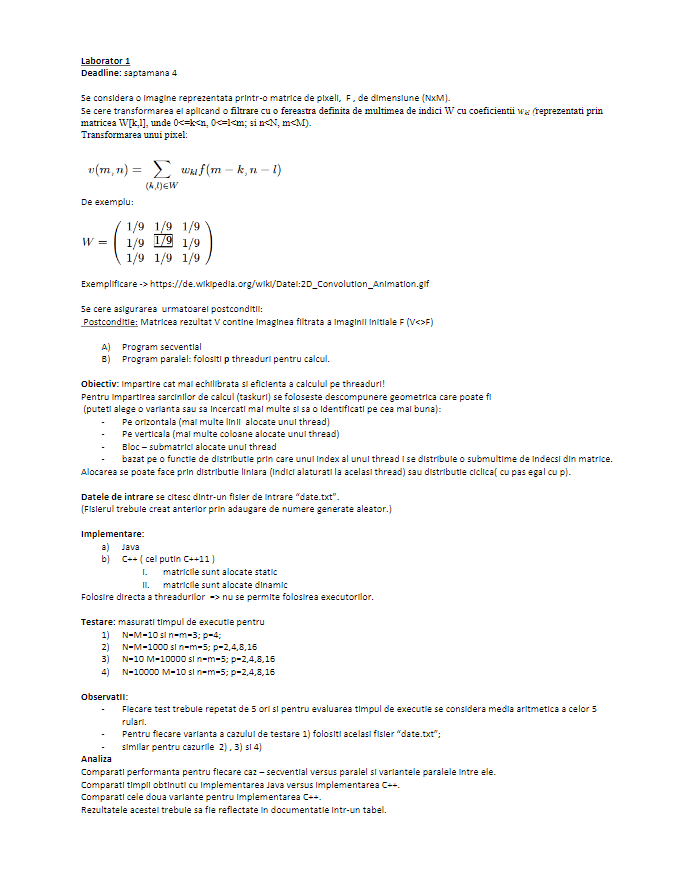
# Documentație Laborator 1



Atât în Java cât și în C++ folosim clasa Utils creeată în Lab 0 din care utilizăm următoarele funcții:

* createFile cu parametri:
  + fname – numele fișierului: string
  + size – numărul de elemente ce vor fi generate: int
  + min – valoarea minimă generată: int
  + max – valoarea maximă generată: int
* ~~checkEquality cu parametri:~~
  + ~~fname1 – numele primului fișier: string~~
  + ~~fname2 – numele celui de-al doilea fișier: string~~
  + ~~type – tipul datelor comparate (int / double): string~~

În rezolvarea problemei s-au folosit următoarele funcții:

1. foo(i: integer, j: integer) – calculează valoarea nouă a pixelului de pe poziția i, j
2. parallel() – realizează execuția paralelă
3. sequencial() – realizează execuția secvențială

## Observații generale

* Pentru distribuirea datelor în threaduri s-a folosit metoda cu saritul peste pas egal cu numărul de threaduri.
* Fiecare caz s-a rulat de 5 ori și s-a făcut media aritmetică a timpilor obținuți

## Observații personale în urma analizei pe tabel

* În Java, apelul secvențial pornește cu un timp mare și scade pe parcurs la alte rulări, pe de altă parte la C++ se întâmplă același lucru la execuția paralelă.
* Cu cât mai multe date, cu atât mai eficientă este execuția cu threaduri față de cea secvențială (în Java, timpul pentru date mari este apropiat, raport aproape 1, pe când în C++ se vede diferența între execuție paralelă și secvențială)
* C++: execuția statică și dinamică au aproximativ același timp de rulare
* Java este uneori mai rapid decât C++ chiar și la date mari, de exemplu Java este mai bun la 1000x1000 pe cand C++ este mai bun pe 10x10000 și 10000x10.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| N | M | n | m | p | Time C++ Static | Time C++ Dinamic | Time Java |
| 10 | 10 | 3 | 3 | Sequencial | 7580 | 7980 | 65200 |
| 10 | 10 | 3 | 3 | 4 | 5336260 | 5495400 | 511980 |
| 1000 | 1000 | 5 | 5 | Sequencial | 83079500 | 98425300 | 29406300 |
| 1000 | 1000 | 5 | 5 | 2 | 54054900 | 62375600 | 25831160 |
| 1000 | 1000 | 5 | 5 | 4 | 38220400 | 42443700 | 26476040 |
| 1000 | 1000 | 5 | 5 | 8 | 32769700 | 35363000 | 23166600 |
| 1000 | 1000 | 5 | 5 | 16 | 34668700 | 41720600 | 26372920 |
| 10 | 10000 | 5 | 5 | Sequencial | 10122700 | 10925500 | 4412180 |
| 10 | 10000 | 5 | 5 | 2 | 11862000 | 11211300 | 6189660 |
| 10 | 10000 | 5 | 5 | 4 | 14838500 | 11212400 | 6192380 |
| 10 | 10000 | 5 | 5 | 8 | 13859700 | 13728000 | 6652460 |
| 10 | 10000 | 5 | 5 | 16 | 14819900 | 16110800 | 9492100 |
| 10000 | 10 | 5 | 5 | Sequencial | 10597200 | 12209400 | 4085080 |
| 10000 | 10 | 5 | 5 | 2 | 11600700 | 11449400 | 5567860 |
| 10000 | 10 | 5 | 5 | 4 | 13059300 | 10053500 | 7277060 |
| 10000 | 10 | 5 | 5 | 8 | 12810300 | 11661900 | 7182180 |
| 10000 | 10 | 5 | 5 | 16 | 16383200 | 17951500 | 8390580 |

|  |  |
| --- | --- |
| Raport Java/C++ | Raport C++ Static / Dinamic |
| 8.601583113 | 0.949874687 |
| 0.095943601 | 0.971041234 |
| 0.353953743 | 0.844086835 |
| 0.477868981 | 0.866603287 |
| 0.692720118 | 0.900496422 |
| 0.706951849 | 0.92666629 |
| 0.760712689 | 0.830973188 |
| 0.435869877 | 0.926520525 |
| 0.521805766 | 1.058039656 |
| 0.417318462 | 1.323400878 |
| 0.479985858 | 1.009593531 |
| 0.640496899 | 0.919873625 |
| 0.385486732 | 0.867954199 |
| 0.479958968 | 1.013214666 |
| 0.557232011 | 1.298980455 |
| 0.560656659 | 1.09847452 |
| 0.512145368 | 0.912636827 |