1. **Užduotis.** **Spalvoti taškai**

Dvimatėje koordinačių plokštumoje yra **n** (1 ≤ n ≤ 12) trijų spalvų taškų. Jungiant vienodos spalvos taškus po tris, galima sudaryti daug skirtingų trikampių. Parašykite programą, kuri kiekvienos spalvos taškų aibėje surastų tuos tris taškus, kuriuos sujungus gautųsi lygiašonis trikampis su didžiausiu įmanomu perimetru.

Pradinių duomenų faile **taskai\_data.txt** pirmoje eilutėje užrašytas visų taškų kiekis **n**. Sekančiose **n** eilučių yra išvardinti visi taškai formatu 'SPALVA X\_KOORDINATE Y\_KOORDINATE'. Spalvų atitinkami kodai: r = raudona, m = mėlyna, z = žalia. Tiek X, tiek Y koordinatė yra sveikieji skaičiai intervale [-100;100].

Rezultatų faile **taskai\_res.txt** išspausdinkite tris eilutes: kiekvienoje eilutėje spalvą (pavyzdyje nurodyta tvarka) bei atitinkamos spalvos lygiašonio trikampio su didžiausiu perimetru perimetrą (vieno skaitmens po kablelio tikslumu). Jeigu tokio trikampio sudaryti negalima, išspausdinkite žodį 'nera' mažosiomis raidėmis. Nenaudokite lietuviškų raidžių.

|  |  |
| --- | --- |
| taskai\_data.txt | taskai\_res.txt |
| 12  r -3 -2  z -6 2  r -5 10  z 0 10  m 0 12  m 1 8  r 2 3  z 4 6  z 4 11  r 2 -3  r 6 0  z 6 2 | Raudona: 16.0  Melyna: nera  Zalia: 32.0 |

1. **Reikalavimai**

* Naudokite sveikųjų, realiųjų skaičių bei simbolių masyvus.
* Formuokite skirtingų spalvų trikampių perimetrų masyvus.
* Pradinių duomenų skaitymo funkcija **void**.
* Atstumo tarp dviejų taškų skaičiavimo funkcija, grąžinanti apskaičiuotą reikšmę per funkcijos vardą.
* Loginė funkcija, tikrinanti, ar iš 3 taškų galima sudaryti lygiašonį trikampį (nepamirškite, kad jungti galima tik vienodos spalvos taškus).
* Funkcija, skaičiuojanti trikampio perimetrą, grąžinanti apskaičiuotą reikšmę per funkcijos vardą.
* Funkcija, surandanti didžiausios reikšmės vietą realiųjų skaičių masyve (indeksą, ne pačią reikšmę).
* Rezultatų spausdinimo funkcija **void**.

1. **Programos tekstas C++ kalba**

#include <iostream>

#include <iomanip>

#include <cmath>

#include <fstream>

using namespace std;

void Skaito(int RX[], int RY[], int ZX[], int ZY[], int MX[], int MY[], int &r, int &z, int &m);

void MasP(int X[], int Y[], int &n, double Z[]);

double Atstumas(int x, int y, int x1, int y1);

bool Lygiasonis(int x, int y, int x1, int y1, int x2, int y2);

double Perimetras(int x, int y, int x1, int y1, int x2, int y2);

void Raso(double R, double Z, double M);

int Didziausias(double A[], int n);

int main(){

int RX[100], RY[100], ZX[100], ZY[100], MX[100], MY[100], r = 0, z = 0, m = 0;

double RP[100], ZP[100], MP[100];

Skaito(RX, RY, ZX, ZY, MX, MY, r, z, m);

MasP(RX, RY, r, RP);

MasP(ZX, ZY, z, ZP);

MasP(MX, MY, m, MP);

Raso(RP[Didziausias(RP, r)], ZP[Didziausias(ZP, z)], MP[Didziausias(MP, m)]);

return 0;

}

void Skaito(int RX[], int RY[], int ZX[], int ZY[], int MX[], int MY[], int &r, int &z, int &m)

{

int n, x, y;

string s;

ifstream df ("taskai\_data.txt");

df >> n;

for (int i = 0; i < n; i++){

df >> s >> x >> y;

if (s == "r"){

RX[r] = x;

RY[r] = y;

r++;

}

else if (s == "z"){

ZX[z] = x;

ZY[z] = y;

z++;

}

else{

MX[m] = x;

MY[m] = y;

m++;

}

}

df.close();

}

void MasP(int X[], int Y[], int &n, double Z[])

{

double pirm;

int ind = 0;

for (int i = 0; i < n - 2; i++){

for (int i1 = i + 1; i1 < n - 1; i1++){

for(int i2 = i1 + 1; i2 < n; i2++){

if(Lygiasonis(X[i], Y[i], X[i1], Y[i1], X[i2], Y[i2])){

Z[ind] = Perimetras(X[i], Y[i], X[i1], Y[i1], X[i2], Y[i2]);

ind++;

}

}

}

}

n = ind;

}

double Atstumas(int x, int y, int x1, int y1)

{

return sqrt(pow(x - x1, 2) + pow(y - y1, 2));

}

bool Lygiasonis(int x, int y, int x1, int y1, int x2, int y2){

if (Atstumas(x, y, x1, y1) == Atstumas(x1, y1, x2, y2) || Atstumas(x1, y1, x2, y2) == Atstumas(x2, y2, x, y) || Atstumas(x2, y2, x, y) == Atstumas(x, y, x1, y1))return true;

else return false;

}

double Perimetras(int x, int y, int x1, int y1, int x2, int y2)

{

return Atstumas(x, y, x1, y1) + Atstumas(x1, y1, x2, y2) + Atstumas(x2, y2, x, y);

}

void Raso(double R, double Z, double M)

{

ofstream rf ("taskai\_res.txt");

rf << "Raudona: ";

if (R > 0){

rf << fixed << setprecision(1) << R << endl;

}

else rf << "nera" << endl;

rf << "Melyna: ";

if (M > 0){

rf << M << endl;

}

else rf << "nera" << endl;

rf << "Zalia: ";

if (Z > 0){

rf << Z;

}

else rf << "nera" << endl;

rf.close();

}

int Didziausias(double A[], int n)

{

int ind = 0;

for (int i = 0; i < n; i++)if(A[i] > A[ind])ind = i;

return ind;

}