**201533661 이승수’s algorithm homework#14 duedate: 2016.11.17.**

**[Clean Campus-graham scan]**

**<code>**

#include <stdio.h>

#include <math.h>

#define INF 1000

#define DEFAULT 0 #include <stdio.h>

#include <math.h>

#define INF 1000

#define DEFAULT 0

struct position{

float x;

float y;

float tan;

int index;

}pos[10];

int posNum = 0;

int stack[10] = {0};

int top=0;

int stackedNum = 0;

FILE \*inF, \*outF;

int fourDivision(float x, float y);

float Distance(int index1, int index2);

float angleCOS(int indexRoot,int indexA, int indexB);

struct position vetorize(int indexRoot, int index);

int ccw(int indexA, int indexB, int indexC);

void swap(int indexA, int indexB);

int pop();

void push(int val);

void printStack();

void main()

{

pos[0].x = 0;

pos[0].y = 0;/\*start at (0,0)\*/

inF = fopen("input.txt","r");

fscanf(inF, "%d", &posNum);

for (int i = 1; i <= posNum; i++)

{

fscanf(inF,"%f %f",&pos[i].x,&pos[i].y);

printf("\n%2.1f %2.1f",pos[i].x,pos[i].y);

pos[i].index = i;

pos[i].tan = pos[i].y / pos[i].x;

}

fclose(inF);

for (int i =posNum; i >0; i--)

{

for (int j = 1; j < i; j++)

{

if (!ccw(0,j,j+1))

{

swap(j, j + 1);

}

}

}

float distance = 1;//banded at (0,0) when start

for (int i = 1; i <= 10; i++)//initialize stack

{

stack[i] = DEFAULT;

}

while (stackedNum<=posNum)

{

push(pos[++stackedNum].index);

if (top <= 2)

{

printf("\n%d,%d",top,stackedNum);

printStack();

continue;

}

if (ccw(stack[top - 2], stack[top - 1], stack[top])<0)

{

pop();

pop();

push(pos[top+2].index);

}

printStack();

}

int cur = 0;

while (cur < top)//calculate distance

{

distance += Distance(stack[cur],stack[cur+1]);

cur++;

}

distance++;//to band at (0,0)

outF = fopen("output.txt", "w");

fprintf(outF,"distance: %4.2f",distance);

fclose(outF);

}

int ccw(int indexA, int indexB, int indexC)

{

struct position tempB;

struct position tempC;

tempB.x = pos[indexB].x - pos[indexA].x;

tempB.y = pos[indexB].x - pos[indexA].y;

tempB.tan = tempB.y / tempB.x;

tempB.index = pos[indexB].index;

tempC.x = pos[indexC].x - pos[indexA].x;

tempC.y = pos[indexC].y - pos[indexA].y;

tempC.tan = tempC.y / tempC.x;

tempC.index = pos[indexC].index;

float tanzentCB = (tempC.tan - tempB.tan) / (1+tempC.tan\*tempB.tan);

float cosCB=angleCOS(indexA,indexB,indexC);

if (tanzentCB\*cosCB > 0)//ccw:dimension 1 or 2

return 1;

else if (tanzentCB\*cosCB < 0)//cw:dimension 3 or 4

return -1;

else

return 0;

}

struct position vetorize(int indexRoot,int index)

{

struct position temp;

temp.x = (pos[indexRoot].x-pos[index].x);

temp.y = (pos[indexRoot].y-pos[index].y);

temp.tan = temp.y/temp.x;

temp.index = pos[index].index;

return temp;

}

float angleCOS(int indexRoot,int indexA,int indexB)

{

float innerProduct = (pos[indexA].x - pos[indexRoot].x)\*(pos[indexB].x - pos[indexRoot].x)+ (pos[indexA].y-pos[indexRoot].y)\*(pos[indexB].y-pos[indexRoot].y);

if (innerProduct == 0)

{

printf("\nERROR:vectorA,B(%d,%d from %d) is in right angle",indexA,indexB,indexRoot);

}

float vectorA = Distance(indexRoot,indexA);

float vectorB = Distance(indexRoot,indexB);

return innerProduct / (vectorA\*vectorB);

}

int fourDivision(float x, float y)

{

if (x > 0 && y > 0)

return 1;

else if (x < 0 && y>0)

return 2;

else if (x < 0 && y < 0)

return 3;

else if (x > 0 && y < 0)

return 4;

else if (y == 0)

return INF;

else if (x == 0)

{

printf("zero");

return 0;

}

return 0;

}

float Distance(int index1,int index2)

{

float difX = pos[index2].x - pos[index1].x;

float difY = (pos[index2].y-pos[index1].y);

return sqrt(pow(difX,2)+pow(difY,2));

}

void swap(int indexA,int indexB)

{

struct position temp;

temp.x = pos[indexA].x;

pos[indexA].x = pos[indexB].x;

pos[indexB].x = temp.x;

temp.y = pos[indexA].y;

pos[indexA].y = pos[indexB].y;

pos[indexB].y = temp.y;

temp.tan = pos[indexA].tan;

pos[indexA].tan = pos[indexB].tan;

pos[indexB].tan = temp.tan;

temp.index = pos[indexA].index;

pos[indexA].index = pos[indexB].index;

pos[indexB].index = temp.index;

}

void push(int val)

{

stack[++top] = val;

}

int pop()

{

int temp = stack[top];

stack[top--] = DEFAULT;

return temp;

}

void printStack()

{

printf("\nStack[");

for (int k = 0; k <= top; k++)

{

printf("%d,",stack[k]);

}

printf("]");

}