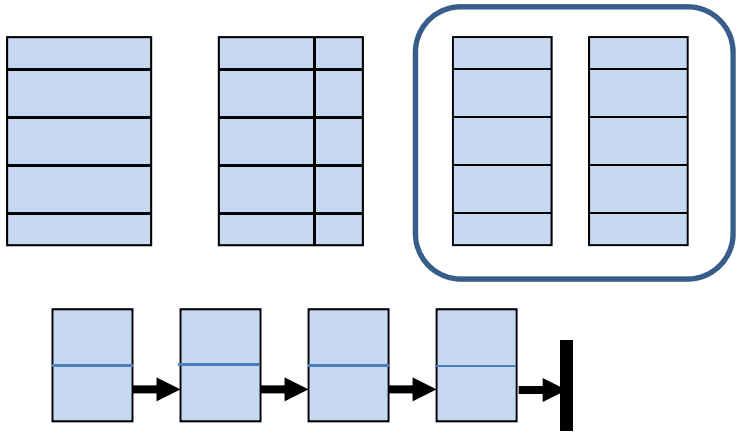
Sequence: iterative view

- **Properties**
  - Collection (values)
  - Ordered (position)
  - {Sorted by value}
  - Duplicate values
- **Attributes**
  - Value
  - Position

- Visualisations & implementations **(value, next ref)**



Sequence - **importance**

- One of the **basic ADTs**
- **Used to represent Sets & Graphs G=(V,E)**
  - List of lists (adjacency list)
  - Array of arrays (adjacency matrix)
- Good intro to the basic operations on a collection (is_empty, add, remove, find, size)
- Good intro to implementation abstraction (attributes & get/set functions) + **RECURSION**

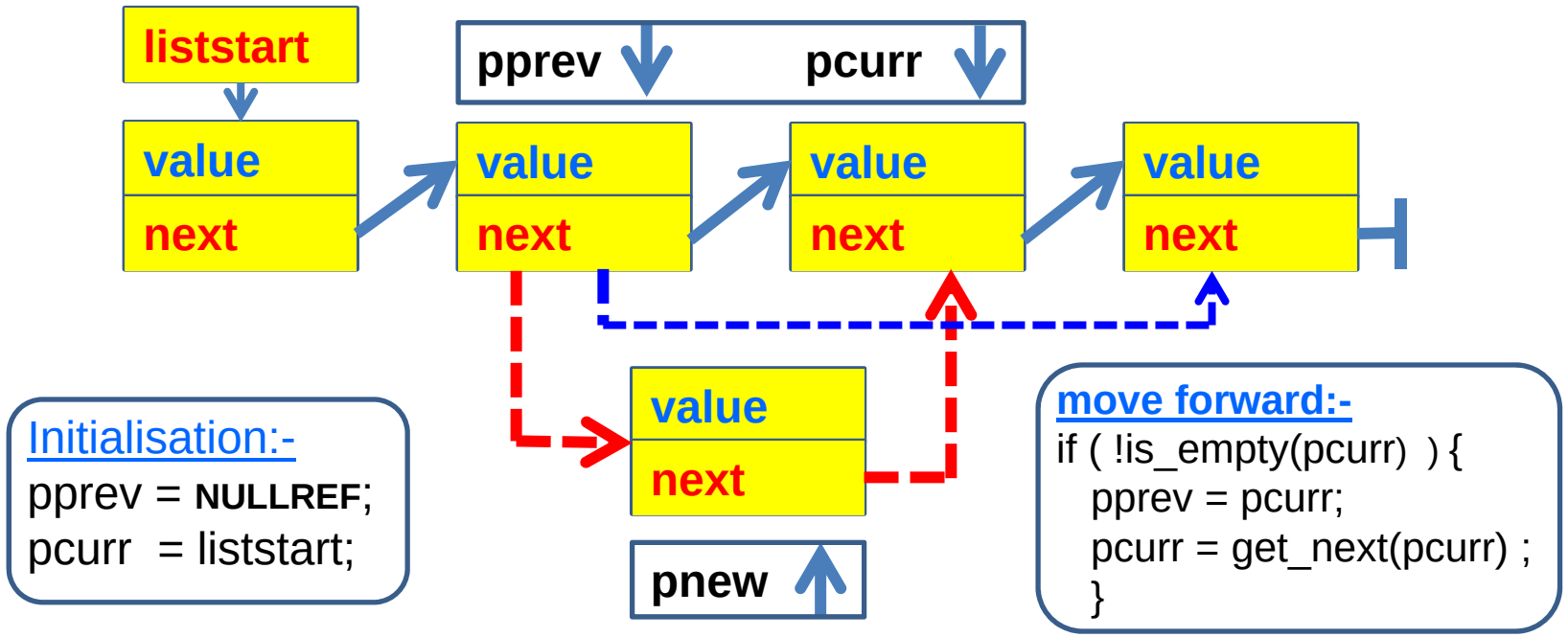
Sequence – **ordered** (position)

- **Position** p must be in range (1..n/n+1)
- **Operations**
  - add_pos(v, p) : **S x v x p ➔ S**
  - rem_pos(p) : **S x p ➔ S**
  - find(v) : **S x v ➔ Boolean**
  - is_empty() : **S ➔ Boolean**
  - size() : **S ➔ integer**

Sequence – **ordered** & **sorted**

- **Operations**
  - add_val(v) **: S x v → S**
  - rem_val(v) **: S x v → S**
  - find(v) **: S x v → Boolean**
  - is_empty() **: S → Boolean**
  - size() **: S → integer**
  - **NB: difference between ORDERED (position) and SORTED (values) (do not confuse these!)**

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)**

The code: Navigation functions

- **Navigation functions (using pprev & pcurr)**

```
void get_Seq_first()   { pprev = NULLREF; pcurr = liststart; }
int    is_Seq_empty()  { return is_empty(pcurr); }
void get_Seq_next()   {
   if ( !is_Seq_empty() ) {            // ➔ pcurr != NULLREF
      pprev = pcurr;
      pcurr  = get_next(pcurr);
      }
   }
```

pprev & pcurr have been hidden (abstracted away)

- **Navigation (iteration) through the sequence – a cliché**

```
get_Seq_first();
while ( !is_Seq_empty() ) { /* process element */ get_Seq_next();   }
```

Handling sequences / lists

- **Iterative method:-**
  - **add “pnew” between pprev & pcurr**

```
void be_add_val(valtype val) {

get_Seq_first();                              // navigate to correct position
  while (!is_Seq_empty() && (val > get_Element_value()) )   get_Seq_next();

 link_in(create_element(val));                // add the new element
  }

valtype    get_Element_value() { return get_value(pcurr); }
```

Handling sequences / lists

- **Iterative method:- find & remove**

```
listref be_find_val( valtype val)  {
  get_Seq_first();
  while (!is_Seq_empty() && (val != get_Element_value()) )  get_Seq_next();
  return get_Current_ref();
  }
```

get_Current_ref() returns the value of **pcurr** – which is a **reference**

pcurr is **NULLREF** (not found) or **refers to an element** (found)

```
void be_rem_val   (valtype val)    { unlink( be_find_val(val) );   }
void be_rem_pos (postype pos)  { unlink( be_find_pos(pos) ); }
```

The “link_in” function

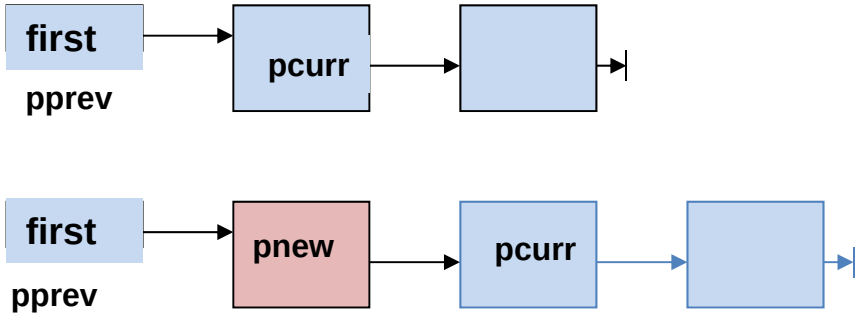
```
void link_in( pnew ) {                          // singly linked list
    set_next(pnew, pcurr);
    if (is_empty(pprev)) liststart = pnew   else set_next(pprev, pnew);
}
void link_in( pnew ) {                          // doubly linked list
    set_prev(pnew, pprev);
    set_next(pnew, pcurr);
    if (is_empty(pprev)) liststart = pnew   else set_next(pprev, pnew);
    if (is_empty(pcurr))  listend  = pnew   else set_prev(pcurr, pnew);
}
```

- pnew is always defined and is thus non-null – but check just the same! TO DO!
- pprev is null on insertion at the beginning - check required
- pcurr is null on insertion at the end - check required

Sequence – add at position p

p = 1 add at beginning

pnew = element; pprev = null; pcurr = 1

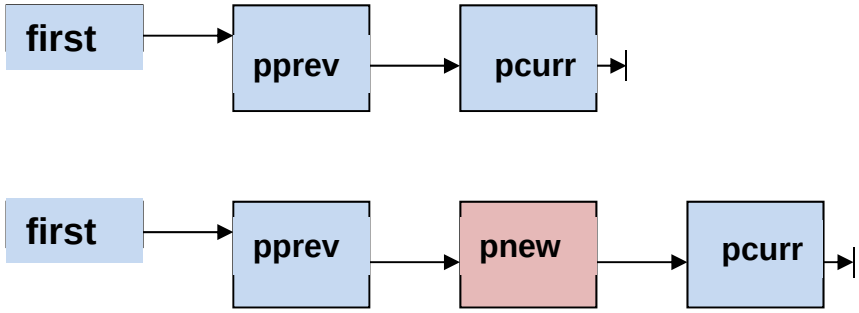


if (is_empty(pprev)) first = pnew; else set_next(pprev, pnew);
set_next(pnew, pcurr);

Sequence – add at position p

p = 2 add in middle

pnew = element; pprev = 1; pcurr = 2

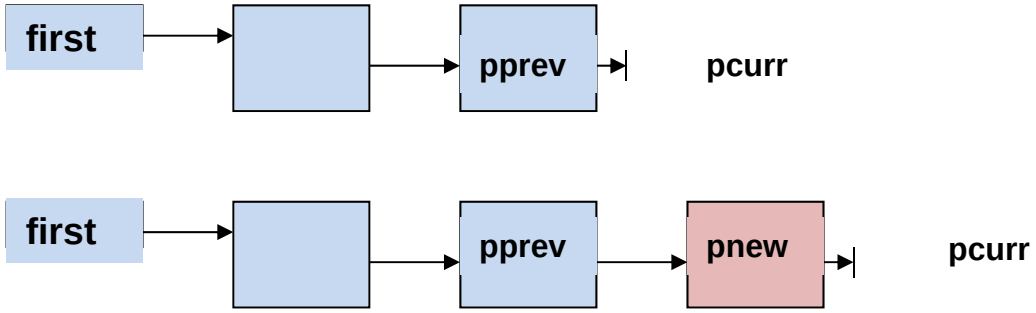


if (is_empty(pprev)) first = pnew; else set_next(pprev, pnew);
set_next(pnew, pcurr);

Sequence – add at position p

p = 3 add at end

pnew = element; pprev = 2; pcurr = null



if (is_empty(pprev)) first = pnew; else set_next(pprev, pnew);
set_next(pnew, pcurr);