Documentatie Laborator 4

**Analiza cerintelor:**

Obiectiv:

· Intelegerea/aprofundarea sablonului “producator-consumator”

· Intelegerea/aprofundarea sincronizarii conditionale

· Intelegerea/aprofundarea excluderii mutuale (granularitatea sectiunilor critice)

Se considera n polinoame reprezentate prin lista de monoame.

Se cere adunarea polinoamelor folosind o implementare multithreading (p threaduri).

Consideratii generale:

- reprezentarea unui polinom in memorie: lista inlantuita (1 nod=1monom) ordonata dupa exponentii monoamelor cu urmatorul INVARIANT (predicat adevarat la orice moment al executiei) de reprezentare:

-monoamele sunt ordonate dupa exponenti

-nu se pasteaza in lista monoame cu coeficient 0;

- nu exista doua noduri (monoame) cu acelasi exponent

- polinoamele se citesc din fisiere – cate un fisier pentru fiecare polinom;

- un fisier contine informatii de tip (coeficient, exponent) pentru fiecare monom al

unui polinom,

- fisierele input se creeaza prin generare de numere aleatoare.

(Conditie: fisierele nu contin monoame cu coeficient egal cu 0 dar nu sunt ordonate dupa exponent!)

Rezolvare:

Se porneste prin crearea unei liste inlantuita - L corespunzatoare unui polinom nul. In final aceasta lista va continue polinomul rezultat.

Metoda A) Implementare secventiala

· Se citeste pe rand din fiecare fisier cate un monom si se adauga in lista rezultat -L (atentie – invariantul trebuie sa ramana adevarat dupa fiecare adaugare de monom).

Metoda B) Implementare paralela – p threaduri

1. Primul thread citeste cate un monom si il adauga intr-o structura de date de tip coada.

(conditie – pentru structura de tip coada NU se admite folosirea unei structuri de date pentru care partea de sincronizare este deja implementata!!!)

2. Celelalte threaduri preiau cate un monom din coada si il aduna la polinomul reprezentat in lista L.

Se continua operatiile 1., 2. pana cand toate monoamele, din toate fisierele, sunt adunate la lista L.

3. Primul thread scrie rezultatul obtinut in lista L intr-un fisier rezultat

(conditie: fisierul nu contine monoame cu coefficient egal cu 0)

Productor-consumator

Sincronizare la nivel de lista!!!

Limbaj: la alegere intre Java si C++

Analiza timpului de executie pentru urmatoarele cazuri:

1) 10 polinoame fiecare cu gradul maxim 1000 si cu maxim 50 monoame

a. p = 4, 6, 8

b. secvential

2) 5 polinoame fiecare cu gradul maxim 10000 si cu maxim 100 monoame

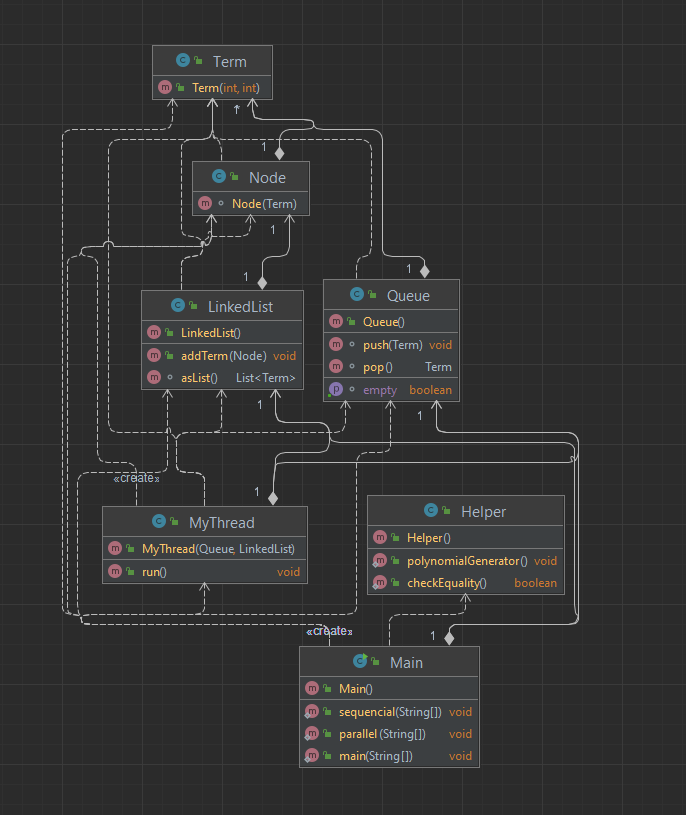
a. p = 4, 6, 8

b. secvential

Analiza: raport Tsecvential/Tparalel

**Proiectare:**

**Java:**

****

**main(String[])-** functia principala de intrare in program

**sequencial(String[])** – functia de calculare secventiala a adunarii polinoamelor

**parallel(String[])** – functia de calculare paralela a adunarii polinoamelor

**Helper –** o clasa ce contine functii ajutatoare

**polynomialGenerator()** – o functie pentru generarea aleatoare de polinoame

**checkEquality()** – o functie pentru verificarea egalitatii fisierelor calculului secvential si paralel

**MyThread** – clasa mea ce extinde Thread

**run()** – functie suprascrisa din Thread in care se ia un monom din coada si se adauga la rezultat (lista inlantuita)

**Queue –** clasa pentru coada

**push(Term)** – adauga un monom in coada

**pop() –** ia un monom din coada

**LinkedList** – o clasa pentru lista inlantuita de monoame, ordonata dupa exponenti

**addTerm(Node)** – adaugarea unui monom la lista daca nu exista termen cu gradul exponentului sau adunarea la coeficientul corespunzator exponentului

**Node** – clasa ce contine monomul si referinta spre nodul urmator

**Term** – monom ce contine exponentul si coeficientul

**Detalii de implementare:**

Se citesc monoamele din fisier si se adauga in coada. La varianta secventiala, se iau pe rand fisierele si se adauga monoamele in lista inlantuita. Daca nu exista deja termen cu acel exponent sau se aduna la coeficient daca exista deja. Apoi se face scrierea rezultatului in fisierul “outputSeq.txt”.

La varianta paralela, primul thread citeste pe rand cate un fisier si adauga monoamele in coada. Restul threadurilor iau din coada cate un monom si il adauga in lista inlantuita. Acesti pasi se repeta pana cand un mai exista fisiere de citit. La final, se scrie rezultatul in fisierul “outputPar.txt”.

**Cazuri de testare:**

| **Input** | **Tip** | **P threads** | **Timp** |
| --- | --- | --- | --- |
| polinoame=10, max exponent=1000, monoame=50 | secvential | 1 | 41.7963 |
| paralel | 4 | 56.90 |
| 6 | 50.89 |
| 8 | 53.97 |
| polinoame=5, max exponent=10000, monoame=100 | secvential | 1 | 49.919 |
| paralel | 4 | 43.81 |
| 6 | 45.633 |
| 8 | 44.58 |

**Analiza rezultatelor:**

Secvential vs Paralel –

In cazul in care exponentul maxim este mai mic, valoarea secventiala este mai buna decat cea paralela Cea paralela se imbunatateste cu cresterea numarului de threaduri.

Iar in cazul in care exponentul maxim este mai mare, valoarea secventiala este mai putin buna decat cea paralela. Cea paralela tinde sa se mentina si cu cresterea thread-urilor.