\*\* 关于快排函数的一些说明 \*\*

qsort,包含在stdlib.h头文件里,函数一共四个参数,没返回值.一个典型的qsort的写法如下

qsort(s,n,sizeof(s[0]),cmp);

其中第一个参数是参与排序的数组名(或者也可以理解成开始排序的地址,因为可以写&s[i]  
这样的表达式,这个问题下面有说明); 第二个参数是参与排序的元素个数; 第三个三数是  
单个元素的大小,推荐使用sizeof(s[0])这样的表达式,下面也有说明 :) ;第四个参数就是  
很多人觉得非常困惑的比较函数啦,关于这个函数,还要说的比较麻烦...

我们来讨论cmp这个比较函数(写成cmp是我的个人喜好,你可以随便写成什么,比如qcmp什么  
的).典型的cmp的定义是

int cmp(const void \*a,const void \*b);

返回值必须是int,两个参数的类型必须都是const void \*,那个a,b是我随便写的,个人喜好.  
假设是对int排序的话,如果是升序,那么就是如果a比b大返回一个正值,小则负值,相等返回  
0,其他的依次类推,后面有例子来说明对不同的类型如何进行排序.

在函数体内要对a,b进行强制类型转换后才能得到正确的返回值,不同的类型有不同的处理  
方法.具体情况请参考后面的例子.

\*\* 关于快排的一些小问题 \*\*

1.快排是不稳定的,这个不稳定一个表现在其使用的时间是不确定的,最好情况(O(n))和最  
坏情况(O(n^2))差距太大,我们一般说的O(nlog(n))都是指的是其平均时间.

2.快排是不稳定的,这个不稳定表现在如果相同的比较元素,可能顺序不一样,假设我们有  
这样一个序列,3,3,3,但是这三个3是有区别的,我们标记为3a,3b,3c,快排后的结果不一定  
就是3a,3b,3c这样的排列,所以在某些特定场合我们要用结构体来使其稳定(No.6的例子就  
是说明这个问题的)

3.快排的比较函数的两个参数必须都是const void \*的,这个要特别注意,写a和b只是我的  
个人喜好,写成cmp也只是我的个人喜好.推荐在cmp里面重新定义两个指针来强制类型转换,  
特别是在对结构体进行排序的时候

4.快排qsort的第三个参数,那个sizeof,推荐是使用sizeof(s[0])这样,特别是对结构体,  
往往自己定义2\*sizeof(int)这样的会出问题,用sizeof(s[0)既方便又保险

5.如果要对数组进行部分排序,比如对一个s[n]的数组排列其从s[i]开始的m个元素,只需要  
在第一个和第二个参数上进行一些修改:qsort(&s[i],m,sizeof(s[i]),cmp);

\*\* 标程,举例说明 \*\*

No.1.手工实现QuickSort  
#include <stdio.h>

int a[100],n,temp;

void QuickSort(int h,int t)  
{  
     if(h>=t) return;  
     int mid=(h+t)/2,i=h,j=t,x;  
     x=a[mid];  
     while(1)  
     {  
         while(a[i]<x) i++;  
         while(a[j]>x) j--;  
         if(i>=j) break;  
         temp=a[i];  
         a[i]=a[j];  
         a[j]=temp;  
     }  
     a[mid]=a[j];  
     a[j]=x;  
     QuickSort(h,j-1);  
     QuickSort(j+1,t);  
     return;  
}

int main()  
{  
     int i;  
     scanf("%d",&n);  
     for(i=0;i<n;i++) scanf("%d",&a[i]);  
     QuickSort(0,n-1);  
     for(i=0;i<n;i++) printf("%d ",a[i]);

     return(0);  
}

No.2.最常见的,对int数组排序  
#include <stdio.h>  
#include <string.h>  
#include <stdlib.h>

int s[10000],n,i;

int cmp(const void \*a, const void \*b)  
{  
     return(\*(int \*)a-\*(int \*)b);  
}

int main()  
{  
     scanf("%d",&n);  
     for(i=0;i<n;i++) scanf("%d",&s[i]);  
      
     qsort(s,n,sizeof(s[0]),cmp);  
      
     for(i=0;i<n;i++) printf("%d ",s[i]);  
      
     return(0);  
}

No.3.对double型数组排序,原理同int

这里做个注释,本来是因为要判断如果a==b返回0的,但是严格来说,两个double数是不可能相等的,只能说fabs(a-b)<1e-20之类的这样来判断,所以这里只返回了1和-1  
#include <stdio.h>  
#include <stdlib.h>

double s[1000];  
int i,n;

int cmp(const void \* a, const void \* b)  
{  
     return((\*(double\*)a-\*(double\*)b>0)?1:-1);  
}

int main()  
{  
     scanf("%d",&n);  
     for(i=0;i<n;i++) scanf("%lf",&s[i]);  
      
     qsort(s,n,sizeof(s[0]),cmp);  
      
     for(i=0;i<n;i++) printf("%lf ",s[i]);  
      
     return(0);  
}

No.4.对一个字符数组排序.原理同int  
#include <stdio.h>  
#include <string.h>  
#include <stdlib.h>

char s[10000],i,n;

int cmp(const void \*a,const void \*b)  
{0  
     return(\*(char \*)a-\*(char \*)b);  
}

int main()  
{  
     scanf("%s",s);  
     n=strlen(s);  
     qsort(s,n,sizeof(s[0]),cmp);  
      
     printf("%s",s);  
     return(0);  
}

No.5.对结构体排序

注释一下.很多时候我们都会对结构体排序,比如校赛预选赛的那个[樱花](http://blog.sina.com.cn/lm/z/sakura/),一般这个时候都在  
cmp函数里面先强制转换了类型,不要在return里面转,我也说不清为什么,但是这样程序会  
更清晰,并且绝对是没错的. 这里同样请注意double返回0的问题

#include <stdio.h>  
#include <stdlib.h>

struct node  
{  
     double date1;  
     int no;  
} s[100];

int i,n;

int cmp(const void \*a,const void \*b)  
{  
     struct node \*aa=(node \*)a;  
     struct node \*bb=(node \*)b;  
     return(((aa->date1)>(bb->date1))?1:-1);  
}

int main()  
{  
     scanf("%d",&n);  
     for(i=0;i<n;i++)  
     {  
         s[i].no=i+1;  
         scanf("%lf",&s[i].date1);  
     }  
     qsort(s,n,sizeof(s[0]),cmp);  
      
     for(i=0;i<n;i++) printf("%d   %lf\n",s[i].no,s[i].date1);  
      
     return(0);  
}

No.6.对结构体排序.加入no来使其稳定(即data值相等的情况下按原来的顺序排)

#include <stdio.h>  
#include <stdlib.h>

struct node  
{  
     double date1;  
     int no;  
} s[100];

int i,n;

int cmp(const void \*a,const void \*b)  
{  
     struct node \*aa=(node \*)a;  
     struct node \*bb=(node \*)b;  
     if(aa->date1!=bb->date1)  
         return(((aa->date1)>(bb->date1))?1:-1);  
     else  
         return((aa->no)-(bb->no));  
}

int main()  
{  
     scanf("%d",&n);  
     for(i=0;i<n;i++)  
     {  
         s[i].no=i+1;  
         scanf("%lf",&s[i].date1);  
     }  
     qsort(s,n,sizeof(s[0]),cmp);

     for(i=0;i<n;i++) printf("%d   %lf\n",s[i].no,s[i].date1);

     return(0);  
}

No.7.对字符串数组的排序(char s[][]型)

#include <stdio.h>  
#include <string.h>  
#include <stdlib.h>

char s[100][100];  
int i,n;

int cmp(const void \*a,const void \*b)  
{  
     return(strcmp((char\*)a,(char\*)b));  
}

int main()  
{  
     scanf("%d",&n);  
     for(i=0;i<n;i++) scanf("%s",s[i]);

     qsort(s,n,sizeof(s[0]),cmp);  
      
     for(i=0;i<n;i++) printf("%s\n",s[i]);  
      
     return(0);  
}

No.8.对字符串数组排序(char \*s[]型)  
#include <stdio.h>  
#include <string.h>  
#include <stdlib.h>

char \*s[100];  
int i,n;

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

{  
     return(strcmp(\*(char\*\*)a,\*(char\*\*)b));  
}

int main()  
{  
     scanf("%d",&n);  
     for(i=0;i<n;i++)  
     {  
         s[i]=(char\*)malloc(sizeof(char\*));  
         scanf("%s",s[i]);  
     }

     qsort(s,n,sizeof(s[0]),cmp);

     for(i=0;i<n;i++) printf("%s\n",s[i]);

     return(0);  
}