Read_sen110_2.c

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

/*
    Function to pull a string var out a string record given a column
    position range for the string (the first character of string is
    referenced as 0).
*/
void getstrbypos(char *buf, int start, int stop, char *res) {
//      char field[100];
        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 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];
        getstrbypos(buf,start,stop,field);
        sscanf(field,"%i",res); /* copy field to res as integer */ }

/*
    Function to pull a double out of a string record given
    a column position range for the double.
*/
void getdoublebypos(char *buf, int start,int stop, double *res) {
        char field[100];
        getstrbypos(buf,start,stop,field);
        sscanf(field,"%lf",res); /* copy field to res as double */ }

typedef struct {
        int counter;
        int congress;
        double xcoordinate;
        int icpsrid;
        int statecode;
        int cd;
        char statenm[9];
        int party;
        char lastname[40];
        char firstname[40];
} kpsorter;

int structcomparison(const void *v1, const void *v2);
int structnamecomparison(const void *s1, const void *s2);
int structintcomparison(const void *t1, const void *t2);

int main(void) {
        FILE *fp;
        FILE *jp;
        double xcoord;
        int cong,icpsr,state,district,party, MAX;
        int i;
        char statename[9], lname[40], fname[40];
        char buf[80];

        jp=fopen("sen110_sort_structure.txt","w");
        if ((fp=fopen("sen110_names_coord.txt","r"))==NULL) {
                printf("Cannot open file!\n");
                exit(1);
        }
/* Open the file and read it as characters to get the number of rows */
        i = 0;
        while ( fgets(buf,80,fp) != NULL) {
                i = i + 1;
        }
        MAX=i;
        kpsorter recordset[MAX];
        fprintf(jp,"Hello Beavis, Number of Records = %d\n",MAX);
        rewind(fp);

         i = 0;
   /* read row from data (up to 80 chars wide) */
        while ( fgets(buf,80,fp) != NULL) {
                printf("%s\n",buf);
                recordset[i].counter=i;

                getintbypos(buf,0,3,&cong);
                recordset[i].congress=cong;
                getdoublebypos(buf,4,9,&xcoord);
                recordset[i].xcoordinate=xcoord;
                getintbypos(buf,10,15,&icpsr);
                recordset[i].icpsrid=icpsr;
                getintbypos(buf,16,24,&state);
                recordset[i].statecode=state;
                getintbypos(buf,25,26,&district);
                recordset[i].cd=district;
                getstrbypos(buf,28,36,recordset[i].statenm);
                strncpy(statename,recordset[i].statenm,9);
                getintbypos(buf,37,39,&party);
                recordset[i].party=party;
                getstrbypos(buf,41,53,recordset[i].lastname);
                strncpy(lname,recordset[i].lastname,13);
                getstrbypos(buf,54,80,recordset[i].firstname);
                strncpy(fname,recordset[i].firstname,27); /* get substring
field from buf */

                fprintf(jp,"Cong   = %i\n",cong);
                fprintf(jp,"Xcoord = %f\n",xcoord);
                fprintf(jp,"ICPSR  = %i\n",icpsr);
                fprintf(jp,"State  = %i\n",state);
                fprintf(jp,"Dist   = %i\n",district);
                fprintf(jp,"Staten = %s\n",statename);
                fprintf(jp,"Party  = %i\n",party);
                fprintf(jp,"lname  = %s\n",lname);
                fprintf(jp,"fname  = %s\n\n",fname);
                i = i+1;
        }
        fprintf(jp,"Hello Beavis, Number of Records = %d\n",i);
        qsort(recordset, MAX, sizeof(kpsorter), structcomparison);
        for(i=0;i<MAX;i++)
        {
                printf("%5d %5d %7.3f %s %s", i, recordset[i].counter,
                       recordset[i].xcoordinate, recordset[i].lastname,
recordset[i].firstname);
                fprintf(jp,"%5d %5d %7.3f %s %s", i, recordset[i].counter,
                       recordset[i].xcoordinate, recordset[i].lastname,
recordset[i].firstname);
        }
        printf("\n\n");

        fprintf(jp,"Hello Beavis, Number of Records = %d\n",i);
        qsort(recordset, MAX, sizeof(kpsorter), structnamecomparison);
        for(i=0;i<MAX;i++)
        {
                printf("%5d %5d %7.3f %s %s", i, recordset[i].counter,
                       recordset[i].xcoordinate, recordset[i].lastname,
recordset[i].firstname);
                fprintf(jp,"%5d %5d %7.3f %s %s", i, recordset[i].counter,
                          recordset[i].xcoordinate, recordset[i].lastname,
recordset[i].firstname);
        }
        printf("\n\n");

        fprintf(jp,"Hello Beavis, Number of Records = %d\n",i);
        qsort(recordset, MAX, sizeof(kpsorter), structintcomparison);
        for(i=0;i<MAX;i++)
        {
                printf("%5d %5d %7.3f %s %s", i, recordset[i].counter,
                       recordset[i].xcoordinate, recordset[i].lastname,
recordset[i].firstname);
                fprintf(jp,"%5d %5d %7.3f %s %s", i, recordset[i].counter,
                          recordset[i].xcoordinate, recordset[i].lastname,
recordset[i].firstname);
        }
        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->xcoordinate < p2->xcoordinate)
                return -1;
        else if (p1->xcoordinate == p2->xcoordinate)
                return 0;
        else
                return 1;
}
int structnamecomparison(const void *s1, const void *s2)
{
        kpsorter *p1=(kpsorter *)s1;
        kpsorter *p2=(kpsorter *)s2;
/* The "->" is special to structures -- it is "aiming" at the data
 * element of the structure */
        int res;
        res = strcmp(p1->lastname, p2->lastname);
        if(res != 0)
                return res;
        else
                return strcmp(p1->firstname, p2-> firstname);
}
int structintcomparison(const void *t1, const void *t2)
{
        kpsorter *p1=(kpsorter *)t1;
        kpsorter *p2=(kpsorter *)t2;
/* The "->" is special to structures -- it is "aiming" at the data
 * element of the structure */
        if(p1->counter < p2->counter)
                 return -1;
        else if(p1->counter == p2->counter)
                        return 0;
        else
                return 1;
}
```

Read_dwnom.c

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

/*
    Function to pull a string var out a string record given a column
    position range for the string (the first character of string is
    referenced as 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 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];
        getstrbypos(buf,start,stop,field);
        sscanf(field,"%i",res); /* copy field to res as integer */ }

/*
    Function to pull a double out of a string record given
    a column position range for the double.
*/
void getdoublebypos(char *buf, int start,int stop, double *res) {
        char field[100];
        getstrbypos(buf,start,stop,field);
        sscanf(field,"%lf",res); /* copy field to res as double */ }

typedef struct {
        int counter;
        int congress;
        int icpsrid;
        int statecode;
        int cd;
        char statenm[9];
        int party;
        char lastname[40];
        double xcoordinate;
        double ycoordinate;
        double sdxcoordinate;
        double sdycoordinate;
        double loglikelihood;
        int nchoices;
        int nerrors;
        double gmp;
} kpsorter;

int structcomparison(const void *v1, const void *v2);
int structnamecomparison(const void *s1, const void *s2);
int structintcomparison(const void *t1, const void *t2);

int main(void) {
        FILE *fp;
        FILE *jp;
        double xcoord, ycoord, sdxcoord, sdycoord, xloglike, xgmp;
        int cong,icpsr,state,district,party, choices, merrors;
        int i, MAX;
        char statename[9], lname[40];
        char buf[100];

        jp=fopen("dwnom_sort_structure.txt","w");
        if ((fp=fopen("hl01111b21_pres.dat","r"))==NULL) {
                printf("Cannot open file!\n");
                exit(1);
        }
/* Open the file and read it as characters to get the number of rows */
        i = 0;
        while ( fgets(buf,100,fp) != NULL) {
                i = i + 1;
        }
        MAX=i;
        printf("Hello Beavis number 1, Number of Records = %d\n",MAX);
        fprintf(jp,"Hello Beavis, Number of Records = %d\n",MAX);
        rewind(fp);
/* Dynamically allocate memory to big data structure */
        kpsorter *recordset;
        recordset = (kpsorter *)malloc(MAX * sizeof(kpsorter));

        i = 0;
   /* read row from data (up to 100 chars wide) */
        while ( fgets(buf,100,fp) != NULL) {
                recordset[i].counter=i;

                getintbypos(buf,0,3,&cong);
                recordset[i].congress=cong;
                getintbypos(buf,4,9,&icpsr);
                recordset[i].icpsrid=icpsr;
                getintbypos(buf,10,12,&state);
                recordset[i].statecode=state;
                getintbypos(buf,13,14,&district);
                recordset[i].cd=district;
                getstrbypos(buf,16,22,recordset[i].statenm);
                strncpy(statename,recordset[i].statenm,9);
                getintbypos(buf,23,27,&party);
                recordset[i].party=party;
                getstrbypos(buf,29,39,recordset[i].lastname);
                strncpy(lname,recordset[i].lastname,11);
                getdoublebypos(buf,40,46,&xcoord);
                recordset[i].xcoordinate=xcoord;
                getdoublebypos(buf,47,53,&ycoord);
                recordset[i].ycoordinate=ycoord;
                getdoublebypos(buf,54,60,&sdxcoord);
                recordset[i].sdxcoordinate=sdxcoord;
                getdoublebypos(buf,61,67,&sdycoord);
                recordset[i].sdycoordinate=sdycoord;
                getdoublebypos(buf,68,79,&xloglike);
                recordset[i].loglikelihood=xloglike;
                getintbypos(buf,80,84,&choices);
                recordset[i].nchoices=choices;
                getintbypos(buf,85,89,&merrors);
                recordset[i].nerrors=merrors;
                getdoublebypos(buf,90,96,&xgmp);
                recordset[i].gmp=xgmp;

                i = i+1;
        }
        printf("Hello Beavis number 2, Number of Records = %d\n",i);
        fprintf(jp,"Hello Beavis, Number of Records = %d\n",i);
        qsort(recordset, MAX, sizeof(kpsorter), structcomparison);
        for(i=0;i<MAX;i++)
        {
                fprintf(jp,"%5d %5d %11s %7.3f %7.3f\n", i,
recordset[i].counter,recordset[i].lastname,
                       recordset[i].xcoordinate, recordset[i].ycoordinate);
        }
        fprintf(jp,"\n\n");

        printf("Hello Beavis number 3, Number of Records = %d\n",i);
        fprintf(jp,"Hello Beavis, Number of Records = %d\n",i);
        qsort(recordset, MAX, sizeof(kpsorter), structnamecomparison);
        for(i=0;i<MAX;i++)
        {
                fprintf(jp,"%5d %5d %11s %7.3f %7.3f\n", i,
recordset[i].counter,recordset[i].lastname,
                         recordset[i].xcoordinate, recordset[i].ycoordinate);
        }
        fprintf(jp,"\n\n");

        printf("Hello Beavis number 4, Number of Records = %d\n",i);
        fprintf(jp,"Hello Beavis, Number of Records = %d\n",i);
        qsort(recordset, MAX, sizeof(kpsorter), structintcomparison);
        for(i=0;i<MAX;i++)
        {
                fprintf(jp,"%5d %5d %11s %7.3f %7.3f\n", i,
recordset[i].counter,recordset[i].lastname,
                         recordset[i].xcoordinate, recordset[i].ycoordinate);
        }
        fprintf(jp,"\n\n");
        free(recordset);
        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->xcoordinate < p2->xcoordinate)
                return -1;
        else if (p1->xcoordinate == p2->xcoordinate)
                return 0;
        else
                return 1;
}
int structnamecomparison(const void *s1, const void *s2)
{
        kpsorter *p1=(kpsorter *)s1;
        kpsorter *p2=(kpsorter *)s2;
/* The "->" is special to structures -- it is "aiming" at the data
 * element of the structure */
        int res;
        res = strcmp(p1->lastname, p2->lastname);
        return res;
}
int structintcomparison(const void *t1, const void *t2)
{
        kpsorter *p1=(kpsorter *)t1;
        kpsorter *p2=(kpsorter *)t2;
/* The "->" is special to structures -- it is "aiming" at the data
 * element of the structure */
        if(p1->counter < p2->counter)
                 return -1;
        else if(p1->counter == p2->counter)
                        return 0;
        else
                return 1;
}
```