Appendiks A
Simple_list.

Simple_list begrebet.

Indsættelse.

Sletning.

Andre operationer.

Eksempel.

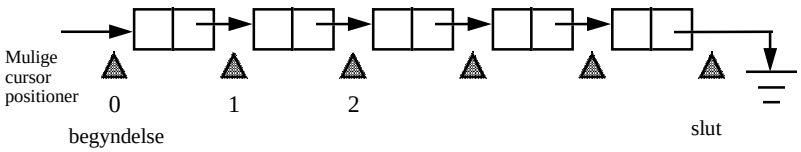
PS1 -- Simple_list

Noter

I dette appendiks beskrives en variation af kædede lister i forhold til klassen LINKED_LIST fra Eiffel biblioteket.

Simple_List[X].

- Nedenfor vises en kædet liste med mulige *cursor positioner*.
- Cursor positioner udpeger *mellemrum* mellem elementer.
- En cursor position kan repræsenteres som referencer til to LINKABLE objekter (“link-kasser”).



- *Cursor positionens nummer* er antallet af elementer før cursoren.
- **Ved indsættelse og sletning efterlades cursor positionens nummer uforandret.**
- L.insert(e); L.delete_after efterlader listen uforandret.

PS1 -- Simple_list

Noter

Det har været et mål med designet af SIMPLE_LIST at opnå et overskuelig men realistisk bud på kædede lister i Eiffel.

Klassen LINKED_LIST fra Eiffel biblioteket er en meget righoldig klasse. I LINKED_LIST udpeger cursor-elementet et element i listen, og det beskrives i LINKED_LIST hvordan cursoren flyttes i forbindelse med de forskellige operationer. I SIMPLE_LIST er cursoren et mellemrum mellem listeelementer. Med dette valgt bliver det meget naturligt at indsætte et element (i et mellemrum). I LINKED_LIST er det meget naturligt at slette det element, som cursoren peger på. Da indsættelse formodentlig er langt hyppigere en sletning, finder vi det af stor værdi, at netop indsættelse er “naturlig”.

Det har været et mål ved designet af SIMPLE_LIST at undgå bevægelser af cursoren ved indsættelse og sletning. Med den fortolkning, at cursoren er et mellemrum relativ til starten af listen, kan dette lade sig gøre (se de kommende slides).

Man kan sige, at cursor begrebet er hævet til et højere abstraktionsniveau i SIMPLE_LIST i forhold til LINKED_LIST. Et mellemrum identificeres ikke ved de to elementer, som er omkring mellemrummet, men derimod relativ til mellemrummets nummer (positionens nummer).

Den sidste observation på denne slide er, at insert og delete_after udført i sekvens alt i alt ikke påvirker listen.

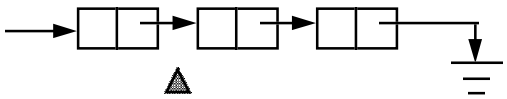
Operationer på Simple_List[X]: Indsættelse.

Insert(e: X)

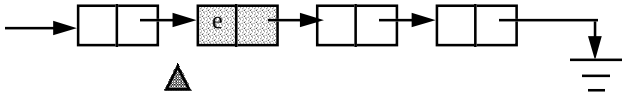
Indsætter e i den nuværende position.
Køretid: O(1).

Cursor positionering forbliver uændret

Før



Efter



Noter

Operationer på Simple_List[X]: Indsættelse i tom liste.

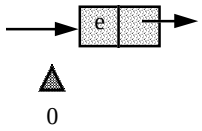
L: Simple_List;
!!L.create -- L er nu den tomme liste.

Før

- Tom liste (position nummer 0).

L.insert(e)

Efter



0

PS1 -- Simple_list

Noter

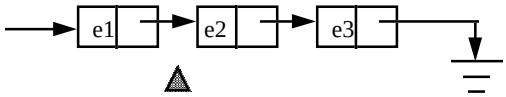
Når vi taler om den tomme liste er det helt naturligt at tale om én mulig cursorposition, som både er begyndelsen og slutningen. Dette svarer til at den tomme LINKED_LIST er både off_left og off_right.

Operationer på Simple_List[X]: Sletning.

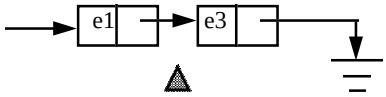
Delete_after

Prebetingelse: Positionen er ikke slutpositionen.
Sletter elementet efter den nuværende position.
Køretid: O(1).
Cursor positionen forbliver uændret.

Før



Efter



PS1 -- Simple_list

Noter

Når vi vil slette et element bliver vi nødt til at udtrykke, om det er elementet før eller efter cursoren (mellemrummet) som ønskes slettet. Da cursoren helt naturligt kan repræsenteres som en reference til elementet *før* mellemrummet (og måske også en reference til elementet efter mellemrummet) er det muligt at slette elementet efter mellemrummet. Men det er ikke muligt i en enkeltkædet liste at slette elementet før mellemrummet. (Årsagen er, at for at slette et element *e* i en kædet liste skal vi have adgang til elementet før *e*).

Øvrige operationer på Simple_List[X]

Retrieve_before: X	Returner elementet før nuværende position. O(1). Prebetingelse: not At_beginning.
Retrieve_after: X	Returnerer el. efter den nuværende position. O(1) Prebetingelse: not At_end.
Advance	Flytter positionen til næste mellemrum. O(1)
Goto_beginning	Flytter positionen til 0. O(1)
Goto_end	Flytter positionen til den største. O(n)
At_beginning: Boolean	Returnerer om positionen er 0. O(1)
At_end: Boolean	Returnerer om positionen er Length. O(1)
Position_number: Integer	Returnerer positionsnummeret.
Is_empty: Boolean	Returnerer hvorvidt listen er tom. O(1)
Length: Integer	Returnerer antallet af elementer.
Append(L:Like Current)	Konkatenerer L til denne liste. Positionering i denne liste bevares. O(Length + L.Length).

PS1 -- Simple_list

Noter

Vi har bevidst forsøgt at benytte en anderledes terminologi i forhold til LINKED_LIST. Derved bliver mere klart ved læsning af et programfragment, om man anvender det ene eller det andet listebegreb.

I det ovenstående er n lig med antallet af elementer i listen, altså lig med resultatet af funktionen Length.

Eksempel på anvendelse.

```
L: Simple_list[Integer]
!!L.create
L.insert(1); L.advance; L.insert(2); L.advance; L.goto_first;

from L.goto_begining
until L.Is_end
loop i := L.retrieve_after;
     udnyt i til et eller andet ;
     L.advance
end
```

PS1 -- Simple_list

Noter