知识点1【类型转换】（了解）

1、无符号和有符号 参加运算  需要将有符号 转换成无符号

2、int和double参加运算 会将int转成从double类型

3、char和short类型 只要参加运算 都会将自己转换成int类型

知识点2【运算符】（了解）

1、运算符的概述

2、算数运算符

1、如果/的所有运算对象都是整数  /的功能就是取整

2、如果/有一个运算对象是实型  /的功能就是除法运算

3、%取余运算符

4、复合运算 +=  -=  *=  /=  %=

3、关系运算符

4、逻辑运算符 &&  ||  !

&&逻辑与:

||逻辑或:

！逻辑非:

5、产生随机数

6、位运算（二进制位运算）

1、& 按位与

2、| 按位或

3、~按位取反

4、^ 按位异或

5、左移<<:左边丢弃 右边补0

5、右移>>:右边丢弃 左边补0（补1）

无符号数：右边丢弃 左边补0

有符号数：

高级应用：

7、三目运算符

8、逗号运算符,

知识点3【优先级】（了解）

知识点4【自增自减运算符 ++ --】（了解）

1、不管是i++ i-- ++i --i 如果是独立的一条语句 那么i++和++i, i--和--i没区别

2、i++ i-- 复合运算 ++ --在运算对象的右边（先使用 后加减）

3、++i --i 复合运算 ++ --在运算对象的左边（先加减 后使用）

知识点5【if控制语句】（重要）

1、如果只在乎项目 某一个结果 需要使用if语句

案例1：键盘输入一个数 判断它能被3整除

2、如果项目有两个结果 但是不会同时出现 需要使用if...else语句

案例1：键盘输入一个数 判断它能否被3整除

3、如果项目有多个结果 但是不会同时出现 需要使用if...else if...else if.....else....语句

案例1：键盘输入一个数 判断它对3的余数

3、如果项目有多个结果 不确定同时出现 需要使用if独立判断每个结果

知识点6【switch选择语句】（重要）

案例1：键盘输入1~7的数值 判断是星期几（周日为7）

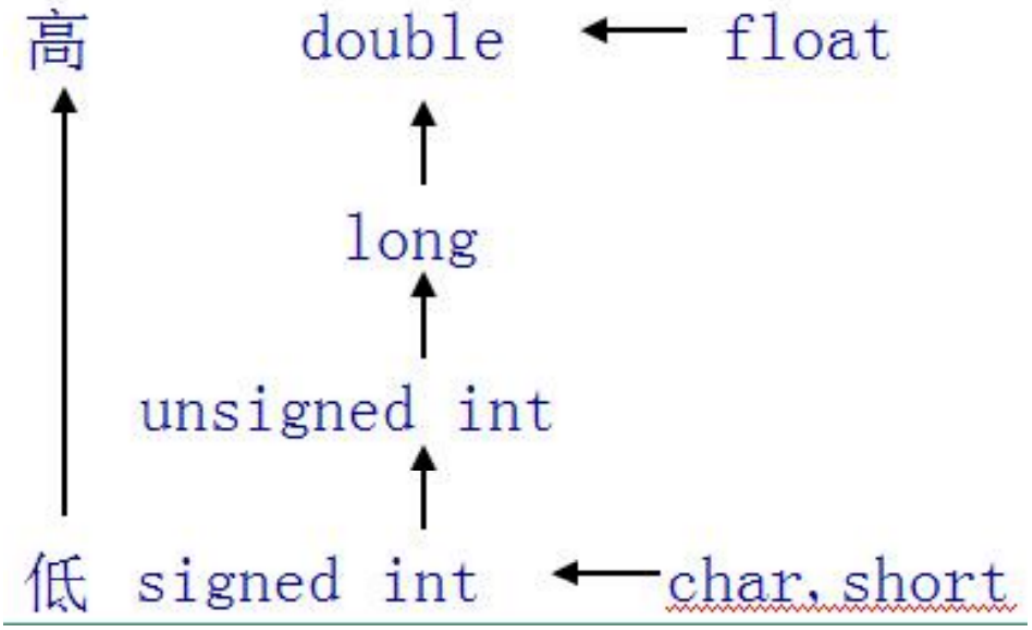
案例2：特殊情况 可以省略break

知识点7【for循环语句】（重要）

# 知识点1【类型转换】（了解）

不同类型数据之间进行混合运算时必然涉及到类型的转换问题。

自动类型转换：保证精度不丢失　将小的类型 转成 大类型




## 1、无符号和有符号 参加运算　需要将有符号 转换成无符号

```c
#include <stdio.h>
int test01()
{
    int data1=-10;
    unsigned int data2 = 6;
    if(data1+data2>0)
    {
        printf(">0\n");
    }
    else
    {
        printf("<=0\n");
    }

    printf("%d\n", data1+data2);
    printf("%u\n", data1+data2);
}
int main(int argc, char *argv[])
```

```
edu@edu: ~/work/c/day02
edu@edu:~/work/c/day02$ gcc 01_test.c
edu@edu:~/work/c/day02$ ./a.out
>0
-4
4294967292
edu@edu:~/work/c/day02$
```

## 2、int和double参加运算 会将int转成从double类型

```c
void test02()
{
    int data1=0;
    double data2=0.0;
    printf("%lu\n", sizeof(data1+data2));//8B
}
```

```
edu@edu: ~/work/c/day02
edu@edu:~/work/c/day02$ gcc 01_test.c
edu@edu:~/work/c/day02$ ./a.out
8
edu@edu:~/work/c/day02$
```

## 3、char和short类型 只要参加运算 都会将自己转换成int类型

```c
void test03()
{
    char ch='a';
    short sh=0;
    printf("%lu\n", sizeof(ch+ch));//4
    printf("%lu\n", sizeof(sh+sh));//4
    printf("%lu\n", sizeof(ch+sh));//4

}
```

```
edu@edu: ~/work/c/day02
edu@edu:~/work/c/day02$ gcc 01_test.c
edu@edu:~/work/c/day02$ ./a.out
4
4
4
edu@edu:~/work/c/day02$
```

强制类型转换：(类型说明符) (表达式)

```
(int)p+1 对p强转成int类型 然后再+1
(int)(p+1)对p+1强转成int类型
```

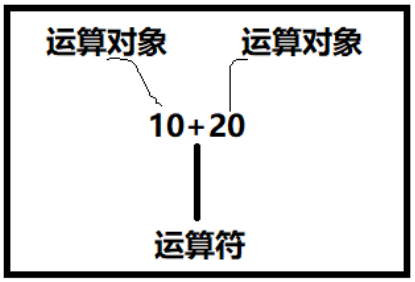
不管是自动类型转换 还是强制类型转换 都是临时。

```c
void test04()
{
    float f = 3.14f;
    int x = 0;
    x = (int)f;
    printf("x = %d, f=%f\n", x, f);
}
```

```
edu@edu: ~/work/c/day02
edu@edu:~/work/c/day02$ gcc 01_test.c
edu@edu:~/work/c/day02$ ./a.out
x = 3, f=3.140000
edu@edu:~/work/c/day02$
```

# 知识点2【运算符】（了解）

## 1、运算符的概述




算术表达式

如果运算符 需要一个运算对象 就叫单目运算符

如果运算符 需要两个运算对象 就叫双目运算符

如果运算符 需要三个运算对象 就叫三目运算符

如果运算符 需要多个运算对象 就叫多目运算符

## 2、算数运算符

```
+ - * / % += -= *= /= %=
```

### 1、如果/的所有运算对象都是整数 /的功能就是取整

```
5/3 == 1 5/2 == 2
```

### 2、如果/有一个运算对象是实型 /的功能就是除法运算

```c
void test05()
{
    printf("%lf\n", 5/2.0);
}
```

```
edu@edu:~/work/c/day02$ gcc 01_test.c
edu@edu:~/work/c/day02$ ./a.out
2.500000
edu@edu:~/work/c/day02$
```

### 3、%取余运算符

```c
5%2 == 1 3%5==3
n%3 == 0,1,2
data%n == 0,1,2,.....n-1
```

**案例1：键盘输入一个数 判断概述能否被3整除**

```c
#include <stdio.h>


int main(int argc, char *argv[])
{
    int num = 0;

    printf("请输入一个整数： ");
    scanf("%d", &num);

    if(num % 3 == 0)
    {
        printf("%d可以被3整除\n", num);
    }
    else
    {
        printf("%d不能被3整除\n", num);
    }

    return 0;
}
```

**案例2：如果rand()函数产生一个随机数>0, 请使用rand()产生60~100的随机数**

```c
rand()%41+60;
```

**案例2：如果rand()函数产生一个随机数>0, 请使用rand()产生'a'~'z'的随机字母**

```c
rand()%26 +'a';
```

### 4、复合运算 += -= *= /= %=

```c
a+=b;//a=a+b;
a*=b;//a=a*b;
a/=b;//a=a/b;
a%=b;//a=a%b;
```

注意：一定要将=号的右边看成一个整体

```c
int a = 3;
a*=4+5;//a = a*(4+5)
```

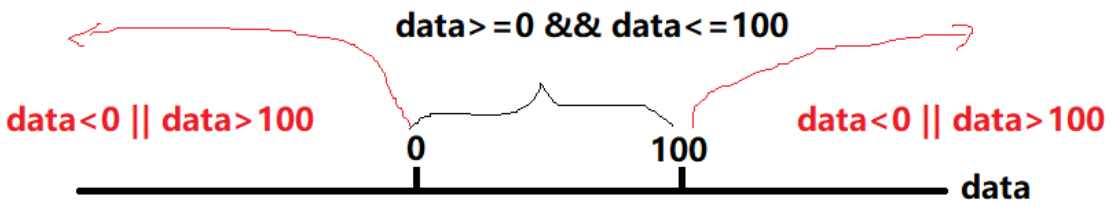
```
3 a的值是：27
```

**案例1：以下表达式执行完 a的值是__0__**

```
1 int a = 12;
2 a+= a-= a*=a;
```

## 3、关系运算符

```
1 （>、<、==、>=、<=、!= ）
```




## 4、逻辑运算符 &&  ||  ！

**&&逻辑与：**

A && B  A为真 且 B为真 整个表达式结果才为真。

A && B  A或B只要有一个为假  整个表达式结果才为假。

注意：逻辑与&&的短路特性：

如果A为假  整个表达式为假， 那么B的真假决定不了整个表达式的结果，所以不会再判断B的真假，就叫“短路特性”

**||逻辑或：**

A || B  A和B只要有一个为真 整个表达式结果为真。

A || B  A和B同时假 整个表达式结果为假。

注意逻辑或||的短路特性：

如果A为真  整个表达式为真， 那么B的真假决定不了整个表达式的结果，所以不会再判断B的真假，就叫“短路特性”

**！逻辑非：**

!真 == 假　　！假==真

在c语言中 除了0为假 其他都为真

```
1 ! 10 == 0 !-10 == 0 !0 == 1
```

## 5、产生随机数

```c
#include<stdlib.h>//srand  rand
#include <time.h>//time
void test06()
{
    //设置随机数种子 设置随机数的基础值
    //time(NULL)获取当前时间（1970~现在的所有秒数）
    srand(time(NULL));

    int num = 0;
    num = rand();
    printf("num = %d\n", num%41+60);
}
int main(int argc, char *argv[])
{
    test06();
    return 0;
}
```

```
edu@edu: ~/work/c/day02
edu@edu:~/work/c/day02$ gcc 01_test.c
edu@edu:~/work/c/day02$ ./a.out
num = 75
edu@edu:~/work/c/day02$ ./a.out
num = 62
edu@edu:~/work/c/day02$ ./a.out
num = 98
edu@edu:~/work/c/day02$ ./a.out
num = 76
edu@edu:~/work/c/day02$ ./a.out
num = 76
edu@edu:~/work/c/day02$ ./a.out
```

# 6、位运算（二进制位运算）

## 1、& 按位与

语法：全1为1  有0为0

特点：和1相与保持不变  和0相与为0

场景：  将指定位 清0

1100 0011

& 1111 0000

---

1100 0000

**案例1：data为1字节  将data的第3,4为清0 其他位保持不变。**

```c
unsigned char data;
data = data & 1110 0111;
data = data & 0xe7;//ok
data &= 0xe7;//ok
```

## 2、| 按位或

语法：有1为1 全0为0

特点：和1或置1，  和0或 保持不变

场景：将指定位 置1

1100 0011

| 1111 0000

---

1111 0011

**案例1：data为1字节  将data的第5,6为置1 其他位保持不变。**

```c
data = data | 0110 0000
data = data | 0x60;//ok
data |= 0x60;//ok
```

## 3、~按位取反

语法：0变1  1变0

~1100 0011 == 0011 1100

**4、^ 按位异或**

语法：相同为0 不同为1

特点：和1异或取反 和0异或保持不变

场景：将指定位 发生翻转

1100 0011

^ 1111 0000

---

0011 0011

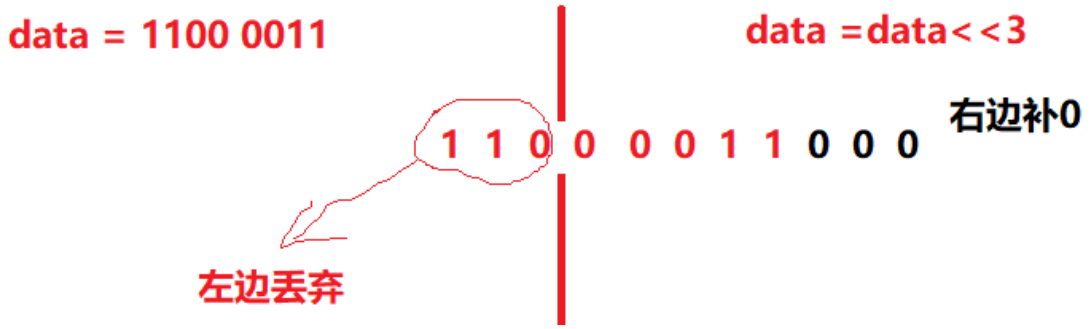
^ 1111 0000

---

1100 0011

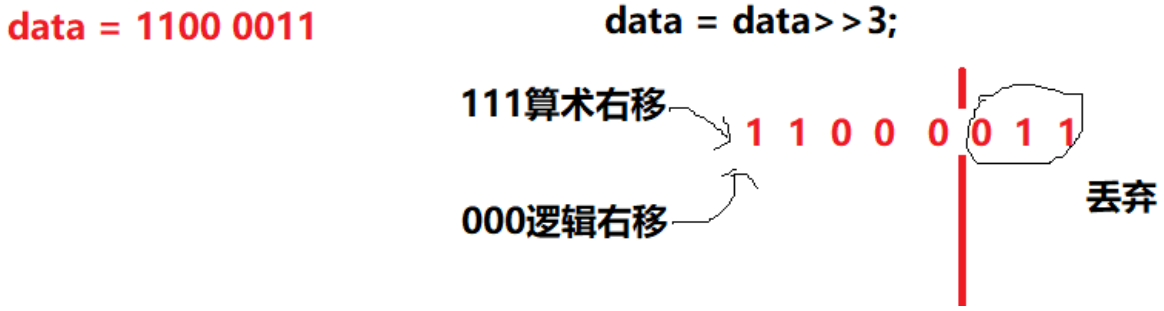
## 5、左移<<:左边丢弃 右边补0

移动的位数 不要超过 自身位的宽度




```
如果：data=0000 0001 data=data<<0; data=0000 0001 == 1==data*2^0
如果：data=0000 0001 data=data<<1; data=0000 0010 == 2==data*2^1
如果：data=0000 0001 data=data<<2; data=0000 0100 == 4==data*2^2
如果：data=0000 0001 data=data<<3; data=0000 1000 == 8==data*2^3
......
如果：data=0000 0001 data=data<<6; data=0100 0000 == 64==data*2^6
```

## 5、右移>>:右边丢弃 左边补0（补1）




算术右移、逻辑右移 都是编译器决定，用户无法确定。

**无符号数：右边丢弃 左边补0**

**有符号数：**

正数：**右边丢弃 左边补0**

负数：**右边丢弃 左边补0（逻辑右移）**

负数：**右边丢弃 左边补1（算术右移）**

编写代码测试 编译器为那种右移：

```
void test07()
{
    char data = -10;//补码：1111 0110
    printf("移动之前内存数据:%#x\n", data);
    data = data>>4;
    if((data&0xff) == 0xff)
    {
        printf("算术右移\n");
    }
    else if((data&0xff) == 0x0f)
    {
        printf("逻辑右移\n");
    }
}
```

```
edu@edu: ~/work/c/day02
edu@edu:~/work/c/day02$ gcc 01_test.c
edu@edu:~/work/c/day02$ ./a.out
移动之前内存数据:0xfffffff6
算术右移
edu@edu:~/work/c/day02$
```

假设data为无符号数

```
如果：data=1000 0000 data=data>>0; data=1000 0000 == 128==data除以2^0
如果：data=1000 0000 data=data>>1; data=0100 0000 == 64==data除以2^1
如果：data=1000 0000 data=data>>2; data=0010 0000 == 32==data除以2^2
如果：data=1000 0000 data=data>>3; data=0001 0000 == 16==data除以2^3
......
如果：data=1000 0000 data=data>>6; data=0000 0010 == 2==data除以2^6
```

## 高级应用：

### 案例1：data为1字节 将data的第3,4为清0 其他位保持不变。

```
data= data & 1110 0111
 1110 0111 == ~(0001 1000) == ~(0001 0000 | 0000 1000)
 ==~(0000 0001<<4 | 0000 0001<<3)
 == ~(0x01<<4 | 0x01<<3);
data &= ~(0x01<<4 | 0x01<<3);//推荐
```

### 案例2：data为1字节 将data的第5,6为置1 其他位保持不变。

```
data = data | 0110 0000
 0110 0000==0100 0000 | 0010 0000==0000 0001<<6 | 0000 0001<<5
 == 0x01<<6 | 0x01<<5
data |=(0x01<<6 | 0x01<<5);//推荐
```

### 案例3：data为1字节 将data的第3,4位清0, 5,6置1 其他位保持不变

```
data = data & ~(0x01<<3|0x01<<4) | (0x01<<5|0x01<<6);
```

## 7、三目运算符

？ ： ---> 表达式1 ？ 值1:值2

如果表达式1为真 整个表达式的值为值1

如果表达式1为假 整个表达式的值为值2

```c
void test08()
{
    int data1=0, data2=0;

    printf("请输入两个int数据:");
    scanf("%d %d", &data1, &data2);

    printf("最大值为:%d\n", data1>data2?data1:data2);
}
int main(int argc, char *argv[])
```

```
edu@edu: ~/work/c/day02
edu@edu:~/work/c/day02$ gcc 01_test.c
edu@edu:~/work/c/day02$ ./a.out
请输入两个int数据:100 200
最大值为:200
edu@edu:~/work/c/day02$
```

## 8、逗号运算符,

```c
void test09()
{
    int x=0,y=0;

    x = 1,2,3;
    y = (1,2,3);

    printf("x = %d, y=%d\n", x, y);
}
int main(int argc, char *argv[])
{
    test09();
    return 0;
```

```
edu@edu: ~/work/c/day02
edu@edu:~/work/c/day02$ gcc 01_test.c
edu@edu:~/work/c/day02$ ./a.out
x = 1, y=3
edu@edu:~/work/c/day02$
```

# 知识点3【优先级】（了解）

优先级数值越小 越先执行

同一优先级 看结合方向

| 优先级别 | 运算符 | 运算形式 | 结合方向 | 名称或含义 |
|---|---|---|---|---|
| 1 | () | (e) | 自左至右 | 圆括号 |
| | [] | a[e] | | 数组下标 |
| | . | x.y | | 成员运算符 |
| | -> | p->x | | 用指针访问成员的指向运算符 |
| 2 | - + | -e | 自右至左 | 负号和正号 |
| | ++ -- | ++x 或 x++ | | 自增运算和自减运算 |
| | ! | ! e | | 逻辑非 |
| | ~ | ~e | | 按位取反 |
| | (t) | (t)e | | 类型转换 |
| | * | *p | | 指针运算，由地址求内容 |
| | & | &x | | 求变量的地址 |
| | sizeof | sizeof(t) | | 求某类型变量的长度 |

| | | | | |
|---|---|---|---|---|
| 3 | * / % | e1 * e2 | 自左至右 | 乘、除和求余 |
| 4 | + - | e1+e2 | 自左至右 | 加和减 |
| 5 | << >> | e1<<e2 | 自左至右 | 左移和右移 |
| 6 | < <= > >= | e1<e2 | 自左至右 | 关系运算(比较) |
| 7 | == != | e1==e2 | 自左至右 | 等于和不等于比较 |
| 8 | & | e1&e2 | 自左至右 | 按位与 |
| 9 | ^ | e1^e2 | 自左至右 | 按位异或 |
| 10 | \| | e1\|e2 | 自左至右 | 按位或 |
| 11 | && | e1 && e2 | 自左至右 | 逻辑与(并且) |
| 12 | \|\| | e1\|\|e2 | 自左至右 | 逻辑或(或者) |
| 13 | ? : | e1? e2: e3 | 自右至左 | 条件运算 |
| 14 | =<br>+= -= *=<br>/= %= >>=<br><<= &= ^=<br>\|= | x=e<br><br>x+=e | 自右至左 | 赋值运算<br><br>复合赋值运算 |
| 15 | , | e1, e2 | 自左至右 | 顺序求值运算 |

# 知识点4【自增自减运算符 ++ --】（了解）

## 1、不管是i++ i-- ++i --i 如果是独立的一条语句 那么i++和++i, i--和--i没区别

```
int i = 3;
//i++;
++i;
printf("i=%d\n", i);//4
```

## 2、i++ i-- 复合运算 ++ --在运算对象的右边（先使用 后加减）

```
int i=3;
int j = 0;
j = i++;//j=i; i++;
printf("i=%d, j=%d\n", i, j);//i=4 j=3
```

## 3、++i --i 复合运算 ++ --在运算对象的左边（先加减 后使用）

```
int i=3;
int j = 0;
j = ++i;//i++; j=i;
printf("i=%d, j=%d\n", i, j);//i=4 j=4
```

# 知识点5【if控制语句】（重要）

顺序结构、选择结构、循环结构

c语言默认都是顺序执行

## 1、如果只在乎项目 某一个结果 需要使用if语句

```
if(条件1)
{
 语句1;
}
条件1为真 执行语句1 如果条件1为假跳过if语句
```

**案例1：键盘输入一个数 判断它能被3整除**

```
void test01()
{
    int num = 0;

    printf("请输入一个int数据:");
    scanf("%d", &num);

    if(num%3 == 0)
    {
        printf("%d能被3整除\n", num);
    }
}
```

```
edu@edu: ~/work/c/day02
edu@edu:~/work/c/day02$ gcc 02_code.c
edu@edu:~/work/c/day02$ ./a.out
请输入一个int数据:9
9能被3整除
edu@edu:~/work/c/day02$ ./a.out
请输入一个int数据:5
edu@edu:~/work/c/day02$
```

## 2、如果项目有两个结果 但是不会同时出现 需要使用if...else语句

```
if(条件1)
{
 语句1;
}
else
{
 语句2;
}
条件1为真 执行语句1 否则执行语句2
```

**案例1：键盘输入一个数 判断它能否被3整除**

```c
#include <stdio.h>
void test01()
{
    int num = 0;

    printf("请输入一个int数据:");
    scanf("%d", &num);

    if(num%3 == 0)
    {
        printf("%d能被3整除\n", num);
    }
    else
    {
        printf("%d不能被3整除\n", num);

    }
}
```

```
edu@edu: ~/work/c/day02
edu@edu:~/work/c/day02$ gcc 02_code.c
edu@edu:~/work/c/day02$ ./a.out
请输入一个int数据:6
6能被3整除
edu@edu:~/work/c/day02$ ./a.out
请输入一个int数据:5
5不能被3整除
edu@edu:~/work/c/day02$
```

## 3、如果项目有多个结果 但是不会同时出现 需要使用if...else if...else if.....else....语句

```c
if(条件1)
{
 语句1;
}
else if(条件2)
{
 语句2;
}
else
{
 语句3;
}
如果条件1 为真 执行语句1， 剩余的条件不会判断
只有条件1 为假 才会判断 条件2，如果条件2为真执行语句2 剩余的条件不会判断
当上面所有条件都不满足 才会执行else，else可以省略
```

**案例1：键盘输入一个数 判断它对3的余数**

```c
#include <stdio.h>
void test01()
{
    int num = 0;

    printf("请输入一个int数据:");
    scanf("%d", &num);

    if(num%3 == 0)
    {
        printf("%d对3余数为0\n", num);
    }
    else if(num%3 ==1)
    {
        printf("%d对3的余数为1\n", num);

    }
    else if(num%3 == 2)
    {
        printf("%d对3的余数为2\n", num);

    }
}
```

```
edu@edu: ~/work/c/day02
edu@edu:~/work/c/day02$ gcc 02_code.c
edu@edu:~/work/c/day02$ ./a.out
请输入一个int数据:5
5对3的余数为2
edu@edu:~/work/c/day02$ ./a.out
请输入一个int数据:4
4对3的余数为1
edu@edu:~/work/c/day02$ ./a.out
请输入一个int数据:6
6对3余数为0
edu@edu:~/work/c/day02$
```

### 3、如果项目有多个结果 不确定同时出现 需要使用if独立判断每个结果

```c
if(条件1)
{
 语句1;
}
if(条件2)
{
 语句2;
}
if(条件3)
{
 语句3;
}
```

# 知识点6【switch选择语句】（重要）

表达式1不能是实型、字符串

case后面的值 必须是 常量表达式 （不能有变量）

```c
switch(表达式1)
{
 case 值1:
 语句1;
 break;
 case 值2:
 语句2;
 break;
 default://可省略
 语句3;
```

```
 break;
}
将表达式1的结果 分别与case后面的值 进行比较 如果相等 就从相应的case下方执行
直到遇到break跳出switch。
建议 每个case都要有一个break
```

## 案例1：键盘输入1~7的数值 判断是星期几（周日为7）

```c
void test02()
{
    int date = 0;

    printf("请输入1~7的数:");
    scanf("%d", &date);

    switch(date)
    {
        case 1:
            printf("星期一\n");
            break;
        case 2:
            printf("星期二\n");
            break;
        default:
            printf("数值无效\n");
            break;
    }
}
```

```
edu@edu: ~/work/c/day02
edu@edu:~/work/c/day02$ gcc 02_code.c
edu@edu:~/work/c/day02$ ./a.out
请输入1~7的数:2
星期二
edu@edu:~/work/c/day02$
```

## 案例2：特殊情况 可以省略break

```c
#include <termios.h>
#include <stdio.h>
#include <unistd.h>
char mygetch( )
{
 struct termios oldt, newt;
 char ch;
 tcgetattr( STDIN_FILENO, &oldt );
 newt = oldt;
 newt.c_lflag &= ~( ICANON | ECHO );
 tcsetattr( STDIN_FILENO, TCSANOW, &newt );
 ch = getchar();
 tcsetattr( STDIN_FILENO, TCSANOW, &oldt );
 return ch;
}
void test02()
{
 char ch='\0';
 ch = mygetch();
 switch(ch)
 {
```

```
 case 'q':
 case 'Q':
 printf("天音波\n");
 break;
 case 'w':
 case 'W':
 printf("铁布衫\n");
 break;
 }
}
int main(int argc, char *argv[])
{
 test02();
 return 0;
}
```

```
edu@edu:~/work/c/day02$ gcc 02_code.c
edu@edu:~/work/c/day02$ ./a.out
天音波
edu@edu:~/work/c/day02$ ./a.out
天音波
edu@edu:~/work/c/day02$
```

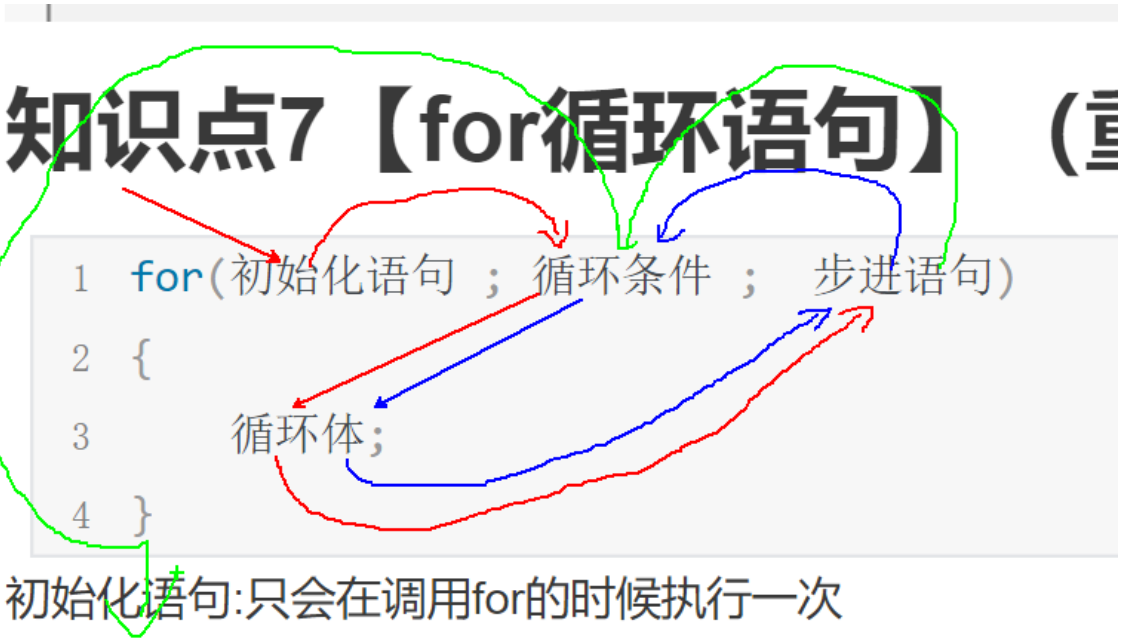
# 知识点7【for循环语句】（重要）

```
for(初始化语句 ; 循环条件 ; 步进语句)
{
 循环体;
}
```

初始化语句:只会在调用for的时候执行一次

循环条件为真 执行循环体，否则跳出循环

步进语句：执行完循环体，自动执行步进语句 然后 才判断循环条件




**案例1：求1~100的和**

```c
void test03()
{
    int i=0;
    int sum=0;

    for(i=1; i<=100; i++)
    {
        sum = sum +i;
    }
    printf("sum = %d\n", sum);
}
int main(int argc, char *argv[])
{
    test03();
    return 0;
}
```

```
edu@edu: ~/work/c/day02
edu@edu:~/work/c/day02$ gcc 02_code.c
edu@edu:~/work/c/day02$ ./a.out
sum = 5050
edu@edu:~/work/c/day02$
```

```c
void test03()
{
    int i=0;
    int sum=0;

    for(i=1; i<=100; i+=2)
    {
        sum = sum +i;
    }
    printf("sum = %d\n", sum);
}
```

```
edu@edu: ~/work/c/day02
edu@edu:~/work/c/day02$ gcc 02_code.c
edu@edu:~/work/c/day02$ ./a.out
sum = 2500
edu@edu:~/work/c/day02$
```

```c
void test03()
{
    int i=0;
    int sum=0;

    for(i=99; i>0; i=i-2)
    {
        sum = sum +i;
    }
    printf("sum = %d\n", sum);
}
```

```
edu@edu: ~/work/c/day02
edu@edu:~/work/c/day02$ gcc 02_code.c
edu@edu:~/work/c/day02$ ./a.out
sum = 2500
edu@edu:~/work/c/day02$
```

## 案例2：循环嵌套循环

```c
for(i=0;i<10;i++)
{
 for(j=0;j<10;j++)
 {
 语句1;
 }
}
```

先写内层循环 然后再写外层循环

**九九乘法口诀表**

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 1×1=1 | | | | | | | | |
| 1×2=2 | 2×2=4 | | | | | | | |
| 1×3=3 | 2×3=6 | 3×3=9 | | | | | | |
| 1×4=4 | 2×4=8 | 3×4=12 | 4×4=16 | | | | | |
| 1×5=5 | 2×5=10 | 3×5=15 | 4×5=20 | 5×5=25 | | | | |
| 1×6=6 | 2×6=12 | 3×6=18 | 4×6=24 | 5×6=30 | 6×6=36 | | | |
| 1×7=7 | 2×7=14 | 3×7=21 | 4×7=28 | 5×7=35 | 6×7=42 | 7×7=49 | | |
| 1×8=8 | 2×8=16 | 3×8=24 | 4×8=32 | 5×8=40 | 6×8=48 | 7×8=56 | 8×8=64 | |
| 1×9=9 | 2×9=18 | 3×9=27 | 4×9=36 | 5×9=45 | 6×9=54 | 7×9=63 | 8×9=72 | 9×9=81 |

先写内层循环 然后再写外层循环

```
edu@edu: ~/work/c/day02
edu@edu:~/work/c/day02$ ./a.out
1*1=1
1*2=2 2*2=4
1*3=3 2*3=6 3*3=9
1*4=4 2*4=8 3*4=12 4*4=16
1*5=5 2*5=10 3*5=15 4*5=20 5*5=25
1*6=6 2*6=12 3*6=18 4*6=24 5*6=30 6*6=36
1*7=7 2*7=14 3*7=21 4*7=28 5*7=35 6*7=42 7*7=49
1*8=8 2*8=16 3*8=24 4*8=32 5*8=40 6*8=48 7*8=56 8*8=64
1*9=9 2*9=18 3*9=27 4*9=36 5*9=45 6*9=54 7*9=63 8*9=72 9*9=81
edu@edu:~/work/c/day02$
```

```c
}

void test04()
{
    int i=6;
    for(i=1;i<=9; i++)
    {
        int j = 0;
        //内层循环
        for(j=1; j<=i;j++)
        {
            printf("%d*%d=%d ",j,i,j*i );
        }
        printf("\n");

    }
}
int main(int argc, char *argv[])
{
    test04();
```

### 案例3：输出

```
*
**
***
****
*****
******
```

contiune:结束本次循环 直接进入下一次循环

```
int i=0,sum=0;
for(i=0; i<=100; i++)
{
 if(i==50)
 continue;
 sum += i;
}
sum== 5000
```

break跳出循环：

```
int i=0,sum=0;
for(i=0; i<=100; i++)
{
 if(i==50)
 break;
 sum += i;
}
sum只加了1~49
```

# 知识点8【while循环语句】（重要）

```
//外部实现 初始化
while(循环条件)
{
 循环语句;
 //内部实现 步进语句
}
```

```
int i=0, sum = 0;//初始化语句

while(i<=100)//循环条件
{
 sum += i;//循环语句
```

```
6
7   i++;//步进语句
8  }
```

continue break一样用于while

```
1  do
2  {
3   //循环体;
4  }while(循环条件);
5  先执行一次循环体 再判断循环条件 来决定 是否下一次循环
```

如果知道循环次数 建议使用for

如果不知道循环次数 但是知道退出条件 建议使用while

# 知识点9【goto 语句 少用】（重要）

```
void test05()
{
    printf("--------001----------\n");
    printf("--------002----------\n");
    goto here;
    printf("--------003----------\n");
    printf("--------004----------\n");
here:
    printf("--------005----------\n");
    printf("--------006----------\n");
}
int main(int argc, char *argv[])
{
```

```
edu@edu: ~/work/c/day02
edu@edu:~/work/c/day02$ gcc 02_code.c
edu@edu:~/work/c/day02$ ./a.out
--------001----------
--------002----------
--------005----------
--------006----------
edu@edu:~/work/c/day02$
```