**Structuri de date**

La acest capitol sunt studiate 4 structuri de date:

- Stiva (Stack - de tip LIFO = Last Input First Output)

- Coada (Queue - de tip FIFO = First Input First Output)

- Arbore Binar De Căutare (Binary Search Tree) - este un arbore binar ordonat

- Arbore Binar de tip Heap.

**Stiva**

Este o structură de date organizată asemenea unei stive de obiecte: orice obiect nou îl punem deasupra (peste) cele deja existente.

Orice obiect luăm din stivă, îl luăm de deasupra celor existente.

Operațiile se notează cu

PUSH(valoare) - adaugă "valoare" în vf. stivei

POP(X) - ia valoarea din vf. stivei (o șterge) și o dă variabilei X.

Dacă stiva e goală, X primește NIL

Exercițiile cu stivă constau într-o succesiune de astfel de operații și o întrebare asupra rezultatului final.

De regulă e indicat să vă desenați stiva (modul în care sunt puse elementele)

Ex: să aplicăm următoarele operații pe o stivă:

PUSH(A), PUSH(M), PUSH(E), POP(X),

PUSH(R), POP(X), PUSH(I), POP(X), POP(X),

POP(X), POP(X), PUSH(C), PUSH(A), PUSH(N)

|  |  |
| --- | --- |
| Operația | Stiva |
| PUSH(A) | |  | | --- | | A | |
| PUSH(M) | |  | | --- | | M  A | |
| PUSH(E) | |  | | --- | | E  M  A | |
| POP(X) | |  | | --- | | M  A |   X=E |
| PUSH(R) | |  | | --- | | R  M  A | |
| POP(X) | |  | | --- | | M  A |   X=R |
| PUSH(I) | |  | | --- | | I  M  A | |
| POP(X) | |  | | --- | | M  A |   X=I |
| POP(X) | |  | | --- | | A |   X=M |
| POP(X) | |  | | --- | |  |   X=A |
| POP(X) | |  | | --- | |  |   X=NIL |
| PUSH(C) | |  | | --- | | C | |
| PUSH(A) | |  | | --- | | A  C | |
| PUSH(N) | |  | | --- | | N  A  C | |

**Coada**

Este o structură de date organizată asemenea unei cozi civilizate: avem două capete, unul pe la care se pun elemente în structură și altul de unde ies elementele din structură.

De regulă se bagă elemente prin dreapta și se scot din stânga.

Operațiile se notează cu

PUSH(valoare) - adaugă "valoare" la dreapta cozii

POP(X) - ia valoarea de la începutul cozii (o șterge) și o dă variabilei X.

Dacă coada e goală, X primește NIL

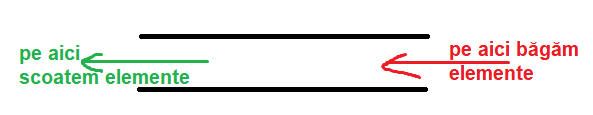
De regulă e indicat să vă desenați coada (modul în care sunt puse elementele)

Ex: să aplicăm următoarele operații pe o coadă:

PUSH(A), PUSH(M), PUSH(E), POP(X),

PUSH(R), POP(X), PUSH(I), POP(X), POP(X),

POP(X), POP(X), PUSH(C), PUSH(A), PUSH(N)



|  |  |
| --- | --- |
| Operația | Coada |
| PUSH(A) | |  | | --- | | A | |
| PUSH(M) | |  |  | | --- | --- | | A | M | |
| PUSH(E) | |  |  |  | | --- | --- | --- | | A | M | E | |
| POP(X) | |  |  | | --- | --- | | M | E |   X=A |
| PUSH(R) | |  |  |  | | --- | --- | --- | | M | E | R | |
| POP(X) | |  |  | | --- | --- | | E | R |   X=M |
| PUSH(I) | |  |  |  | | --- | --- | --- | | E | R | I | |
| POP(X) | |  |  | | --- | --- | | R | I |   X=E |
| POP(X) | |  | | --- | | I |   X=R |
| POP(X) | |  | | --- | |  |   X=I |
| POP(X) | |  | | --- | |  |   X=NIL |
| PUSH(C) | |  | | --- | | C | |
| PUSH(A) | |  |  | | --- | --- | | C | A | |
| PUSH(N) | |  |  |  | | --- | --- | --- | | C | A | N | |

**Arborii binari de căutare**

Sunt arbori binari (orice nod are cel mult doi fii) cu următoarea proprietate: valoarea asociată fiecărui nod al arborelui este mai mare sau egală cu valoarea din fiul său stâng (dacă există) și strict mai mică decât valoarea din fiul său drept.

Cel mai adesea, exercițiile cu arbori binari propun construirea unui arbore binar de căutare plecând de la niște valori.

Construirea se face astfel: cu prima valoare formăm rădăcina arborelui.

Luăm apoi valoare cu valoare și pe fiecare, plecând din rădăcină, o comparăm cu nodul curent. Dacă este mai mică sau egală mergem pe stânga, dacă nu, pe dreapta. În primul loc liber găsit în acest fel punem nodul.

Exemplu: Să construim un arbore binar de căutare din literele: A, M, E, R, I, C, A, N

|  |  |
| --- | --- |
| A |  |
| M |  |
| E |  |
| R |  |
| I |  |
| C |  |
| A |  |
| N |  |

Deci ultimul reprezintă arborele binar de căutare asociat șirului de litere A,M,E,R,I,C,A,N

Acest arbore are adâncimea (sau înălțimea 3)

Se mai poate cere să-l parcurgem în preordine (RSD), inordine (SRD), postordine (SDR):

Hai să vedem ce formă au aceste parcurgeri:

1) Preordine (RSD): A A M E C I R N

2) Inordine (SRD): A A C E I M N R (!!interesting fact: un arbore binar de căutare parcurs în inordine ne dă literele ordonate alfabetic)

3) Postordine (SDR): A C I E N R M A