DA-ALG1000

Obligatorisk oppgave 1: Prosjektrapport

Av: Martin Kowalik Gran

Forord

Dedikerer denne oppgaven til Alva Sophia, som har klart å holde seg frisk og i barnehagen hele den alternative arbeidsuka. Takket være et oppsving i immunforsvarets evne til å bekjempe alle mulige bakterier og virus har rapporten og programmet blitt ferdigstilt uten nevneverdig stress

Innholdsfortegnelse

1.0 Innledning

Denne rapporten omhandler den første obligatoriske programmeringsoppgaven i faget DA-MAD1000 våren 2015. Oppgaven var å skrive et program som kunne lagre data i en «Singel-linked liste». Programmet og rapporten er skrevet av Martin Kowalik Gran uten hjelp av andre. Rapporten inneholder bestillingen (oppgavetekst), beskrivelse av fremgangsmåten av utviklingen av programmet, beskrivelse av løsningen på programmet brukerveiledning for det ferdige programmet, UML og testing av programmet.

1.1 Oppgaveteksten

Skriv et program der du tar utgangspunkt i klassene Node og SingleLinkedList (forelesning nr. 5) og benytter disse til programmeringen. Du skal ha et hovedprogram (main), som ligger i egen klasse. Dette skal inneholde en meny (liste av tilgjengelige kommandoer) med tanke på å legge inn data, fjerne data, skrive ut og andre funksjoner for en lenket liste. I menyen listes alle tilgjengelige kommandoer opp. Programmet skal avsluttes ved eget valg. Listen skal ikke være sortert og det kan forekomme like tall. Det er ikke nødvendig med en GUI-løsning. Følgende funksjoner skal implementeres:

1: Slett det første elementet i listen.
2: Slett det siste elementet i listen.
3: Slett et element med oppgitt verdi fra listen (slett kun første forekomst).
4: Slett alle element med oppgitt verdi fra listen.
5: Legg til et element med oppgitt verdi i starten av listen.
6: Legg til et element med oppgitt verdi i slutten av listen.
7: Legg til et element etter et element med oppgitt verdi (gjøres én gang).
8: Legg til et element foran et element med oppgitt verdi (gjøres én gang).
9: Skriv ut lengden på listen.
10: Tell opp antall forekomster av element med gitt verdi i lista, dette antallet skrives ut.
11: Skriv ut hele listen. Maks 5 elementer pr. linje.
12: Slett hele listen. Hvor mange elementer som ble slettet, skrives ut.

Du skal levere oppgaven med en rapport. Kildekoden for filen(e) din(e) skal legges ved denne rapporten. (Foreslår en zip-fil eller lignende slik at jeg får kun én fil i Fronter) Du må gjerne jobbe sammen i gruppe med flere (og da levere som en gruppeinnlevering, maks 3 personer pr. gruppe). Det skal ikke gjøres endringer i datafeltene for klassene Node og SingleLinkedList. Alle metode-navn og variabler skal være på engelsk. Oppgaven skal leveres innen innleveringsfrist. Skulle det være situasjoner som krever utsettelse, skal det du har gjort før tidsfristen leveres samt at søknad om utsettelse leveres meg på e-post.

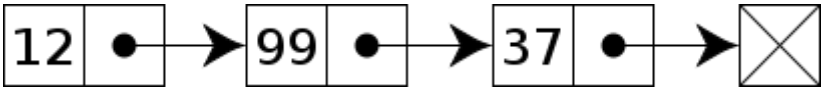
2.0 Bakgrunnskunnskap for oppgaven

2.1Hva er lenkede lister?

En lenket liste er en enkel datastruktur som består av en «kjede» med data. Disse lagres i objekter av typen «noder». Noder inneholder pekere til andre noder. I tillegg inneholder noder «data». Avhengig av formålet kan ulike typer data plasseres i noden. Figur 1. viser en lenket liste med noder som inneholder data av typen «integer», samt en peker til neste node i listen.

En lenket liste inneholder også en startpeker «head» som peker på første noden i listen. Lenkede lister kan også inneholde en sluttpeker «tail», som peker på siste elementet i listen.

En «singel linked liste» er en lenket datastruktur der nodene kun inneholder en peker. Denne peker alltid til neste node i lenken. Den siste noden i lenken peker enten mot «null». I tilfeller av tomme singel linkede lister peker «head» mot «null». Figur 1. illustrer en typisk lenket liste.



Figur 1: Figuren viser en datastruktur av typen «Singel linked liste».

Fordelen med lenkede lister kontra «array-lister» er rask innsetting og sletting av data. I tillegg bruker lenkede lister er dynamiske og avsetter ikke minne i forveien som er tilfellet med array-lister.

Ulempene med singel-linkede lister er mangelen på indeksering og direkte oppslag. Skal man ha tilgang til element n, må man gå gjennom foregående elementer. Dette øker tiden det tar å lese av ønsket data.

3.0 Beskrivelse av programmet

3.1 Beskrivelse av fremgangsmåten

For å løse oppgaven ble oppgaveteksten nøye lest. Ut fra oppgaveteksten ble det definert hvilke funksjoner programmet skulle inneholde. Deretter ble programmet skrevet. Det ble opprettet klasser kalt «Node» og «LinkedList». I tillegg ble det opprettet et testprogram kalt «TestOblig1». Klassen LinkedList inneholdt alle metodene for å håndtere

funksjonene programmet skulle utføre. Klasse Node inneholdt peker ti neste node og mulighet til å lagre data som en «Int-variabel». TestOblig1 skulle håndtere menyfunksjonen og inneholde «main», men underveis ble det bestemt å dele funksjonen mellom TestOblig1 og klassen MenuItems som inneholder menyvalgene som en tabell av typen String, samt en metode for å skrive disse.

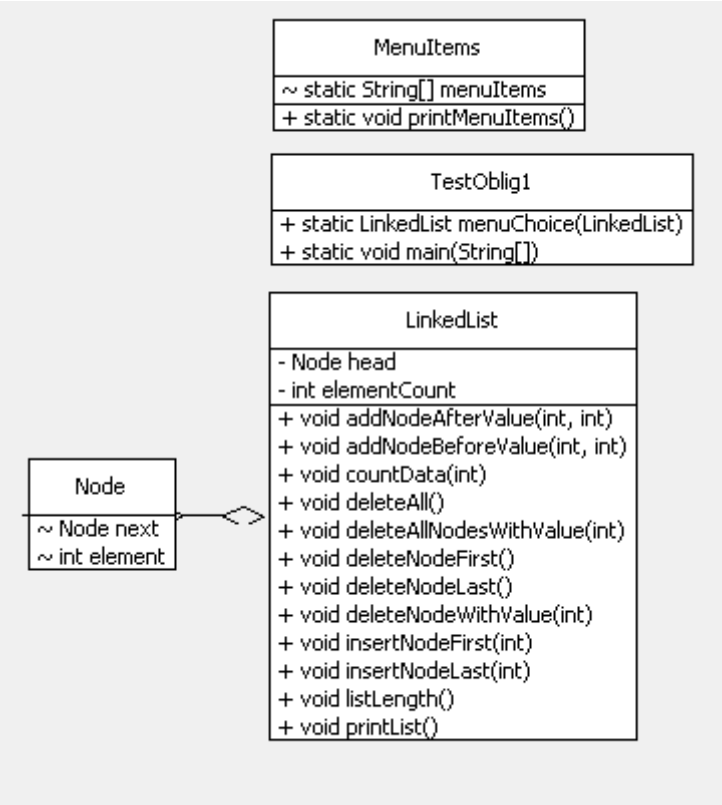
Da en førsteversjon av programmet var klart ble det gjennomført testing av de ulike funksjonene. De fleste funksjonene viste seg å ha problemer ved ulike forutsetninger. Blant annet å slette elementer i en tom liste, eller slette elementer som ikke fantes i listen gav feil. Derfor ble det laget et testskjema der alle funksjonene ble systematisk testet på forutsetningene: listen er tom, node funnet i starten, node ikke funnet og node funnet på slutten. Programmet ble forbedret helt til alle funksjonene fungerte under alle forutsetninger. Deretter ble svarene skrevet ned i et eget skjema (se tabell 1. under feiltesting).

3.2 Beskrivelse av løsningen

Programmet løser oppgaven uten en GUI-løsning. Ut fra oppgaveteksten har det ikke vist seg å være nødvendig med en avansert løsning på bestillingen. Programmet tilfredsstiller en robust måte å lagre data i en lenkestruktur. Det klarer også å utføre alle funksjonskravene so er satt til programmet. Dette er bekreftet gjennom testing. Menyfunksjonen gjør at man enkelt kan modifisere den opprettede lenke-listen.

4.0 Brukerveiledning

4.1 UML



Figur 2: Figuren viser UML for programmet med variabler og metoder.

4.2 Informasjon for bruk

Programmet inneholder fire klasser. Det gis en innføring i bruk av de ulike metodene i under de ulike klassene.

4.2.1 TestOblig1

Inneholder metoden main og metoden menuChoice(LinkedList list).

menuChoice(LinkedList list) er en metode som håndterer menyfunksjonen.

- LinkedList list – objekt av typen LinkedList som menyfunksjonene skal utføres på.
- menuChoice() har ulike menyvalg gjennom en switch-funksjon. Disse kan velges gjennom å taste tallene 1-13.
    1. Slett det første elementet i listen
        - Det første elementet i lista slettes.
        - Feilmelding gis om listen er tom.
    2. Slett det siste elementet i listen

    - Det siste elementet i lista slettes. Feilmelding gis om lista er tom.
3. Slett et element med oppgitt verdi fra listen fra listen (slett kun første forekomst)
    - Her ber programmet input i form av heltall (int) verdien på noden som skal slettes.
4. Slett alle elementene med oppgitt verdi fra listen
    - Her ber programmet input i form av heltall (int) verdien på nodene som skal slettes.
5. Legg til et element med oppgitt verdi i starten av listen
    - Her ber programmet om antallet noder du vil legge til. Denne inputen gis iform av heltall (int). Nodene legges foran hverandre. I tillegg må verdien på hver enkelt node skrives inn som heltall (int)
6. Legg til et element med oppgitt verdi i slutten av listen
    - Her ber programmet om antallet noder du vil legge til. Denne inputen gis iform av heltall (int). Nodene legges bak hverandre. I tillegg må verdien på hver enkelt node skrives inn som heltall (int)
7. Legg til et element etter et element med oppgitt verdi (gjøres en gang)
    - Det må først spesifiseres verdien på noden du vil legge til. Dette gjøres i form av heltall (int)
    - Det må spesifiseres verdi på noden du vil legge verdien etter.
8. Legg til et element foran et element med oppgitt verdi (gjøres en gang)
    - Det må først spesifiseres verdien på noden du vil legge til. Dette gjøres i form av heltall (int)
    - Det må spesifiseres verdi på noden du vil legge verdien foran.
9. Skriv ut lengden på listen
    - Antallet noder i listen skrives ut.
10. Tell opp antall forekomster av element med gitt verdi i lista
    - Antallet noder med bestemt verdi telles og skrives ut.
11. Skriv ut hele listen.
    - Hele listen skrives ut fra første til siste element.
12. Slett hele listen.
13. Vis valgene i menyen en gang til
    - Menyvalgene skrives på nytt så disse kan leses nedover i programmet.
0. Avslutt programmet

4.2.2 Node

Objekt av typen Node. Inneholder variablen element av typen int. Her lagres dataene i noden.

Inneholder også peker til neste node i next av typen Node.

4.2.3 MenuItems

Inneholder et array av typen String. Denne tabellen inneholder tekst for menyvalgene. Inneholder også metoden

printMenuItems – metoden skriver ut menyvalgene.

4.2.3 LinkedList

LinkedList er klassen som objekter av typen LinkedList opprettes fra. Klassen inneholder metodene:

deleteNodeFirst() – sletter første noden i lenken

deleteNodeLast() – sletter siste noden i lenken

deleteNodeWithValue(int data) – sletter første node i lenken med gitt verdi.

- Int data – variabel av typen Int som definerer verdien på noden som skal slettes.

deleteAllNodesWithValue(int data) – sletter alle noder med gitt verdi.

- Int data – variabel av typen Int som definerer verdien på nodene som skal slettes.

insertNodeFirst() – setter inn ny node først i listen.

insertNodeLast() – setter inn ny node sist i listen.

addNodeAfterValue(int nodeData int data) – setter inn ny node etter en node med gitt verdi.

- Int nodeData – variabel av typen Int som definerer verdien på nodene man søker etter.
- Int data – variabel av typen Int som definerer verdien på den nye noden.

addNodeBeforeValue(int nodeData int data) - setter inn ny node foran en node med gitt verdi.

countData(int data) – teller alle noder med gitt verdi.

printList() – skriver ut listen.

deleteAll() – sletter hele listen.

5.0 Feiltesting

Det ble gjennomført en systematisk feiltesting av programmets funksjoner under ulike forutsetninger. I tabell 1 kan man se hvilke forutsetninger de ulike funksjonene har bestått. Grønne felter betyr at testkriteriet ikke er relevant for metoden. Som man kan lese av tabellen består funksjonene alle testforutsetningene.

Tabell1: *Tabellen viser de ulike testkriteriene som funksjonene har bestått.*

Menyvalg	Metode i SingelLinkedList	Lista er tom	Funnet i starten	Ikke funnet	Funnet på slutten
Slett første element i lista	deleteNodeFirst()	ok			
slett siste element i lista	deleteNodeLast()	ok			
slett første forekomst av element med oppgitt verdi fra lista	deleteNodeWithValue(int data)	ok	ok	ok	ok
Slett alle element med oppgitt verdi fra lista	deleteAllNodesWithValue(int data)	ok	ok	ok	ok
legg til et element med oppgitt verdi i starten av lista	insertNodeFirst()	ok			
legg til et element med oppgitt verdi i slutten av lista	insertNodeLast()	ok			
legg til et element etter et element med oppgitt verdi (gjøres en gang)	addNodeAfterValue(int nodeData int data)	ok	ok	ok	ok
legg til et element foran et element med oppgitt verdi (gjøres en gang)	addNodeBeforeValue(int nodeData int data)	ok	ok	ok	ok
Skriv ut lengden på listen	listLength()	ok			
Tell opp antall forekomster av element med gitt verdi i lista, dette antallet skrives ut	countData(int data)	ok	ok	ok	ok
Skriv ut hele listen. Maks 5 elementer pr linje	printList()	ok			
Slette hele listen. Antall slettede elementer skrives ut	deleteAll()	ok			

6.0 Konklusjon

Oppgaven ble løst innen fastsatt tid. Programmet utfører de oppgavene som skal utføres og tilbyr noen ekstrafunksjonaliteter under menyvalgene. Det er vektlagt mye tid til feilretting i oppgaven for å overkomme alle kjente utfordringer ved singellenkede lister. Rapport og brukerveiledning gir nok innsikt til å kunne bruke programmet for bruker og eller andre utviklere.