

# 第5章 循环控制

——流程的转移控制与用穷举法求解问题

---

# 本节要讨论的主要问题

- C语言中用于实现流程转移控制的语句有哪几种？
- `break`语句和`continue`语句对循环过程的影响有什么不同？



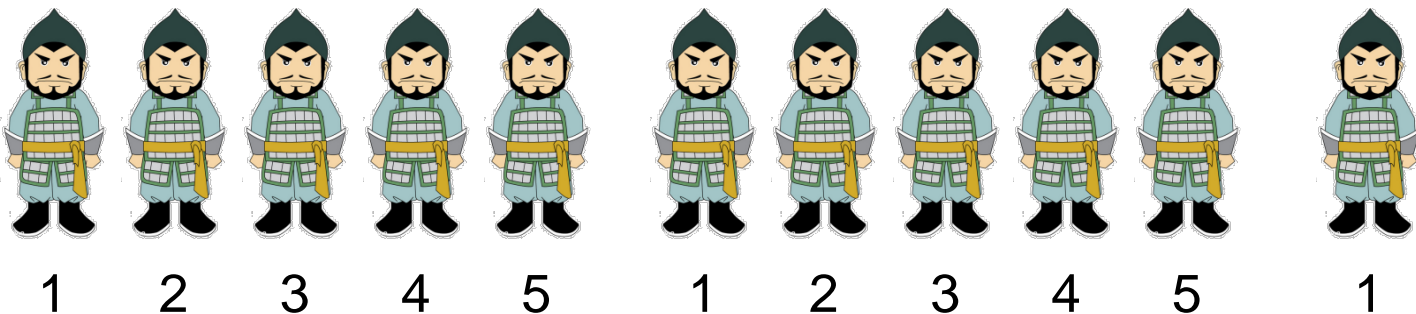
# 从韩信点兵谈起

- 韩信有一队兵，按从1至5排队报数，最末一个士兵报的数为1；按从1至6报数，最末一个士兵报的数为5；按从1至7报数，最末一个士兵报的数为4；最后再按从1至11报数，最末一个士兵报的数为10。请问韩信至少有多少兵。
- 穷举法
- 确定问题的输入和输出
  - \* 输入：无；输出：士兵至少 $x$ 人
- 确定穷举对象：士兵数 $x$
- 确定搜索范围： $x$ 从1开始试验
- 如何确定判定条件？



# 韩信点兵

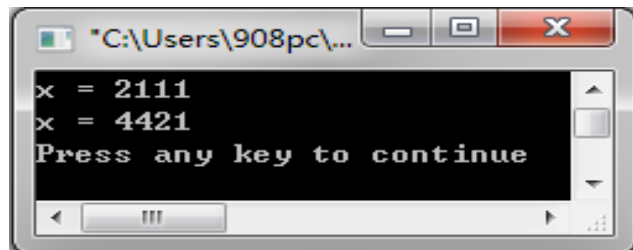
- 韩信有一队兵，按从1至5排队报数，最末一个士兵报的数为1；按从1至6报数，最末一个士兵报的数为5；按从1至7报数，最末一个士兵报的数为4；最后再按从1至11报数，最末一个士兵报的数为10。请问韩信至少有多少兵。
- 确定判定条件：
  - \* 按从1至5排队报数，最末一个士兵报的数为1？
    - \*  $x$ 被5整除，余数为1
  - \*  $x$ 被5、6、7、11整除，余数为1、5、4、10



# 韩信点兵

```
#include <stdio.h>
int main()
{
    int x;
    for (x=1; x < 5000 ;x++)
    {
        if (x%5==1 && x%6==5 && x%7==4 && x%11==10)
        {
            printf("x = %d\n", x);
        }
    }
    return 0;
}
```

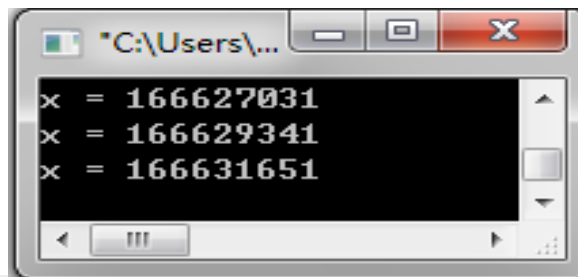
错在哪里？



# 韩信点兵

```
#include <stdio.h>
int main()
{
    int x;
    for (x=1; ;x++)
    {
        if (x%5==1 && x%6==5 && x%7==4 && x%11==10)
        {
            printf("x = %d\n", x);
        }
    }
    return 0;
}
```

错在哪里？



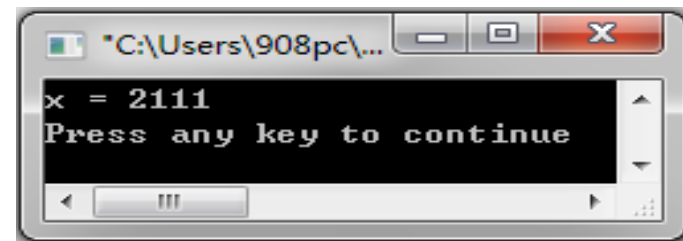
```
x = 166627031
x = 166629341
x = 166631651
```

# 韩信点兵

```
#include <stdio.h>
int main()
{
    int x;
    for (x=1; ;x++)
    {
        if (x%5==1 && x%6==5 && x%7==4 && x%11==10)
        {
            printf("x = %d\n", x);
            goto END;
        }
    }
    END: ;
    return 0;
}
```

无条件转向语句

标号（标识符）后面必须有语句，哪怕是空语句



# 韩信点兵

```
#include <stdio.h>
int main()
{
    int x;
    for (x=1; ;x++)
    {
        if (x%5==1 && x%6==5 && x%7==4 && x%11==10)
        {
            printf("x = %d\n", x);
            break;
        }
    }
    return 0;
}
```

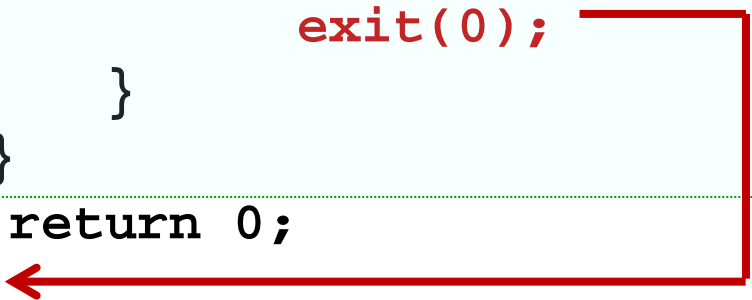
本质是受限的goto语句，  
跳转的位置限定为紧接着  
循环语句后的第一条语句





# 韩信点兵

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
    int x;
    for (x=1; ;x++)
    {
        if (x%5==1 && x%6==5 && x%7==4 && x%11==10)
        {
            printf("x = %d\n", x);
            exit(0);
        }
    }
    return 0;
}
```



终止整个程序的执行强制返回操作系统。当其参数为0时，表示程序正常退出，非0时表示程序出现某种错误后退出



# 韩信点兵

```
#include <stdio.h>
int main()
{
    int    x;
    int    find = 0;          /*置为假*/
    for (x=1; !find ;x++)
    {
        if (x%5==1 && x%6==5 && x%7==4 && x%11==10)
        {
            printf("x = %d\n", x);
            find = 1; /*置为真*/
        }
    }

    return 0;
}
```

find == 0?  
find != 1?

使用标志变量使  
程序可读性更好

# 韩信点兵

```
#include <stdio.h>
```

```
int main()
```

```
{
```

```
    int    x = 0;
```

```
    do{
```

```
        x++;
```

```
    }while ( !(x%5==1 && x%6==5 && x%7==4 && x%11==10) );
```

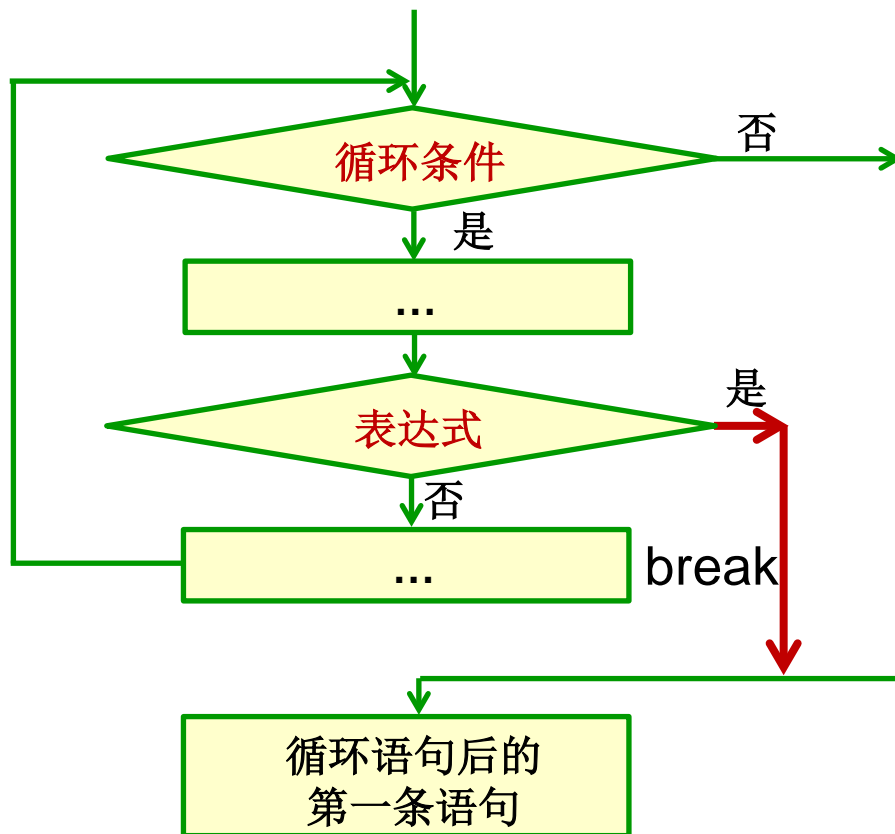
```
    printf("x = %d\n", x);
```

```
    return 0;
```

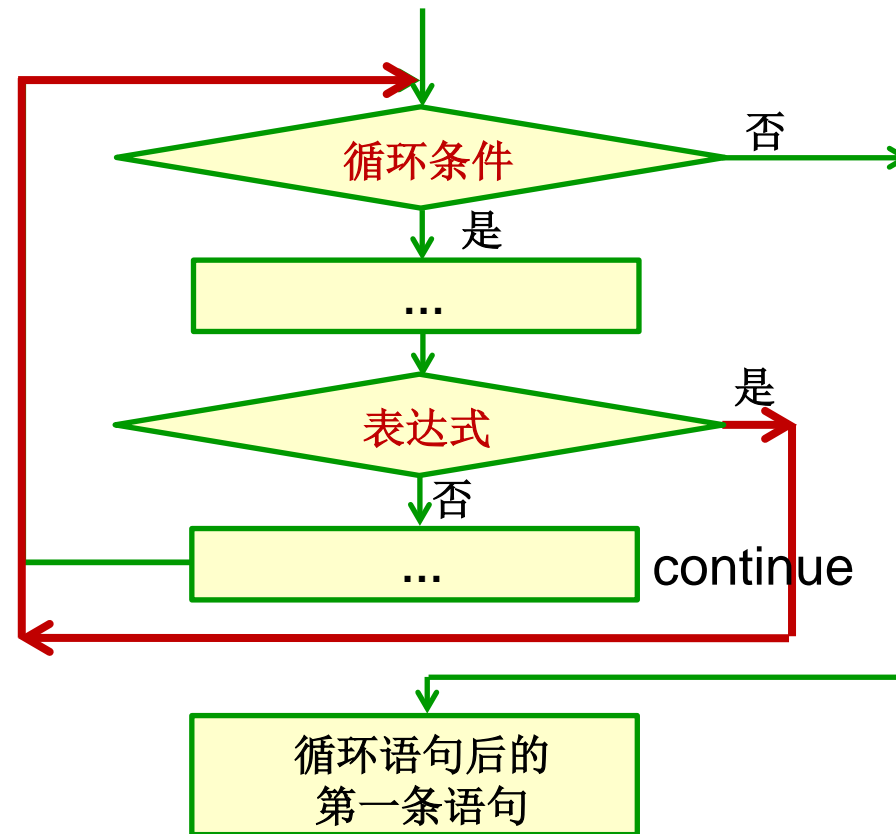
```
}
```

条件控制的循环，没有  
搜索上限时，如果无解，  
那么…？

# continue与break的区别



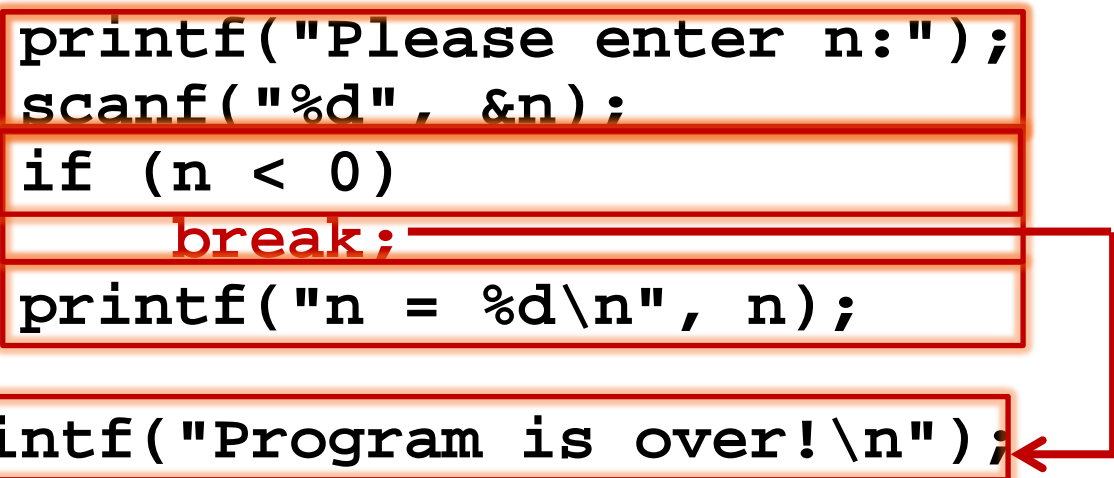
退出一层循环或switch，  
转到闭合循环之后的那一点



中断此次循环，  
开始下一次循环

# break与continue对循环的影响

```
#include <stdio.h>
int main()
{
    int i, n;
    for (i=1; i<=5; i++)
    {
        printf("Please enter n:");
        scanf("%d", &n);
        if (n < 0)
            break;
        printf("n = %d\n", n);
    }
    printf("Program is over!\n");
    return 0;
}
```



n

-10

Please enter n:10✓

n = 10

Please enter n: -10✓

Program is over!

# break与continue对循环的影响

```
#include <stdio.h>
int main()
{
    int i, n;
    for (i=1; i<=5; i++) ←
    {
        printf("Please enter n:");
        scanf("%d", &n);
        if (n < 0)
            continue;
        printf("n = %d\n", n);
    }
    printf("Program is over!\n");
    return 0;
}
```

n

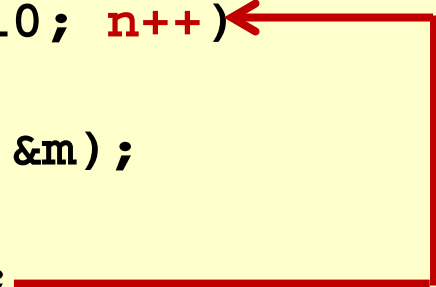
30

```
Please enter n:10✓
n = 10
Please enter n: -10✓
Please enter n:20✓
n = 20
Please enter n: -20✓
Please enter n:30✓
n = 30
Program is over!
```

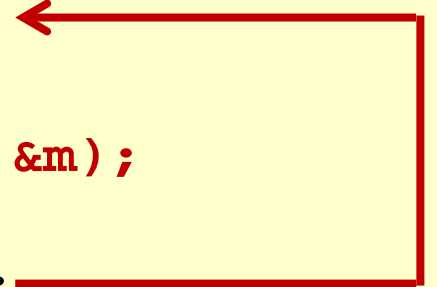
# continue对循环的影响

- 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++;
}
```



# Evil goto's ?

```
START_LOOP: ←  
if (fStatusOk)  
{  
    if (fDataAvaiable)  
    {  
        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: ←
```

这样使用goto，使程序  
迅速退化为垃圾代码



尽量避免使用goto语句，  
尤其不要使用过多的  
goto语句标号，只允许在  
一个单入口单出口的模块  
内向前跳转



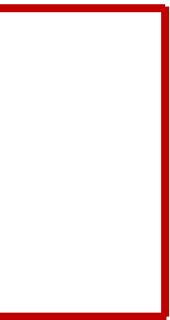


# Evil goto's ? Maybe Not...

## ■ 两种适合使用goto语句的情况

- \* 跳向共同的出口位置，进行退出前的处理工作
- \* 跳出多重循环的一条捷径
- \* {...

```
        {...  
            {...  
                goto Error;  
            }  
        }  
    }  
Error: ←
```



# 用goto语句跳向共同的出口位置

```
void Init(void)
{
    char *p1 = NULL;
    char *p2 = NULL;
    char *p3 = NULL;

    p1 = (char*)malloc(256);
    if (p1 == NULL)
        goto Exit;

    p2 = (char*)malloc(256);
    if (p2 == NULL)
        goto Exit;

    p3 = (char*)malloc(256);
    if (p3 == NULL)
        goto Exit;
```

/\*正常处理的代码\*/

```
Exit:
    if (p1 != NULL)
        free(p1);
```

用goto转向同一语句标  
号处进行相同的错误处理

```
return;
```

```
}
```



# 讨论

```
#include <stdio.h>
int main()
{
    int x = 1;
    int find = 0;          /*置找到标志变量为假*/
    while (!find)
    {
        if (x%5==1 && x%6==5 && x%7==4 && x%11==10)
        {
            printf("x = %d\n", x);
            find = 1; /*置找到标志变量为真*/
            x++;
        }
    }
    return 0;
}
```

这个韩信点兵  
程序错在哪里？



# 讨论

```
#include <stdio.h>
int main()
{
    int x = 1;
    int find = 0;          /*置找到标志变量为假*/
    do{
        find = (x%5==1 && x%6==5 && x%7==4 && x%11==10);
        x++;
    }while (!find);
    printf("x = %d\n", x);
    return 0;
}
```

这个韩信点兵  
程序错在哪里？

