<=1000000)
    {
        s = Func(n);
        printf("%ld\n", s);
    }
    else
    {
        printf("Input error!\n");
    }
    return 0;
}

```

```

//函数功能：计算1!+2!+...+n!的末6位数
long Func(int n)
{
    int i;
    long s = 0, f = 1;
    for (i=1; i<=n; i++)
    {
        f = f * i;
        s = s + f;
    }
    return s % MOD;
}

```

- 从键盘任意输入一个数n ( $0 < n \leq 1000000$ )，编程计算并输出  $S=1!+2!+\dots+n!$  的末6位（不含前导0）。
- 这个程序能否得到你期望的结果？



```

#include <stdio.h>
#define MOD 1000000
long Func(int n);
int main()
{
    int n;
    long s;
    printf("Input n:");
    scanf("%d", &n);
    if (n>0 && n<=1000000)
    {
        s = Func(n);
        printf("%ld\n", s);
    }
    else
    {
        printf("Input error!\n");
    }
    return 0;
}

```

```

//函数功能：计算1!+2!+...+n!的末6位数
long Func(int n)
{
    int i;
    long s = 0, f = 1;
    if (n > 24) n = 24;
    for (i=1; i<=n; i++)
    {
        f = f * i % MOD; //保留阶乘值末6位
        s = (s + f) % MOD; //保留阶乘和末6位
    }
    return s;
}

```

- 从键盘任意输入一个数n ( $0 < n \leq 1000000$ )，编程计算并输出  $S=1!+2!+\dots+n!$  的末6位（不含前导0）。
- 这个程序能否得到你期望的结果？



# 飞机游戏V5.0: 敌机下移

```
void UpdateWithoutInput() //与用户输入无关的更新
{
    if (bulletX > -1)
    {
        bulletX--; //子弹移动, 超出屏幕上边界不显示
    }
    if (bulletX == enemyX && bulletY == enemyY) //击中敌机
    {
        enemyX = -1; //敌机消失
    }
    //用来控制敌机向下移动的速度, 每隔10次循环才移动一次敌机
    static int speed = 0; //静态局部变量, 仅初始化1次
    if (speed < 10)
    {
        speed++; //记录本函数被调用的次数
    }
    else if (speed == 10) //每执行10次函数调用后
    {
        enemyX++; //敌机下移1次
        speed = 0; //重新开始计数
    }
}
```

# 飞机游戏V6.0: 随机产生移动的敌机

```
void UpdateWithoutInput() //与用户输入无关的更新
{
    srand(time(NULL));
    if (bulletX > -1)
    {
        bulletX--; //子弹移动, 超出屏幕上边界不显示
    }
    if (bulletX == enemyX && bulletY == enemyY) //击中敌机
    {
        enemyX = -1; //敌机消失, 隔10帧后重新出现在屏幕顶端
        enemyY = rand() % width; //水平位置随机生成
    }
    if (enemyX > high) //敌机跑出屏幕下边界时产生新的敌机
    {
        enemyX = 0;
        enemyY = rand() % width;
    }
    if (planeX == enemyX && planeY == enemyY) //撞上敌机则游戏结束
    {
        printf("Game over!\n");
        system("pause");
        exit(0);
    }
    //用来控制敌机向下移动的速度, 每隔几次循环才移动一次敌机
    .....
}
```

# 飞机游戏V7.0: 记录游戏得分

```
void UpdateWithoutInput() //与用户输入无关的更新
```

```
{
    srand(time(NULL));
    if (bulletX > -1)
    {
        bulletX--; //子弹移动, 超出屏幕上边界不显示
    }
    if (bulletX == enemyX && bulletY == enemyY) //击中敌机
    {
        score++;
        enemyX = -1; //敌机消失
        enemyY = rand() % width;
    }
    if (enemyX > high) //敌机
    {
        enemyX = 0;
        enemyY = rand() % width;
    }
    if (planeX == enemyX && planeY == enemyY)
    {
        printf("Game over!\n");
        system("pause");
        exit(0);
    }
    .....
}
```

```
//全局变量
```

```
.....
int score;
void Initialize() //数据的初始化
{
    .....
    score = 0;
}
```

```
void Show() //显示画面
{
    .....
    printf("得分: %d\n", score);
}
```