SDA - Stive și Cozi

Daniel Chiș - 2022, UPB, ACS, An I, Seria AC

Stive - Stacks

Stivă - Stack

O stivă este un tip de date abstract (abstract data type). O stivă permite operații doar la un capăt al ei. La orice moment din timp, putem avea acces doar la elementul din capătul stivei.

Astfel, structura de date merge pe principiul, ultimul intrat primul care iese. Elementul adăugat ultimul, e primul accesat.

Operația de inserție se numește **PUSH**

Operația de scoatere se numește **POP**

Stiva poate fi realizată cu vectori, liste etc. Pot să fie fixe ca dimensiune sau dinamice.

LIFO
LAST IN FIRST OUT

Push

- Pas 1 - verific dacă stiva este plină
- Pas 2 - dacă este plină arunc eroare
- Pas 3 - dacă nu este plină, incrementez **top** să pointeze către următorul spațiu liber
- Pas 4 - adaug elementul către poziția din stivă unde este **top**
- Pas 5 - return success

```
1  begin procedure push: stack, data
2
3      if stack is full
4          return null
5      endif
6
7      top ← top + 1
8      stack[top] ← data
9
10 end procedure
11
```

Pop

- Pas 1 - verific dacă stiva este goală
- Pas 2 - dacă este goală arunc eroare
- Pas 3 - dacă nu este goală, accesez elementul la locul în care **top** pointează
- Pas 4 - decrementăm valoarea lui **top** cu 1
- Pas 5 - return success

```
1  begin procedure pop: stack
2
3      if stack is empty
4          return null
5      endif
6
7      data ← stack[top]
8      top ← top - 1
9      return data
10
11 end procedure
12
```

Cozi - Queues

Cozi - Queues

O coadă este o structură de date abstractă, asemănătoare cu stiva. Spre deosebire de stive, cozile sunt deschise la amândouă capetele. Un capăt e folosit pentru inserție (enqueue) și unul pentru extracție dequeue).

Cozile funcționează pe principiul primul care intră, primul care iese.

Ca și stivele, coada poate fi realizată cu vectori, liste etc. Pot să fie fixe ca dimensiune sau dinamice.

Pentru a realiza operații cu cozile, avem nevoie de doi pointeri care să rețină pozițiile de început (**front**) și sfârșit (**rear**).

FIFO
FIRST IN FIRST OUT

Adăugare - Enqueue

- Pas 1 - verific dacă coada este plină
- Pas 2 - dacă este plină arunc eroare
- Pas 3 - dacă nu este plină, incrementez **rear** să pointeze către următorul spațiu liber
- Pas 4 - adaug elementul către poziția din stivă unde este **rear**
- Pas 5 - return success.

```
1  procedure enqueue(data)
2
3      if queue is full
4          return overflow
5      endif
6
7      rear ← rear + 1
8      queue[rear] ← data
9      return true
10
11 end procedure
12
```

Extracție - Dequeue

- Pas 1 - verific dacă coada este goală
- Pas 2 - dacă este goală arunc eroare
- Pas 3 - dacă nu este goală, accesez elementul la locul în care **front** pointezează
- Pas 4 - incremetăm valoarea lui **front** cu 1
- Pas 5 - return success

```
1  procedure dequeue
2
3      if queue is empty
4          return underflow
5      end if
6
7      data = queue[front]
8      front ← front + 1
9      return true
10
11 end procedure
12
```

Stivă/Cozi - Alte operații

Pe lângă push și pop mai sunt:

- peek() - vizualizează elementul de la capul stivei fără să-l scoată/vizualizează elementul de la front
- isFull() - verifică dacă stiva este plină
- isEmpty() - verifică dacă stiva este goală

Exerciții

Exerciții

1. Implementați o coadă și o stivă la care să folosiți toate operațiile prezentate în laborator (push/pop/enqueue/dequeue/ismempty/isfull/peek) **2p**.
2. Implementați două stive folosind un singur vector. Trebuie să folosiți operațiile de push si pop ca să umpleți stivele și să le goliți. **3.5p**
3. Realizați o coadă folosind 2 stive. **3.5p**

Exerciții FIIR

1. Implementați o coadă și o stivă la care să folosiți toate operațiile prezentate în laborator (push/pop/enqueue/dequeue/ismempty/isfull/peek) .

Extra Puncte FIIR

https://www.codecademy.com/learn/learn-c - pana pe 19 aprilie

Recuperati pana la 25% din laborator