1. **Užduotis.** **Sportininkai**

Tarptautinis Olimpinis Komitetas (TOK) sportininkams rekomenduoja maisto sudėtinių dalių pasiskirstymą per parą: angliavandeniai turėtų sudaryti **a** procentų, riebalai – **r** procentų, baltymai – **b** procentų gaunamų kilokalorijų (kcal). Žinomi sportininkų duomenys: ūgis, masė ir suvartojamas energijos kiekis kcal per parą.

Parašykite programą, kuri rastų aukščiausio, žemiausio, lengviausio ir sunkiausio sportininko masę, ūgį, suvartojamos energijos kiekį kcal per parą ir kūno masės indeksą (KMI). Sportininko kūno masės indeksas (KMI), apskaičiuojamas pagal Adolphe Quetelet formulę: KMI = masė (kg) / ūgis2 (cm2).

Jeigu yra keli vienodai aukščiausi (žemiausi, sunkiausi, lengviausi) sportininkai, tai spausdinami visų duomenys.

Pradinių duomenų faile **sportininkai\_data.txt**:

* pirmoje eilutėje įrašyti trys sveikieji skaičiai **a**, **r**, **b** – rekomenduojamas angliavandenių, baltymų ir riebalų kiekis procentais;
* antroje failo eilutėje įrašytas krepšinio komandos sportininkų skaičius **n**;
* tolesnėse **n** eilučių įrašyta po du realiuosius skaičius: sportininko kūno masė (kg), ūgis (m) ir sveikasis skaičius – per parą suvartojamas energijos kiekis (kcal).

Rezultatų faile **sportininkai\_res.txt** turi būti įrašyta:

* pirmosiose n eilučių – kiekvieno sportininko eilės numeris ir kūno masės indeksas (KMI), apvalinamas iki sveikojo skaičiaus;
* tolesnėse eilutėse turi būti spausdinama:
  + aukščiausio sportininko duomenys: eilės numeris, masė, ūgis, suvartojamas energijos kiekis, KMI. Jei yra keli tokie sportininkai, tuomet turi būti spausdinami visų sportininkų duomenys;
  + žemiausio sportininko duomenys: eilės numeris, masė, ūgis, suvartojamas energijos kiekis, KMI. Jei yra keli tokie sportininkai, tuomet turi būti spausdinami visų sportininkų duomenys;
  + lengviausio sportininko duomenys: eilės numeris, masė, ūgis, suvartojamas energijos kiekis, KMI. Jei yra keli tokie sportininkai, tuomet turi būti spausdinami visų sportininkų duomenys;
  + sunkiausio sportininko duomenys: eilės numeris, masė, ūgis, suvartojamas energijos kiekis, KMI. Jei yra keli tokie sportininkai, tuomet turi būti spausdinami visų sportininkų duomenys.

|  |  |
| --- | --- |
| sportininkai\_data.txt | sportininkai\_res.txt |
| 60 25 15  12  82.00 1.85 3608  98.00 1.98 4073  92.00 1.95 3893  80.00 1.90 3541  125.00 2.08 4853  107.00 2.04 4358  107.00 2.05 4367  97.00 2.03 4094  117.00 2.11 4677  98.00 2.05 4138  93.00 1.98 3946  85.00 1.88 3650 | 1 24  2 25  3 24  4 22  5 29  6 26  7 25  8 24  9 26  10 23  11 24  12 24  9 117.00 2.11 4677 26  1 82.00 1.85 3608 24  4 80.00 1.90 3541 22  5 125.00 2.08 4853 29 |

1. **Reikalavimai**

* Naudokite sveikųjų ir realiųjų skaičių masyvus.
* Sukurkite pradinių duomenų skaitymo procedūrą (**void**).
* Sukurkite didžiausios reikšmės paieškos funkciją. Ją panaudokite ieškodami aukščiausio ir sunkiausio sportininko.
* Sukurkite mažiausios reikšmės paieškos funkciją. Ją panaudokite ieškodami žemiausio ir lengviausio sportininko.
* Sukurkite rezultatų rašymo į failą procedūrą (**void**).

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

#include <iostream>

#include <iomanip>

#include <cmath>

#include <fstream>

using namespace std;

void Skaito(int &a, int &b, int &r, int &n, double M[], double U[], int E[]);

void Kmi(int n, double M[], double U[], double K[]);

void Maziausias(double A[], int n, int M[], int &ind);

void Didziausias(double A[], int n, int M[], int &ind);

void Raso(double K[], int &n, double M[], int MM[], int mm, int MD[], int md, int UM[], int um, int UD[], int ud, int E[], double U[]);

int main(){

int a, b, r, n, MM[100], MD[100], UM[100], UD[100], mm = 0, md = 0, ud = 0, um = 0, E[100];

double M[100], U[100], K[100];

Skaito(a, b, r, n, M, U, E);

Kmi(n, M, U, K);

Maziausias(M, n, MM, mm);

Didziausias(M, n, MD, md);

Maziausias(U, n, UM, um);

Didziausias(U, n, UD, ud);

Raso(K, n, M, MM, mm, MD, md, UM, um, UD, ud, E, U);

return 0;

}

void Skaito(int &a, int &b, int &r, int &n, double M[], double U[], int E[])

{

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

df >> a >> b >> r >> n;

for (int i = 0; i < n; i++)df >> M[i] >> U[i] >> E[i];

df.close();

}

void Kmi(int n, double M[], double U[], double K[])

{

for(int i = 0; i < n; i++)K[i] = M[i] / pow(U[i], 2);

}

void Maziausias(double A[], int n, int M[], int &ind)

{

int m = 0;

M[0] = 0;

ind++;

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

if (A[m] > A[i]){

ind = 0;

m = i;

M[ind] = i;

ind++;

}

else if(A[m] == A[i]){

M[ind] = i;

ind++;

}

}

}

void Didziausias(double A[], int n, int M[], int &ind)

{

int m = 0;

M[0] = 0;

ind++;

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

if (A[m] < A[i]){

ind = 0;

m = i;

M[ind] = i;

ind++;

}

else if(A[m] == A[i]){

M[ind] = i;

ind++;

}

}

}

void Raso(double K[], int &n, double M[], int MM[], int mm, int MD[], int md, int UM[], int um, int UD[], int ud, int E[], double U[])

{

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

for (int i = 0; i < n; i++)rf << i + 1 << " " << fixed << setprecision(0) << K[i] << endl;

for (int i = 0; i < ud; i++)rf << fixed << setprecision(2) << UD[i] + 1 << " " << M[UD[i]] << " " << U[UD[i]] << " " << E[UD[i]] << " " << fixed << setprecision(0) << K[UD[i]] << endl;

for (int i = 0; i < um; i++)rf << fixed << setprecision(2) << UM[i] + 1 << " " << M[UM[i]] << " " << U[UM[i]] << " " << E[UM[i]] << " " << fixed << setprecision(0) << K[UM[i]] << endl;

for (int i = 0; i < mm; i++)rf << fixed << setprecision(2) << MM[i] + 1 << " " << M[MM[i]] << " " << U[MM[i]] << " " << E[MM[i]] << " " << fixed << setprecision(0) << K[MM[i]] << endl;

for (int i = 0; i < md; i++)rf << fixed << setprecision(2) << MD[i] + 1 << " " << M[MD[i]] << " " << U[MD[i]] << " " << E[MD[i]] << " " << fixed << setprecision(0) << K[MD[i]] << endl;

rf.close();

}