C**库函数总结**

## C 库函数主要指那些由美国国家标准协会（ANSI）或国际标准化组织(ISO)发布的标准中规定的库函数，按照标准 C 的要求来进⾏行行 C 语⾔言编程是很重要的，因为这样你的代码才有可能跨平台使⽤用。

ctype.h

ctype.h 这个头⽂文件主要定义了了⼀一批 C 语⾔言字符分类函数，所有的函数都只有⼀一个参数，且参数和返回值均为 int 类型。下⾯面是简单的函数介绍：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **函数名** | **说明** | **函数名** | **说明** | **函数名** | **说明** |
| isalpha | 是否为字⺟母 | isdigit | 是否为数字 | isalnum | 是否为数字或字⺟母 |
| iscntrl | 是否为控制字符 | isgraph | 是否为图形⽂文字 | isupper | 是否为⼤大写字⺟母 |
| islower | 是否为⼩小写字⺟母 | tolower | 转换为⼩小写字⺟母 | toupper | 转换为⼤大写字⺟母 |
| isprintf | 是否为可打印字符 | ispunct | 是否为标点符号 | isspace | 是否为空⽩白 |
| isxdigit | 是否为⼗十六进制的数字 |  |  |  |  |

stdio.h

stdio.h 这个头⽂文件应该是⼤大多数⼈人接触 C 语⾔言的时候第⼀一个认识的 C 库函数。这个函数最常⽤用，但是也很复杂。它定义了了三种类型，⼀一些宏和很多的输⼊入输出函数。

1. 定义的三种类型：
   1. size\_t 是由 sizeof 关键字产⽣生的⽆无符号整类型。

## FILE 是⼀一个结构体类型，记录了了控制流需要的所有信息，包括它的⽂文件定位符、指向相关缓冲的指针、记录是否发⽣生了了读/写错误的错误提示符和记录⽂文件是 否结束的⽂文件结束符。

* 1. fpos\_t 包含可以唯⼀一指定⽂文件中的每⼀一个位置所需的所有信息。

1. 定义了了⼀一些常量量：
   1. NULL 空 值
   2. \_IOFBF 表示完全缓冲 \_IOLBF 表示线缓冲 \_IONBF 表示⽆无缓存
   3. BUFSIZ 是 setbuf 函数所使⽤用的缓冲区的⼤大⼩小

## EOF 是负整数,该表达式由⼏几个函数返回来说明⽂文件的结束，即⼀一个流输⼊入结束了了(END OF FILE)

* 1. FOPEN\_MAX (20)同时打开的⽂文件的最⼤大数量量
  2. FILENAME\_MAX ⽂文件名的最⼤大⻓长度
  3. L\_tmpnam 整数，最⼤大⻓长度的临时⽂文件名
  4. SEEK\_CUR 取得⽬目前⽂文件位置 SEEK\_END 将读写位置移到⽂文件尾时 SEEK\_SET 将读写位置移到⽂文件开头
  5. TMP\_MAX 表示 tmpnam 函数可以⽣生成的单独⽂文件名的最⼤大数⽬目
  6. `stderr 标准错误流，默认为屏幕, 可输出到⽂文件 stdin 标准输⼊入流，默认为键盘 stdout 标准输出流，默认为屏幕

## ⼀一些相关的函数

* 1. ⽂文件操作函数 （4个）

int remove(const char \*filename); 删除⽂文件

int rename(const char \*old, const char \*new); 重命名⽂文件

FILE \*tmpfile(void); 创建⼀一个临时的⼆二进制⽂文件，并通过模式“wb+”打开。char \*tmpname（char \*s）; ⽣生成⼀一个字符串串，这个字符串串是⼀一个有效的⽂文件名。

## ⽂文件访问函数 (6个)

FILE\* fopen(const char \*filename, const char \*mode); 打开名字为

filename 指向的⽂文件，并且把这个⽂文件和⼀一个流相关联。

FILE\* freopen(const char \*filename, const char \*mode, FILE

\*stream); 打开名字为 filename 指向的⽂文件，并且把它和 stream 指向的流关联在⼀一起。

int fclose(FILE \*stream); 使 stream 指向的流被清空，并且和流相关联的

## ⽂文件被关闭。

int fflush(FILE \*stream); 对 stream 指向的流执⾏行行清空⾏行行为。

void setbuf(FILE \*stream, char \*buf); 除了了没有返回值，等价于函数

setvbuf 。

void setvbuf(FILE \*stream, char \*buf, int mode, size\_t size); 设定

stream 缓冲的⽅方式。

## 格式化的输⼊入输出函数 （9个）

int fprintf(FILE \*stream, const char \*format, ...); int printf(const char \*format, ...);

int sprintf(char \*s, const char \*format, ...);

## 返回传送的字符的数⽬目。

int fscanf(FILE \*stream, const char \*format, ...); int scanf(const char \*format, ...);

int sscanf(char \*s, const char \*format, ...);

## 如果在任何转换之前发⽣生了了输⼊入失败，返回 EOF ；否则返回赋值的输⼊入项的数

⽬目。

int vfprintf(FILE \*stream, const char \*format, va\_list arg); int vprintf(const char \*format, va\_list arg);

int vsprintf(char \*s, const char \*format, va\_list arg);

等价于对应的 printf 函数，不不过可变参数表⽤用 arg 代替。

## 字符输⼊入/输出函数 （11个）

int fgetc(FILE \*stream); 从 stream 指向的输⼊入流中读取下⼀一个字符。

int getc(FILE \*stream); 等价于函数 fgetc 。

int getchar(void); 等价于⽤用参数 stdin 调⽤用函数 getc 。

int fputc(int c, FILE \*stream); 把字符写到指向的输出流中指定的位置处。

int putc(int c, FILE \*stream); 等价于 fputc。

int putchar(int c); 等价于把stdout作为第⼆二个参数调⽤用的 putc。

char \*fgets(char \*s, int n, FILE \*stream); 从 stream 指向的流中读取字符。

int fputs(const char \*s, FILE \*stream); 把 s 指向的串串写⼊入 stream 指向的流中。

char \*gets(char \*s); 从 stdin 指向的输⼊入流中读取若⼲干个字符，并将其保留留到 s 指向的数组中，直到遇到⽂文件结束符或者读取⼀一个换⾏行行符。

int puts(const char \*s); 把 s 指向的串串写到 stdout 指向的流中，并且在输出最后添加⼀一个换⾏行行符。

int ungetc(int c, FILE \*stream); 把 c 指定的字符（转换为 unsigned

char 类型）退回到 stream 指向的输⼊入流中。

## 直接输⼊入/输出函数 （2个）

size\_t fread(void \*ptr, size\_t size, size\_t nmemb, FILE \*stream);

从 stream 指向的流中读取最多 nmemb 个元素到 ptr 指向的数组中 。

size\_t fwrite(const void \*ptr, size\_t size, size\_t nmemb, FILE

\*stream);

从 ptr 指向的数组中读取最多 nmemb 个元素并将其写到 stream 指向的流中。

## ⽂文件定位函数 (5个)

int fgetpos(FILE \*stream, fpos\_t \*pos);

把 stream 指向的流的⽂文件定位符的当前值存储到 pos 指向的对象中。

int fsetpos(FILE \*stream, const fpos\_t \*pos);

根据 pos 指向的对象的值来设置 stream 指向的流的⽂文件定位符，

long int ftell(FILE \*stream);

获得 stream 指向的流的⽂文件定位符的当前值。

int fseek(FILE \*stream, long int offset, int whence);

为 stream 指向的流设置⽂文件定位符。

void rewind(FILE \*stream);

## 把 stream 指向的流的⽂文件定位符设置在⽂文件的开始位置，等价于

(void)fseek(stream, 0L, SEEK\_SET); ，只不不过流的错误指示符也被清零。

## 错误处理理函数 (4个)

void clearerr(FILE \*stream);

清空 stream 指向的流的⽂文件结束符和错误指示符。

int feof(FILE \*stream);

测试 stream 指向的流的⽂文件结束符。当且仅当 stream 流设置了了⽂文件结束符时函数返回⼀一个⾮非0值。

int ferror(FILE \*stream);

测试 stream 指向的流的错误指示符。当且仅当 stream 流设置了了错误指示符时函数返回⼀一个⾮非0值。

void perror(const char \*s);

## 把整数表达式 errno 中的错误编号转换为⼀一条错误消息。

stdlib.h

是标准⼯工具库。包含了了C语⾔言的中最常⽤用的系统函数

1. 定义了了五个宏
   1. NULL 空
   2. EXIT\_FAILURE 失败状态码
   3. EXIT\_SUCCESS 成功状态码
   4. RAND\_MAX rand 的最⼤大返回值
   5. MB\_CUR\_MAX 多字节字符中的最⼤大字节数

## 定义了了四个类型：

* 1. size\_t 是由 sizeof 关键字产⽣生的⽆无符号整类型。
  2. wchar\_t 是⼀一个整型，标识⼀一个宽字节字符，例例如 L’x’ 的类型就是

wchar\_t 。

* 1. div\_t 是结构体类型 作为 div 函数的返回类型
  2. ldiv\_t 是⼀一个结构类型，是函数 ldiv 的返回值类型。

## 定义了了六类函数：

* 1. 字符串串函数

double atof(const char \*nptr); 将字符串串转换成浮点型数

int atoi(const char \*nptr); 将字符串串转换成整型数

long int atol(const char \*nptr); 将字符串串转换成⻓长整型数

double strtod(const char \*nptr, char \*\*endptr); 将字符串串转换成浮点数long int strtol(const char \*nptr, char \*\*endptr, int base); 将字符串串转换成⻓长整型数

unsigned long int strtoul(const char \*nptr, char \*\*endptr, int

base); 将字符串串转换成⽆无符号⻓长整型数

## 内存控制函数

void \*calloc(size\_t nmemb, size\_t size); 配置内存空间

void free(void \*ptr); 释放原先配置的内存

void \*malloc(size\_t size); 配置内存空间

void \*realloc(void \*ptr, size\_t size); 重新分配主存

## 环境函数

int atexit(void (\*func)(void));

## 注册func指向的函数，该函数在程序正常终⽌止的时候被调⽤用。

void exit(int status);

## 使程序正常终⽌止。

void abort(void);

## 使程序异常终⽌止，除⾮非捕获了了信号SIGABRT且信号处理理程序没有返回。

char \*getenv(const char \*name);

## 搜索宿主环境提供的环境表，来查找⼀一个能和name指向的串串匹配的串串。

int system(const char \*string);

## 把sring指向的串串传递给宿主环境，然后命令处理理程序按照实现定义的⽅方式执⾏行行 它。

* 1. 搜索和排序函数

void bsearch(const void \*key, const void \*base, size\_t nmemb, size\_t size, int (\*compare)(const void , const void \*));

搜索⼀一个拥有 nmemb 个元素的数组，来查找与 key 指向的对象匹配的元素，

void qsort(void base, size\_t nmemb, size\_t size, int (\*compare) (const void , const void \*));

根据 compare 所指向的⽐比较函数将数组内容排列列成升序。

## 数学函数

int abs(int j); 计算整型数的绝对值

long int labs(long int j); 将两个整数相除, 返回商和余数

div\_t div(int number, int denom); 取⻓长整型绝对值

ldiv\_t ldiv(long int number, long int denom); 两个⻓长整型数相除, 返回商和余数

int rand(void); 随机数发⽣生器器

void srand(unsigned int seed); 设置随机数种⼦子

## 多字节函数

int mblen(const char \*s, size\_t n); 根据locale的设置确定字符的字节数int mbtowc(wchar\_t \*pwc, const char \*s, size\_t n); 把多字节字符转换为宽字符

int wctomb(char \*s, wchar\_t wchar); 把宽字符转换为多字节字符size\_t mbstowcs(wchar\_t \*pwcs, const char \*s, size\_t n); 把多字节字符串串转换为宽字符串串

size\_t wcstombs(char \*s, const wchar\_t \*pwcs, size\_t n); 把宽字符串串转换为多字节字符串串

# string

## 包含了了C语⾔言的最常⽤用的字符串串操作函数

1. 定义了了⼀一个宏： NULL 空
2. 定义了了⼀一个变量量： size\_t

## 定义了了五类函数：

* 1. 名字以mem开头的函数对任意的字符序列列进⾏行行操作。其中⼀一个参数指向字符串串的 起始位置，另⼀一个参数对元素的个数尽⾏行行计数。

void \*memmove(void \*s1, const void \*s2, size\_t n); void \*memcpy(void \*s1, const void \*s2, size\_t n);

## memmove和memcpy均从s2指向的对象中复制n个字符到s1指向的对象中，返回

s1的值。不不同之处在于，当复制发⽣生在两个重叠的对象中，memcpy对这种⾏行行为未定义，memmove则可以正确执⾏行行。

int memcmp(const void \*s1, const void \*s2, size\_t n);

## ⽐比较s1和s2指向的对象中前n个字符。

void \*memchr(const void \*s, int c, size\_t n);

## 返回指向定位的字符的指针，如果没有返回空指针。

void \*memset(void \*s, int c, size\_t n);

## 把c的值复制到s指向的对象的前n个字符的每个字符中，返回s的值。

* 1. 名字以strn开头的函数对⾮非空字符序列列进⾏行行操作。

char \*strcpy(char \*s1, const char \*s2);

## 包括终⽌止的空字符，返回s1的值

char \*strcat(char \*s1, const char \*s2);

## 包括终⽌止的空字符，返回s1的值。

int strcmp(const char \*s1, const char \*s2);

## ⽐比较s1和s2指向的串串。

char \*strchr(const char \*s, int c);

## 终⽌止的空字符被认为串串的⼀一部分。

char \*strrchr(cosnt char \*s, int c);

## 确定c在s指向的串串中最后⼀一次出现的位置。r意为right⼀一侧。

char \*strstr(const char \*s1, const char \*s2);

## s2指向的串串的字符序列列在s1指向的串串中第⼀一次出现的位置。

size\_t strspn(const char \*s1, const char \*s2);

## 计算s1指向的字符串串中完全由s2指向的字符串串中的字符组成的最⼤大初始段的⻓长度。也就是说，它从s1的开头搜索⼀一个和s2中的任意元素都不不匹配的字符，返回 此字符的下标。

size\_t strcspn(const char \*s1, const char \*s2);

## 查找两个字符串串第⼀一个相同的字符在s1中的位置，也就是The first char both in s1 and s2，如果没有则返回终⽌止的空字符的索引。

strcspn和strchr很相似，但它匹配的是任意⼀一个字符集⽽而不不是⼀一个字符。

char \*strpbrk(const char \*s1, const char \*s2);

## 确定s2指向的串串中的任意的字符在s1中第⼀一次出现的位置。

char \*strtok(char \*s1, const char \*s2);

## 分解字符串串为⼀一组字符串串。s1为要分解的字符串串，s2为分隔符字符串串。

* 1. 名字以str开头的函数对空字符结尾的字符序列列进⾏行行操作。

char \*strncpy(char \*s1, const char \*s2, size\_t n);

## 从s2指向的数组中复制最多n个字符（包括空字符），不不会在s1的末尾⾃自动添加空字符；如果达到n个字符之前遇到了了空字符，则复制完空字符后停⽌止，并在s1指 向的数组后⾯面添加空字符，直到写⼊入了了n个字符。

简单来说，复制，直到遇到空字符或达到n个字符，如果此时复制的字符数未达到

n，就填充空字符，直到字符数达到n。

char \*strncat(char \*s1, const char \*s2, size\_t n);

## 从s2指向的数组中将最多n个字符（空字符及其后⾯面的字符不不添加）添加到s1指 向的串串的结尾。在最后的结果后⾯面加上⼀一个空字符。

简单来说，cat，直到遇到空字符或达到n个字符，最后的结果⼀一定以空字符结 尾。

int strncmp(const char \*s1, const char \*s2, size\_t n);

## 对s1和s2指向的数组中的最多n个字符进⾏行行⽐比较（空字符后⾯面的字符不不参加⽐比较）。

* 1. 其他类

size\_t strlen(const char \*s);

## 计算s指向的串串的⻓长度，不不包括结尾的空字符。

char \*strerror(int errnum);

## 将errnum中的错误编号对应到⼀一个错误信息串串。

* 1. 与区域设置有关

int strcoll(const char \*s1, const char \*s2);

## 功能同strcmp，只是⽐比较时串串都被解释为适合当前区域设置的类别LC\_COLLATE 的形式。

size\_t strxfrm(char \*s1, const char \*s2, size\_t n);

## 如果区域选项是”POSIX”或者”C”，那么strxfrm()同⽤用strncpy()来拷⻉贝字符串串是等价的。

**熟练程序员的层次：**

assert.h

这个头⽂文件只定义了了⼀一个宏： assert

## 如果条件为假，assert将输出⼀一些信息并调⽤用abort函数退出程序。

⼀一个可能的实现：

C++

#define NDEBUG #include <assert.h>

int main(int argc, char\* argv[]) { int a = 12;

int b = 24; assert( a > b );

printf("a is larger than b!");

return 0;

}

上⾯面的程序会发现程序中⽌止， printf 并未执⾏行行，且有这样的输出： main: Assertion 'a

> b' failed. 原因就是因为 a 其实⼩小于 b ，导致断⾔言失败， assert 输出错误信息，并调

⽤用 abort() 中⽌止了了程序执⾏行行。

# limits.h

## 头⽂文件定义了了整型变量量的⼀一些极限值和设置。

CHAR\_BIT ⼀一个ASCII字符⻓长度 SCHAR\_MIN 字符型最⼩小值

SCHAR\_MAX 字符型最⼤大值 UCHAR\_MAX ⽆无符号字符型最⼤大值

## CHAR\_MIN / CHAR\_MAX char字符的最⼤大最⼩小值，如果char字符正被表示有符号整数。它们的值就跟有符号整数⼀一样。否则char字符的最⼩小值就是0，最⼤大值就是⽆无符号char字符的最⼤大值 。 MB\_LEN\_MAX ⼀ 一 个 字 符 所 占 最 ⼤ 大 字 节 数 SHRT\_MIN 最 ⼩ 小 短 整 型 SHRT\_MAX 最⼤大短整形 USHRT\_MAX 最⼤大⽆无符号短整型

INT\_MIN 最⼩小整型 INT\_MAX 最⼤大整形UINT\_MAX 最⼤大⽆无符号整型 LONG\_MIN 最⼩小⻓长整型LONG\_MAX 最⼤大⻓长整型 ULONG\_MAX ⽆无符号⻓长整型

# stddef.h

## 此头⽂文件定义了了3个类型和2个宏，其中⼀一些在其他头⽂文件中也有定义。 定义了了三个类型

ptrdiff\_t 两个指针相减的结果的类型，有符号整型，⼀一般来说是int或long的typedef 。 size\_t 是 sizeof 操 作 符 的 结 果 的 类 型 ， ⽆ 无 符 号 整 型 。 wchar\_t 是⼀一个整型，标识⼀一个宽字节字符，例例如Lʼxʼ的类型就是wchar\_t。

定义了了两个宏

NULL 空指针常量量。

offsetof(type, member-designator) 展开为⼀一个size\_t类型的整值常量量表达式。

# time.h

## 作时间的函数。

1. 定义了了两个宏
   1. NULL
   2. CLOCKS\_PER\_SEC 使 clock 函数的返回值 CLOCKS\_PER\_SEC 的单位是秒

## 定义了了四个类型

声明的类型有size\_t、clock\_t、time\_t和struct tm。

* 1. size\_t 在 stddef.h 中已介绍过。
  2. clock\_t 是 clock 函数的返回值类型
  3. time\_t 是 time 函数的返回值类型，
  4. struct tm 保存了了⼀一个⽇日历时间的各组成部分，⽐比如年年⽉月⽇日时分秒等。

## 定义了了两类函数

* 1. 时间获取函数

time\_t time(time\_t \*timer);

## 取得⽬目前的时间

clock\_t clock(void);

## 确定处理理器器使⽤用的时间。

* 1. 时间转换函数

struct tm \*gmtime(const time\_t \*timer); 把⽇日期和时间转换为(GMT)时间struct tm \*localtime(const time\_t \*timer); 取得当地⽬目前时间和⽇日期char \*ctime(const time\_t \*timer); 把⽇日期和时间转换为字符串串

time\_t mktime(struct tm \*timeptr); 将时间结构数据转换成经过的秒数

char \*asctime(const struct tm \*timeptr); 将时间和⽇日期以字符串串格式表示

size\_t strftime(char \*s, size\_t maxsize, const char \*format, const struct tm \*timeptr); 将时间格式化

## 其他函数

double difftime(time\_t time1, time\_t time0);

## 计算两个⽇日历时间之差： time1-time0， 返回以秒为单位的差值。这个函数似乎简单到没有存在的必要，直接执⾏行行减法不不就得了了。

**优秀程序员的层次：**

float.h

float 头⽂文件类似 limit 头⽂文件，主要是浮点型数值的相关定义。

在所有实例例⾥里里 FLT 指的是 float ， DBL 是 double ， LDBL 指的是 long double .

|  |  |
| --- | --- |
| **变量量** | **定义** |
| FLT\_ROUNDS | 定义浮点型数值四舍五⼊入的⽅方式，-1是不不确定，0是向0，1是向最近，2是向正⽆无穷⼤大，3是负⽆无穷⼤大 |
| FLT\_RADIX 2 | 定义指数的基本表示（⽐比如base-2是  ⼆二进制，base-10是⼗十进制表示法，16 是⼗十六进制） |
| FLT\_MANT\_DIG，DBL\_MANT\_DIG，LDBL\_MANT\_DIG | 定义数值⾥里里数字的个数 |
| FLT\_DIG 6，DBL\_DIG 10，LDBL\_DIG 10 | 在四舍五⼊入之后能不不更更改表示的最⼤大⼩小数位 |
| FLT\_MIN\_EXP，DBL\_MIN\_EXP，LDBL\_MIN\_EXP | FLT\_RADIX 的指数的最⼩小负整数值 |
| FLT\_MIN\_10\_EXP -37，DBL\_MIN\_10\_EXP  -37，LDBL\_MIN\_10\_EXP -37 | 10进制表示法的的指数的最⼩小负整数值 |
| FLT\_MAX\_EXP ,DBL\_MAX\_EXP ,LDBL\_MAX\_EXP | FLT\_RADIX 的指数的最⼤大整数值 |
| FLT\_MAX\_10\_EXP +37 ,DBL\_MAX\_10\_EXP  ,LDBL\_MAX\_10\_EXP +37 +37 | 10进制表示法的的指数的最⼤大整数值 |
| FLT\_MAX 1E+37，DBL\_MAX 1E+37，LDBL\_MAX 1E+37 | 浮点型的最⼤大限 |
| FLT\_EPSILON 1E-5，DBL\_EPSILON 1E-9，LDBL\_EPSILON 1E-9 | 能表示的最⼩小有符号数 |

# math.h

## math.h是 C 语⾔言内的数学函数库。

|  |  |  |  |
| --- | --- | --- | --- |
| **函数** | **含义** | **函数** | **含义** |
| **三⻆角函数** |  |  |  |
| double sin(double x); | 正弦 | double cos(double x); | 余弦 |
| double tan(double x); | 正切 | \*cot三⻆角函数，可以使⽤用tan(PI/2-x)来实现。 |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **函数** | **含义** | **函数** | **含义** |
| **反三⻆角函数** |  |  |  |
| double asin(double x); | 结果介于[-PI/2, PI/2] | double acos(double x); | 结果介于[0, PI] |
| double atan(double x); | 反正切(主值),  结果介于[-PI/2, PI/2] | double atan2(double y,double); | 反正切(整圆值),  结果介于[-PI, PI] |
| **双曲三⻆角函数** |  |  |  |
| double sinh(double x); | 计算双曲正弦 | double cosh(double x); | 计算双曲余弦 |
| double tanh(double x); | 计算双曲正切 |  |  |
| **指数与对数** |  |  |  |
| double exp(double x); | 求取⾃自然数e的幂 | double sqrt(double x); | 开平⽅方 |
| double log(double x); | 以e为底的对数 | double log10(double x); | 以10为底的对数 |
| double pow(double x, double y); | 计算以x为底数的y次幂 | float powf(float x, float y); | 与pow⼀一致，输⼊入与输出皆为浮点数 |
| **取整** |  |  |  |
| double ceil(double); | 取上整 | double floor(double); | 取下整 |
| **标准化浮点数** |  |  |  |
| double frexp(double f, int \*p); | 标准化浮点数, f = x \* 2^p, 已知f求x, p  ( x介于[0.5, 1] ) | double ldexp(double x, int p); | 与frexp相反, 已知  x, p求f |
| **取整与取余** |  |  |  |
| double modf(double, double\*); | 将参数的整数部分通过指针回传, 返回⼩小数部分 | double fmod(double, double); | 返回两参数相除的余数 |