**Implementare**

**Secvențial**

* Se definește o structură de date pentru o **listă dublu înlănțuită**, în care fiecare nod conține un **Participant**. Structura **Participant** are trei proprietăți: id, punctaj și țară.
* La **inserarea unui element în listă**, se verifică hash-ul participantului:
  + Dacă hash-ul este prezent într-un **vector de hash-uri** (care reține participanții eliminați), nu se efectuează nicio acțiune, ceea ce indică faptul că participantul a fost eliminat din concurs.
  + Dacă participantul a fost deja adăugat în listă, se incrementează scorul său, urmat de **resortarea listei**.
  + Dacă este prima sa apariție, participantul este adăugat în listă și lista este **sortată**.
* La introducerea unui participant cu **punctajul -1**, acesta este tratat ca **eliminat**:
  + Hash-ul său este adăugat în vectorul de hash-uri pentru participanți eliminați.
  + Dacă participantul există deja în listă, acesta este **eliminat**.

**Paralel**

* Structurile descrise în secțiunea secvențială sunt completate cu mecanisme de **sincronizare** pentru a gestiona execuția concurentă.
* Se introduce o structură suplimentară pentru implementarea unei **cozi cu blocări mutuale**. Blocările sunt aplicate în următoarele cazuri:
  + La **adăugarea** unui element în coadă (push).
  + La **extragerea** unui element din coadă (pop).
  + La interogarea pentru a verifica dacă **coada este goală** sau dacă s-au încheiat citirile din fișiere.
* Structura pentru lista dublu înlănțuită este extinsă cu **blocări mutuale** pentru operații care implică:
  + **Adăugarea** unui element în listă.
  + **Ștergerea unui nod** sau **sortarea listei**, propagând blocările necesare.
* Se implementează modelul **producător-consumator**:
  + **Producătorii** citesc participanți din fișiere și îi adaugă în coadă.
  + **Consumatorii** extrag participanți din coadă și îi procesează:
    - Dacă participantul nu este eliminat (conform vectorului de hash-uri), este adăugat în listă sau i se actualizează scorul.
    - Operațiile asupra listei sunt sincronizate prin blocări mutuale pentru a asigura consistența datelor.

Acest model permite rularea simultană a mai multor fire de execuție, cu protejarea resurselor partajate prin excludere mutuală

|  |  |  |
| --- | --- | --- |
| **Nr readers** | **Nr workers** | **Timp executie total** |
| Secvential | - | 186.4129 |
| 1 | 3 | 440.4 |
| 1 | 5 | 400 |
| 1 | 7 | 436 |
| 1 | 15 | 420.4 |
| 2 | 2 | 361.2 |
| 2 | 4 | 412.3 |
| 2 | 6 | 393.6 |
| 2 | 14 | 403.7 |

**Analiza:**

Performanța programului secvențial este semnificativ mai bună, acest lucru fiind explicat de blocările de tip excludere mutuală din implementarea multithreading-ului. Aceste blocări apar la nivelul cozii și al listei pentru fiecare operație – fie că este vorba despre adăugarea de elemente în coadă, extragerea acestora, verificarea dacă coada este goală, identificarea sfârșitului citirilor din fișiere, adăugarea în listă (care include și sortarea listei) sau ștergerea unui nod. Astfel, pe baza datelor obținute, se poate concluziona că programul secvențial este de cel puțin două ori mai rapid decât cel paralel.