```
/*
 * file.c
 * Aquiring data from datafiles
 */

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

#include "main.h"
#include "file.h"
#include "convert.h"

/*
 * get[x|y|z]Num: return the number of x,y or z values
 * given the type of plot
 */
long getxNum (struct dataset *ds)
{
return ds->X * ds->Y;
}

long getyNum (struct dataset *ds)
{
return ds->X * ds->Y;
}

long getzNum(struct dataset *ds)
{
return ds->X * ds->Y;
}

struct dataset *scaleDataSet(struct dataset *ds)
{
double min, max;
int i;

/* Need to find Max and Min value regardless of axis */
min = ds->x[0]; max = ds->x[0];
```

```
for (i=0; i<getxNum(ds); i++) {
if (ds->x[i] < min)
min = ds->x[i];
if (ds->x[i] > max)
max = ds->x[i];
}
for (i=0; i<getyNum(ds); i++) {
if (ds->y[i] < min)
min = ds->y[i];
if (ds->y[i] > max)
max = ds->y[i];
}
for (i=0; i<getzNum(ds); i++) {
if (ds->z[i] < min)
min = ds->z[i];
if (ds->z[i] > max)
max = ds->z[i];
}

/* Scale all values so they lie in a unit cube */
max = max - (max + min)/2; min = min - (max + min)/2;

for (i=0; i<getxNum(ds); i++) {
ds->x[i] = (ds->x[i] - (max + min)/2) * (1/max);
}
for (i=0; i<getyNum(ds); i++) {
ds->y[i] = (ds->y[i] - (max + min)/2) * (1/max);
}
for (i=0; i<getzNum(ds); i++) {
ds->z[i] = (ds->z[i] - (max + min)/2) * (1/max);
}

return ds;
}

void destroyDataSet(struct dataset *ds)
{
if (ds == NULL)
return;
if (ds->x != NULL)
```

```
free(ds->x);
if (ds->y != NULL)
free(ds->y);
if (ds->z != NULL)
free(ds->z);

free(ds);
}

struct dataset *readDataFile(FILE *fh)
{
struct dataset *ds;
char buf[256];
char tmp[256];
int i,j;
int line=0;
int file_valid = 1;
char type[5];
int notpara = 0;
double *t;
double v;


/* Initialise the data set structure */
printf("Initialising the data structure...");
ds = malloc(sizeof(struct dataset));
ds->x = NULL;
ds->y = NULL;
ds->z = NULL;
printf("done\n");


/* Is this parametric data ? If not, flag it */
fgets(buf, sizeof(buf) - 1, fh);
if (strncmp(buf, "PARA", 4) == 0) {
printf("Type PARA\n");
notpara = 0;
} else {
notpara = 1;
strncpy(type, buf, 4);
```

```
}

fgets(buf, sizeof(buf) - 1, fh);

/* Line should be of form 'X,Y'; X,Y > 0 */
i=0; j=0;
while( i < strlen(buf) && buf[i] != ',' ) {
tmp[j] = buf[i];
i++; j++;
}

ds->X = atoi(tmp);
i++; j=0;

while( i < strlen(buf) ) {
tmp[j] = buf[i];
i++; j++;
}

ds->Y = atoi(tmp);

/* Y is optional.. so still could be 0 */
if (ds->X <= 0 || ds->Y < 0) {
file_valid = 0;
printf("readDataFile: File invalid; Line %d\n", line);
return NULL;
}

/* Fix for KURV type, set Y=1 so room to store dataset */


/* Allocate memory for dataset */
printf("Allocate memory for dataset...");

t = realloc(ds->x, getxNum(ds) * sizeof(double));
printf("Num X values: %ld\n", getxNum(ds) );
if (t == NULL) {
destroyDataSet(ds);
printf("readDataFile: failed while reallocating data");
return NULL;
```

```
}
ds->x = t;

t = realloc(ds->y, getyNum(ds) * sizeof(double));
if (t == NULL) {
destroyDataSet(ds);
printf("readDataFile: failed while reallocating data");
return NULL;
}
ds->y = t;

t = realloc(ds->z, getzNum(ds) * sizeof(double));
if (t == NULL) {
destroyDataSet(ds);
printf("readDataFile: failed while reallocating data");
return NULL;
}
ds->z = t;

printf("done\n");

/* Convert dataset to PARA if required */
if (notpara == 1) {
convertToPara(type, ds);
}


/* x values */
for (i=0; i< getxNum(ds); i++) {
fgets(buf, sizeof(buf) - 1, fh);
line++;
v = strtod(buf, NULL);
ds->x[i] = v;
}
/* y values */
for (i=0; i< getyNum(ds); i++) {
fgets(buf, sizeof(buf) - 1, fh);
line++;
v = strtod(buf, NULL);
ds->y[i] = v;
```

```
}
/* z values */
for (i=0; i< getzNum(ds); i++) {
fgets(buf, sizeof(buf) - 1, fh);
v = strtod(buf, NULL);
ds->z[i] = v;
}

/* fclose(fh); */

/* Scale all the values */
ds = scaleDataSet(ds);

return ds;
}
```