L6: Tabela de dispersie

Să se implementeze în C++ un anumit **container de date** (*TAD*) folosind o anumită reprezentare (indicată) și o **tabelă de dispersie** (TD) ca *structură de date*, cu o anumită metodă (indicată) pentru rezolvarea coliziunilor:

- Liste independente
- Liste întrepătrunse
- Adresare deschisă

Interfețele pentru TAD-uri și testele sunt disponibile în secțiunea Files/Class materials/Laborator/Teme laborator/ Interfete TAD (proiecte C++) si teste.

1. **TAD Matrice** - reprezentare sub forma unei matrice rare (triplete de forma <*linie*, *coloană*, *valoare*> (*valoare*$\neq 0$)), memorate folosind o TD cu rezolvare coliziuni prin liste independente.
2. **TAD Matrice** - reprezentare sub forma unei matrice rare (triplete de forma <*linie*, *coloană*, *valoare*> (*valoare*$\neq 0$)), memorate folosind o TD cu rezolvare coliziuni prin liste întrepătrunse.
3. **TAD Matrice** - reprezentare sub forma unei matrice rare (triplete de forma <*linie*, *coloană*, *valoare*> (*valoare*$\neq 0$)), memorate folosind o TD cu rezolvare coliziuni prin adresare deschisă, verificare pătratică.
4. **TAD Colecție** – reprezentare memorând toate elementele, folosind o TD cu rezolvare coliziuni prin liste întrepătrunse.
5. **TAD Colecție** – reprezentare memorând toate elementele, folosind o TD cu rezolvare coliziuni prin adresare deschisă, dispersia dublă.
6. **TAD Colecție** – reprezentare prin perechi de forma (*element*, *frecvență*), folosind o TD cu rezolvare coliziuni prin liste independente.
7. **TAD Colecție** – reprezentare prin perechi de forma (*element*, *frecvență*), folosind o TD cu rezolvare coliziuni prin liste întrepătrunse.
8. **TAD Colecție** – reprezentare prin perechi de forma (*element*, *frecvență*), folosind o TD cu rezolvare coliziuni prin adresare deschisă, verificare pătratică.
9. **TAD Mulțime** – reprezentare folosind o TD cu rezolvare coliziuni prin liste independente.
10. **TAD Mulțime** – reprezentare folosind o TD cu rezolvare coliziuni prin liste întrepătrunse.
11. **TAD Mulțime** – reprezentare folosind o TD cu rezolvare coliziuni prin adresare deschisă, dispersia dublă.
12. **TAD Dicționar** – reprezentare sub forma de perechi (*cheie*, *valoare*), folosind o TD cu rezolvare coliziuni prin liste independente.
13. **TAD Dicționar** – reprezentare sub forma de perechi (*cheie*, *valoare*), folosind o TD cu rezolvare coliziuni prin liste întrepătrunse.

14. **TAD Dicționar** – reprezentare sub forma de perechi (*cheie*, *valoare*), folosind o TD cu rezolvare coliziuni prin adresare deschisă, verificare pătratică.
15. **TAD Dicționar Ordonat** – reprezentare sub forma unei TD cu rezolvare coliziuni prin liste independente.
16. **TAD Dicționar Ordonat** – reprezentare sub forma unei TD cu rezolvare coliziuni prin liste întrepătrunse.
17. **TAD Dicționar Ordonat** – reprezentare sub forma unei TD cu rezolvare coliziuni prin adresare deschisă, dispersia dublă.
18. **TAD MultiDicționar** – memorarea tuturor perechilor de forma (*cheie*, *valoare*), reprezentare sub forma unei TD cu rezolvare coliziuni prin liste independente.
19. **TAD MultiDicționar** – memorarea tuturor perechilor de forma (*cheie*, *valoare*), reprezentare sub forma unei TD cu rezolvare coliziuni prin liste întrepătrunse.
20. **TAD MultiDicționar** – memorarea tuturor perechilor de forma (*cheie*, *valoare*), reprezentare sub forma unei TD cu rezolvare coliziuni prin adresare deschisă, verificare pătratică.
21. **TAD MultiDicționar** – memorarea *cheilor* distincte (fiecare *cheie* va memora un vector dinamic cu *valorile* asociate *cheii*), reprezentare sub forma unei TD cu rezolvare coliziuni prin liste independente. Pentru implementarea vectorului de valori, se va putea folosi *vector* din STL.
22. **TAD MultiDicționar** – memorarea *cheilor* distincte (fiecare *cheie* va memora un vector dinamic cu *valorile* asociate *cheii*), reprezentare sub forma unei TD cu rezolvare coliziuni prin liste întrepătrunse. Pentru implementarea vectorului de valori, se va putea folosi *vector* din STL.
23. **TAD MultiDicționar** – memorarea *cheilor* distincte (fiecare *cheie* va memora un vector dinamic cu *valorile* asociate *cheii*), reprezentare sub forma unei TD cu rezolvare coliziuni prin adresare deschisă, dispersia dublă. Pentru implementarea vectorului de valori, se va putea folosi *vector* din STL.
24. **TAD MultiDicționar Ordonat** – memorarea tuturor perechilor de forma (*cheie*, *valoare*), reprezentare sub forma unei TD cu rezolvare coliziuni prin liste independente.
25. **TAD MultiDicționar Ordonat** – memorarea tuturor perechilor de forma (*cheie*, *valoare*), reprezentare sub forma unei TD cu rezolvare coliziuni prin liste întrepătrunse.
26. **TAD MultiDicționar Ordonat** – memorarea tuturor perechilor de forma (*cheie*, *valoare*), reprezentare sub forma unei TD cu rezolvare coliziuni prin adresare deschisă, verificare pătratică.
27. **TAD MultiDicționar Ordonat**– memorarea *cheilor* distincte (fiecare *cheie* va memora un vector dinamic cu *valorile* asociate *cheii*), reprezentare sub forma unei TD cu rezolvare coliziuni prin liste independente. Pentru implementarea vectorului de valori, se va putea folosi *vector* din STL.
28. **TAD MultiDicționar Ordonat**– memorarea *cheilor* distincte (fiecare *cheie* va memora un vector dinamic cu *valorile* asociate *cheii*), reprezentare sub forma unei TD cu cu rezolvare coliziuni

prin liste întrepătrunse. Pentru implementarea vectorului de valori, se va putea folosi *vector* din STL. Pentru implementarea vectorului de valori, se va putea folosi *vector* din STL.

29. **TAD MultiDicționar Ordonat**– memorarea *cheilor* distincte (fiecare *cheie* va memora un vector dinamic cu *valorile* asociate *cheii*), reprezentare sub forma unei TD cu cu rezolvare coliziuni prin adresare deschisă, dispersia dublă. Pentru implementarea vectorului de valori, se va putea folosi *vector* din STL.

---

30. **TAD Dictionar** – reprezentare folosind o TD cu dispersie Cuckoo (*Cuckoo hashing*)
31. **TAD Dictionar** - reprezentare folosind o TD cu rezolvare coliziuni prin liste independente interconectate (**Linked Hash Table** - Java)