Inhoud

[2.1 Specificatie 2](#_Toc86597446)

[2.2 Implementatie van een gelinkte lijst 3](#_Toc86597447)

[2.2.1 Implementatie van een knoop 3](#_Toc86597448)

[2.2.2 Implementatie van een gelinkte lijst 3](#_Toc86597449)

[2.2.3 Het gebruik van ankercomponenten 6](#_Toc86597450)

[2.3 Dubbel gelinkte lijsten 6](#_Toc86597451)

[2.4 Beschrijving en implementatie van Stapels 7](#_Toc86597452)

[2.4.1 Beschrijving van een Stapel 7](#_Toc86597453)

[2.4.2 Implementatie van een Stapel 8](#_Toc86597454)

[2.5 Toepassingen van Stapels 9](#_Toc86597455)

[2.5.1 Controleren van haakjes 9](#_Toc86597456)

[2.5.2 Waardebepaling van een rekenkundige uitdrukking 10](#_Toc86597457)

Hoofdstuk 2 Gelinkte lijsten

Elementen van een array bezetten steeds opeenvolgende geheugenplaatsen wat ervoor zorgt dat men de elementen kan aanspreken in een constante tijd. Anderzijds zorgt dit ervoor dat het minder eenvoudig is om elementen ergens in het midden toe te voegen. Gelinkte lijsten zijn een voorbeeld van een dynamische datastructuur waarbij de samenhang tussen de verschillende componenten gerealiseerd kan worden dmv pointers.

Voordelen gelinkte lijsten:

* Het toevoegen of verwijderen van elementen in een constante tijd gebeurt, op voorwaarde dat men reeds weet waar het element moet worden toegevoegd of verwijderd.
* De gelinkte lijst legt a priori geen beperkingen op aan het aantal elementen dat aan deze lijst kan toegevoegd worden.

# 2.1 Specificatie

Een gelinkte bestaat uit een aantal knopen die via een kettingstructuur aan elkaar geschakeld zijn.

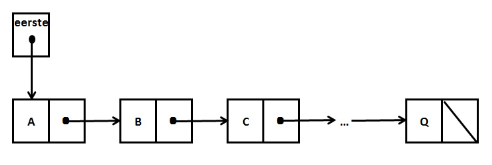
Een knoop bestaat uit twee velden:

* Een data-veld: waarin de data is opgeslagen
* Een volgende-veld: dat verwijst naar de volgende knoop in de lijst

De laatste knoop bevat een wijzer null. De eerste “eerste”.

De belangrijkste basisbewerkingen voor een enkelvoudige gelinkte lijst zijn:

* Zoek(): zoekt de positie van de knoop met als data-veld het argument
* Verwijder(): verwijdert de knoop die volgt na de opgegeven knoop en geeft de waarde van het data-veld van de verwijderde knoop weer
* voegToe(): voegt een knoop toe aan een opgegeven knoop, het data-veld krijgt de waarde van het argument



# 2.2 Implementatie van een gelinkte lijst

Een gelinkte lijst wordt voorgesteld door een ketting van aaneengeschakelde knopen.

Een knop bestaat uit:

* data veld
* referentie veld: volgende

## 2.2.1 Implementatie van een knoop

De klasse in pseudocode:

Afbeelding met tekst

Automatisch gegenereerde beschrijvingDe gedefinieerde Knoop is een datastructuur die een element van een niet nader gedefinieerde klasse Element bevat

**Algoritme voor de constructor**

De constructor Knoop maakt een nieuw object van de klasse Knoop aan.

Afbeelding met tekst

Automatisch gegenereerde beschrijving

## 2.2.2 Implementatie van een gelinkte lijst

De klasse Knoop wordt als inner class van de klasse GelinkteLijst geïmplementeerd. De methodes van de klasse GelinkteLijst hebben toegang tot te velden van Knoop.

**De klasse GelinkteLijst:**

Afbeelding met tekst

Automatisch gegenereerde beschrijving De constructor GelinkteLijst maakt een nieuw object van de klasse GelinkteLijst aan.

**Algoritme voor het opzoeken van een element x**.

Het opzoeken van een knoop met dataveld x in een gelinkte lijst l gebeurt door de knopen van de lijst een voor een te overlopen.

Afbeelding met tekst

Automatisch gegenereerde beschrijving

Als eerste Knoop wordt de Knoop waar de referentie eerste naar verwijst bekeken. Indien de waarde van het dataveld niet overeenstemt met x wordt de volgende knoop bekeken. Deze knoop is eenvoudig te vinden door de ketting te volgen via het referentieveld.

Afbeelding met tekst

Automatisch gegenereerde beschrijving

De uitvoeringstijd is afhankelijk van de positie van x.

* in het slechtste geval: heel de lijst wordt doorlopen en x wordt niet gevonden. => lineaire uitvoeringstijd
* in het beste geval: x wordt direct gevonden => constante uitvoeringstijd
* gemiddelde geval: x zit in de helft => lineaire uitvoeringstijd

Het zoeken van een element x gebeurt dus niet efficiënter dan in een gewone array. In de praktijk is het volgen van pointers zelfs trager.

**Algoritme voor het verwijderen van een knoop**

De functie verwijder() verwacht als inputwaarde een referentie ref. Deze verwijst naar de knoop in de gelinkte lijst die het te verwijderen element bevat.

Afbeelding met tekst

Automatisch gegenereerde beschrijving

Afbeelding met keukenapparaat

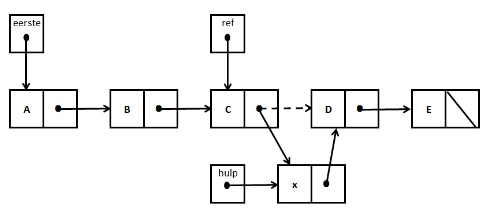
Automatisch gegenereerde beschrijving

verwijderen van de knoop na de gegeven knoop

**Algoritme voor het toevoegen van een knoop**

De methode voegToe() voegt een nieuwe knoop toe na de knoop die wordt aangeduid door een gegeven referentie.

* Creëer een nieuwe knoop hulp
* Stockeer de waarde x in het data-veld van de knoop hulp
* Laat hulp.volgende verwijzen naar ref.volgende
* Laat ref.volgende verwijzen naar hulp



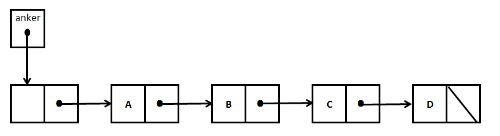
Afbeelding met tekst

Automatisch gegenereerde beschrijving

## 2.2.3 Het gebruik van ankercomponenten

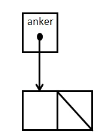
De methodes verwijder en voegtoe zijn enkel bruikbaar voor algemene gevallen. Het speciale geval om een element toe te voegen aan een lege lijst of een element toe te voegen vooraan in de lijst is niet mogelijk met de besproken methode. Zelfde geld voor verwijder.

Daarom voegen we een ankercomponent toe. Dat is een extra Knoop die helemaal vooraan aan de gelinkte lijst wordt toegevoegd. Het dataveld ervan is leeg, het referentieveld verwijst naar de eerste knoop van de gelinkte lijst. Het maakt dus geen deel uit van de eigen gelinkte lijst maar dient om te implementatie te vereenvoudigen.



Door dit toe te voegen heeft elke knoop een voorganger. => ook de eerste knoop kan verwijderd worden of op de eerste positie kan er een nieuwe knoop toegevoegd worden.

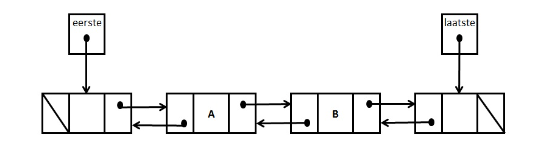
Een lege gelinkte lijst bestaat nu uit een enkele knoop = de ankercomponent.



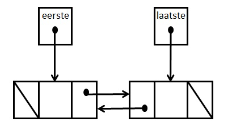
# 2.3 Dubbel gelinkte lijsten

In een enkelvoudige gelinkte lijst is de eerste knoop van de lijst vlot bereikbaar. De laatste knoop bereiken kost echter tijd lineair in de lengte van de lijst.

Oplossing: elke knoop moet twee referenties bijhouden. Een referentie volgende en een referentie vorige. Zo een lijst wordt een dubbel gelinkte lijst genoemd.



In een dubbel gelinkte lijst wordt meestal gewerkt met twee ankercomponenten.

Testen of de lijst leeg is kan als volgende:

Eerste.voglende = laatste

Of

Laatste.vorige = eerste

# 2.4 Beschrijving en implementatie van Stapels

## 2.4.1 Beschrijving van een Stapel

De naam Stapel of Stack is gekozen naar analogie met een stapel boeken.

Indien we een boek van een stapel nodig hebben, is het meest voorhand liggende boek het bovenste of eerste. Een boek dat zich op een andere plaats bevindt in de stapel, is alleen bereikbaar als we de bovenliggende boeken verwijderd hebben.

Principe Stack:

* Een element dat we willen toevoegen aan een Stapel, komt steeds bovenaan de reeds bestaande stapel.
* Enkel het bovenste element van de stapel kan verwijderd worden. Dit element wordt de top van de stapel genoemd.

Een stapel is maw een LIFO of last-in-first-out structuur

Voor de implementatie van deze structuur zullen we gebruik maken van de klasse Stack. Hier zijn de basisbewerkingen gedefinieerd:

* Stack(): constructor, maakt een nieuwe stapel aan
* Empty(): controleert of de stapel al dan niet leeg is
* Push(): voegt een nieuw element toe bovenaan een stapel, het toegevoegde element wordt de nieuwe top van de stapel
* Pop(): verwijdert het bovenste element van een stapel en retourneert het verwijderde element.
* Peek(): geeft het bovenste element van de stapel terug zonder het te verwijderen.

## 2.4.2 Implementatie van een Stapel

We tonen nu hoe een stapel kan geïmplementeerd worden als een gelinkte lijst waarbij enkel de top bereikbaar is.

Afbeelding met tekst

Automatisch gegenereerde beschrijvingDe klasse Knoop wordt al inwendige klasse van de klasse Stack geïmplementeerd. De methodes van de klasse Stack hebben toegang tot de velden van Knoop.

Afbeelding met tekst

Automatisch gegenereerde beschrijvingAfbeelding met tekst

Automatisch gegenereerde beschrijvingAfbeelding met tekst

Automatisch gegenereerde beschrijving

Afbeelding met tekst

Automatisch gegenereerde beschrijvingAfbeelding met tekst

Automatisch gegenereerde beschrijving

# 2.5 Toepassingen van Stapels

## 2.5.1 Controleren van haakjes

Compilers controleren steeds of de haakjes allemaal correct zijn en generen een foutboodschap wanneer dit niet het geval is.

**Basisidee**:

Voor de controle van de haakjes worden twee zaken onderzocht.

1. Controle of elk openend haakje juist een overeenkomstig sluitend haakje heeft. Maw, de symbolen moeten in paren voorkomen
2. De volgorderegels moeten in acht genomen worden. De haakjes mogen genest zijn, maar paren mogen elkaar niet overlappen

We willen met behulp van een stapel een algoritme ontwerpen dat de vereffening van symbolen verifieert.

* Maak een lege stapel aan
* Doorloop alle symbolen een voor een. Indien het symbool een haakje is moet er een opdracht uitgevoerd worden
  + Als het een open symbool is: op stapel plaatsen push()
  + Als het een sluit symbool is, zijn er 2 situaties:
    - De stapel is leeg: er wordt een foutmelding gegenereerd aangezien er geen corresponderend open symbool is
    - De stapel is niet leeg: het top element wordt van de stapel gehaald, pop() dit symbool wordt vergeleken met het ingelezen symbool, indien deze niet overeenkomen wordt er een foutmelding gegeven.
* Als alle karakters zijn ingelezen en de stapel niet leeg is, dan wordt er een fout melding gegeven. Afbeelding met tekst

  Automatisch gegenereerde beschrijving

## 2.5.2 Waardebepaling van een rekenkundige uitdrukking

Een wiskunde uitdrukking wordt normaal gezien op deze manier beschreven:

1 + 2 x 3

Dit is de infix-notatie. Echter kunnen deze niet rechtstreeks door de compiler vertaald worden. De postfix-notatie is beter geschikt hiervoor.

Het staat ook bekend onder de benaming RPN( reverse Polish Notation)

Bv 3 4 x

De binaire vermenigvuldigingsoperator staat na zijn operanden en werkt in op de twee argumenten ervoor nl 3 en 4. De betekenis van deze postfix is dus 3 x 4 = 12

Afbeelding met tafel

Automatisch gegenereerde beschrijving

Er zijn een aantal afspraken:

* De operatoren x en / hebben een hogere prioriteit dan + en –
* Prioriteiten kunnen door gebruik van haakjes worden aangepast
* De prioriteit van x en / is gelijk. Als ze beide in een uitdrukking staan, worden ze uitgevoerd in de volgorde waarin je ze tegenkomt, dus van links naar rechts.

**Het bepalen van de waarde van een postfix-uitdrukking**

Een machine kan de waarde gemakkelijk bepalen met behulp van een stapel. De berekening van de uitdrukking mbv een stapel gaat als volgt:

* De uitdrukking wordt van links naar rechts doorlopen
* Als er een operand wordt ontmoet dan wordt dit op de stapel geplaatst
* Als een operator wordt ontmoet dan worden de twee laats gestapelde operanden van de stapel gehaald en de bewerking met die twee operanden uitgevoerd, het resultaat ervan wordt op de stapel geplaatst.
* Deze stappen worden herhaald totdat de volledige uitdrukking is doorlopen
* Als deze volledig doorlopen is, bestaat de stapel nog maar uit 1 element, de waarde van de postfix-uitdrukking

Dit algoritme is uitvoerbaar in lineaire tijd, aangezien voor elk symbool uit de uitdrukking maximaal een constant aantal stapel-bewerkingen moet uitgevoerd worden, waarbij elke bewerking een constante tijd vraagt.