



中山大學
SUN YAT-SEN UNIVERSITY

循环结构

中山大学计算机学院



讲课人：陈武辉

目录

CONTENTS

01

while循环

02

Do...while循环

03

for循环

04

循环嵌套

05

break/continue

历史人物： Ada Lovelace

- 请检索 Ada Lovelace
- 为什么要知道这个名字？

在1842年，Ada Lovelace 在朋友英国数学家、发明家兼机械工程师查尔斯·巴贝其发明的分析机上编写算法和程序，被称为“世界上第一个程序员”。

程序的基本结构元素：**顺序，分支，循环**



奥古斯塔 · 爱达 · 金，勒芙蕾丝伯爵夫人

循环结构 (Loop)

有条件地重复执行程序片段，就是**循环**。例如：

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

要点：

- ① while 和 if 一样是控制语句执行顺序的关键字。其语义也直观易懂
- ② i++ 是完成 += 运算后 i 再加一

4950

注：你可能有更好解法，如梯形公式 $(a + b) * h / 2$ 。但它是最直观的算法，while 作为关键字指出，当后面条件为 true，则不断执行后面复合语句，直到条件为 false。

while循环

```
while(表达式)
{
    语句块
}
```

```
while(表达式)
    语句
```

在**while**循环结构下，首先判断**循环条件表达式**是否为真(即给定的条件是否成立)，如果为真，就执行后面的语句/语句块，也称为**循环体**。

之后不断的重复上述过程，先判断**循环条件表达式**是否为真，为真就执行**循环体**。

直到判断**循环条件表达式**为假，不再执行**循环体**，离开**循环结构**。



while循环

while(表达式)

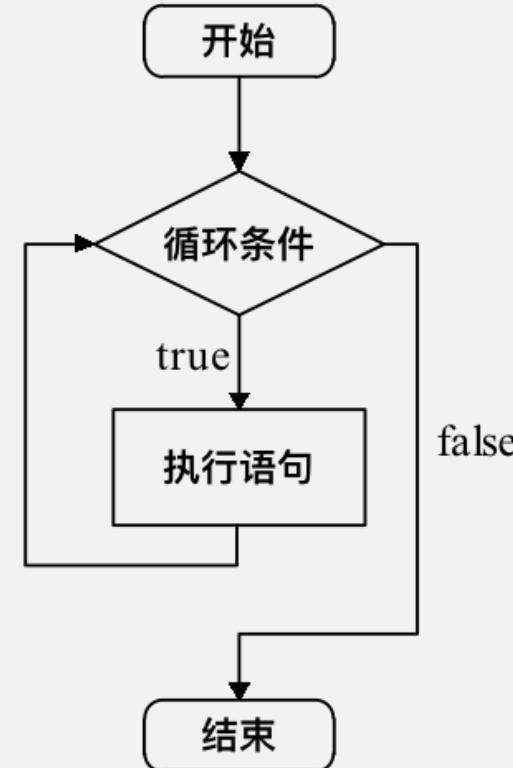
{

语句块

}

while(表达式)

语句



练习 - while循环 – 死循环

修改 loop程序，请问下面程序的输出各是什么？

```
#include<stdio.h>

int main() {
    int sum = 0;
    int i = 1;
    while (i < 100) {
        sum += i;
        i++;
    }
    printf("The sum is %d",sum);
}
```

4950

```
#include<stdio.h>

int main() {
    int sum = 0;
    int i = 1;
    while (i < 100) {
        sum += i;
    }
    printf("The sum is %d",sum);
}
```

循环体与循环条件逻辑上不配合，可能导致循环条件永远无法满足，称为**死循环**



while循环 – 字符计数

```
/*while count string*/
#include <stdio.h>

int main(){
    int n=0;
    printf("Input a string:");
    while(getchar()!='\n') n++;

    printf("Number of characters: %d\n", n);
    return 0;
}
```

字符串

循环 – 技巧 - 使用标志变量控制循环

```
/*while-add-learning*/
#include<stdio.h>
#include<stdbool.h>
#include<stdlib.h>
#include<time.h>

int main() {
    //give seed
    srand((unsigned)time(NULL)); //seed

    int i,j,sum;
    bool flag = true; //标志变量
```

- ① 复习 stdbool.h 使用
- ② 使用标志变量控制循环的技巧
- ③ 进阶理解 scanf 如何处理字符

```
while (flag) {
    i = rand() % 10;
    j = rand() % 10;

    printf("%d + %d = ",i,j);
    scanf("%d",&sum);

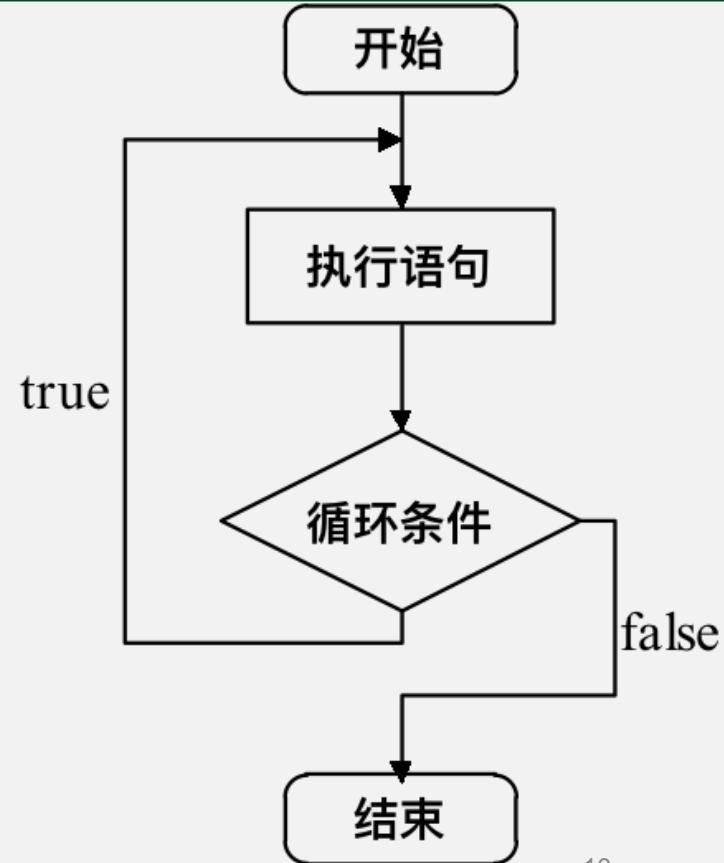
    if (i + j == sum)
        printf("恭喜你，答对了！\n");
    else
        printf("Ooops，再努力一次。\\n");

    char c = getchar();
    if (c=='q'||c=='Q')
        flag = false;
}
```

do...while循环

```
do  
{    语句块  
}  
while(表达式);
```

在do...while循环结构下，先执行一次指定的循环体语句，然后判断表达式，当表达式的值为真时，返回重新执行循环体语句，如此反复，直到判断到表达式的值为假，结束循环。



do...while循环

sum的值有什么不同？

```
int i = 11, sum = 0;
while(i <= 10)
{
    sum += i;
    i++;
}
printf("%d\n", sum);
```

```
int i = 11, sum = 0;
do
{
    sum += i;
    i++;
}
while(i <= 10);
printf("%d\n", sum);
```

do...while循环

sum的值有什么不同？

```
int i = 11, sum = 0;
while(i <= 10)
{
    sum += i;
    i++;
}
printf("%d\n", sum);
```

至少要执行一次“语句块”

```
int i = 11, sum = 0;
do
{
    sum += i;
    i++;
}
while(i <= 10);
printf("%d\n", sum);
```

do...while循环 – 案例研究

```
/*dowhile guess number*/
#include<stdio.h>
#include<stdlib.h>
#include<time.h>

int main() {
    //give a number between 0 .. 99
    srand((unsigned)time(NULL)); //seed
    int target = rand() % 100;

    int guess = -1;
    do {
        printf("Guess a magic number between 0 and 99:");
        scanf("%d",&guess);

        if (guess > target)
            printf("Your guess is too high\n");
        else if (guess < target)
            printf("Your guess is too low\n");
        else
            printf("Yes! game over\n");
    } while (guess != target);
}
```

应用：猜数小游戏

在{}里定义
只能在{}里用
局部变量
不可以

- ① 语句 `int guess = -1;` 写在循环体内可以吗？
- ② 课后练习：可以用 `while` 循环解决问题吗？



for循环

```
for(表达式1; 表达式2; 表达式3)  
{  
    语句块  
}
```

表达式1:

设置初始值, 只执行一次, 可以为零个、一个或多个变量设置初始值。

表达式2:

循环条件表达式, 根据表达式的真假来决定是否继续执行循环。

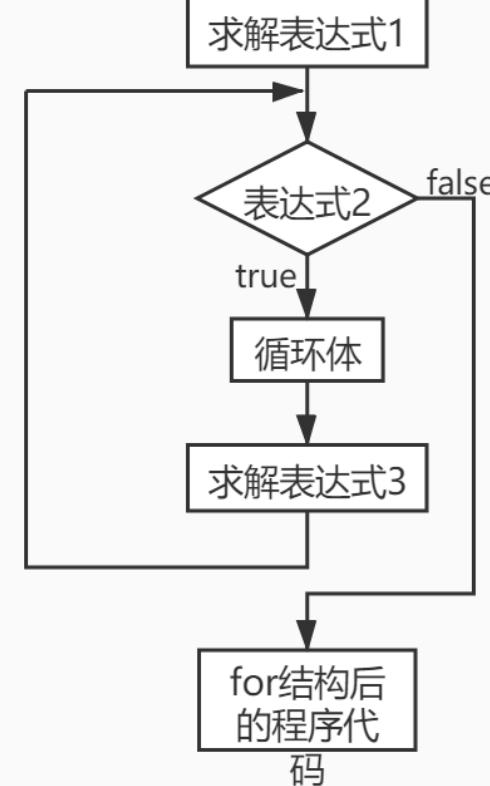
表达式3:

修改循环变量, 如使循环变量增加, 它是执行完循环体后才进行的。



for循环

```
for(表达式1; 表达式2; 表达式3)  
{  
    语句块  
}
```



for循环

for(循环变量赋初值; 循环条件; 循环变量增值)

```
{  
    语句块  
}
```

最常见的for循环语句形式

值得注意的是表达式3，要使循环变量趋于使循环结构”结束”的方向进行，避免死循环

for循环

```
int sum = 0;  
for(int i = 1; i <= 10; i++)  
{  
    sum += i;  
}
```

等价于

```
int i = 1, sum = 0;  
while(i <= 10)  
{  
    sum += i;  
    i++;  
}
```

for 是一种特殊的 while 循环?
那为什么需要 for 语句呢?



for循环 – pk while

```
/*for loop*/
#include<stdio.h>

int main() {
    int sum = 0;
    for (int i = 1; i < 100; i++) {
        sum += i;
    }
    printf("The sum is %d",sum);
}
```

```
#include<stdio.h>

int main() {
    int sum = 0;
    int i = 1;
    while (i < 100) {
        sum += i;
        i++;
    }
    printf("The sum is %d",sum);
}
```

假设在**循环体不存在修改循环变量的语句** ...

- ① 现实中有大量问题需要枚举有明确开始、结束数字或事物的序列
- ② 使用 for 语句符合自然语言语义，语法简明（开始，结束，增量）
- ③ 易于调试，不易出现死循环

for循环 – 循环变量定义位置

```
/*for loop*/  
#include<stdio.h>  
  
int main() {  
    int sum = 0;  
    int i;  
    for (i = 1; i < 100; i++) {  
        sum += i;  
    }  
    printf("To %d, the sum is %d",i,sum);  
}
```

```
/*for loop*/  
#include<stdio.h>  
  
int main() {  
    int sum = 0;  
    for (int i = 1; i < 100; i++) {  
        sum += i;  
    }  
    printf("To %d, the sum is %d",i,sum);  
}
```

i 只能在 for 循环

里面使用

两个程序等价吗?

for循环

```
int sum = 0;  
for(int i = 1; ; i++)  
{  
    sum += i;  
}
```

等价于

```
int i = 1, sum = 0;  
while(1)  
{  
    sum += i;  
    i++;  
}
```

都是死循环



for循环

```
int sum = 0;  
for(int i = 1; i <= 10; )  
{  
    sum += i;  
    i++;  
}
```

```
int sum = 0,i = 1;  
for(; i <= 10; )  
{  
    sum += i;  
    i++;  
}
```

等价于

```
int i = 1, sum = 0;  
while(i <= 10)  
{  
    sum += i;  
    i++;  
}
```

for循环

3个表达式可以同时省略

```
for( ; ; )
```

```
while(1)
```

for循环

“表达式1”和“表达式3”可以是一个简单表达式也可以是逗号表达式

```
for( sum=0,i=1; i<=100; i++ ) sum=sum+i;
```

```
for( i=0,j=100; i<=100; i++,j-- ) k=i+j;
```

while循环

```
#include <stdio.h>
int main(){
    int n=0;
    printf("Input a string:");
    while(getchar()!="\n") n++;
    printf("Number of characters: %d\n", n);
    return 0;
}
```

while循环

```
#include <stdio.h>
int main(){
    int n=0;
    printf("Input a string:");
    \\while(getchar()!="\n") n++;
    for( n=0; (getchar())!="\n"; n+=1 );
    printf("Number of characters: %d\n", n);
    return 0;
}
```

循环嵌套

```
while(表达式1)
{
    while(表达式2)
    {...}
}
```

```
for(表达式1.1;表达式1.2; 表达式1.3)
{
    for(表达式2.1;表达式2.2; 表达式2.3)
    {...}
}
```

一个循环体内又包含另一个完整的循环结构，称为**循环的嵌套**

循环嵌套（Nest）

```
for(int i = 1;i <= 2; i++)
{
    for(int j = 1;j <= 3; j++)
        printf("i = %d,j = %d, i*j = %d\n",i,j,i*j);
}
```

输出结果:

```
i = 1,j = 1,i*j = 1
i = 1,j = 2,i*j = 2
i = 1,j = 3,i*j = 3
i = 2,j = 1,i*j = 2
i = 2,j = 2,i*j = 4
i = 2,j = 3,i*j = 6
```

在该结构中，首先进入第一层循环 $i = 1$ ，满足循环条件 $i \leq 2$ ，执行循环体的内容，而循环体是一个单独的循环，于是执行这个单独的循环，当执行完单独的循环，在执行 $i++$



循环嵌套（Nest）

输出一个 4×4 的整数矩阵

1	2	3	4
2	4	6	8
3	6	9	12
4	8	12	16

问题求解：

因为只能按行输出，因此：

使用 i 代表行，如 $1..4$

使用 j 代表列，如 $1..4$

则 i 行第 j 个数是 $i * j$



循环嵌套（Nest）

```
/*nested print matrix*/
#include <stdio.h>

int main()
{
    const int n = 4;
    for(int i = 1; i <= n; i++){
        for(int j = 1; j <= n; j++){
            printf("%-4d", i*j);
        }
        printf("\n");
    }
    return 0;
}
```

修改常数n为5，运行程序。
思考：这里为什么要定义常数？

1	2	3	4
2	4	6	8
3	6	9	12
4	8	12	16

循环嵌套（Nest）

输出九九乘法表

1*1=1

2*1=2 2*2=4

3*1=3 3*2=6 3*3=9

4*1=4 4*2=8 4*3=12 4*4=16

5*1=5 5*2=10 5*3=15 5*4=20 5*5=25

6*1=6 6*2=12 6*3=18 6*4=24 6*5=30 6*6=36

7*1=7 7*2=14 7*3=21 7*4=28 7*5=35 7*6=42 7*7=49

8*1=8 8*2=16 8*3=24 8*4=32 8*5=40 8*6=48 8*7=56 8*8=64

9*1=9 9*2=18 9*3=27 9*4=36 9*5=45 9*6=54 9*7=63 9*8=72 9*9=81

循环嵌套（Nest）

输出九九乘法表

```
/*nest 99 table*/
#include <stdio.h>

int main(){
    int i, j;

    for(i=1; i<=9; i++){ //外层for循环
        for(j=1; j<=i; j++){ //内层for循环
            printf("%d*%d=%-2d ", i, j, i*j);
        }
        printf("\n");
    }

    return 0;
}
```

break/continue语句

```
break;
```

break语句可以使当前流程跳出switch结构，继续执行switch结构后的下一条语句。

break语句还用来从循环体中跳出整个循环结构，即提前结束循环，执行循环结构后的下一条语句。

```
continue;
```

有时候并不希望终止整个循环，只是想提前结束本次循环，而接着执行下次循环

break/continue语句

```
for(int i = 1;i <= 10; i++)  
{  
    if(i == 5) break;  
    printf("i = %d\n",i);  
}
```

i = 1
i = 2
i = 3
i = 4

这个for循环结构的作用为打印数字1-10

第一个程序中：

当i = 5时，满足if语句的判断，执行break语句。
此后跳出整个循环，循环结束。

```
for(int i = 1;i <= 10; i++)  
{  
    if(i == 5) continue;  
    printf("i = %d\n",i);  
}
```

i = 1
i = 2
i = 3
i = 4
i = 6
i = 7
i = 8
i = 9
i = 10

第二个程序中：

当i = 5时，满足if语句的判断，执行continue语句，此后跳过本次循环中continue语句后面语句的执行，转入下一次循环。

break/continue语句 – 练习

求满足 $1+2+3+\dots+n \leq 100$ 条件的 n 最大值

建议在 loop.c 基础上修改



中山大學

SUN YAT-SEN UNIVERSITY

谢谢

中山大学计算机学院



编制人：课题组