The **ADT** & a Virtual Machine

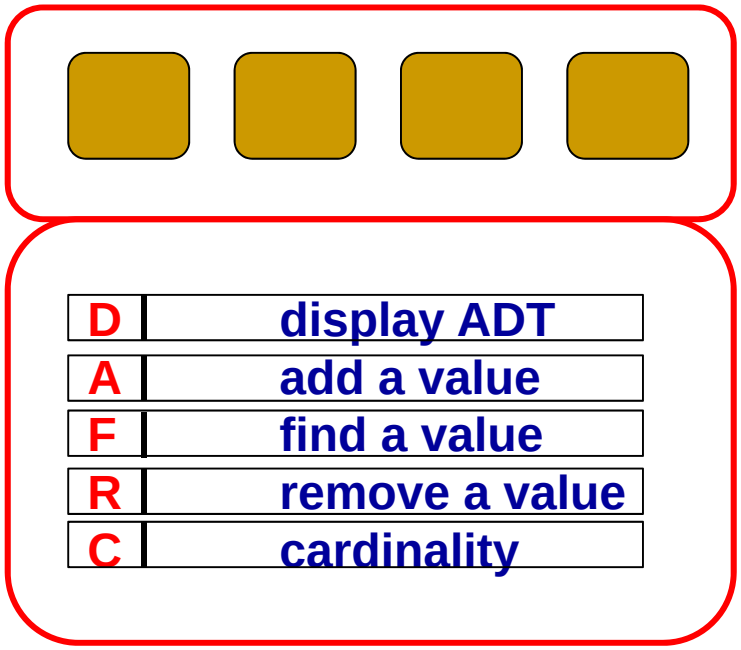
ADT = Abstract Data Type

- UI (user interface) – **menu**
- **ADT = a virtual machine**
- **Menu ➔ User Dialog**

>enter value:

- UI – menu based
  - d – display ADT
  - a – add a value
  - f – find a value
  - r – remove a value
  - n – number of elements (cardinality)

- A sequence
- ADT + operations

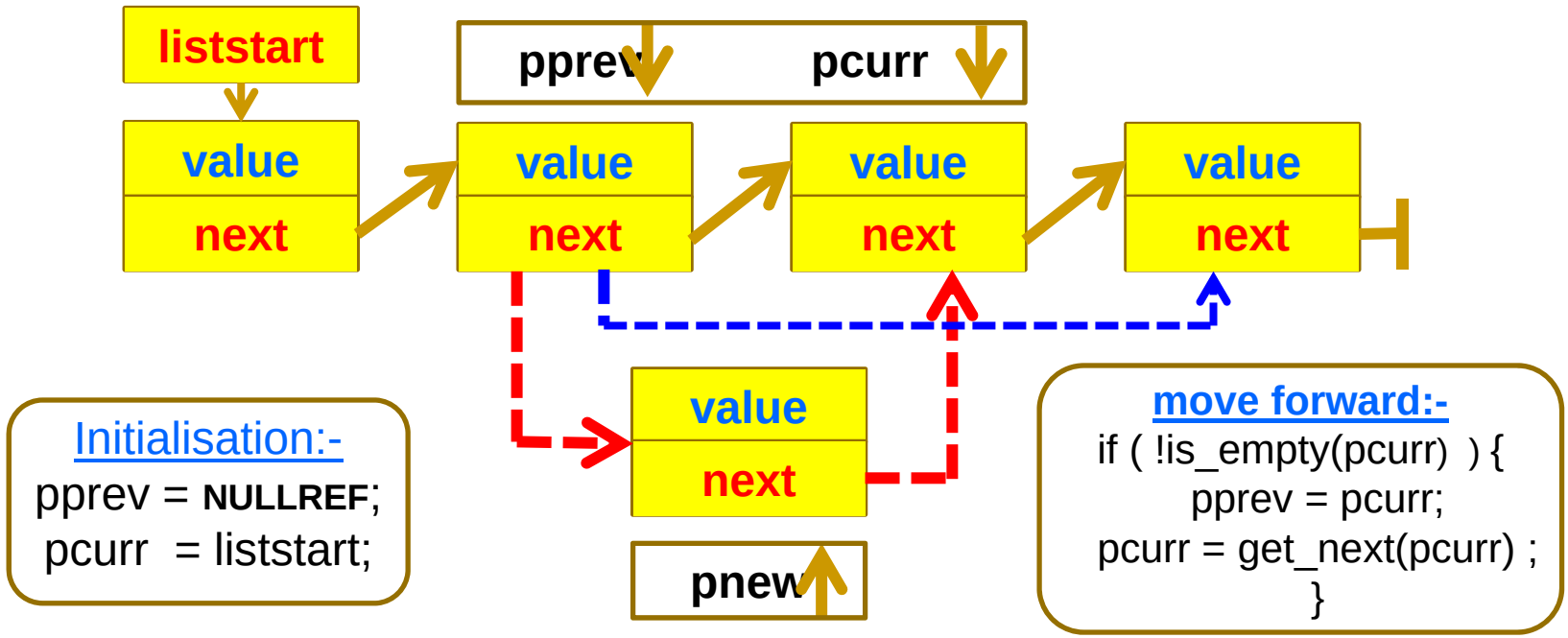


ADT: sequence (linear; ordered)

- **Properties:** a collection of ordered entities
- **Relationships:** successor (E_n, E_{n+1}) (next)
  predecessor (E_{n-1}, E_n) (previous)
- **Operations**
  - As for collections
  - **Sorting & Searching**
- **Implementations**: struct+ptrs (linked lists) / arrays

  NB: the implementation is a sequence hence can use recursion!
- **Used for**: hashing, heaps, implementing graphs

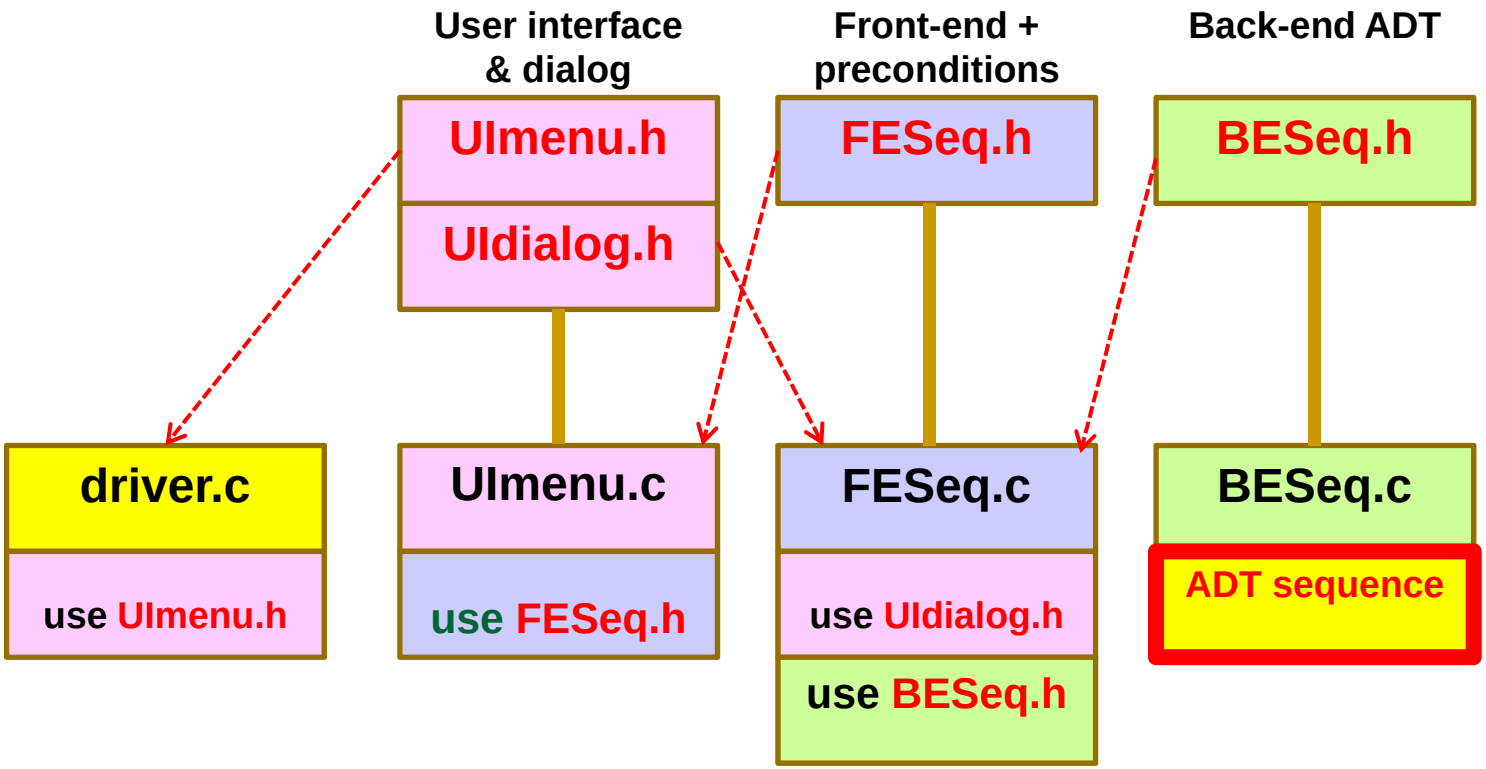
The role of pprev, pcurr, pnew



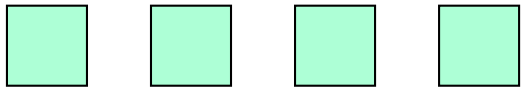
(pprev, pcurr) move as a pair along the list **(used in add/ find /remove)**

pnew is inserted between pprev and pcurr **(used in add)**

ADTseq – implementation UI/FE/BE

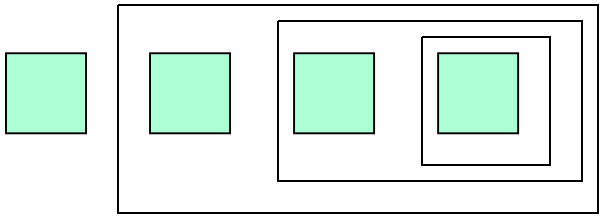


Sequence: Iterative view

- Picture: 
- Position: 1 2 3 4
- Operations: first, next
- Support: pprevious, pcurrent, pnew
  linkin(), unlink()
- Navigation: is_seq_empty(), get_seq_first()
  get_seq_next()
- Collection: add, find, remove, cardinality (size)
  display

Sequence: Recursive view

- Picture:



- Definition: S ::= H T | ¤; H ::= element; T ::= S
- Operations: head(S), tail(S) **deconstruction**
  cons(H,T) **reconstruction**
  is_empty(S)
- Collection: add, find, remove, cardinality (size)
  display