# C语言标准库学习

Qsort(int\* , num, sizeof(),mycmp)

Bsearch()

# 一：<time.h>

## time\_t

适合存储日历时间的类型，以秒为单位。

## struct tm

这是一个用来保存时间和日期的结构。

struct tm {

int tm\_sec; /\* 秒，范围从 0 到 59        \*/

int tm\_min; /\* 分，范围从 0 到 59        \*/

int tm\_hour; /\* 小时，范围从 0 到 23        \*/

int tm\_mday; /\* 一月中的第几天，范围从 1 到 31    \*/

int tm\_mon; /\* 月，范围从 0 到 11        \*/

int tm\_year; /\* 自 1900 年起的年数        \*/

int tm\_wday; /\* 一周中的第几天，范围从 0 到 6    \*/

int tm\_yday; /\* 一年中的第几天，范围从 0 到 365    \*/

int tm\_isdst; /\* 夏令时                \*/

};

## [time\_t time(time\_t \*timer)](https://www.runoob.com/cprogramming/c-function-time.html)

计算当前日历时间，并把它编码成 time\_t 格式。

应用：

time\_t rawtime;

time( &rawtime );

## struct tm \*localtime(const time\_t \*timer)

可已将time()函数获取的time\_t类型变量变成**struct tm \*的格式。**

## char \*asctime(const struct tm \*timeptr)

可以将**struct tm \*转化为一个字符串指针。**

## double difftime(time\_t time1, time\_t time2)

返回time1和time2的秒数

## 基本运用：

## 1.【打印本地时间】

int main()

{

time\_t timer;

struct tm\* tmer;

char\* str;

time(&timer);

tmer = localtime(&timer);

str = asctime(tmer);

printf("%s\n",str);

}

## 2.计算时间差

#include<time.h>

#ifdef \_WIN32

#include <Windows.h>

#else

#include <unistd.h>

#endif

int main()

{

time\_t start;

time\_t end;

time(&start);

Sleep(10000);

time(&end);

printf("end:%ld,start:%ld,%d\n",end, start, end - start);

printf("%f\n",difftime(end,start));

}

# 二：<Windows.h>

## Sleep(int ms)

延时函数，window是以ms为单位

# 三：<stdlib.h>

注意：都是item个数在前，也就是nitem, sizeof(item)在后。因此最好malloc也写成malloc(nitem \* sizeof(item))

## Void\* malloc(size\_t size)

分配所需的内存空间，并返回一个指向它的指针。

## Void calloc(size\_t nitem, size\_t sizeofitem)

分配所需的内存空间，并返回一个指向它的指针。

## void \*bsearch(const void \*key, const void \*base, size\_t nitems, size\_t size, int (\*compar)(const void \*, const void \*))

## [void qsort(void \*base, size\_t nitems, size\_t sizeofitem, int (\*compar)(const void \*, const void\*))](https://www.runoob.com/cprogramming/c-function-qsort.html)

qsort对字符串排序，要求第一个参数中元素在原列表位置是可以改的，如直接对参数“dcf”进行调用会失败。

下面这个例子是OK的。



1. 必要辅助的比较函数

注意

MyCmp函数时最好不要写return \*pa -\*pb，不安全容易溢出。

Int MyCmp(const void\* a, const void\* b)

{

Int \*pa = (int\*)a;

Int \*pb = (int\*)b;

Return \*pa > \*pb;

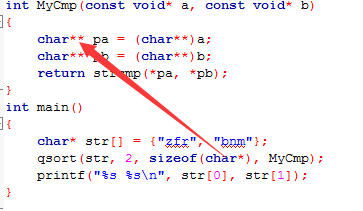
}

当函数返回值>0时，元素a往后排，即升序；【记住这个】

当函数返回值<0时，元素a往前排，即降序；

当函数返回值=0时，元素a，b位置不确定；

(2)比较字符串数组时，注意元素传递上去相当于指针会升级，MyCmp的参数需要更高一级的指针。Str的元素是char\*型的，通过参数传递过去其是char\*\*的。



以下的这个例子同理：

Char\*\*\* tickets = [["MUC", "LHR"], ["JFK", "MUC"], ["SFO", "SJC"], ["LHR", "SFO"]]

int MyCmp(const void\* a, const void\* b)

{

    char\*\*\* pa = (char\*\*\*)a;

    char\*\*\* pb = (char\*\*\*)b;

    int ret = strcmp((\*pa)[1], (\*pb)[1]);

    return ret;

}

qsort(tickets, ticketsSize, sizeof(char\*\*), MyCmp);

## double atof(const char \*str)

如果没有执行有效的转换返回0.00。

例子 double res = atof(“random”);因为不是类似1.2223这种字符串，因此转换不了

# 四：<string.h>

# 五：<assert.h>

# 六：[<stddef.h>](https://www.runoob.com/cprogramming/c-standard-library-stddef-h.html)

# 七：<limits.h>

## INT\_MIN & INT\_MAX

-2147483648 (-2^31)

2147483647 (2^31 - 1)

## LONG\_LONG\_MAX && LONG\_LONG\_MIN

9223372036854775807 (2^63 - 1)

-9223372036854775808 (-2^63)

与LONG\_MAX是不同的，其实LONG\_MAX == INNT\_MAX