## Proiect Laborator

### Programarea calculatoarelor și limbaje de programare

### C

### Mijache Nicușor

### Mai 2024

## 1. Introducere

### 1.1 Enunțul problemei

Imaginați-vă un scenariu în care trei angajați trebuie să caute informații specifice în rafturi pline de cărți. Pentru a asigura o repartizare corectă și eficientă a muncii, cărțile trebuie distribuite celor trei angajați fără a fi necesară mutarea sau împărțirea lor în grămezi separate. Soluția cea mai simplă este împărțirea raftului în trei secțiuni distincte, fiecare secțiune fiind alocată unui angajat.

Dar cum putem asigura că împărțirea este echitabilă? Dacă fiecare carte ar avea aceeași dimensiune, soluția ar fi simplă: împărțim raftul în secțiuni egale. De exemplu:

100 100 100 – 100 100 100 – 100 100 100

Astfel, fiecare persoană ar avea de gestionat 300 de pagini.

Provocarea apare atunci când cărțile variază în dimensiune. Utilizând aceeași metodă pentru un raft cu cărți de dimensiuni diferite, distribuția devine inegală:

100 200 300 – 400 500 600 – 700 800 900

În acest caz, prima secțiune are 600 de pagini, iar ultima conține 2400 de pagini, ceea ce nu este echitabil.

Pentru a obține o încărcătură de muncă mai echilibrată, o împărțire mai bună ar putea arăta astfel:

100 200 300 400 500 – 600 700 – 800 900

În acest aranjament, cea mai grea sarcină este de 1700 de pagini, iar cea mai ușoară de 1300 de pagini. Acest exemplu ilustrează necesitatea unei abordări strategice pentru distribuirea echilibrată a sarcinilor atunci când cărțile au dimensiuni variabile.

### 1.2 Descrierea problemei

Trei angajați trebuie să caute informații într-un raft plin de cărți. Pentru a împărți munca echitabil, raftul trebuie divizat în trei secțiuni, câte una pentru fiecare angajat. Problema apare din cauza dimensiunilor variabile ale cărților, care fac împărțirea inegală. De exemplu, împărțind cărțile după număr, putem avea secțiuni cu 600, 1500 și 2400 de pagini. Soluția optimă implică o strategie de împărțire care să echilibreze numărul total de pagini în fiecare secțiune pentru a distribui munca uniform.

## 2. Algoritmi

### 2.1 Descrierea algoritmilor folosiți

1. **Sortarea cărților în ordine descrescătoare**:
   * Funcția qsort este folosită pentru a sorta array-ul de cărți în ordine descrescătoare după numărul de pagini.
   * Funcția de comparație cmp returnează diferența dintre două valori de tip int în ordine descrescătoare.
2. **Distribuirea cărților către angajați folosind o abordare greedy**:
   * Algoritmul greedy presupune alocarea fiecărei cărți angajatului care are în acel moment cel mai mic număr total de pagini.
   * Gasim indexul gramezii cu cel mai putin numar de pagini.
   * Adaugam cartea urmatoare in gramada.
3. **Procesul principal**:
   * Programul citește numărul de cărți și numărul de pagini pentru fiecare carte.
   * Cărțile sunt sortate în ordine descrescătoare.
   * Fiecare carte este alocată gramezii cu cele mai puține pagini folosind abordarea greedy.
   * Rezultatele finale sunt afișate, iar memoria alocată este eliberată.

### 2.2 Pseudocod

functie cmp(a, b)  
 return (b - a)  
  
functie main()  
 deschide fisierul "date.in" pentru citire si atribuie-l lui fin  
 deschide fisierul "date.out" pentru scriere si atribuie-l lui fout  
  
 int n = 0  
 aloca memorie pentru un vector v de dimensiune n  
 aloca memorie pentru un vector gramada de dimensiune n  
  
 citeste n din fin  
 pentru i de la 0 la n-1  
 citeste v[i] din fin  
  
 qsort(v, n, sizeof(int), cmp)  
  
 int gr[3] = {0, 0, 0}  
  
 pentru i de la 0 la n-1  
 int min = 0  
 daca gr[1] < gr[min]  
 min = 1  
 daca gr[2] < gr[min]  
 min = 2  
  
 gr[min] += v[i]  
 gramada[i] = min + 1  
  
 scrie "Gramada 1:" in fout  
 pentru i de la 0 la n-1  
 daca gramada[i] == 1  
 scrie v[i] in fout  
 scrie "=> Suma: " + gr[0] + "\n" in fout  
  
 scrie "Gramada 2:" in fout  
 pentru i de la 0 la n-1  
 daca gramada[i] == 2  
 scrie v[i] in fout  
 scrie "=> Suma: " + gr[1] + "\n" in fout  
  
 scrie "Gramada 3:" in fout  
 pentru i de la 0 la n-1  
 daca gramada[i] == 3  
 scrie v[i] in fout  
 scrie "=> Suma: " + gr[2] + "\n" in fout  
  
 elibereaza memorie pentru v  
 elibereaza memorie pentru gramada  
  
 return 0

## 3. Descrierea aplicației

Această aplicație este un program simplu scris în limbajul de programare C, care demonstrează o abordare de distribuire echitabilă a cărților angajaților.

### 3.1 Funcționalități principale

1. **Citirea și sortarea cărților**: Utilizatorul introduce numărul de cărți și numărul de pagini pentru fiecare carte. Cărțile sunt sortate în ordine descrescătoare în funcție de numărul de pagini.
2. **Distribuirea cărților**: Cărțile sunt distribuite angajaților folosind o abordare “greedy” care încearcă să mențină un echilibru între numărul total de pagini gestionate de fiecare angajat.
3. **Afișarea rezultatelor**: La final, se afișează numărul total de pagini gestionate de fiecare angajat, împreună cu lista cărților alocate fiecăruia.

### 3.3 Observații

* Aplicația are ca scop ilustrarea unui algoritm simplu pentru distribuirea echitabilă a cărților între angajați.
* Nu există gestionarea erorilor sau validarea intrării în acest cod simplu.
* Utilizarea limbajului C face dificilă gestionarea memoriei și prelucrarea datelor în comparație cu alte limbaje de programare moderne.

## 4. Concluzii

În concluzie, acest program reprezintă

o demonstrație simplă a modului în care pot fi distribuite echitabil resursele între diferiți angajați. Deși eficient în gestionarea resurselor într-un mediu simplu, acesta prezintă limite în ceea ce privește gestionarea erorilor și adaptabilitatea la scenarii mai complexe. Cu toate acestea, oferă o platformă de pornire utilă pentru înțelegerea conceptelor de bază și pentru dezvoltarea ulterioară a unor aplicații mai complexe de gestionare a resurselor.