循环执行

Ⅲ、循环执行

定义：某些代码会被重复执行。

分类：

一、For

1、格式

For(1；2；3)

语句 4;//执行顺序：1→2(2 成立→4→3/不成立退出循环)。3 执行过代表一次循环完成

//1 只执行一次；4 完了一定执行 3；3 完了一定执行 2.

2、执行流程【重点】

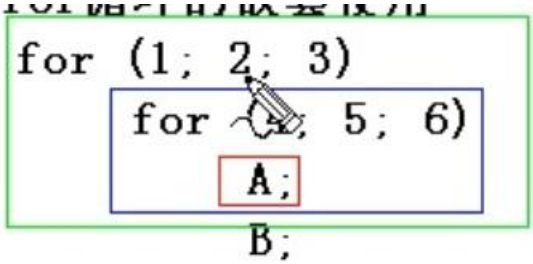
单个 for 循环的使用

1→2(2 成立→4→3/不成立退出循环)。3 执行过代表一次循环完成

多个 for 循环的嵌套使用

```
For(1；2；3)        //1
  For(4；5；6)      //2
     A;             //3
     B;             //4
```

整体式两个语句，1，2，3 是第一个语句；4 是第二个语句



等价于（整体式两个语句）

```
For（1; 2; 3）
    {
    For(4; 5; 6)
        {
            A;
        }
    }
 B;
```

先执行 1→2（成立执行 4→5（成立→A→6→5（成立→A→6），不成立→3））→2（成立继续执行，不成立执行 B）

3、范围问题

For 也是只能控制一个语句，或者使用{}括起来

4、举例【详情见 task】

1+2+3+...+100

1+1/2+1/3+...+1/100

二、Wlile

1、执行流程

格式：

While（表达式）

语句；//执行表达式，成立执行语句，然后继续执行表达式。

//while 默认也只能控制一个语句

2、与 for 的相互比较

For（1； 2； 3）

A;

等价于：

1;

Whlie（2）

{

A;

3;

}

For 与 while 可以相互转换，但 for 的逻辑性更强，推荐多使用 for。

3、While 举例

从键盘输入一个数字，如果该数字是回文数，

则返回 yes，否则返回 no;

回文数：争着写和倒着写都一样

比如说：121，12321，32123 都是回文数。

4、什么时候使用 while，什么时候使用 for

没法说，用多了，自然就知道了。

三、Do...wlile

1、格式：

Do

{

.....

}while（表达式）;

2、执行顺序：先执行.....然后执行表达式，如果表达式成立，继续循环，执行.....

3、与 for、while 的比较

Do...while 并不等价于 for，也不等价于 while。

主要用于人机交互。

5、例子：

一元二次方程（为什么用于人机交互：用户和机器之间的相互交流）

```c
#include<stdio.h>
#include<math.h>
int main(void) {
    double a, b, c;
    double delta;
    double x1, x2;
    char ch;

    do {
        printf("请输入a b c的值(中间用空格隔开):\n");
        scanf("%lf %lf %lf", &a, &b, &c);
        delta = b*b-4*a*c;

        if (delta > 0) {
            x1 = (-b - sqrt(delta))/(2*a);
            x2 = (-b + sqrt(delta))/(2*a);
            printf("该方程有两个解x1 = %lf,  x2 = %lf\n", x1, x2);
        } else if (delta == 0) {
            x1 = (-b - sqrt(delta))/(2*a);
            x2 = x1;
            printf("该方程有一个解: x1=x2=%lf\n", x1);
        } else
            printf("无解！\n");
        printf("您想继续吗？(Y/N)");
        scanf(" %c", &ch);//%c 前面必须得加一个空格，原因略。
    } while ('y'==ch || 'Y'==ch);

    return 0;
}
```

break 和 continue

Break：【重点】

Break 如果用于循环是用来终止循环；

Break 如果用于 switch，则用于终止 switch；

Break 不能直接用于 if，除非 if 属于循环内部的一个子句：

例子：

```c
#include <stdio.h>
int main(void) {
    int i;
    /*  switch(2) {
            case 2:
                printf("haha!\n");
                break;//ok break可以用于switch
        }
    */
    for (i=1; i<3; i++) {
        if (3>2)
            break;//break虽然是if内的子句，但break终止的却是外部的for
        printf("haha!\n");//这个语句永远不会输出
    }

    return 0;
}
```

在多层循环中，break 只能终止距离它最近的循环：

例子：

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

    for (i=0; i<3; ++i) {
        for (j=0; j<4; ++j)
            break;//break只能终止距离它最近的循环
        printf("同志们好！\n");
    }

    return 0;
}
```

在多层 switch 嵌套中，break 只能终止距离它最近的 switch：

例子：

```c
#include<stdio.h>
int main(void) {
    int y, a, b = 0;
    int x = 1;
    switch(x) //第一个switch
    {
        case 1:
            switch(y) //第二个switch
            {
            case 0:
                a++;
                break;//终止第二个switch
            case 1:
                b++;
                break;
            }
            b = 100;
            break;//终止第一个switch
        case 2:
            a++;
            b++;
            break;
    }
    printf("%d %d\n", a, b);
    return 0;
}
```

最终输出结果是：1 100

Continue

用于跳过本次循环余下的语句；

转去判断是否需要执行下次循环。

例子 1：

```c
for (1; 2; 3)
{
    A;
    B;
    continue; //如果执行该语句，则执行完该语句后，会执行语句3，C和D都会被跳过去,C和D不会被执行
    C;
    D;
}
```

例子 2：

```c
while (表达式)
{
    A;
    B;
    continue;  //如果执行该语句，则执行完该语句后，
               //会执行表达式，C和D都会被跳过去,C和D不会被执行
    C;
    D;
}
```