qsort（）函数是以快速排序为基础并且可以将任何类型的数据以你想要的方式进行排序。

---

std::qsort

Defined in header <cstdlib>

```
void qsort( void *ptr, std::size_t count,
            std::size_t size, /* c-compare-pred */* comp );
void qsort( void *ptr, std::size_t count,
            std::size_t size, /* compare-pred */* comp );
```

(1)

```
extern "C" using /* c-compare-pred */ = int(const void*, const void*);
extern "C++" using /* compare-pred */ = int(const void*, const void*);
```

(2) (exposition only*)

Sorts the given array pointed to by `ptr` in ascending order. The array contains `count` elements of `size` bytes. Function pointed to by `comp` is used for object comparison.

If `comp` indicates two elements as equivalent, their order is unspecified.

If the type of the elements of the array is not a *PODType*(until C++11) *TriviallyCopyable* type(since C++11), the behavior is undefined.

在C++官网cppreference.com上可以找到以上结果，翻译过来就是使用qsort要包含文件。

这个函数有四个参数qsort(void * base , size_t num, size_t size, int (*cmp)(const void* ,const void*))，第一个参数base 是一个指针，输入要排序数组的首元素的地址。

第二个参数num代表了这个数组的元素个数，第三个参数size代表了数组中的每个元素所占的空间（以字节为单位）。

第四个参数cmp（compare function）是一个函数指针，指向一个有两个参数返回类型为int的函数，我们可以在这个函数里面规定这些数据应该如何排序，如果如果第一个参数小于第二个参数，则返回负整数值，如果第一个参数大于第二个参数，则返回正整数值，如果参数等效，则返回零。

那么接下来有条件的话请将下面的代码复制到你的编译器上，听我解释。

```
#define _CRT_SECURE_NO_WARNINGS 1

#include
#include
```

```
#include

#if 1
void print(int* a, int len)
{
    for (int i = 0; i < len; i++)
    {
        printf("%d ", a[i]);
    }
}

int cmp_by_int(const int* p1, const int* p2)
{
    return  *p2 - *p1;
}

void test1()
{
    int arr[] = { 2,8,6,4,9,3,1,5,7,0 };
    int len = sizeof(arr) / sizeof(arr[0]);

    qsort(arr, len, sizeof(arr[0]), cmp_by_int);
    print(arr, len);
}
/*****************************************************************/
struct stu
{
    char name[20];
    int age;
};

int cmp_by_stu_name(const void * p1, const void * p2)
{
    return strcmp(((struct stu*)p1)->name, ((struct stu*)p2)->name);
}

int cmp_by_stu_age(const void* p1, const void* p2)
{
    return ((struct stu*)p1)->age - ((struct stu*)p2)->age;
}

void test2()
{
    struct stu arr[3] = { {"zhangsan",18},{"lisi",20},{"wangwu",19} };
    int len = sizeof(arr) / sizeof(arr[0]);

    qsort(arr, len, sizeof(arr[0]), cmp_by_stu_name);

    for (int i = 0; i < len; i++)
        printf("%s ", arr[i].name);
```

```
    qsort(arr, len, sizeof(arr[0]), cmp_by_stu_age);

    for (int i = 0; i < len; i++)
        printf("%d ", arr[i].age);
}
/*****************************************************************/
void bubble_sort(int* a, int len)
{
    for (int i = 0; i < len; i++)
    {
        for (int j = 0; j < len - 1 - i; j++)
        {
            if (a[j] > a[j+1])
            {
                a[j] = a[j] ^ a[j+1];
                a[j+1] = a[j] ^ a[j+1];
                a[j] = a[j] ^ a[j+1];
            }
        }
    }
}

void swap(char * p1,char * p2, size_t wide)
{
    for (int i = 0; i < wide; i++)//每个字节分开交换
    {
        char* tmp = p1;
        p1 = p2;
        p2 = tmp;
        p1++;
        p2++;
    }
}

void bubble_qsort(void * base,size_t len,size_t wide ,int (*cmp)(const void*
,const void* ))
{
    for (int i = 0; i < len; i++)
    {
        for (int j = 0; j < len - 1 - i; j++)
        {
            if (cmp((char*)base + j * wide, (char*)base + (j + 1) * wide) > 0)//将
第j和第j+1个元素的首地址传去cmp
            {
                swap((char*)base + j * wide, (char*)base + (j + 1) * wide,
wide);//交换函数
            }
        }
    }
}
void test3()
```

```
{
    int arr[] = { 5,9,1,2,3,7 };
    int sz = sizeof(arr) / sizeof(arr[0]);
    bubble_sort(arr, sz,sizeof(arr[0]),cmp_by_int);
    print(arr, sz);
}
/*****************************************************************/
int main()
{
    test1();//对整型排序
    printf("\n");
    test2();//对结构体排序
    printf("\n");
    test3();//bubble_qsort模拟实现qsort

    return 0;
}
#endif
```

第一段和第二段代码（用/*.....*/分段）分别展示了使用qsort对整型和结构体进行排序。你可以从这里了解到关于qsort具体是如何传参，如何使用，你也可以自己试着用qsort来进行排序。

第三段代码则是把我们熟悉的冒泡排序进行了模拟实现qsort，如果你想要对qsort的理解更加深刻或许你可以删掉此代码然后自己写一遍，这会使你的编程思维得到提升，并且会让你对qsort的实现有一个更加全面的认识。（学习的重点应该放在思维的训练上，而不是单纯的这个函数应该如何使用）

好了本期博客到这里就结束了，如果有什么错误，欢迎指出，如果有帮助，点个赞，谢谢。