Chap 4. Linked Lists

Singly linked lists and chains

- Figure 4.1: non-sequential list representation
- Figure 4.2: usual way to draw a linked list
- Figure 4.3: inserting into a linked list
- Figure 4.4: delete GAT

Representing chains in C

- Example 4.1 [List of words]
    - Declarations // § 4.2
    - Statements // § 4.2
    - Figure 4.5: referencing the fields of a node
- Example 4.2 [two-node linked list]
    - Definition of node structure // § 4.2
    - Program 4.1: create a two-node list
    - Figure 4.6: A two-node list
- Program 4.2: simple insert into front of list
- Figure 4.7: inserting into an empty and nonempty list
- Program 4.3: deletion from a list
    - Figure 4.8 and 4.9
- Program 4.4: printing a list

Linked stacks and queues

- Figure 4.11: linked stack and queue
- Declarations, initial/boundary conditions // §4.3
    - Stacks
    - Queues
- Program 4.5: add to a linked stack
- Program 4.6: delete from a linked stack
- Program 4.7: add to the rear of a linked queue
- Program 4.8: delete from the front of a linked queue

Polynomials

- Polynomial representation // §4.4.1
  - Type declarations
  - polyNode
  - examples
  - Figure 4.12
- Adding polynomials
  - Figure 4.13
  - Program 4.9
  - Program 4.10
- Erasing polynomials
  - Program 4.11
- Circular list representation of polynomials
  - Figure 4.14
  - Program 4.12
  - Program 4.13
  - Program 4.14
- Adding two polynomials represented as circular lists with header nodes
  - Figure 4.15
  - Program 4.15

Additional list operations

- Declarations // §4.5.1
- Program 4.16: Inverting a singly linked list
- Program 4.17: Concatenating singly linked lists
- Program 4.18: Inserting at the front of a list
- Program 4.19: Finding the length of a circular list

Doubly linked lists

- Figure 4.21
- Declarations // §4.8
- Ptr = ptr->llink->rlink = ptr->rlink->llink
- Figure 4.22
- Program 4.26
- Program 4.27
- Figure 4.23