1. **Užduotis. Civilinio karo simuliavimas**

Kovose gali dalyvauti 5 tipo kareiviai, kuris kiekvienas turi savo puolimo bei gynybos taškus.

Programos užduotis - pasakyti ar besiginanti pusė (gynėjas) laimėjo karą ir kiek gynėjas prarado kievieno tipo kareivių.

**Karių lentelė:**

|  |  |  |
| --- | --- | --- |
| **Kario pavadinimas** | **Puolimo taškai** | **Gynybos taškai** |
| Pretorionas | 20 | 64 |
| Legionierius | 54 | 54 |
| Lankininkas | 44 | 64 |
| Barbaras | 82 | 12 |
| Centurionas | 137 | 62 |

**Skaičiavimų paaiškinimas**:

Laimėtojas nustatomas lyginant puolimo stiprumą (PS) ir gynybos tvirtumą (GT).

Jeigu PS yra didesnis arba lygus GT, laikoma, kad puolėjas laimėjo kovą. Jeigu GT yra didesnis už PS – laimėjo gynėjas.

**PS yra visų kareivių puolimų taškų suma:**

|  |  |
| --- | --- |
| Duomenų failas | Skaičiavimas |
| 10 | 10 (kareivių skaičius iš duomenų failo) \* 20 ( kareivio puolimo taškai) = 200 |
| 0 | 0 \* 54 = 0 |
| 0 | 0 \* 44 = 0 |
| 500 | 500 \* 82 = 41000 |
| 100 | 100 \* 137 = 13700 |
| **BPP** | **54900** |

**GT yra visų kareivių gynimo taškų suma:**

|  |  |
| --- | --- |
| Duomenų failas | Skaičiavimas |
| 10 | 10 (kareivių skaičius iš duomenų failo) \* 64 ( kareivio gynimo taškai) = 640 |
| 5 | 5 \* 54 = 270 |
| 30 | 30 \* 64 = 1920 |
| 3000 | 3000 \* 12 = 36000 |
| 0 | 0 \* 62 = 0 |
| **BPP** | **38830** |

Siekiant nustatyti gynėjo prarastų karių skaičių, reikia apskaičiuoti santykį efektyvumą (SE).

**Santykinis efektvumas skaičiuojamas taip:**

Jeigu PS yra lygus arba didesnis už GT:

SE = GT / PS.

Jeigu GT didesnis už PS:

SE = PS / GT.

SE yra suapvalinamas iki 4 skaičių po kablelio.

**Gynėjo aukų skaičiavimas:**

Jeigu gynėjas pralaimėjo, tai aukos (per kario tipą) skaičiuojamos taip:

KarioAukos = KariųSkaičius \* (1 - SE)

Jeigu gynėjas laimėjo, tai aukos (per kario tipą) skaičiuojamos taip:

KarioAukos = KariųSkaičius \* SE

**Įvesties paaiškinimas:**

Duomenų failas turi 10 eilučių, kuriose nuo 1 iki 5 eilutės yra parašyti puolėjo karių skaičius, o nuo 6 iki 10 gynėjo karių skaičius.

|  |  |
| --- | --- |
| Duomenų failas (data\_input.txt) | Paaiškinimas |
| 1 (pirma puolėjo eilutė) | Puolėjo pretorionų skaičius |
| 2 | Puolėjo legionierių skaičius |
| 3 | Puolėjo lankininkų skaičius |
| 4 | Puolėjo barbarų skaičius |
| 5 (paskutinė puolėjo eilutė) | Puolėjo centurionų skaičius |
| 6 (pirma gynėjo eilutė) | Gynėjo pretorionų skaičius |
| 7 | Gynėjo legionierių skaičius |
| 8 | Gynėjo lankininkų skaičius |
| 9 | Gynėjo barbarų skaičius |
| 10 (paskutinė gynėjo eilutė) | Gynėjo centurionų skaičius |

**Išvesties paaiškinimas:**

|  |  |
| --- | --- |
| Išvesties failas (data\_output.txt) | Paaiškinimas |
| true | Gynėjas laimėjo |
| 41 | Gynėjas prarado 41 pretorioną |
| 0 | Gynėjas neprarado legionierių |
| 10 | Gynėjas prarado 10 lankininkų |
| 0 | Gynėjas neprarado barbarų |
| 0 | Gynėjas neprarado centurionų |

|  |  |
| --- | --- |
| data\_input.txt | data\_output.txt |
| 0  0  0  0  5000  500  100  0  0  0 | false  437  874  0  0  0 |

1. **Reikalavimai**

* Nenaudoti masyvų.
* Apvalinti pajėgumų santykį iki 4 skaičiaus po kableliu.
* Skaičiuojant gynėjo aukų skaičių, reikšmę apvalinti naudojant **floor** funkciją.
* Naudokite įvesties failo vardą **data\_input.txt**, o išvesties **data\_output.txt.**
* Skaičių tipų konversijai naudoti **static\_cast <>()**.
* Naudoti **void** funkciją santikinio efektyvumui (**SE**) skaičiavimui, kuri rezultatą grąžintų per parametrus.

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

#include <iostream>

#include <iomanip>

#include <cmath>

#include <fstream>

using namespace std;

void Santykis(double pt, double gt, double &SE);

int main(){

double P, La, Le, B, C, P1, La1, Le1, B1, C1, pt, gt;

double SE;

ifstream df ("data\_input.txt");

ofstream rf ("data\_output.txt");

df >> P >> Le >> La >> B >> C >> P1 >> Le1 >> La1 >> B1 >> C1;

pt = P \* 20 + Le \* 54 + La \* 44 + B \* 82 + C \* 137;

gt = P1 \* 64 + Le1 \* 54 + La1 \* 64 + B1 \* 12 + C1 \* 62;

Santykis(pt, gt, SE);

if (pt >= gt){

rf << "false" << endl;

SE = 1 - SE;

}

else{

rf << "true" << endl;

}

P1 = P1 \* SE;

Le1 = Le1 \* SE;

La1 = La1 \* SE;

B1 = B1 \* SE;

C1 = C1 \* SE;

rf << floor(P1) << endl << floor(Le1) << endl << floor(La1) << endl << floor(B1) << endl << floor(C1) << endl;

df.close();

rf.close();

return 0;

}

void Santykis(double pt, double gt, double &SE)

{

if (pt >= gt)SE = gt / pt;

else SE = pt / gt;

SE = round(SE \* 10000) / 10000;

}