Sort_structure_1.c

```
/*
sort_structure.c -- shows simple double sort using qsort
*/
#include <stdlib.h>
#include <stdio.h>

#define MAX 100
/* The "typedef" keyword creates a synonym for a structure -- in this
 * case "kpsorter".  Hence, when the word "kpsorter" is used below the
 * compiler knows that you mean "struct kpsorter".  Also "kpsorter"
 * becomes a word like "int", "double", or "char" so you can declare
 * something as a structure of type kpsorter (see below)*/
typedef struct {int icount;
                double data;
} kpsorter;

int structcomparison(const void *v1, const void *v2);

int main(void)
{
        int i;
        double arr[MAX], arrsort[MAX];
/* Declare recordset as a structure defined by the label kpsorter */
        kpsorter recordset[MAX];
        srand(17);
        printf("RAND_MAX=%10d\n", RAND_MAX);            /* check operating
system limit value */
        for(i=0;i<MAX;i++)
        {
                arr[i] = rand()/((double)RAND_MAX + 1); /* fill arr[,] with
random doubles*/
                arrsort[i]=arr[i];
                recordset[i].icount=i;
                recordset[i].data=arr[i];
        }
        qsort(recordset, MAX, sizeof(kpsorter), structcomparison);
        for(i=0;i<MAX;i++)
        {
                printf("%10d %10d  arr[%10.6f]  arrsort[%10.6f]\n", i,
recordset[i].icount, arrsort[i], recordset[i].data);
        }
        printf("\n\n");
        return(0);
}
int structcomparison(const void *v1, const void *v2)
{
/* Again, "kpsorter" is a declaration of a type like "int" or "double"
 * Here we declare two pointers p1 and p2 of type "kpsorter" and set
 * them equal to the pointers v1 and v2.  Note the "(kpsorter *)" CASTS
 * v1 and v2 as pointers of type structure as defined by kpsorter above*/
        kpsorter *p1=(kpsorter *)v1;
        kpsorter *p2=(kpsorter *)v2;
/* The "->" is special to structures -- it is "aiming" at the data
```

```
 * element of the structure */
        if(p1->data < p2->data)
                return -1;
        else if (p1->data == p2->data)
                return 0;
        else
                return 1;
}
```

Sort_structure_2.c

```
/*
sort_structure.c -- shows simple double sort using qsort
*/
#include <stdlib.h>
#include <stdio.h>

#define MAX 100
// Declare structure here -- the structure tag is "kpsorter"
struct kpsorter {
        int icount;
        double data;
};

int structcomparison(const void *v1, const void *v2);

int main(void)
{

/* Declare Structure here of type kpsorter -- you need to use the
 * "struct" keyword because "typedef" was not used above -- recordset
 * is now a structure according to the kpsorter template above*/
        struct kpsorter recordset[MAX];
         int i;
        double arr[MAX], arrsort[MAX];
        srand(17);
        printf("RAND_MAX=%10d\n", RAND_MAX);            /* check operating
system limit value */
        for(i=0;i<MAX;i++)
        {
                arr[i] = rand()/((double)RAND_MAX + 1); /* fill arr[,] with
random doubles*/
                arrsort[i]=arr[i];
                recordset[i].icount=i;
                recordset[i].data=arr[i];
        }
/* Either of these sorting commands will work*/
//      qsort(recordset, MAX, sizeof(struct kpsorter), structcomparison);
        qsort(recordset, MAX, sizeof(recordset[0]), structcomparison);
        for(i=0;i<MAX;i++)
        {
                printf("%10d %10d  arr[%10.6f]  arrsort[%10.6f]\n", i,
recordset[i].icount, arrsort[i], recordset[i].data);
        }
        printf("\n\n");
        return(0);
}
int structcomparison(const void *v1, const void *v2)
{
/* these two sets of commands do the same thing: This */
//      struct kpsorter *p1=(struct kpsorter *)v1;
//      struct kpsorter *p2=(struct kpsorter *)v2;
/* and this: */
        struct kpsorter *p1, *p2;
```

```
        p1=(struct kpsorter *)v1;  /* the "(struct kpsorter *)" CASTS v1 and
v2 */
        p2=(struct kpsorter *)v2;  /*  as pointers of type structure as
                                   *  defined by kpsorter above */
        if(p1->data < p2->data)
                return -1;
        else if (p1->data == p2->data)
                return 0;
        else
                return 1;
}
```

Read_sen110.c

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

/*
    Function to pull a string variable out of a string record given a column
    position range for the string (the first character of string is
    referenced as 0).  The function then returns the string with a
    terminating character (0)
*/
void getstrbypos(char *buf, int start, int stop, char *res) {
        strncpy(res,buf+start,stop-start+1); /* get substring field from buf
*/
        res[stop-start+1] = 0; /* Add end of string character */
}

/*
    Function to pull and an integer out of a string record given
    a column position range for the integer.
*/
void getintbypos(char *buf, int start,int stop, int *res) {
        char field[100];
/* Call function getstrbypos to extract the character string that will
Be converted to an integer.  This character string is returned in field*/
        getstrbypos(buf,start,stop,field);
/*SSCANF - formatted input conversion -- "sscanf(s, format [, p1, p2, ...] )"
reads input from the string "s" and assigns it to the areas of memory pointed
to by the pointers "p1", "p2", and so on. The "format" string indicates how
to interpret the characters being read in.  Below “field” is our character
string; “%i”, is our format statement, and “res” is our memory address.
 */
        sscanf(field,"%i",res); /* copy field to res as integer */
}


int main(void) {
        FILE *fp;
        int cong,icpsr,state,district,party;
        char statename[10], lname[20], fname[50];
        char buf[80];

        if ((fp=fopen("sen110_names.txt","r"))==NULL) {
                printf("Cannot open file!\n");
                exit(1);
        }

   /* read row from data (up to 80 chars wide) */
        while ( fgets(buf,80,fp) != NULL) {
   /* write row from data (up to 80 chars wide) to screen */
                printf("%s\n",buf);

                getintbypos(buf,0,3,&cong);
                getintbypos(buf,4,15,&icpsr);
```

```
                getintbypos(buf,16,24,&state);
                getintbypos(buf,25,26,&district);
                getstrbypos(buf,28,36,statename);
                getintbypos(buf,37,39,&party);
                getstrbypos(buf,41,53,lname);
                getstrbypos(buf,54,80,fname);

                printf("Cong   = %i\n",cong);
                printf("ICPSR  = %i\n",icpsr);
                printf("State  = %i\n",state);
                printf("Dist   = %i\n",district);
                printf("Staten = %s\n",statename);
                printf("Party  = %i\n",party);
                printf("lname  = %s\n",lname);
                printf("fname  = %s\n\n",fname);
        }
        return(0);
}
```