第五章
循环控制

五院三教　许岩

引入

谁做的好事？

班级中有四位同学A、B、C、D，其中一位做了好事，不留名，表扬信来了之后，队长问这四位是谁做的好事。

- A说：不是我。
- B说：是C。
- C说：是D。
- D说：他胡说。

已知：三个人说的是真话，一个人说的是假话。现在请你根据这些信息，编写程序找出做了好事的人。

分析事件真相的可能性

可能性	A	B	C	D
第1种	T	F	F	F
第2种	F	T	F	F
第3种	F	F	T	F
第4种	F	F	F	T

T：Ture；F：False

如何实现推理？

- 用枚举法验证各可能性

按照题目包含的四种可能性，**逐一**对每一种可能性进行**检验**，即：依据该可能性，对四个人的话进行检验，看看有几句是真话。如果不满足三句为真的条件，即不符合题目要求，就抛弃这种可能性，换下一个可能性再试。

引入

如何实现推理？

➢ **用枚举法验证各可能性**

按照四种可能性，逐一对每种可能性去测试四个人的话，

例1 已知：一个整数的平方能被2整除，
求证： 这个数是偶数。

证明：设整数a的平方能被2整除.

假设a不是偶数，则a是奇数，

不妨设a=2m+1（m是整数）

$\therefore a^2 = 4m(m+1)+1$

$\therefore$ a^2是奇数，与已知矛盾

假定“做好事者是A”

如何表示语义？

- 用枚举法验证各可能性

要将自然语言表达的语义（即四个人说的四句话的意思），用计算机可以处理（运算）的式子来表示，只有这样，才能同通过程序语言编程来指挥计算机进行处理。

如何表示语义？

- **判断性的命题如何表示？**

说话人	说的话	写成关系表达式
A	这个人不是我	Thisman != ‘A’
B	是C	Thisman == ‘C’
C	是D	Thisman == ‘D’
D	他（C）胡说	Thisman != ‘D’

逐一测试4种可能

（1）假定thisman是A，代入四句话中

说话人	说的话	关系表达式	
A	Thisman != ‘A’	‘A’ != ‘A’	FALSE
B	Thisman == ‘C’	‘A’==‘C’	FALSE
C	Thisman == ‘D’	‘A’==‘D’	FALSE
D	Thisman != ‘D’	‘A’！ =‘D’	TRUE

逐一测试4种可能 ➢ （2）假定thisman是B，代入四句话中

说话人	说的话	关系表达式	
A	Thisman != ‘A’	‘B’ != ‘A’	TURE
B	Thisman == ‘C’	‘B’==‘C’	FALSE
C	Thisman == ‘D’	‘B’==‘D’	FALSE
D	Thisman != ‘D’	‘B’！ =‘D’	TRUE

逐一测试4种可能

（3）假定thisman是C，代入四句话中

说话人	说的话	关系表达式	
A	Thisman != ‘A’	‘C’ != ‘A’	TURE
B	Thisman == ‘C’	‘C’==‘C’	TURE
C	Thisman == ‘D’	‘C’==‘D’	FALSE
D	Thisman != ‘D’	‘C’！ =‘D’	TRUE

逐一测试4种可能

（4）假定thisman是D，代入四句话中

说话人	说的话	关系表达式	
A	Thisman != ‘A’	‘D’ != ‘A’	TURE
B	Thisman == ‘C’	‘D’==‘C’	FALSE
C	Thisman == ‘D’	‘D’==‘D’	TURE
D	Thisman != ‘D’	‘D’！ =‘D’	FALSE

如何数真话的数量

```
char thisman;
int sum;
thisman='D';

sum=(thisman!='A')+(thisman=='C')+
    (thisman=='D')+(thisman!='D');

if(sum==3)
    printf("this man is D");
else
    printf("this man is NOT D");
```

出现“重复代码”时的风险和隐患

1. 如果使用的算法确实时一样的，那么，当情况发生变化而需要修改时，就需要在多个地方进行修改，而这些地方有可能并不集中，这样就**容易遗漏**
2. 代码长了增加**阅读负担**，对代码逻辑的**理解更加困难**
3. 维护代码的人，在进行优化时需要仔细对比，导致**维护工作量加大**

- 如果你想让计算输出（

Printf（“

- 如果你想让计算输出十

Printf（“0 1 0 1 0

1 0 1”）；

- 如果你想让计算输出一千个 0 1，你该怎么做？

```
Printf（“0 1”）；
Printf（“0 1”）；
Printf（“0 1”）；
Printf（“0 1”）；
Printf（“0 1”）；
Printf（“0 1”）；
Printf（“0 1”）；
Printf（“0 1”）；
Printf（“0 1”）；
Printf（“0 1”）；
```

5.1 while语句

5.2 do-while语句

5.3 for语句

5.4 break和continue语句

5.5 几种循环的比较

5.6 循环的嵌套

循环控制：

就是在给定的条件成立时反复执行某一程序段，被反复执行的程序段称为循环体。

- 在C语言中可以用以下语句来实现循环：

1. 用while语句；
2. 用do--while语句；
3. 用for语句。

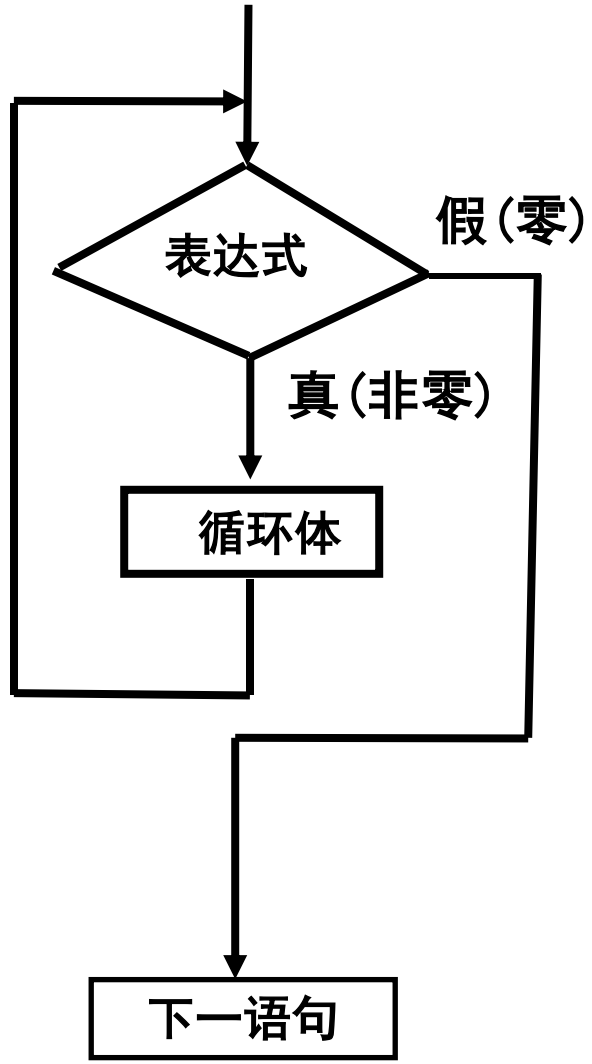
5.1 while语句

❖while 语句的形式：

while（表达式）

语句；

当表达式为真（非0）时，执行其中的内嵌语句（循环体），再判断表达式的值，如此反复；直到表达式的值为假跳出该循环。



5.1 while语句

while(表达式) 语句;

如：k=1;

while(k<=100) {s=s+k;k++;}

循环控制表达式　循环控制变量　循环体

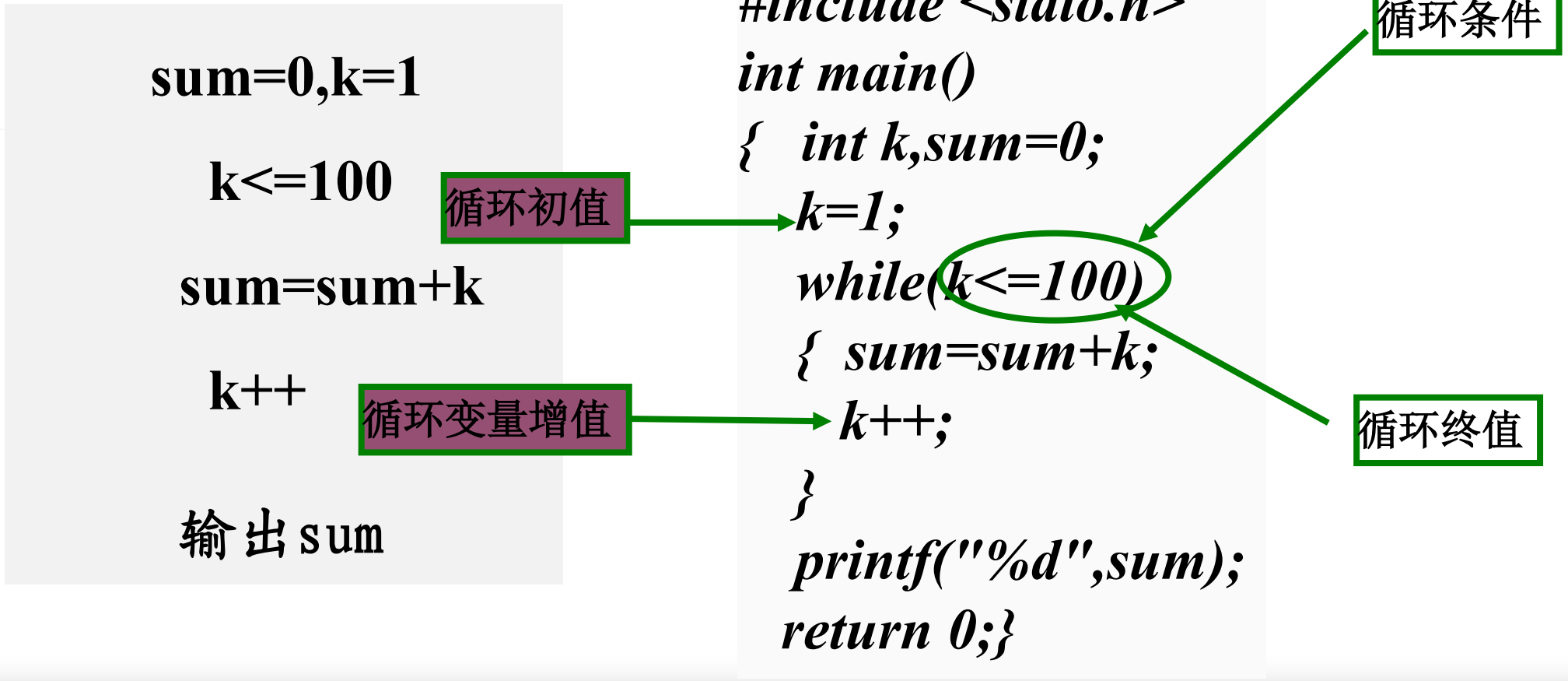
注意：

- 若循环体包含一条语句以上，应以复合语句形式出现
- 循环前必须给循环控制变量赋初值
- 循环体中必须有改变循环控制变量值的语句

  （使循环趋向结束）
- 循环体可以为空

5.1 while语句

例 用while循环求 $\sum_{n=1}^{100} n$



5.1 while语句

【例】求100以内的正奇数、偶数之和

分析:

- 偶数和放在even变量中

even=2+4+6+....+100

- 奇数和放在odd变量中

odd=1+3+5...+99

- 计数器为n，初值为1

```
while(n<100)
 {
      odd = odd+n;
      even=even+(n+1);
      n = n+2;
}
```

```
int main ( )
{
   int n=1,odd=0,even=0;
   while(n<=100)
  {
   if(n%2==0) even+=n;
   else odd+=n;
   n++;
 }
 printf("odd=%d,even=
          %d",odd,even);
 return 0;
}
```

5.1 while语句

【例】统计从键盘输入的一行字符的个数（以回车键作为输入结束标记）。

```
#include <stdio.h>
 int main()
{
    char ch;int num=0;
    ch=getchar();
    while(ch!='\n')
    {
        num++;
        ch=getchar();
    }
    printf("num=%d\n",num);
   return 0;
}
```

判断是否输入结束

5.1 while语句

说明:

- 循环体有可能一次也不执行。
- 循环体中一般有改变条件表达式的语句。
- while（表达式）后面没有分号。
- 无限循环：while（1）

  循环体；

5.1 while语句

```c
#include <stdio.h>

int main() {
    int input;

    while (1) { // 无限循环
        printf("请输入一个数字（输入0退出）：");
        scanf("%d", &input);

        if (input == 0) { // 出口条件
            printf("程序已退出。\n");
            break; // 跳出循环
        }

        printf("你输入的数字是：%d\n", input);
    }

    return 0;
}
```

请写出下面程序运行结果。 [填空1]

```
int main()
{
    int  x=0,s=0;
    while(!x!=0)
        s+=++x;
    printf("s=%d",s);
     return 0;

}
```

5.2 do-while语句

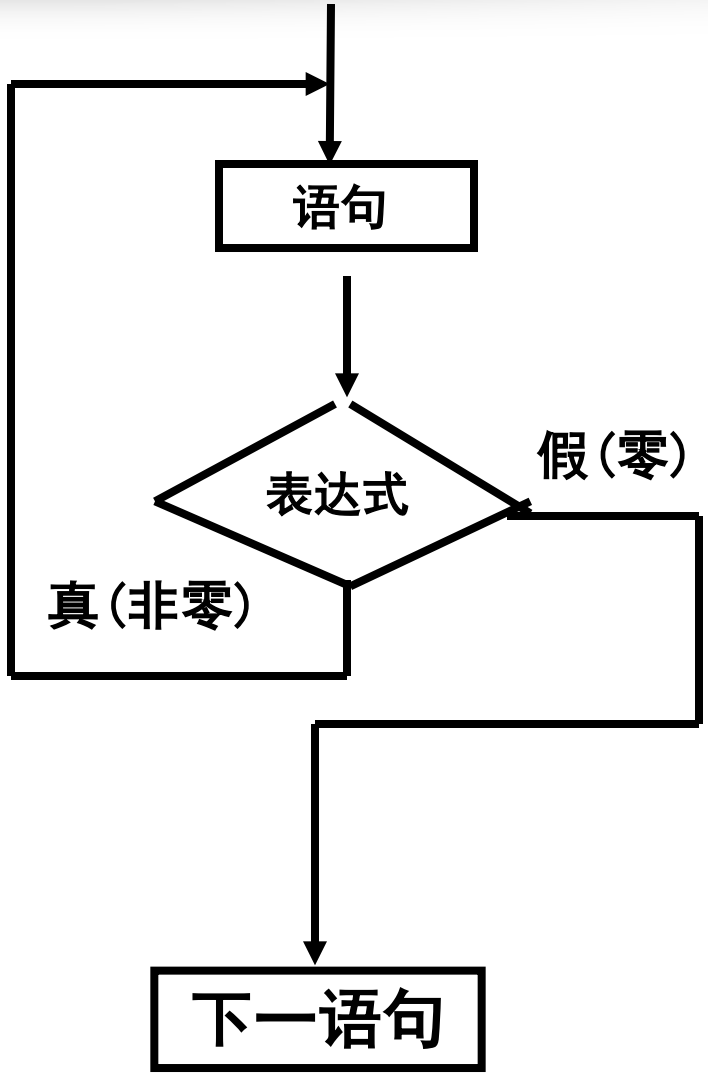
1、 do--while的形式：

do

循环体语句

while（表达式）；

不可省略！切记！



注意点与while语句循环相同
（初始值，终止值，步长变化）

2、说明：

- 先执行语句，后判断表达式。
- 第一次条件为真时，while,do-while等价；

第一次条件为假时，二者不同(后者至少执行了一次循环体)。

5.2 do-while语句

例 用do-while循环求 $\sum_{n=1}^{100} n$

```
#include <stdio.h>
int main()
{   int i,sum=0;
    i=1;
    do
   {
      sum=sum+i ;
      i++ ;
   }while(i<=100);
    printf("%d",sum);
    return 0;}
```

先做后判

```
#include <stdio.h>
int main()
{   int i,sum=0;
    i=1;
    while(i<=100)
    {  sum=sum+i;
       i++;
    }
    printf("%d",sum);
    return 0;}
```

先判后做

5.2 do-while语句

注意：

- 在if、while语句中，表达式后面都没有分号，而在do-while语句的表达式后面则必须加分号。

- do-while和while语句相互替换时，要注意修改循环控制条件 。

请写出下面程序的运行结果。 [填空1]

```
#include<stdio.h>
 int main()
 {
        int y=10;
        do{ y--;}
        while(--y);
        printf(“%d\n”,y--);
        return 0;
  }
```

5.3 for语句

1、for的形式：

for （表达式1；表达式2；表达式3）语句

表达式1：用于循环开始前为循环变量设置初始值。

表达式2：控制循环执行的条件，决定循环次数。

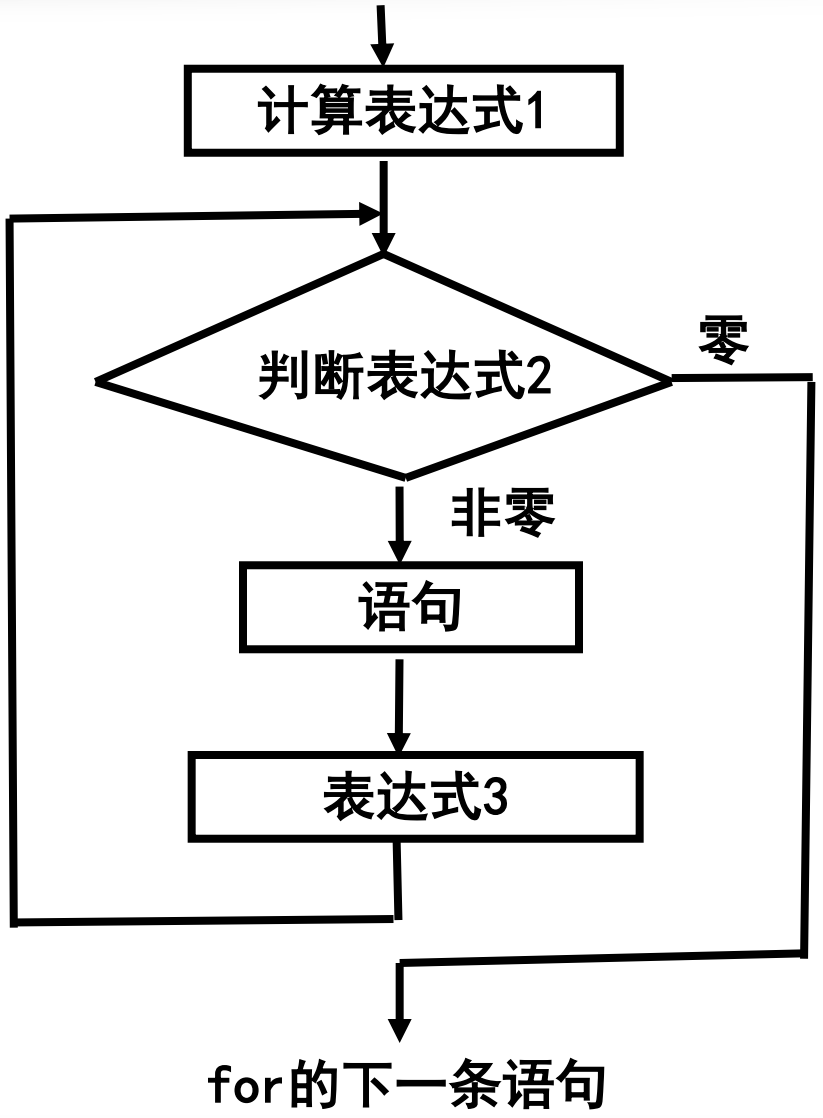
表达式3：循环控制变量修改表达式。

语句： 被重复执行的语句。

5.3 for语句

for（表达式1；表达式2；表达式3）
　　语句

- 表达式1在进入循环之前求解（循环变量赋初值）
- 表达式3是循环体的一部分



例如：

```
for(i=1;i<=100;i++)
        sum=sum+i;
```

它相当于以下语句：

```
i=1;                    ——→ 表达式1;
while (i<=100)          ——→ while（表达式2）
{
    sum=sum+i;
    i++;                ——→ 表达式3;
}
```

程序运行结果

```
#include<stdio.h>
  int main()
 {
        int  i , n,sum;
        for(i=1;i<6;i++)
            sum+=i;
        printf("%d\n",sum);
        return 0;
 }
```

A.15 B.6 C.不确定 D.0

5.3 for语句

编程实现求区间 (m,n)之间（包含 m 和 n ）所有能被 3 整除并且不能被 4 整除的整数之和。

```
#include<stdio.h>
  int main()
 {
        int  i , m,n,sum=0;
      scanf(“%d,%d”,&m,&n);
        for(i=0;i>=m&&i<=n;i++)
            {if(i%3==0&&i%4!=0)
               {sum=sum+i;  }
      }
        printf(“%d\n”,sum);
        return 0;}
```

3、 说明

- 三个表达式都可以是逗号表达式。

```
for（i=1；i<=100;i++,i++） sum=sum+i;
```

```
for（i=1；i<=100;i=i+2） sum=sum+i;
```

- 三个表达式都是任选项，都可以省略，但要注意省略表达式后，分号间隔符不能省略。

5.3 for语句

(1)省略表达式1，即： for （ ;表达式2;表达式3)

```
i=1;
for （ ; i<=100; i++)
sum+=i;
```

分号不能省

(2)省略表达式2，即：for（表达式1； ；表达式3)

等同于：表达式1; while(1) {…表达式3;}

```
for （ i=1 ; ; i++)
{ if(i>100) break; }
```

5.3 for语句

(3)省略表达式3，即：for （表达式1;表达式2;）

等同于：表达式1; while(表达式2) （死循环）

```
for （i=1; i<=100; ）
    {…i++; … }
```

(4)省略表达式1、2、3，即：for（； ；）

等同于： while (1)，会无限循环（死循环）

```
int i=1;
for （  ;  ;  ）
{if(i>100) break;
     sum=sum+i;  i++;  }
```

5.3 for语句

总结：在省略某个表达式时，应在适当位置进行循环控制的必要操作，以保证循环的正确执行。

例 读程序，判断程序的功能。

```
#include <stdio.h>
int main()
{
   char c;
   for(;(c=getchar())!='\n';)
   putchar(c);
   putchar('\n');
   return 0;  }
```

读入一个字符，
当它不是回车时就输出

注意：getchar()仅当遇到回车的时候才开始执行，
从键盘缓冲区中取字符。

例 读程序，判断程序的功能。

```
#include <stdio.h>
int main() {
    int ch;
    printf("请输入（按Ctrl+Z/Ctrl+D结束）：\n");
    while ((ch = getchar()) != EOF) {
        putchar(ch); // 原样输出所有输入，包括换行
    }
    printf("输入结束！\n");
    return 0;
}
```

循环控制程序设计要点：

- 正确确定循环体：
  - 找出循环体

- 对循环体循环的次数进行控制：
  - 循环控制变量三要素（初值、终值、步长）

- 累加器问题
- 累乘器问题
- 枚举法
- 递推法

典型例题分析

【例】 求累加和1+2+3+…+1000

属于“累加器”类型问题。

基本方法:

（1）在进入累加前先给累加器赋初值（一般为0）；

（2）用循环语句实现累加；

for（循环变量赋初值；循环条件；循环变量改变规律）

（3）循环体语句的设计。

累加器当前值=累加器原值+循环变量当前值 ；

典型例题分析

参考程序：

```
main()
{
    long int  k,s;
    s=0;                          累加器赋初值
    for(k=1;k<=1000;k++)
        s=s+k;                    累加
    printf(" s=%ld ",s);
}
```

典型例题分析

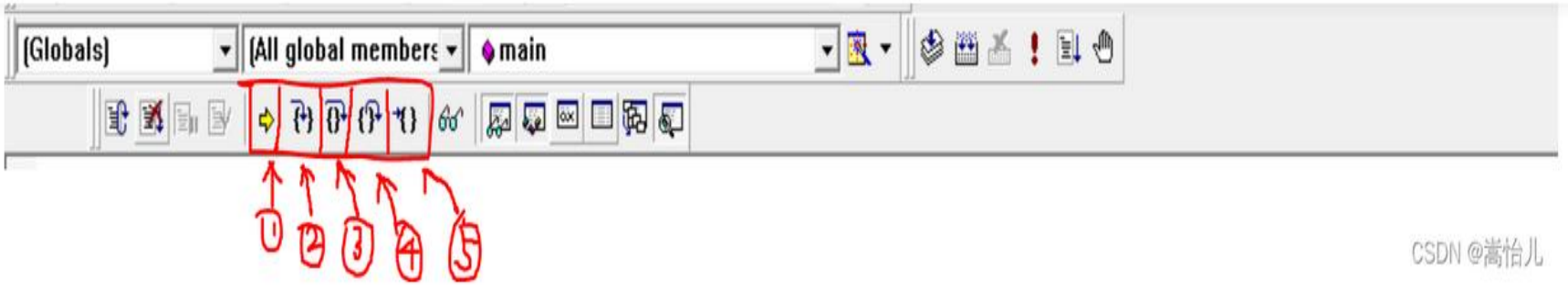
【例】 $s = \frac{2}{1} - \frac{3}{2} + \frac{5}{3} - \frac{8}{5} \cdots$ 求前10项之和

```
#include<stdio.h>
void main()
{
        float s=0,f1=2,f2=1,f=1,t,n;        /*累加器赋初值*/
        for(n=1;n<=10;n++){
                s=s+f*f1/f2;
                /*累加器当前值=累加器原来的值+新的要加的数据*/
                f=f*(-1);t=f2;f2=f1;f1=f1+t;
                /*为求下一个要加的数据做准备*/
        }
        printf("s=%f\n",s);
}
```

控制加/减运算

重新计算分子

重新计算分母



CSDN @嵩怡儿

①：这个黄色箭头相当一个书签，点击可以跳转到正在调试的地方

②：指进入到函数内部调试，一般调试自定义的函数会用到，而一般的库函数就没必要进入其内部调试了，因为那样会跳转到汇编指令，相对来说就底层了，没必要。

③：指跳过函数调试，一般遇到库函数就点这个跳过就好了。

④：指跳出当前调试的函数。

⑤：跳转到光标所指的地方进行调试。

例：求累乘积。如：1×2×3×...×100

属于“累乘器”类型问题。

基本方法：

（1）给累乘器赋初值，一般为1；

（2）用循环语句实现累乘；

for（循环变量赋初值；循环条件；循环变量改变规律）

（3）循环体设计。

累乘器当前值=累乘器原值*循环变量当前值；

参考程序：

```
void main()
{
    double  s=1;
    int k;
    for(k=1;k<=100;k++)
        s=s*k;
    printf(" s=%lf ",s);
}
```

整数连乘结果一定是整数，而本例中结果数值相当大，用long型都无法存放，因此将存放累乘结果的变量s定义为double型。

累乘器赋初值

累乘

例 用0--9这十个数字可以组成多少无重复数字的三位数?

➢编程方法： “枚举法”

按问题本身的性质，一一列举出该问题所有可能的解，并在逐一列举的过程中，检验每个可能解是否是问题的真正解，若是，我们采纳这个解，否则抛弃它。对于所列举的值，既不能遗漏也不能重复。

```
#include <stdio.h>
 void main()          /*a,b,c代表百位、十位、个位*/
{
        int x,a,b,c,num=0;
/*num存放满足条件的数的个数，注意num要赋初值*/
        for(x=100;x<=999;x++)
        {
                a=x/100;b=x/10%10;c=x%10;
                if(a!=b&&a!=c&&b!=c)
                {
                        num++;
                        printf("%5d",x);
                }
        }
        printf("\nnumber=%d",num);
}
```

【例】输入任意一个整数，将其逆序输出，

例如输入1234，输出4321。

```
#include <stdio.h>
void main()
{
        long y,n;
        scanf("%ld",&y)
        while(y!=0) {
                n=y%10;
                printf("%ld",n);
                y=y/10;
        }
  }
```

5.4 典型例题分析

【例】 裴波那契数列的第1、2项分别为1、1，以后各项的值均是其前两项之和。求前30项菲波那契数。

编程方法：“递推法”

所谓递推法就是从初值出发，归纳出新值与旧值间的关系，直到求出所需值为止。新值的求出依赖于旧值，不知道旧值，无法推导出新值。数学上递推公式正是这一类问题。

5.4 典型例题分析

【例】 裴波那契数列的第1、2项分别为1、1，以后各项的值均是其前两项之和。求前30项菲波那契数。

- f1--第一个数　f2--第二个数　f3--第三个数

  f1=1; f2=1; f3=f1+f2;

- 以后只要改变f1,f2的值，即可求出下一个数.

  f1=f2;f2=f3; f3=f1+f2;

参考程序:

```
#include <stdio.h>
int main()
{
    int  f1=1, f2=1, f3;
    int  k;
    printf(" %d\t%d\t ", f1,f2);
     for(k=3;  k<=30;  k++)
     {
            f3=f1+f2;
            printf(" %d\t   ",f3);
            f1=f2; f2=f3;   return 0;
     }  }
```

例　给一年级的小学生出10道100以内的加法练习题。

分析: 1、出题

2、回答

3、判断对错

思考: 1、做对几题

2、总成绩

```
#include “stdlib.h”
main( )
{ int i ,x ,y ,z ; int count=0 ,score=0 ;
  randomize();
  for(i=1; i<=10; i++)
     {x=random(99); y=random(99);
      printf(“%d+%d=”, x , y );
      scanf(“%d”, &z);
      if(z==x+y){printf(“\nright!\n”);
                 count++; score+=10;}
      else printf(“\nwrong!\n”) ;}
   printf(“%d,%d\n”,count,score);
}
```

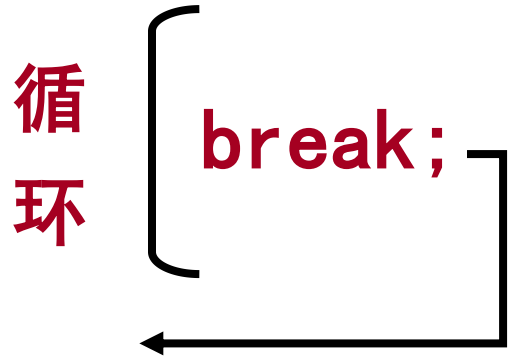
5.5 break语句和continue语句

辅助控制语句

--break语句和continue语句

(1) 语句形式:

break;



(2)作用:

- 结束break所在的 switch语句。
- 结束当前循环，跳出break所在的三种循环结构。

【例】 求300以内能被17整除的最大的数。

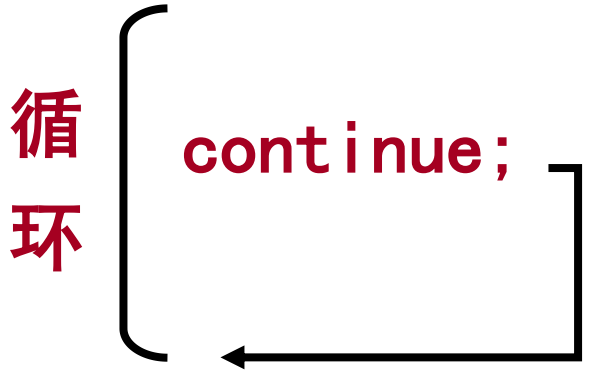
```
#include <stdio.h>
int main()
{       int x,k;
        for(x=300;x>=1;x--)
                if(x%17==0)
                 break;
        printf("x=%d\n",x);
        return 0;  }
```

找到满足条件的最大数，结束循环

2、continue 语句

（1）语句形式：

continue;



（2）语句作用：

仅用于循环语句中，用来结束本次循环。

（3）语句执行流程：

continue语句可以结束本次循环，即不再执行循环体中continue 语句之后的语句，转入下一次循环条件的判断与执行。

5.5 break语句和continue语句

【例】 求300以内能被17整除的所有整数，并打印结果。

```
#include <stdio.h>
 int main()
{
        int x,k;
        for(x=1;x<=300;x++)
       {
              if(x%17!=0) continue;
              printf("%d\t",x);
        }
        return 0;
}
```

请写出程序执行结果。 [填空1]

```
void main()
{
        int a, b;
        for(a=1,b=1; a<=10; a++)
        {
                if(b>=10) break;
                if(b%3==1)
                {
                  b+=3;
                  continue;
                }
                a++;
        }
        printf("a=%d\n",a);
}
```

5.5 break语句和continue语句

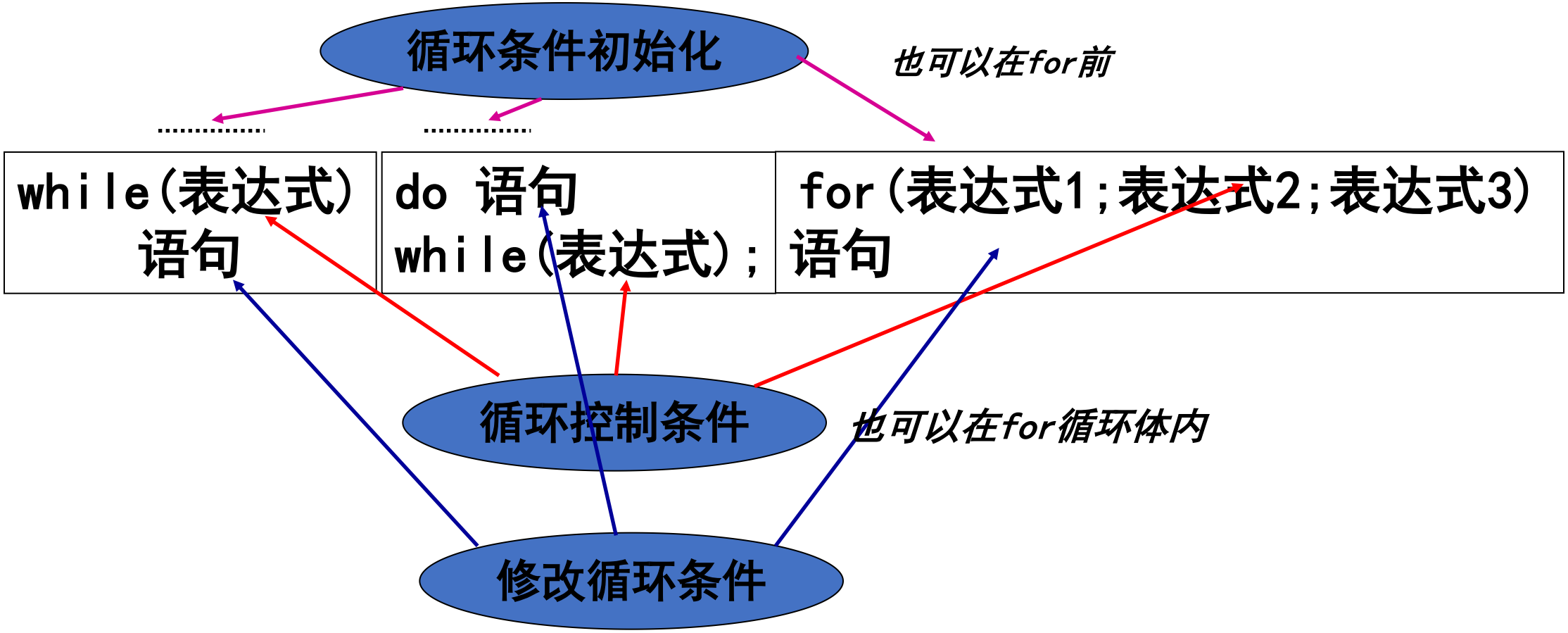
【例】 分析以下程序的运行结果。

```
#include <stdio.h>
 int main()
{
   int a, b;
   for(a=1,b=1; a<=10; a++)
  {
       if(b>=10) break;
   if(b%3==1)
       {
          b+=3; continue;
        }
       a++;
   }
   printf( “a=%d\n",a);
   return 0;
}
```

程序运行结果:

a=4

5.6 几种循环比较


循环条件初始化
也可以在for前
while(表达式)
语句
do 语句
while(表达式);
for(表达式1;表达式2;表达式3)
语句
循环控制条件
也可以在for循环体内
修改循环条件