导线测量程序设计报告

## 题目要求

设计程序进行附合导线计算，数据从文件中读入，计算角度闭合差、全长闭合差和各坐标的平差值。

## 需求分析

输入：./data.txt

格式：（角度使用 度,分,秒 表示；坐标使用 X,Y 表示。同一行的各数据间用空格隔开）

第一行：测站数n

第二行：已知方位角（2）

第三行：观测角（左角）（n）

第四行：边长（n-1）

第五行：已知坐标（2）

格式范例：

6

237,59,30 46,45,24

99,01,00 167,45,36 123,11,24 189,20,36 179,59,18 129,27,24

225.85 139.03 172.57 100.07 102.48

2507.69,1215.63 2166.74,1757.27

输出：./out.txt

内容：输出内容有角度闭合差、导线相对闭合差和各导线坐标。

范例：

角度闭合差=-0.000175

导线相对闭合差K=1/3523.809524

导线坐标：

(2507.69,1215.63)

(2299.83,1303.80)

(2186.29,1383.97)

(2192.45,1556.40)

(2179.74,1655.64)

(2166.74,1757.27)

数学模型：

先通过已知方位角和观测角计算角度闭合差，然后计算改正数和改正角以计得各坐标方位角。

通过坐标方位角与距离计算出坐标增量和相对闭合差，并计算改正。最后计算改正后坐标值。

## 方案设计

数据结构：

使用数组表示各数据

角度使用弧度以double记录，当输入输出时转化成度分秒形式使用int[3]记录。

坐标使用double[2]记录。

边长使用double记录。

主要数据结构如下：

double \*gcj;//观测角

double \*jdgz;//角度改正值

double \*gzj;//改正后角度

double \*fwj;//方位角

double (\*zb)[2];//坐标

double (\*zbzl)[2];//坐标增量

double (\*zbgz)[2];//坐标改正

double (\*gzzl)[2];//改正后增量

double \*bc;//边长

主要算法：

程序中先将文件内容导入，在导入过程中先根据读到的测站数动态分配储存数据的内存。随后将其他数据读入内存并转换成相应的形式保存在变量中。

计算角度闭合差并与允许值进行对比。

进行角度改正。

计算方位角。

计算全长闭合差。

进行坐标改正。

计算改正后坐标。

将所需内容输出到文件。

模块分解：

程序中分为将主要算法和主要流程。主要流程在count()函数中。

函数设计：

int Round(double num) //四舍五入到整数

double Round2(double num) //四舍五入到小数点后两位

double DMSToRed(int angle[])//度分秒转弧度

void RedToDMS(double red, int angle[])//弧度转度分秒

void printDMS(char word[], int angle[])//输出度分秒

void printRedToDMS(char word[], double red) //输出度分秒

void refreshDegWithDMS(double &red) //刷新使用角度刷新弧度,防止产生累积差

void ZBZS(double zb[2], double bc, double fwj) //坐标正算

void strSplit(char str[], const char spl[], char \*\*ret) //字符分割

void StrToDoubles(char str[], const char spl[], double num[]) //字符串转doubles

void StrToReds(char str[], const char spl[], double num[]) //字符串转Reds

void StrToXYs(char str[], const char spl[], double num[][]) //字符串转x,y

void printXY(char str[], double zb[2])//输出坐标

double Modulo(double zb[2])//取模

## 程序实现：

程序源代码:

#include <stdio.h>

#include <iostream.h>

#include <math.h>

#include <string.h>

#include <stdlib.h>

#define PI 3.141592653589793238462643383279502884197169399375105820974944

//四舍五入到整数

int Round(double num){

return (int)(num>0 ? num+0.5: num-0.5);

}

//四舍五入到小数点后两位

double Round2(double num){

return (double)Round(num\*100)/100;

}

//输入度分秒

void GetDMS(char word[], int angle[]){

printf("%s",word);

scanf("%d,%d,%d",&angle[0],&angle[1],&angle[2]);

//检查输入

if(angle[0]>=360){

angle[0] -= 360;

}else if(angle[0]<0){

angle[0] += 360;

}

if(angle[2]>=60 || angle[2]<0 || angle[1]>=60 || angle[1]<0){

printf("输入错误，请重新输入\n");

GetDMS(word, angle);

}

}

//输入坐标

void GetXY(char word[], double XY[]){

printf("%s",word);

scanf("%lf,%lf",&XY[0],&XY[1]);

}

//输入整数

void GetInt(char word[], int &data){

printf("%s",word);

scanf("%d",&data);

}

//输入浮点数

void GetDouble(char word[], double &data){

printf("%s",word);

scanf("%lf",&data);

}

//度分秒转秒

int DMSToSec(int deg[]){

return deg[2] + deg[1]\*60 + deg[0]\*3600;

}

//秒转度分秒

void SecToDMS(int sec, int ret[]){

//检查范围

//if(sec >= 360\*60\*60){

// sec -= 360\*60\*60;

//}else if(sec < 0){

// sec += 360\*60\*60;

//}

//转换

ret[0] = sec/3600;

ret[1] = (sec%3600)/60;

ret[2] = (sec%60);

}

//角度加法

void AngAdd(int res[], int a[], int b[]){

SecToDMS(DMSToSec(a) + DMSToSec(b), res);

/\*

int i;

for(i=0;i<3;i++){

res[i] = a[i] + b[i];

}

for(i=2;i>0;i--){

if(res[i]>60){

res[i] =- 60;

res[i-1]++;

}

}

\*/

}

//角度减法

void AngMin(int res[], int a[], int b[]){

SecToDMS(DMSToSec(a) - DMSToSec(b), res);

}

//////////////////////////////////////////////////////////////////

//度分秒转弧度

double DMSToRed(int angle[]){

return (angle[0]<0?-1:1) \* (abs(angle[0]\*3600) + angle[1]\*60 + angle[2]) \* PI / 180 / 3600;

}

//弧度转度分秒

void RedToDMS(double red, int angle[]){

double secd = red \* 180 \* 3600 / PI;

int sec = Round(secd);

angle[0] = sec/3600;

angle[1] = (sec%3600)/60;

angle[2] = sec%60;

//double sec = red \* 180 \* 3600 / PI;

//angle[0] = sec/3600;

//angle[1] = (sec-angle[0]\*3600)/60;

//angle[2] = Round(sec-angle[0]\*3600-angle[1]\*60);

}

//输入度分秒

double GetDMSToRed(char word[]){

int angle[3];

printf("%s",word);

scanf("%d,%d,%d",&angle[0],&angle[1],&angle[2]);

//检查输入

if(angle[0]>=360){

angle[0] -= 360;

}else if(angle[0]<0){

angle[0] += 360;

}

if(angle[2]>=60 || angle[2]<0 || angle[1]>=60 || angle[1]<0){

printf("输入错误，请重新输入\n");

GetDMS(word, angle);

}

return DMSToRed(angle);

}

//输出度分秒

void printDMS(char word[], int angle[]){

printf("%s(%d,%d,%d)\n",word,angle[0],angle[1],angle[2]);

}

//输出度分秒

void printRedToDMS(char word[], double red){

int angle[3];

RedToDMS(red, angle);

printDMS(word, angle);

}

//刷新使用角度刷新弧度,防止产生累积差

void refreshDegWithDMS(double &red){

int angle[3];

RedToDMS(red, angle);

red = DMSToRed(angle);

}

//坐标正算

void ZBZS(double zb[2], double bc, double fwj){

zb[0] = bc \* cos(fwj);

zb[1] = bc \* sin(fwj);

}

//字符分割

void strSplit(char str[], const char spl[], char \*\*ret){

char \*p;

p = strtok(str, spl);

int i = 0;

while(p)

{

ret[i++] = p;

p = strtok(NULL, spl);

}

}

//字符串转doubles

void StrToDoubles(char str[], const char spl[], double num[]){

char \*ret[10] = {"","","","","","","","","",""};

strSplit(str,spl,ret);

int i = 0;

for(i=0;ret[i]!="";i++){

num[i] = atof(ret[i]);//char[]转double

}

}

//字符串转Reds

void StrToReds(char str[], const char spl[], double num[]){

char \*ret[10] = {"","","","","","","","","",""};

char \*cang[3] = {"","",""};

int ang[3];

strSplit(str,spl,ret);

for(int i=0;ret[i]!="";i++){

strSplit(ret[i],",",cang);

for(int j=0;j<3;j++){

ang[j] = atof(cang[j]);//char[]转double

}

num[i] = DMSToRed(ang);

}

}

//字符串转x,y

void StrToXYs(char str[], const char spl[], double num[][]){

char \*ret[10] = {"","","","","","","","","",""};

char \*cang[2] = {"",""};

double xy[2];

strSplit(str,spl,ret);//空格分割

for(int i=0;ret[i]!="";i++){

strSplit(ret[i],",",cang);//二次分割

for(int j=0;j<2;j++){

xy[j] = atof(cang[j]);//char[]转double

}

memcpy(num[i],xy,sizeof(xy));

}

}

void printXY(char str[], double zb[2]){

cout << str << "(" << zb[0] << "," << zb[1] << ")" << endl;

}

double Modulo(double zb[2]){

return sqrt(zb[0]\*zb[0] + zb[1]\*zb[1]);

}

//计算闭合导线

void Count(){

int i;

int j;

int sta;//测站数(坐标数)max:10

//int lor;//左角或右角

double \*gcj;//观测角

double \*jdgz;//角度改正值

double \*gzj;//改正后角度

double \*fwj;//方位角

double (\*zb)[2];//坐标

double (\*zbzl)[2];//坐标增量

double (\*zbgz)[2];//坐标改正

double (\*gzzl)[2];//改正后增量

double \*bc;//边长

double dxcd = 0;//导线长度

double bhc;//角度闭合差

double bhcyxz;//角度闭合差允许值

double qcbhc[2];//导线全长闭合差

double qcbhcyxz[2];//导线全长闭合差允许值

/\*

//手动输入数据

GetInt("测站数(坐标数)：", sta);

GetInt("左角+1，右角-1：", lor);

printf("请输入2个已知角:\n");

fwj[0] = GetDMSToRed("");

fwj[sta-1] = GetDMSToRed("");

//检查输入

//printRedToDMS("",fwj[0]);

//printRedToDMS("",fwj[sta-1]);

printf("请输入%d个观测角:\n",sta-1);

for(i=1;i<sta;i++){

gcj[i] = GetDMSToRed("");

}

//检查输入

//for(i=1;i<sta;i++){

// printRedToDMS("",gcj[i]);

//}

printf("请输入4个已知坐标:\n");

GetXY("B:", zb[0]);

GetXY("A:", zb[1]);

GetXY("C:", zb[sta-1]);

GetXY("D:", zb[sta]);

printf("请输入%d个边长:\n",sta-2);

for(i=1;i<sta-1;i++){

GetDouble("", bc[i]);

dxcd += bc[i];

}

\*/

FILE \*fp = NULL;//文件指针

char buff[255];//缓冲区

double nbuf[255] = {0};//数组缓存

fp = fopen("data.txt", "r");

fgets(buff, 255, (FILE\*)fp);

sta = atof(buff) + 1;

//fscanf(fp,"%d",&sta);

cout << "测站数:" << sta << endl;

//根据测站数动态分配主要变量大小

gcj = (double\*)calloc(sta, sizeof(double));

jdgz = (double\*)calloc(sta, sizeof(double));

gzj = (double\*)calloc(sta, sizeof(double));

fwj = (double\*)calloc(sta, sizeof(double));

zb = new double[sta+1][2];

zbzl = new double[sta][2];

zbgz = new double[sta][2];

gzzl = new double[sta][2];

bc = (double\*)calloc(sta, sizeof(double));

fgets(buff, 255, (FILE\*)fp);

StrToReds(buff, " ", nbuf);

fwj[0] = nbuf[0];

fwj[sta-1] = nbuf[1];

cout << "已知角:" << endl;

printRedToDMS("",fwj[0]);

printRedToDMS("",fwj[sta-1]);

fgets(buff, 255, (FILE\*)fp);

StrToReds(buff, " ", nbuf);

for(i=1;i<sta;i++){

gcj[i] = nbuf[i-1];

}

cout << "观测角:" << endl;

for(i=1;i<sta;i++){

printRedToDMS("", gcj[i]);

}

fgets(buff, 255, (FILE\*)fp);

StrToDoubles(buff, " ", nbuf);

for(i=1;i<sta-1;i++){

bc[i] = nbuf[i-1];

dxcd += bc[i];

}

cout << "边长:" << endl;

for(i=1;i<sta-1;i++){

cout << bc[i] << endl;

}

fgets(buff, 255, (FILE\*)fp);

StrToXYs(buff, " ", zb);

memcpy(zb[sta-1],zb[1],sizeof(zb[1]));//memcpy(a,b,sizeof(b));//b的元素复制给a

memcpy(zb[1],zb[0],sizeof(zb[0]));

cout << "坐标:" << endl;

cout << "(" << zb[1][0] << "," << zb[1][1] << ")" << endl;

cout << "(" << zb[sta-1][0] << "," << zb[sta-1][1] << ")" << endl;

fclose(fp);

cout << "导入完毕！" << endl << endl;

//导入完毕

cout << "边长和=" << dxcd << endl;

//计算角度闭合差

bhc = fwj[0] - fwj[sta-1];

for (i = 1; i < sta; i++)

{

//printRedToDMS("", bhc);//中间步骤检查

refreshDegWithDMS(bhc);

bhc += gcj[i] - PI;

}

printRedToDMS("角度闭合差=",bhc);

int baseNum[] = {0,0,40};

bhcyxz = DMSToRed(baseNum)\*sqrt(sta-1);

printRedToDMS("闭合差允许值=±",bhcyxz);

if(bhc > bhcyxz){

cout << "角度闭合差超限！" << endl;

//return 0;

}else cout << "角度闭合差在限差范围内√" << endl;

//进行角度改正

jdgz[0] = -bhc / (sta - 1);

//不考虑改正值不平均分配的情况。

for(i = 1; i < sta; i++){

gzj[i] = gcj[i] + jdgz[0];

fwj[i] = fwj[i-1] + gzj[i] - PI;

}

cout << endl << "改正角：" << endl;

for(i=1;i<sta;i++){

printRedToDMS("",gzj[i]);

}

cout << endl << "坐标方位角：" << endl;

for(i=0;i<sta;i++){

printRedToDMS("",fwj[i]);

}

//计算导线全长闭合差

for(j=0;j<2;j++) qcbhc[j] = zb[1][j] - zb[sta-1][j];

cout << endl << "坐标增量：" << endl;

for(i=1;i<sta-1;i++){

ZBZS(zbzl[i], bc[i], fwj[i]);//计算坐标增量

for(j=0;j<2;j++) zbzl[i][j] = Round2(zbzl[i][j]);

printXY("",zbzl[i]);

for(j=0;j<2;j++) qcbhc[j] += zbzl[i][j];//计算闭合差

}

printXY("导线闭合差(fx,fy):", qcbhc);

cout << "导线全长闭合差f=" << Round2(Modulo(qcbhc)) << endl;

cout << "导线相对闭合差K=" << "1/" << dxcd/Round2(Modulo(qcbhc)) << endl;

cout << "允许相对闭合差K允=" << "1/2000" << endl;

if(dxcd/Round2(Modulo(qcbhc)) < 2000){

cout << "相对闭合差超限！" << endl;

//return 0;

}else cout << "相对闭合差在限差范围内√" << endl;

//进行坐标改正

cout << endl << "坐标增量改正：" << endl;

for(i=1;i<sta-1;i++){

for(j=0;j<2;j++) zbgz[i][j] = Round2(-qcbhc[j] \* bc[i] / dxcd);

printXY("",zbgz[i]);

for(j=0;j<2;j++) gzzl[i][j] = zbzl[i][j] + zbgz[i][j];

}

cout << endl << "改正后坐标增量：" << endl;

for(i=1;i<sta-1;i++){

printXY("",gzzl[i]);

}

for(i=1;i<sta-1;i++){

for(j=0;j<2;j++) zb[i+1][j] = zb[i][j] + gzzl[i][j];

}

cout << endl << "坐标：" << endl;

for(i=1;i<sta;i++){

printXY("",zb[i]);

}

//输出文件

fp = fopen("out.txt", "w");

fprintf(fp, "角度闭合差=%lf\n", bhc);

if(bhc > bhcyxz){fprintf(fp, "角度闭合差超限！");}

fprintf(fp, "导线相对闭合差K=1/%lf\n", dxcd/Round2(Modulo(qcbhc)));

if(dxcd/Round2(Modulo(qcbhc)) < 2000){fprintf(fp, "相对闭合差超限！");}

fprintf(fp, "导线坐标：\n");

for(i=1;i<sta;i++){

fprintf(fp, "(%.2lf,%.2lf)\n", zb[i][0], zb[i][1]);

}

fclose(fp);

cout << "写入完毕，已输出到out.txt" << endl << endl;

return;

}

int main(){

Count();

return 0;

}

算例运行结果

输入:

6

237,59,30 46,45,24

99,01,00 167,45,36 123,11,24 189,20,36 179,59,18 129,27,24

225.85 139.03 172.57 100.07 102.48

2507.69,1215.63 2166.74,1757.27

输出:

角度闭合差=-0.000175

导线相对闭合差K=1/3523.809524

导线坐标：

(2507.69,1215.63)

(2299.83,1303.80)

(2186.29,1383.97)

(2192.45,1556.40)

(2179.74,1655.64)

(2166.74,1757.27)

## 总结

效果评价

该程序能正确计算附和导线平差,并使用了动态内存分配.关键算法中使用四舍六入计算,使得计算的结果最接近于手工计算的结果.

改进设想

在程序中增加输入格式引导.增加格式检查使其能识别格式错误并抛出异常.