Never be tired to repeat.

C语言程序设计

C Programming

6

C控制语句：循环

理论课程

厦門大學

XIAMEN UNIVERSITY

信息学院 黄 炜

(特色化示范性软件学院) 博士,副教授

School of Informatics Wei Huang

内容要点

- 循环的要素
- 循环的流程
- 关系运算符
- 循环的灵活用法
- 循环的中途退出
- 数组初步

目录

循环的三个要素

- 一般的循环具有三个要素
  - 初始状态的设定
  - 循环的条件
  - 循环状态的改变
- 特殊的循环
  - 死循环：循环条件永远为真，无法停止
  - 无意义的循环：循环条件永远为假，无法进入循环

三种类型的循环

- 三种类型的循环
  - do-while 循环的循环体至少执行一次

选用何种循环应根据实际而定，使得代码简单

```
[初始状态;]
while (<循环条件表达式>)
{
    [循环语句（含状态的改变）]
}
```

```
[初始状态;]
do
{
    [循环语句（含状态的改变）]
}
while (<循环条件表达式>);
```

```
for ([初始状态]; [循环条件表达式]; [循环状态的改变])
{
    [循环语句（可包含状态的改变）]
}
```

```c
/* summing.c -- sums integers entered interactively */
#include <stdio.h>
int main(void) {
    long num;
    long sum = 0L;      /* initialize sum to zero   */
    int status;
    printf("Please enter an integer to be summed ");
    printf("(q to quit): ");
    status = scanf("%ld", &num);
    while (status == 1) {  /* == means "is equal to"   */
        sum = sum + num;
        printf("Please enter next integer (q to quit): ");
        status = scanf("%ld", &num);
    }
    printf("Those integers sum to %ld.\n", sum);
    retur
}
```

初始状态

循环条件表达式
（==表示判断是否相等）

状态改变

```
Please enter an integer to be summed (q to quit): 55↲
Please enter next integer (q to quit): 33↲
Please enter next integer (q to quit): 235↲
Please enter next integer (q to quit): q↲
Those integers sum to 323.
```

```c
// when.c -- when a loop quits
#include <stdio.h>
int main(void)
{
    int n = 5;

    while (n < 7)                       // line 7
    {
        printf("n = %d\n", n);
        n++;                            // line 10
        printf("Now n = %d\n", n);      // line 11
    }
    printf("The loop has finished.\n");

    return 0;
}
```

```
n = 5
Now n = 6
n = 6
Now n = 7
The loop has finished.
```

目录

循环的流程

- 循环符合条件则进循环体；否则跳过循环体。

```
[初始状态;]
while (<循环条件表达式>) {
    [循环语句（含状态的改变）]
}
[循环体之外的语句]
```

```
[初始状态;]
while (<循环条件表达式>) {
    [循环语句（含状态的改变）]
}
[循环体之外的语句]
```

```
[初始状态;]
do {
    [循环语句（含状态的改变）]
}
while (<循环条件表达式>);
[循环体之外的语句]
```

```
[初始状态;]
do {
    [循环语句（含状态的改变）]
}
while (<循环条件表达式>);
[循环体之外的语句]
```

说明：🟢表示条件满足和语句成功执行，🔴表示条件不满足。

循环的流程

- 循环符合条件则进循环体；否则跳过语句体。

```
for ([初始状态]; [循环条件表达式]; [循环状态的改变]) {
    [循环语句（可包含状态的改变）]
}
[循环体之外的语句]
```

1 2 5 8 4 7 3 6 9

```
for ([初始状态]; [循环条件表达式]; [循环状态的改变]) {
    [循环语句（可包含状态的改变）]
}
[循环体之外的语句]
```

1 2 3

说明：🟢表示条件满足和语句成功执行，🔴表示条件不满足。

循环的流程

- 循环的终止
  - 符合条件则进入循环语句体；否则跳过语句体。

死循环（永不停止）

```
int index = 1;
while (index < 5)
    printf("%d\n", index);
```

状态不改变

循环3次

```
int index = 1;
while (++index < 5)
    printf("%d\n", index);
```

循环$2^{31}+1$次

```
int index = 1;
while (--index < 5)
    printf("%d\n", index);
```

不循环（循环0次）

```
int index = 10;
while (++index < 5)
    printf("%d\n", index);
```

```c
/* while1.c -- watch your braces       */
/* bad coding creates an infinite loop */
#include <stdio.h>
int main(void)
{
    int n = 0;

    while (n < 3)
        printf("n is %d\n", n);
        n++;
    printf("That's all this program does\n");

    return 0;
}
```

C程序中，空白字符（空格、制表符、回车）是相同的。不能用缩进来判断循环体的范围。正确的缩进，便于读者理解。

```
n is 0
n is 0
n is 0
n is 0
（永不停止）
```

循环的语句体范围

- 循环的语句体范围
  - 在while/for后最近（do-while间）的语句体或一条语句
  - 如果循环的内容是一条语句，可以不用花括号分隔。
    - 注意不要多加分号，因为空语句“;”也是一条语句

```
while (a<5) {
    a++;
    printf("%d",a);
}
```

```
for (i=0; i<5; i++) {
    a++;
    printf("%d",a);
}
```

```
do {
    a++;
    printf("%d",a);
} while (a<5);
```

```
while (a<5)
    a++;
printf("%d",a);
```

```
for (i=0; i<5; i++)
    a++;
printf("%d",a);
```

```
while (a<5);
a++;
```

此处有分号

循环的语句体范围

- 循环的语句体范围
  - 循环的内容是在while或for之后最近的（或do-while之间的）一个语句体或一条语句，用花括号分隔，缩进一层。
  - 如果循环的内容是一条语句，可以不用花括号分隔。
    - 注意不要多加分号，因为空语句“;”也是一条语句

```
while (a<5)
    a++;
printf("%d",a);
```

```
for (i=0; i<5; i++)
    a++;
printf("%d",a);
```

```
do
    a++;
while (a<5);
printf("%d",a);
```

```
while (a<5);
a++;
```

```
for (i=0; i<5; i++);
a++;
```

此处有分号

```c
/* while2.c -- watch your semicolons */
#include <stdio.h>
int main(void)
{
    int n = 0;

    while (n++ < 3);                /* line 7 */
    printf("n is %d\n", n);         /* line 8 */
    printf("That's all this program does.\n");

    return 0;
}
```

在不该打分号的地方应注意。
如应有分号，应另起一行引起注意。

```
n is 4
That's all this program does.
```

目录

不确定循环和计数循环

- 两种循环

循环	特点	示例代码

不确定循环	不能预先知道循环了多少次	`for (int i = 0; str[i] != '\0'; i++) {`
		`  str[i] = str[i] - 'a' + 'A';`
		`}`

计数循环	存在计数器，初始化，判断条件，更新计数	`for (int i = 0; i < N; i++) {`
		`  str[i] = str[i] - 'a' + 'A';`
		`}`

```
for (初始状态; 循环表达式; 状态改变) {
    循环体;
}
```

当初始状态、循环条件、状态改变可以在40字符内写完时，用for语句书写，否则用while。

```c
// sweetie1.c -- a counting loop
#include <stdio.h>

int main(void)
{
    const int NUMBER = 22;
    int count = 1;                          // initialization

    while (count <= NUMBER)                 // test
    {
        printf("Be my Valentine!\n");       // action
        count++;                            // update count
    }

    return 0;
}
```

```
Be my Valentine!
Be my Valentine!
Be my Valentine!
（此处省略数行）
Be my Valentine!
```

```c
// sweetie2.c -- a counting loop using for
#include <stdio.h>
int main(void)
{
    const int NUMBER = 22;
    int count;

    for (count = 1; count <= NUMBER; count++)
        printf("Be my Valentine!\n");

    return 0;
}
```

仔细检查开始和结束的条件
注意是<=和<之间的差异。

```
Be my Valentine!
Be my Valentine!
Be my Valentine!
（此处省略数行）
Be my Valentine!
```

for循环的使用方法

- 语句之间可以互换，表达式之间可以互换
  - 但表达式不可以使用语句替换；语句可以使用表达式语句

方法	示例
减量运算符	`for (i = 5; i >= 0; --i)`
一次加减多步	`for (i = 5; i >= 0; i -= 2)`
用其它类型计数	`for (i = 'a'; i <= 'z'; ++i)`
其它循环条件	`for (i = 5; i * i * i >=1; i -= 2)`
算术或几何增加	`for (i = 0; i <= 10; i *= 1.2)`
用复杂的表达式（控制在一行）	`for (printf("Enter a number: "); scanf("%d",&num); printf("Enter again: "));`
任何表达式为空	`for (; i >= 0; i -= 2)` 或 `for (;;)`

前、后自增的效率问题

扩展内容

- ++i 相当于下列代码（效率较高）

```
i = i + 1;
return i;
```

- i++ 相当于下列代码

```
j = i;
i = i + 1;
return j;
```

- 使用编译器的优化选项，会将该效率差别消除

```c
/* for_cube.c -- using a for loop to make a table of cubes */
#include <stdio.h>
int main(void)
{
    int num;

    printf("    n   n cubed\n");
    for (num = 1; num <= 6; num++)
        printf("%5d %5d\n", num, num*num*num);

    return 0;
}
```

```
    n   n cubed
    1       1
    2       8
    3      27
    4      64
    5     125
    6     216
```

序列求和

- 题目：序列求和
  - 古希腊哲学家Zeno（芝诺）说：一支箭从弓箭手手里射出去，理论上这只箭永远到达不了靶子上。一开始这只箭飞到了弓箭手与靶子的一半位置，接着飞到该位置与靶子的一半位置（1/4），接着是（1/8），以此类推，它永远是在某个点与靶子中间的位置，它永远到不了靶子上。
- 思路
  - 输入循环次数，每次循环p*=2，计算t=t+p。

```c
/* zeno.c -- series sum */
#include <stdio.h>
int main(void) {
    int t_ct;        // term count
    double time, power_of_2;
    int limit;
    printf("Enter the number of terms you want: ");
    scanf("%d", &limit);
    for (time=0, power_of_2=1, t_ct=1; t_ct <= limit;
                                t_ct++, power_of_2 *= 2.0) {
        time += 1.0/power_of_2;
        printf("time = %f when terms = %d.\n", time, t_ct);
    }
    return 0;
}
```

```
Enter the number of terms you want: 7↲
time = 1.000000 when terms = 1.
time = 1.500000 when terms = 2.
time = 1.750000 when terms = 3.
（此处省略数行）
time = 1.984375 when terms = 7.
```

```c
/* do_while.c -- exit condition loop */
#include <stdio.h>
int main(void) {
    const int secret_code = 13;
    int code_entered;
    do {
        printf("To enter the triskaidekaphobia therapy club,\n");
        printf("please enter the secret code number: ");
        scanf("%d", &code_entered);
    } while (code_entered != secret_code);
    printf("Congratulations! You are cured!\n");
    return 0;
}
```

```
To enter the triskaidekaphobia therapy club,
please enter the secret code number: 32↲
To enter the triskaidekaphobia therapy club,
please enter the secret code number: 23↲
To enter the triskaidekaphobia therapy club,
please enter the secret code number: 13↲
Congratulations! You are cured!
```

```c
/* entry.c -- entry condition loop */
#include <stdio.h>
int main(void)
{
    const int secret_code = 13;
    int code_entered;
    printf("To enter the triskaidekaphobia therapy club,\n");
    printf("please enter the secret code number: ");
    scanf("%d", &code_entered);
    while (code_entered != secret_code)
    {
        printf("To enter the triskaidekaphobia therapy club,\n");
        printf("please enter the secret code number: ");
        scanf("%d", &code_entered);
    }
    printf("Congratulations! You are cured!\n");
    return 0;
}
```

目录

循环的中止（break）

- 循环运行到break语句，跳出语句体。

```
[初始状态;]
while (<条件表达式>)
{
    [循环语句]
    if (<退出条件>)
      break;
    [循环语句]
}
[循环体之外的语句]
```

```
for ([初始状态]; [条件表达式]; [状态改变])
{
    [循环语句（可包含状态的改变）]
    if (<退出条件>)
      break;
    [循环语句（含状态的改变）]

}
[循环体之外的语句]
```

说明：🟢表示条件满足和语句成功执行，🔴表示条件不满足。

循环的继续（continue）

- continue语句中断本层循环，跳到本次循环结束
  - while循环跳入判断表达式；for循环跳到状态改变语句

```
[初始状态;]
while (<条件表达式>)
{
    [循环语句]
    if (<退出条件>)
      continue;
    [循环语句]
}
[循环体之外的语句]
```

```
for ([初始状态]; [条件表达式]; [状态改变])
{
    [循环语句]
    if (<退出条件>)
      continue;
    [循环语句（含状态的改变）]

}
[循环体之外的语句]
```

说明：🟢表示条件满足和语句成功执行，🔴表示条件不满足。

嵌套循环（nest loop）

- 循环内部可以嵌套循环

```
for (row = 0; row < ROWS; row++) {
    for (ch = 'A'; ch < ('A' + CHARS); ch++)
        printf("%c", ch);
    printf("\n");
}
```

- 循环中的中止和继续语句只能退出当前循环

```
for (i = 0; i < 10; i++) {
    for (j = 0; j < 10; j++) {
        if ((i * j) % 2 == 0)
            continue;
        if ((i * j) % 13 == 0)
            break;
        printf("(%d, %d) ", i, j);
    }
    printf("\n");
}
```

```c
/* rows1.c -- uses nested loops */
#include <stdio.h>
#define ROWS  6
#define CHARS 10
int main(void) {
    int row;
    char ch;

    for (row = 0; row < ROWS; row++) {
        for (ch = 'A'; ch < ('A' + CHARS); ch++)
            printf("%c", ch);
        printf("\n");
    }

    return 0;
}
```

```
ABCDEFGHIJ
ABCDEFGHIJ
ABCDEFGHIJ
ABCDEFGHIJ
ABCDEFGHIJ
ABCDEFGHIJ
```

```c
// rows2.c -- using dependent nested loops
#include <stdio.h>
int main(void)
{
    const int ROWS = 6;
    const int CHARS = 6;
    int row;
    char ch;

    for (row = 0; row < ROWS; row++) {
        for (ch = ('A' + row); ch < ('A' + CHARS); ch++)
            printf("%c", ch);
        printf("\n");
    }

    return 0;
}
```

内外层循环的条件之间不必相互独立

```
ABCDEF
BCDEF
CDEF
DEF
EF
F
```

```c
/* skippart.c  -- uses continue to skip part of loop */
#include <stdio.h>
int main(void) {
    const float MIN = 0.0f;
    const float MAX = 100.0f;
    float score;
    float total = 0.0f;
    int n = 0;
    float min = MAX;
    float max = MIN;
    printf("Enter the first score (q to quit): ");
    while (scanf("%f", &score) == 1) {
        if (score < MIN || score > MAX) {
            printf("%0.1f is an invalid value. Try again: ", score);
            continue;  // jumps to while loop test condition
        }
        printf("Accepting %0.1f:\n", score);
        min = (score < min)? score: min;
        max = (score > max)? score: max;
        total += score;
```

越界判断

求最大（小）值的方法是将当前值与最大（小）值判断，如果更大（小），则记录下来

```
        n++;
        printf("Enter next score (q to quit): ");
    }
    if (n > 0) {
        printf("Average of %d scores is %0.1f.\n", n, total / n);
        printf("Low = %0.1f, high = %0.1f\n", min, max);
    }
    else
        printf("No valid scores were entered.\n");
    return 0;
}
```

有经验的程序员在使用除数之前会判断除数是否为0

在程序无法得到正确答案时，程序应提供错误信息

```
Enter the first score (q to quit): 123↲
123.0 is an invalid value. Try again: 59↲
Accepting 59.0:
Enter next score (q to quit): 100↲
Accepting 100.0:
Enter next score (q to quit): 85↲
Accepting 85.0:
Enter next score (q to quit): 83↲
Accepting 83.0:
Enter next score (q to quit): 0↲
Accepting 0.0:
Enter next score (q to quit): q↲
Average of 5 scores is 65.4.
Low = 0.0, high = 100.0
```

```c
/* break.c -- uses break to exit a loop */
#include <stdio.h>
int main(void) {
    float length, width;

    printf("Enter the length of the rectangle:\n");
    while (scanf("%f", &length) == 1) {
        printf("Length = %0.2f:\n", length);
        printf("Enter its width:\n");
        if (scanf("%f", &width) != 1)
            break;
        printf("Width = %0.2f:\n", width);
        printf("Area = %0.2f:\
        printf("Enter the leng
    }
    printf("Done.\n");

    return 0;
}
```

有经验的程序员在用户输入错误时停止运算

```
Enter the length of the rectangle:
15↲
Length = 15.00:
Enter its width:
30↲
Width = 30.00:
Area = 450.00:
Enter the length of the rectangle:
q↲
Done.
```

循环计算数值的整数次幂

- 应设计一个计算次幂的函数
- 循环以输入是否合法为条件
  - 有些人以输入-1或0为退出条件，错的
  - 正常输入应该返回正确的结果
- 有一定的提示信息

```c
// power.c -- raises numbers to integer powers
#include <stdio.h>
double power(double n, int p); // ANSI prototype
int main(void) {
    double x, xpow;
    int exp;
    printf("Enter a number and the positive integer power");
    printf(" to which\nthe number will be raised. Enter q");
    printf(" to quit.\n");
    while (scanf("%lf%d", &x, &exp) == 2) {
        xpow = power(x, exp);   // function call
        printf("%.3g to the power %d is %.5g\n", x, exp, xpow);
        printf("Enter next pair of numbers or q to quit.\n");
    }
    printf("Hope you enjoyed this power trip -- bye!\n");
    return 0;
}
double power(double n, int p) {  // function definition
    double pow = 1;
    int i;
    for (i = 1; i <= p; ++i)
        pow *= n;
    return pow;
}
```

函数应声明，再使用，有定义

有经验的程序员先判断scanf()的返回值再处理

```
Enter a number and the positive integer
power to which
the number will be raised. Enter q to quit.
3 5↵
3 to the power 5 is 243
Enter next pair of numbers or q to quit.
q↵
Hope you enjoyed this power trip -- bye!
```

无条件跳转（goto）

- 无条件跳转语句跳到指定的标签处
  - 不允许
- 一次返回多个循环可使用goto语句
  - 滥用goto会使程序难以理解

无条件跳转（goto）

goto

功能	无条件跳转	
格式	goto *tag*;	
参数	*tag*	同一个函数内的标签。
说明	1. 无条件跳转的标签仅限于同一个函数内。 2. 尽量避免使用回向goto和长距离goto，即尽量避免跳转回之前语句的标签，避免跳转到太远的标签。 3. 结构化编程建议使用其他语句替代goto。	

使用场景

- 代码异常处理

```c
int mystrlen(char* str) {
    int ret=0;
    if (str == NULL) {
        ret = -1;
        goto error_tag;
    }
    if (*str == 0) {
        ret = 0;
        goto error_tag;
    }
    while (*str != 0) {
        ret++;
        str++;
    }
    return ret;
error_tag:
    fprintf(stderr, "%s", "Empty.\n");
    return 0;
}
```

- 一次性跳出多层循环

```c
while (exp1) {
    while (exp2) {
        if (exp3) {
            goto out;
        }
    }
}
out: printf("Exit!\n");
```

目录

循环前未设置初始状态

- 每次进入循环前应设置初始状态，否则结果不可控
  - 循环最后设置初始状态是不好的编程习惯

```
for (int i = 0; i < 10; i++) {
    for (int j = 0; j < i; j++)
        s += arr[j];
    printf("S-%d: %d\n", i, s);
    s = 0;
}
```

循环后设置初始状态是不好的编程习惯

```
for (int i = 0; i < 10; i++) {
    int s = 0;
    for (int j = 0; j < i; j++)
        s += arr[j];
    printf("S-%d: %d\n", i, s);
}
```

尽量在最小适用范围声明变量并初始化

循环使用的场合

- 流程大致相同，代码位置相邻
- 仅有次序区别，或没有区别

```
switch (month) {
    case 1: total = 0; break;
    case 2: total = 31; break;
    case 3: total = 31 + 28; break;
    ...
    case 12: total = ...; break;
}
```

流程大致相同，仅有次序区别

```
int month_days[] = { 31,28,31,30,
    31,30,31,31,30,31,30,31 };
int total = 0;

for (int i = 0; i < month - 1; i++) {
    total += month_days[i];
}
```

循环的终止条件错误常引发答案或者运行错误

循环的起止

- 循环的起止条件的书写应顺应变量的物理含义
  - 变量名和变量的使用应具备物理意义，否则误导读者

```
int month_days[] = { 31,28,31,30,31,
    30,31,31,30,31,30,31 };
int total = 0;

for (int i = 1; i < month; i++) {
    total += month_days[i - 1];
}
```

变量i是数组下标，故意以1开始，下标使用i-1不直观

```
int month_days[] = { 31,28,31,30,
    31,30,31,31,30,31,30,31 };
int total = 0;

for (int i = 0; i < month - 1; i++) {
    total += month_days[i];
}
```

循环变量为数组下标的，一般从0开始，循环条件使用小于号而不是小于等于号。

目录

C语言程序设计

C Programming

6

谢谢观看

理论课程

厦門大學

XIAMEN UNIVERSITY

信息学院 黄 炜

(特色化示范性软件学院) 博士,副教授

School of Informatics Wei Huang