**Documentație**

**C++ Static:**

**Analiza Cerințelor**

**- Tema:** Considerand ca se da o matrice F(n,m) si o matrice de convolutie C(k,k) se cere sa se calculeze matricea V(n,m) rezultata in urma aplicarii convolutiei cu matricea de convolutie C pe matricea F.

A) Program secvential

B) Program paralel: folositi p threaduri pentru calcul.

**- Cerințe:**

- Citirea unei matrice și a unei matrice de convoluție dintr-un fișier.

- Implementarea a patru metode de convoluție: secvențială, pe linii, pe coloane și vectorizată.

- Utilizarea unui număr variabil de thread-uri pentru metodele paralele.

- Măsurarea și afișarea timpului de execuție pentru fiecare metodă.

Constrângeri:

- Programul trebuie să ruleze corect pentru orice dimensiune a matricei și a matricei de convoluție.

**Proiectare**

**- Structuri de Date:**

- `int mat[MAX\_N][MAX\_M]`: Matricea principală.

- `int convMat[MAX\_N][MAX\_M]`: Matricea de convoluție.

- `int newMat[MAX\_N][MAX\_M]`: Matricea rezultată după aplicarea convoluției.

- **Partiționare pe Threaduri:**

- Fiecare metodă paralelă împarte sarcina între thread-uri în mod diferit:

- `linii()`: Împarte matricea pe rânduri.

- `coloane()`: Împarte matricea pe coloane.

- `vectorizare()`: Împarte matricea pe elemente.

- **Clase:**

- `ThreadLinii`: Clasă pentru procesarea pe rânduri.

- `ThreadColoane`: Clasă pentru procesarea pe coloane.

- `ThreadVectorizare`: Clasă pentru procesarea pe elemente.

- **Funcții:**

- `readInput()`: Citește datele din fișier.

- `secvential()`: Realizează convoluția secvențial.

- `linii()`: Realizează convoluția pe rânduri folosind thread-uri.

- `coloane()`: Realizează convoluția pe coloane folosind thread-uri.

- `vectorizare()`: Realizează convoluția pe elemente folosind thread-uri.

- `main()`: Punctul de intrare al programului, gestionează argumentele și apelează funcțiile corespunzătoare.

**C++ Dinamic:**

**Analiza Cerințelor**

- **Tema**: Considerand ca se da o matrice F(n,m) si o matrice de convolutie C(k,k) se cere sa se calculeze matricea V(n,m) rezultata in urma aplicarii convolutiei cu matricea de convolutie C pe matricea F.

A) Program secvential

B) Program paralel: folositi p threaduri pentru calcul.

- **Cerințe:**

- Citirea unei matrice și a unei matrice de convoluție dintr-un fișier.

- Implementarea a patru metode de convoluție: secvențială, pe linii, pe coloane și vectorizată.

- Utilizarea unui număr variabil de thread-uri pentru metodele paralele.

- Măsurarea și afișarea timpului de execuție pentru fiecare metodă.

**- Constrângeri:**

- Programul trebuie să ruleze corect pentru orice dimensiune a matricei și a matricei de convoluție.

**Proiectare**

**- Structuri de Date:**

- `int \*\*mat`: Matricea principală.

- `int \*\*convMat`: Matricea de convoluție.

- `int \*\*newMat`: Matricea rezultată după aplicarea convoluției.

- **Partiționare pe Threaduri:**

- Fiecare metodă paralelă împarte sarcina între thread-uri în mod diferit:

- `linii()`: Împarte matricea pe rânduri.

- `coloane()`: Împarte matricea pe coloane.

- `vectorizare()`: Împarte matricea pe elemente.

- **Clase:**

- `ThreadLinii`: Clasă pentru procesarea pe rânduri.

- `ThreadColoane`: Clasă pentru procesarea pe coloane.

- `ThreadVectorizare`: Clasă pentru procesarea pe elemente.

- **Funcții:**

- `readInput()`: Citește datele din fișier.

- `secvential()`: Realizează convoluția secvențial.

- `linii()`: Realizează convoluția pe rânduri folosind thread-uri.

- `coloane()`: Realizează convoluția pe coloane folosind thread-uri.

- `vectorizare()`: Realizează convoluția pe elemente folosind thread-uri.

- `main()`: Punctul de intrare al programului, gestionează argumentele și apelează funcțiile corespunzătoare.

-**Cazuri de testare C++:**

**-Analiza performanțelor:**

-Putem observa că numărul de Threaduri are o influență mare asupra timpului de execuție, diferența fiind sesizată în cazurile de testare cu dimensiuni mari de matrici. (i.e. 10000x10000, 2 Threads – împărțire pe linii – 19437ms, iar cu 16 Threads – împărțire pe linii – 11271ms, avem o diferență de 8166ms, adică aproximativ 8 secunde).

- Diferența dintre alocarea dinamică și statică a matricilor este de asemenea sesizabilă (i.e 10000x10000, static – secvențial – 26490ms, iar dinamic – secvențial – 29283ms, avem o diferență de 2793ms, adică aproximativ 3 secunde, programul cu alocare statică rulând mai rapid, decât cel cu alocare dinamică).