C 学习笔记

2023 年 10 月 7 日

目录

1 C 标准库函数大全

1.1 <ctype.h>

序号	函数原型	功能
1	int iscntrl(int c)	判断字符 c 是否为控制字符
2	int isalnum(int c)	判断字符 c 是否为字母或数字
3	int isalpha(int c)	判断字符 c 是否为英文字母
4	int isascii(int c)	判断字符 c 是否为 ascii 码
5	int isblank(int c)	判断字符 c 是否为 TAB 或空格
6	int isdigit(int c)	判断字符 c 是否为数字
7	int isgraph(int c)	判断字符 c 是否为除空格外的可打印字符
8	int islower(int c)	判断字符 c 是否为小写英文字母
9	int isprint(int c)	判断字符 c 是否为可打印字符（含空格）
10	int ispunct(int c)	判断字符 c 是否为标点符号
11	int isspace(int c)	判断字符 c 是否为空白符
12	int isupper(int c)	判断字符 c 是否为大写英文字母
13	int isxdigit(int c)	判断字符 c 是否为十六进制数字
14	int toascii(int c)	将字符 c 转换为 ascii 码
15	int tolower(int c)	将字符 c 转换为小写英文字母
16	int toupper(int c)	将字符 c 转换为大写英文字母

1.2 <math.h>

序号	函数原型	功能
1	float fabs(float x)	求浮点数 x 的绝对值
2	int abs(int x)	求整数 x 的绝对值
3	float acos(float x)	求 x（弧度表示）的反余弦值
4	float asin(float x)	求 x（弧度表示）的反正弦值
5	float atan(float x)	求 x（弧度表示）的反正切值
6	float atan2(float y, float x)	求 y/x（弧度表示）的反正切值
7	float ceil(float x)	求不小于 x 的最小整数
8	float cos(float x)	求 x（弧度表示）的余弦值
9	float cosh(float x)	求 x 的双曲余弦值
10	float exp(float x)	求 e 的 x 次幂
11	float floor(float x)	求不大于 x 的最大整数
12	float fmod(float x, float y)	计算 x/y 的余数
13	float frexp(float x, int *exp)	把浮点数 x 分解成尾数和指数
14	float ldexp(float x, int exp)	返回 $x * 2^{exp}$ 的值
15	float modf(float num, float *i)	将浮点数 num 分解成整数部分和小数部分

16	float hypot(float x, float y)	对于给定的直角三角形的两个直角边，求其斜边的长度
17	float log(float x)	计算 x 的自然对数
18	float log10(float x)	计算 x 的常用对数
19	float pow(float x, float y)	计算 x 的 y 次幂
20	float pow10(float x)	计算 10 的 x 次幂
21	float sin(float x)	计算 x（弧度表示）的正弦值
22	float sinh(float x)	计算 x（弧度表示）的双曲正弦值
23	float sqrt(float x)	计算 x 的平方根
24	float tan(float x)	计算 x（弧度表示）的正切值
25	float tanh(float x)	求 x 的双曲正切值

1.3 <stdio.h>

序号	函数原型	功能
1	int printf(char *format...)	产生格式化输出的函数
2	int getchar(void)	从键盘上读取一个键，并返回该键的键值
3	int putchar(char c)	在屏幕上显示字符 c
4	FILE *fopen(char *filename, char *type)	打开一个文件
5	FILE *freopen(char *filename, char *type,FILE *fp)	打开一个文件，并将该文件关联到 fp 指定的流
6	int fflush(FILE *stream)	清除一个流
7	int fclose(FILE *stream)	关闭一个文件
8	int remove(char *filename)	删除一个文件
9	int rename(char *oldname, char *newname)	重命名文件
10	FILE *tmpfile(void)	以二进制方式打开暂存文件
11	char *tmpnam(char *sptr)	创建一个唯一的文件名
12	int setvbuf(FILE *stream, char *buf, int type, unsigned size)	把缓冲区与流相关
13	int fprintf(FILE *stream, char *format[, argument,...])	传送格式化输出到一个流中
14	int scanf(char *format[,argument,...])	执行格式化输入
15	int fscanf(FILE *stream, char *format[,argument...])	从一个流中执行格式化输入
16	int fgetc(FILE *stream)	从流中读取字符
17	char *fgets(char *string, int n, FILE *stream)	从流中读取一字符串
18	int fputc(int ch, FILE *stream)	送一个字符到一个流中
19	int fputs(char *string, FILE *stream)	送一个字符到一个流中
20	int getc(FILE *stream)	从流中取字符

21	int getchar(void)	从 stdin 流中读字符
22	char *gets(char *string)	从流中取一字符串
23	int putchar(int ch)	在 stdout 上输出字符
24	int puts(char *string)	送一字符串到流中
25	int ungetc(char c, FILE *stream)	把一个字符退回到输入流中
26	int fread(void *ptr, int size, int nitems, FILE *stream)	从一个流中读数据
27	int fwrite(void *ptr, int size, int nitems, FILE *stream)	写内容到流中 int fseek
28	(FILE *stream, long offset, int fromwhere)	重定位流上的文件指针
29	long ftell(FILE *stream)	返回当前文件指针
30	int rewind(FILE *stream)	将文件指针重新指向一个流的开头
31	int fgetpos(FILE *stream)	取得当前文件的句柄
32	int fsetpos(FILE *stream, const fpos_t *pos)	定位流上的文件指针
33	void clearerr(FILE *stream)	复位错误标志
34	int feof(FILE *stream)	检测流上的文件结束符
35	int ferror(FILE *stream)	检测流上的错误
36	void perror(char *string)	系统错误信息

1.4 <stdlib.h>

序号	函数原型	功能
1	char *itoa(int i)	把整数 i 转换成字符串
2	void exit(int retval)	结束程序
3	double atof(const char *s)	将字符串 s 转换为 double 类型
4	int atoi(const char *s)	将字符串 s 转换为 int 类型
5	long atol(const char *s)	将字符串 s 转换为 long 类型
6	double strtod (const char*s,char **endp)	将字符串 s 前缀转换为 double 型
7	long strtol(const char*s,char **endp,int base)	将字符串 s 前缀转换为 long 型
8	unsinged long strtol(const char*s,char **endp,int base)	将字符串 s 前缀转换为 unsinged long 型
9	int rand(void)	产生一个 0 RAND_MAX 之间的伪随机数
10	void srand(unsigned int seed)	初始化随机数发生器
11	void *calloc(size_t nelem, size_t elsize)	分配主存储器
12	void *malloc(unsigned size)	内存分配函数
13	void *realloc(void *ptr, unsigned newsize)	重新分配主存
14	void free(void *ptr)	释放已分配的块

15	void abort(void)	异常终止一个进程
16	void exit(int status)	终止应用程序
17	int atexit(atexit_t func)	注册终止函数
18	char *getenv(char *envvar)	从环境中取字符串
19	void *bsearch(const void *key, const void *base, size_t *nelem, size_t width, int(*fcmp)(const void *, const *))	二分法搜索函数
20	void qsort(void *base, int nelem, int width, int (*fcmp)())	使用快速排序例程进行排序
21	int abs(int i)	求整数的绝对值
22	long labs(long n)	取长整型绝对值
23	div_t div(int number, int denom)	将两个整数相除, 返回商和余数
24	ldiv_t ldiv(long lnumer, long ldenom)	两个长整型数相除, 返回商和余数

1.5 <time.h>

序号	函数原型	功能
1	clock_t clock(void)	确定处理器时间函数
2	time_t time(time_t *tp)	返回当前日历时间
3	double difftime(time_t time2, time_t time1)	计算两个时刻之间的时间差
4	time_t mktime(struct tm *tp)	将分段时间值转换为日历时间值
5	char *asctime(const struct tm *tblock)	转换日期和时间为 ASCII 码
6	char *ctime(const time_t *time)	把日期和时间转换为字符串
7	struct tm *gmtime(const time_t *timer)	把日期和时间转换为格林尼治标准时间
8	struct tm *localtime(const time_t *timer)	把日期和时间转变为结构
9	size_t strftime(char *s,size_t smax,const char *fmt, const struct tm *tp)	根据 fmt 的格式要求将 *tp 中的日期与时间转换为指定格式

1.6 <string.h>

序号	函数原型	功能
1	int bcmp(const void *s1, const void *s2, int n)	比较字符串 s1 和 s2 的前 n 个字节是否相等
2	void bcopy(const void *src, void *dest, int n)	将字符串 src 的前 n 个字节复制到 dest 中
3	void bzero(void *s, int n)	置字节字符串 s 的前 n 个字节为零
4	void *memccpy(void *dest, void *src, unsigned char ch, unsigned int count)	由 src 所指内存区域复制不多于 count 个字节到 dest 所指内存区域，如果遇到字符 ch 则停止复制

5	void *memcpy(void *dest, void *src, unsigned int count)	由 src 所指内存区域复制 count 个字节到 dest 所指内存区域
6	void *memchr(void *buf, char ch, unsigned count)	从 buf 所指内存区域的前 count 个字节查找字符 ch
7	int memcmp(void *buf1, void *buf2, unsigned int count)	比较内存区域 buf1 和 buf2 的前 count 个字节
8	int memicmp(void *buf1, void *buf2, unsigned int count)	比较内存区域 buf1 和 buf2 的前 count 个字节但不区分字母的大小写
9	void *memmove(void *dest, const void *src, unsigned int count)	由 src 所指内存区域复制 count 个字节到 dest 所指内存区域
10	void *memset(void *buffer, int c, int count)	把 buffer 所指内存区域的前 count 个字节设置成字符 c
11	void setmem(void *buf, unsigned int count, char ch)	把 buf 所指内存区域前 count 个字节设置成字符 ch
12	void movmem(void *src, void *dest, unsigned int count)	由 src 所指内存区域复制 count 个字节到 dest 所指内存区域
13	char *stpcpy(char *dest,char *src)	把 src 所指由 NULL 结束的字符串复制到 dest 所指的数组中
14	char *strcat(char *dest,char *src)	把 src 所指字符串添加到 dest 结尾处 (覆盖 dest 结尾处的’\0’) 并添加’\0’
15	char *strchr(char *s,char c)	查找字符串 s 中首次出现字符 c 的位置
16	int strcmp(char *s1,char * s2)	比较字符串 s1 和 s2
17	int stricmp(char *s1,char * s2)	比较字符串 s1 和 s2，但不区分字母的大小写
18	int strcspn(char *s1,char *s2)	在字符串 s1 中搜寻 s2 中所出现的字符
19	char *strdup(char *s)	复制字符串 s
20	int strlen(char *s)	计算字符串 s 的长度
21	char *strlwr(char *s)	将字符串 s 转换为小写形式
22	char *strupr(char *s)	将字符串 s 转换为大写形式
23	char *strncat(char *dest,char *src,int n)	把 src 所指字符串的前 n 个字符添加到 dest 结尾处 (覆盖 dest 结尾处的’\0’) 并添加’\0’
24	int strcmp(char *s1,char * s2，int n)	比较字符串 s1 和 s2 的前 n 个字符
25	int strnicmp(char *s1,char * s2，int n)	比较字符串 s1 和 s2 的前 n 个字符但不区分大小写
26	char *strncpy(char *dest, char *src, int n)	把 src 所指由 NULL 结束的字符串的前 n 个字节复制到 dest 所指的数组中
27	char *strpbrk(char *s1, char *s2)	在字符串 s1 中寻找字符串 s2 中任何一个字符相匹配的第一个字符的位置，空字符 NULL 不包括在内
28	char *strrev(char *s)	把字符串 s 的所有字符的顺序颠倒过来（不包括空字符 NULL）
29	char *strset(char *s, char c)	把字符串 s 中的所有字符都设置成字符 c
30	char *strstr(char *haystack, char *needle)	从字符串 haystack 中寻找 needle 第一次出现的位置（不比较结束符 NULL)

31	char *strtok(char *s, char *delim)	分解字符串为一组标记串。s 为要分解的字符串，delim 为分隔符字符串
32	int strnicmp(char *s1,char * s2，int n)	比较字符串 s1 和 s2 的前 n 个字符但不区分大小写

2 常用缩写

原词	缩写	释义
A		
absolute	abs	绝对值
address	addr	地址
administration	admin	管理员
action	act	动作
algorithm	algo	算法
allocate	alloc	分配
android	adr	安卓操作系统
application	app	应用
argument	arg	参数
assemble	asm	汇编
asynchronization	async	异步
attribute	attr	属性
authentication	auth	校验
authentication code	authcode	校验码
average	avg	平均
B		
back	bk	回退
bitmap	bm	位图
bottom	btm	底部
button	btn	按钮
buffer	buf	缓存
block	blk	块
between	btw	之间
C		
calculate	calc	计算
character	char	字符
change	chg	改变
check	chk	检查
clear	clr	清除
click	clk	点击
client	cli	客户端
clock	clk	时钟
command line interface	cli	命令行接口
content	cont	内容
context	ctx	上下文
common	comm	通用
config	conf	配置

column	col	列
command	cmd	命令
communication	comu	通信
compare	cmp	比较
complete	cpl	完成
connect	conn	连接
construct	cons	构建
consumer	csm	消费者
control	ctrl	控制
convert	conv	转换
coordinates	coord	坐标
copy	cpy	拷贝
count	cnt	计数
counter	cter	计数器
current	cur	当前
cursor	csr	光标
cycle	cycl	周期，循环
cylinder	cyl	圆柱体
D		
database	db	数据库
dependency	dep	依赖
debug	dbg	调试
decode	deco	解码
decrease	decre	减少
default	def	默认
degree	deg	度数，程度
delay	dly	延迟
delete	del	删除
descriptor	desc	描述符
deserialize	dese	反序列化
destination	dst	目的地
device	dev	设备
dictionary	dict	字典
different	diff	区别
digit	dig	数字
dimension	dim	维度
direct	dirt	直接的
directory	dir	目录
disable	dis	使失效
display	disp	显示
divide	div	除以

dialog	dlg	对话
document	doc	文档
double	dbl	两倍
driver	drv	驱动
duplicate	dup	重复
dynamic	dyn	动态
E		
effective	eff	生效的
electric	elec	电子的
execute	exec	执行
executable file	exe	可执行文件
enviroment	env	环境
error	err	错误
extension	ext	扩展
encode	enc	编码
enable	en	使可能
engine	eng	引擎
equal	eq	相等的
ethernet	eth	以太网
exception	excep	异常
expand	expa	扩展
expiration	expi	过期
exponent	exp	指数
exposure	expo	曝光
F		
feature	fea	特征
field	fld	字段
frequency	freq	频率
flag	flg	标识
frame	frm	框架
fraction	fract	小数
full	ful	全量
G		
greater than	gt	大于
greater equal	ge	大于等于
group	grp	组
generate	gen	产生
H		
horizontal	hori	水平
high	hi	高的
handler	hdler	处理者

handle	hdl	处理
I		
increment	inc	增量
ineffective	ineff	未生效的
infomation	info	信息
identifier	id	标识符
index	idx	索引
input	in	输入
image	img	图片
implement	imp	实现
increase	incre	增加
initialization	init	初始化
insert	isrt	插入
instance	ins	实例
instruction	istr	指令
interrupt	intr	中断
interval	intv	间隔
invalid	inv	无效的
invert	invt	颠倒
L		
less than	lt	小于
less equal	le	小于等于
length	len	长度
level	lv	等级，档位
library	lib	库
link	lnk	连接
limit	lim	限制
list	lst	列表
local	lcl	本地的
lock	lk	上锁
logic	lgc	逻辑的
M		
magnitude	mag	巨大
mailbox	mbox	邮箱
make	mk	做
manager	mgr	管理器
mantissa	mant	尾数
manufacturer	mft	制造商
markdown	md	文本标记语言
marshal	mar	序列化
unmarshal	unmar	反序列化

mask	msk	掩码
maximum	max	最大值
message	msg	消息
memory	mem	内存
middle	mid	中间
minimum	min	最小值
multiply	mul	乘
mutex	mut	互斥锁
move	mov	移动
measure	meas	测量
multiplex	mul	多路复用
N		
negative	neg	负的
number	num	数量
neutral	neut	中立
next	nxt	下一个
O		
object	obj	对象
offset	ofs	偏移
operate	opr	操作，运算
optimize	opt	优化
origin	org	起源
organization	organ	组织
output	out	输出
overrun	ovr	溢出
P		
package	pkg	包
password	pass	密码
parameter	param	参数
performance	perf	性能
period	perd	时期
permutation	permut	排列
permit, permission	perm	许可
picture	pic	图片
point	pt	点
position	pos	位置
positive	posi	积极的
power	pwr	电源
previous	prev	先前的
prefix	pre	前缀
print	prt	打印

priority	prio	优先级
process	proc	进程
product	prod	产品
profile	pf	用户画像; 用户配置
project	proj	项目
program	prog	程序
property	prop	属性
protocol	proto	协议
protect	prot	保护
proxy	prx	代理
public	pub	公共的
Q		
quality	qlty	质量
quarter	quar	四分之一
R		
read	r	读
ready	rdy	就绪
reactive	react	有反应的
recall	rcl	召回
receive	rcv	接收
rectangle	rect	长方形
recursion	recu	递归
reference	ref	引用
register	reg	注册
region	rgn	领域
release	rel	发布
repeat	rpt	重复
repository	repo	仓库
request	req	请求
reserve	resv	保留
reset	rst	重置
resource	res	资源
response	rsp	响应
result	rslt	结果
resume	resu	重新开始
return	ret	返回
reverse	revs	反转
rotate	rot	旋转
S		
sample	smp	样本
scale	scal	比例

scan	sca	扫描
schedule	sch	计划
scheduler	scher	调度器
screen	scr	屏幕
search	srch	搜索
second	sec	秒
section	sect	节
segment	seg	段
select	sel	选择
semaphore	sem	信号量
sequence	seq	序列
server	svr	服务
serialize	seri	序列化
signal	sig	信号
source	src	来源
specification	spec	说明
standard	std	标准
statistic	stat	统计
status	sts	状态
storage	sto	存储
strategy	stra	策略
stream	stm	流
subscribe	subs	订阅
subtract	sub	减去
summation	sum	总和
success	succ	成功
suspend	susp	挂起
switch	sw	开关
synchronization	sync	同步
system	sys	系统
T		
tabbar	tab	标签栏
table	tbl	表
task	tsk	任务
temporary	tmp	临时的
terminate	trmt	终止
test	tst	测试
text	txt	文本
threshold	thr	阈值
timestamp	ts	时间戳
toggle	tgl	切换

toolbar	toolbar	工具栏
total	tot	所有
translate	tran	翻译，转换
Transmit	tx	发送
trigger	trig	触发器
U		
unknown	unk	未知的
unlock	unlk	解锁
update	upd	更新
upgrade	upg	升级
utility	util	工具
V		
value	val	值
variable	var	变量
version	ver	版本
vertical	vert	垂直的
vector	vect	向量
visible	vis	可见的
voltage	vol	电压
virus	vir	病毒
W		
write	w	写
window	win	窗口

3 不常用功能

3.1 位域操作

```
struct mybitfields
 {
  unsigned short a : 4;
  unsigned short b : 5;
  unsigned short c : 7;
 } test;%小端模式，越早声明的在低字节处
*(uint16_t*)&test = 0x1f2;  %赋值
 uint16_t aa = *(uint16_t*)&test;  %取值

union test_union
{
  struct mybitfields test1;
  uint16_t    test2;
};%使用联合体可以方便赋值和取值
```

位域操作

3.2 lowbit 原理

```
uint8_t lowbit(uint8_t x)
{
  uint8_t temp=x;
  x = ~x;
  x++;
  return temp &= x;
}
```

只保留最低位的 1 及其后面的 0: 先取反，第一个 1 变为 0，后面的 0 变为 1；再加 1，第一个 1 变回 1；再与相反数 &，则只保留第一位的 1

```
while(x)
{
  x-=lowbit(x)
  ans++;
}
```

利用 lowbit 统计 1 的个数

4 IAR 功能备忘

4.1 空指令

```
asm("nop");
```
空指令

4.2 文件路径

路径	说明
$PROJ_DIR$	工程文件 *.ewp 所在目录
$PROJ_DIR$\..\	代表 ewp 的上一层目录

4.3 用户手册

iar 及官方工具的用户手册在以下路径：

C:\Program Files\IAR Systems\Embedded Workbench 9.1\arm\doc