

MAT1030 – Diskret Matematikk

Forelesning 2: Kontrollstrukturer, tallsystemer, basis

Dag Normann

Matematisk Institutt, Universitetet i Oslo

20. januar 2010

(Sist oppdatert: 2010-01-20 12:28)



Kapittel 1: Algoritmer (fortsettelse)

Kontrollstrukturer

Kontrollstrukturer

I går innførte vi **pseudokoder** og **kontrollstrukturer**.

Kontrollstrukturer

I går innførte vi **pseudokoder** og **kontrollstrukturer**. Vi hadde tre typer grunninstruksjoner:

Kontrollstrukturer

I går innførte vi **pseudokoder** og **kontrollstrukturer**. Vi hadde tre typer grunninstruksjoner:

- Input variabel

Kontrollstrukturer

I går innførte vi **pseudokoder** og **kontrollstrukturer**. Vi hadde tre typer grunninstruksjoner:

- Input variabel
- variabel \leftarrow term

Kontrollstrukturer

I går innførte vi **pseudokoder** og **kontrollstrukturer**. Vi hadde tre typer grunninstruksjoner:

- Input variabel
- variabel \leftarrow term
- Output variabel

Kontrollstrukturer

Kontrollstrukturer

Vi hadde fem kontrollstrukturer

Kontrollstrukturer

Vi hadde fem kontrollstrukturer

- **If ... then**

Kontrollstrukturer

Vi hadde fem kontrollstrukturer

- **If ... then**
- **If ... then ... else**

Kontrollstrukturer

Vi hadde fem kontrollstrukturer

- **If ... then**
- **If ... then ... else**
- **While ... do**

Kontrollstrukturer

Vi hadde fem kontrollstrukturer

- **If ... then**
- **If ... then ... else**
- **While ... do**
- **Repeat ... until**

Kontrollstrukturer

Vi hadde fem kontrollstrukturer

- **If ... then**
- **If ... then ... else**
- **While ... do**
- **Repeat ... until**
- **For ... to ... do**

Kontrollstrukturer

Vi hadde fem kontrollstrukturer

- **If ... then**
- **If ... then ... else**
- **While ... do**
- **Repeat ... until**
- **For ... to ... do**

Vi skal se på noen flere eksempler på pseudokoder.

Kontrollstrukturer

Kontrollstrukturer

Det er ingen som vet om algoritmen som er beskrevet i den neste pseudokoden vil **terminere** for alle input.

Kontrollstrukturer

Det er ingen som vet om algoritmen som er beskrevet i den neste pseudokoden vil **terminere** for alle input. Det betyr at den muligens **ikke** er en algoritme i bokas forstand.

Kontrollstrukturer

Det er ingen som vet om algoritmen som er beskrevet i den neste pseudokoden vil **terminere** for alle input. Det betyr at den muligens **ikke** er en algoritme i bokas forstand. Den forutsetter at vi kan skille mellom partall og oddetall.

Kontrollstrukturer

Det er ingen som vet om algoritmen som er beskrevet i den neste pseudokoden vil **terminere** for alle input. Det betyr at den muligens **ikke** er en algoritme i bokas forstand. Den forutsetter at vi kan skille mellom partall og oddetall.

Eksempel (Ubegrenset while-løkke)

Kontrollstrukturer

Det er ingen som vet om algoritmen som er beskrevet i den neste pseudokoden vil **terminere** for alle input. Det betyr at den muligens **ikke** er en algoritme i bokas forstand. Den forutsetter at vi kan skille mellom partall og oddetall.

Eksempel (Ubegrenset while-løkke)

1. Input x [$x \geq 1$ heltall.]

Kontrollstrukturer

Det er ingen som vet om algoritmen som er beskrevet i den neste pseudokoden vil **terminere** for alle input. Det betyr at den muligens **ikke** er en algoritme i bokas forstand. Den forutsetter at vi kan skille mellom partall og oddetall.

Eksempel (Ubegrenset while-løkke)

1. Input x [$x \geq 1$ heltall.]
2. $y \leftarrow 0$

Kontrollstrukturer

Det er ingen som vet om algoritmen som er beskrevet i den neste pseudokoden vil **terminere** for alle input. Det betyr at den muligens **ikke** er en algoritme i bokas forstand. Den forutsetter at vi kan skille mellom partall og oddetall.

Eksempel (Ubegrenset while-løkke)

1. Input x [$x \geq 1$ heltall.]
2. $y \leftarrow 0$
3. **While** $x > 1$ **do**

Kontrollstrukturer

Det er ingen som vet om algoritmen som er beskrevet i den neste pseudokoden vil **terminere** for alle input. Det betyr at den muligens **ikke** er en algoritme i bokas forstand. Den forutsetter at vi kan skille mellom partall og oddetall.

Eksempel (Ubegrenset while-løkke)

1. Input x [$x \geq 1$ heltall.]
2. $y \leftarrow 0$
3. **While** $x > 1$ **do**
 - 3.1. **If** x er partall **then**

else

Kontrollstrukturer

Det er ingen som vet om algoritmen som er beskrevet i den neste pseudokoden vil **terminere** for alle input. Det betyr at den muligens **ikke** er en algoritme i bokas forstand. Den forutsetter at vi kan skille mellom partall og oddetall.

Eksempel (Ubegrenset while-løkke)

1. Input x [$x \geq 1$ heltall.]
2. $y \leftarrow 0$
3. **While** $x > 1$ **do**
 - 3.1. **If** x er partall **then**
 - 3.1.1. $x \leftarrow \frac{x}{2}$
 - else**

Kontrollstrukturer

Det er ingen som vet om algoritmen som er beskrevet i den neste pseudokoden vil **terminere** for alle input. Det betyr at den muligens **ikke** er en algoritme i bokas forstand. Den forutsetter at vi kan skille mellom partall og oddetall.

Eksempel (Ubegrenset while-løkke)

1. Input x [$x \geq 1$ heltall.]
2. $y \leftarrow 0$
3. **While** $x > 1$ **do**
 - 3.1. **If** x er partall **then**
 - 3.1.1. $x \leftarrow \frac{x}{2}$
 - else**
 - 3.1.1. $x \leftarrow 3x + 1$

Kontrollstrukturer

Det er ingen som vet om algoritmen som er beskrevet i den neste pseudokoden vil **terminere** for alle input. Det betyr at den muligens **ikke** er en algoritme i bokas forstand. Den forutsetter at vi kan skille mellom partall og oddetall.

Eksempel (Ubegrenset while-løkke)

1. Input x [$x \geq 1$ heltall.]
2. $y \leftarrow 0$
3. **While** $x > 1$ **do**
 - 3.1. **If** x er partall **then**
 - 3.1.1. $x \leftarrow \frac{x}{2}$
 - else**
 - 3.1.1. $x \leftarrow 3x + 1$
- 3.2. $y \leftarrow y + 1$

Kontrollstrukturer

Det er ingen som vet om algoritmen som er beskrevet i den neste pseudokoden vil **terminere** for alle input. Det betyr at den muligens **ikke** er en algoritme i bokas forstand. Den forutsetter at vi kan skille mellom partall og oddetall.

Eksempel (Ubegrenset while-løkke)

1. Input x [$x \geq 1$ heltall.]
2. $y \leftarrow 0$
3. **While** $x > 1$ **do**
 - 3.1. **If** x er partall **then**
 - 3.1.1. $x \leftarrow \frac{x}{2}$
 - else**
 - 3.1.1. $x \leftarrow 3x + 1$
 - 3.2. $y \leftarrow y + 1$
4. Output y

Kontrollstrukturer

Kontrollstrukturer

Vi kan finne en enkel pseudokode for å finne ledd nr. n i følgen

1, 1, 2, 3, 5, 8, 13, 21, 34, 55, \dots

av **Fibonacci**-tall:

Kontrollstrukturer

Vi kan finne en enkel pseudokode for å finne ledd nr. n i følgen

1, 1, 2, 3, 5, 8, 13, 21, 34, 55, \dots

av **Fibonacci**-tall:

Eksempel (Fibonacci)

Kontrollstrukturer

Vi kan finne en enkel pseudokode for å finne ledd nr. n i følgen

1, 1, 2, 3, 5, 8, 13, 21, 34, 55, \dots

av **Fibonacci**-tall:

Eksempel (Fibonacci)

1. Input n [$n \geq 1$ heltall]

Kontrollstrukturer

Vi kan finne en enkel pseudokode for å finne ledd nr. n i følgen

1, 1, 2, 3, 5, 8, 13, 21, 34, 55, \dots

av **Fibonacci**-tall:

Eksempel (Fibonacci)

1. Input n [$n \geq 1$ heltall]
2. $x \leftarrow 1$

Kontrollstrukturer

Vi kan finne en enkel pseudokode for å finne ledd nr. n i følgen

1, 1, 2, 3, 5, 8, 13, 21, 34, 55, \dots

av **Fibonacci**-tall:

Eksempel (Fibonacci)

1. Input n [$n \geq 1$ heltall]
2. $x \leftarrow 1$
3. $y \leftarrow 1$

Kontrollstrukturer

Vi kan finne en enkel pseudokode for å finne ledd nr. n i følgen

1, 1, 2, 3, 5, 8, 13, 21, 34, 55, \dots

av **Fibonacci**-tall:

Eksempel (Fibonacci)

1. Input n [$n \geq 1$ heltall]
2. $x \leftarrow 1$
3. $y \leftarrow 1$
4. **For** $i = 2$ **to** n **do**

Kontrollstrukturer

Vi kan finne en enkel pseudokode for å finne ledd nr. n i følgen

1, 1, 2, 3, 5, 8, 13, 21, 34, 55, \dots

av **Fibonacci**-tall:

Eksempel (Fibonacci)

1. Input n [$n \geq 1$ heltall]
2. $x \leftarrow 1$
3. $y \leftarrow 1$
4. **For** $i = 2$ **to** n **do**
 - 4.1. $z \leftarrow x$

Kontrollstrukturer

Vi kan finne en enkel pseudokode for å finne ledd nr. n i følgen

1, 1, 2, 3, 5, 8, 13, 21, 34, 55, \dots

av **Fibonacci**-tall:

Eksempel (Fibonacci)

1. Input n [$n \geq 1$ heltall]
2. $x \leftarrow 1$
3. $y \leftarrow 1$
4. **For** $i = 2$ **to** n **do**
 - 4.1. $z \leftarrow x$
 - 4.2. $x \leftarrow x + y$

Kontrollstrukturer

Vi kan finne en enkel pseudokode for å finne ledd nr. n i følgen

1, 1, 2, 3, 5, 8, 13, 21, 34, 55, \dots

av **Fibonacci**-tall:

Eksempel (Fibonacci)

1. Input n [$n \geq 1$ heltall]
2. $x \leftarrow 1$
3. $y \leftarrow 1$
4. **For** $i = 2$ **to** n **do**
 - 4.1. $z \leftarrow x$
 - 4.2. $x \leftarrow x + y$
 - 4.3. $y \leftarrow z$

Kontrollstrukturer

Vi kan finne en enkel pseudokode for å finne ledd nr. n i følgen

1, 1, 2, 3, 5, 8, 13, 21, 34, 55, \dots

av **Fibonacci**-tall:

Eksempel (Fibonacci)

1. Input n [$n \geq 1$ heltall]
2. $x \leftarrow 1$
3. $y \leftarrow 1$
4. **For** $i = 2$ **to** n **do**
 - 4.1. $z \leftarrow x$
 - 4.2. $x \leftarrow x + y$
 - 4.3. $y \leftarrow z$
5. Output x

Kontrollstrukturer

Kontrollstrukturer

- Parenteser i forskjellige former brukes mye i matematikk, informatikk og spesielt i programmer.

Kontrollstrukturer

- Parenteser i forskjellige former brukes mye i matematikk, informatikk og spesielt i programmer.
- Eksempelvis, i \LaTeX forekommer “parenteser” som f.eks.

Kontrollstrukturer

- Parenteser i forskjellige former brukes mye i matematikk, informatikk og spesielt i programmer.
- Eksempelvis, i \LaTeX forekommer “parenteser” som f.eks.

`\begin{center} ... \end{center}`

Kontrollstrukturer

- Parenteser i forskjellige former brukes mye i matematikk, informatikk og spesielt i programmer.
- Eksempelvis, i \LaTeX forekommer “parenteser” som f.eks.

`\begin{center} ... \end{center}`

og

Kontrollstrukturer

- Parenteser i forskjellige former brukes mye i matematikk, informatikk og spesielt i programmer.
- Eksempelvis, i \LaTeX forekommer “parenteser” som f.eks.

`\begin{center} ... \end{center}`

og

`\begin{itemize} ... \end{itemize}`

Kontrollstrukturer

- Parenteser i forskjellige former brukes mye i matematikk, informatikk og spesielt i programmer.
- Eksempelvis, i \LaTeX forekommer “parenteser” som f.eks.

`\begin{center} ... \end{center}`

og

`\begin{itemize} ... \end{itemize}`

og det er viktig at de står riktig i forhold til hverandre.

Kontrollstrukturer

- Parenteser i forskjellige former brukes mye i matematikk, informatikk og spesielt i programmer.
- Eksempelvis, i \LaTeX forekommer “parenteser” som f.eks.

`\begin{center} ... \end{center}`

og

`\begin{itemize} ... \end{itemize}`

og det er viktig at de står riktig i forhold til hverandre.

- Dette kontrolleres når dokumentet eller programmer kompileres.

Kontrollstrukturer

- Parenteser i forskjellige former brukes mye i matematikk, informatikk og spesielt i programmer.
- Eksempelvis, i \LaTeX forekommer “parenteser” som f.eks.

`\begin{center} ... \end{center}`

og

`\begin{itemize} ... \end{itemize}`

og det er viktig at de står riktig i forhold til hverandre.

- Dette kontrolleres når dokumentet eller programmer kompileres.
- Det neste eksemplet på en pseudokode er en parentes-sjekker.

Kontrollstrukturer

Kontrollstrukturer

- Vi kan kontrollere om en liste venstre og høyreparenteser

$((()((())))()((())))()$

er lovlig eller ikke, ved å telle opp og ned – opp ved (og ned ved)
– fra venstre mot høyre.

Kontrollstrukturer

- Vi kan kontrollere om en liste venstre og høyreparenteser

$((()((())))(())((())))(())$

er lovlig eller ikke, ved å telle opp og ned – opp ved (og ned ved)
– fra venstre mot høyre.

- Hvis vi til slutt ser at vi har like mange parenteser av hvert slag, og aldri underveis har flere) enn (, er uttrykket i orden.

Kontrollstrukturer

- Vi kan kontrollere om en liste venstre og høyreparenteser

$((()((())))()((())))()$

er lovlig eller ikke, ved å telle opp og ned – opp ved (og ned ved)
– fra venstre mot høyre.

- Hvis vi til slutt ser at vi har like mange parenteser av hvert slag, og aldri underveis har flere) enn (, er uttrykket i orden.
- I det neste eksemplet gir vi to input, lengden av uttrykket og sekvensen av parenteser.

Kontrollstrukturer

- Vi kan kontrollere om en liste venstre og høyreparenteser

$((()((())))(())((())))(())$

er lovlig eller ikke, ved å telle opp og ned – opp ved (og ned ved)
– fra venstre mot høyre.

- Hvis vi til slutt ser at vi har like mange parenteser av hvert slag, og aldri underveis har flere) enn (, er uttrykket i orden.
- I det neste eksemplet gir vi to input, lengden av uttrykket og sekvensen av parenteser.
- Vi sjekker uttrykket fra venstre mot høyre.

Kontrollstrukturer

- Vi kan kontrollere om en liste venstre og høyreparenteser

$$((()((())))(()((())))(())$$

er lovlig eller ikke, ved å telle opp og ned – opp ved (og ned ved) – fra venstre mot høyre.

- Hvis vi til slutt ser at vi har like mange parenteser av hvert slag, og aldri underveis har flere) enn (, er uttrykket i orden.
- I det neste eksemplet gir vi to input, lengden av uttrykket og sekvensen av parenteser.
- Vi sjekker uttrykket fra venstre mot høyre.
- Vi bruker variabelen `val` til å holde orden på om sekvensen så langt er i orden.

Kontrollstrukturer

- Vi kan kontrollere om en liste venstre og høyreparenteser

$$((()((())))(()((())))(())$$

er lovlig eller ikke, ved å telle opp og ned – opp ved (og ned ved) – fra venstre mot høyre.

- Hvis vi til slutt ser at vi har like mange parenteser av hvert slag, og aldri underveis har flere) enn (, er uttrykket i orden.
- I det neste eksemplet gir vi to input, lengden av uttrykket og sekvensen av parenteser.
- Vi sjekker uttrykket fra venstre mot høyre.
- Vi bruker variabelen `val` til å holde orden på om sekvensen så langt er i orden.
- Vi bruker variabelen `y` til å telle overskuddet av (.

Kontrollstrukturer

Kontrollstrukturer

1. Input n [Lengden av uttrykket, antall parenteser totalt]

Kontrollstrukturer

1. Input n [Lengden av uttrykket, antall parenteser totalt]
2. Input $x_1 \cdots x_n$ [En liste av venstre og høyreparenteser]

Kontrollstrukturer

1. Input n [Lengden av uttrykket, antall parenteser totalt]
2. Input $x_1 \cdots x_n$ [En liste av venstre og høyreparenteser]
3. $y \leftarrow 0$

Kontrollstrukturer

1. Input n [Lengden av uttrykket, antall parenteser totalt]
2. Input $x_1 \cdots x_n$ [En liste av venstre og høyreparenteser]
3. $y \leftarrow 0$
4. $val \leftarrow \text{JA}$

Kontrollstrukturer

1. Input n [Lengden av uttrykket, antall parenteser totalt]
2. Input $x_1 \cdots x_n$ [En liste av venstre og høyreparenteser]
3. $y \leftarrow 0$
4. $val \leftarrow \text{JA}$
5. **For** $i = 1$ **to** n **do**

Kontrollstrukturer

1. Input n [Lengden av uttrykket, antall parenteser totalt]
2. Input $x_1 \cdots x_n$ [En liste av venstre og høyreparenteser]
3. $y \leftarrow 0$
4. $val \leftarrow \text{JA}$
5. **For** $i = 1$ **to** n **do**
 - 5.1. **If** $x_i = ($ **then**

else

Kontrollstrukturer

1. Input n [Lengden av uttrykket, antall parenteser totalt]
2. Input $x_1 \cdots x_n$ [En liste av venstre og høyreparenteser]
3. $y \leftarrow 0$
4. $val \leftarrow \text{JA}$
5. **For** $i = 1$ **to** n **do**
 - 5.1. **If** $x_i = ($ **then**
 - 5.1.1. $y \leftarrow y + 1$
 - else**

Kontrollstrukturer

1. Input n [Lengden av uttrykket, antall parenteser totalt]
2. Input $x_1 \cdots x_n$ [En liste av venstre og høyreparenteser]
3. $y \leftarrow 0$
4. $val \leftarrow \text{JA}$
5. **For** $i = 1$ **to** n **do**
 - 5.1. **If** $x_i = ($ **then**
 - 5.1.1. $y \leftarrow y + 1$
 - else**
 - 5.1.2. **If** $y = 0$ **then**

Kontrollstrukturer

1. Input n [Lengden av uttrykket, antall parenteser totalt]
2. Input $x_1 \cdots x_n$ [En liste av venstre og høyreparenteser]
3. $y \leftarrow 0$
4. $val \leftarrow \text{JA}$
5. **For** $i = 1$ **to** n **do**
 - 5.1. **If** $x_i = ($ **then**
 - 5.1.1. $y \leftarrow y + 1$
 - else**
 - 5.1.2. **If** $y = 0$ **then**
 - 5.1.2.1. $val \leftarrow \text{NEI}$

Kontrollstrukturer

1. Input n [Lengden av uttrykket, antall parenteser totalt]
2. Input $x_1 \cdots x_n$ [En liste av venstre og høyreparenteser]
3. $y \leftarrow 0$
4. $val \leftarrow \text{JA}$
5. **For** $i = 1$ **to** n **do**
 - 5.1. **If** $x_i = ($ **then**
 - 5.1.1. $y \leftarrow y + 1$
 - else**
 - 5.1.2. **If** $y = 0$ **then**
 - 5.1.2.1. $val \leftarrow \text{NEI}$
 - else**

Kontrollstrukturer

1. Input n [Lengden av uttrykket, antall parenteser totalt]
2. Input $x_1 \cdots x_n$ [En liste av venstre og høyreparenteser]
3. $y \leftarrow 0$
4. $val \leftarrow \text{JA}$
5. **For** $i = 1$ **to** n **do**
 - 5.1. **If** $x_i = ($ **then**
 - 5.1.1. $y \leftarrow y + 1$
 - else**
 - 5.1.2. **If** $y = 0$ **then**
 - 5.1.2.1. $val \leftarrow \text{NEI}$
 - else**
 - 5.1.2.2. $y \leftarrow y - 1$

Kontrollstrukturer

1. Input n [Lengden av uttrykket, antall parenteser totalt]
2. Input $x_1 \cdots x_n$ [En liste av venstre og høyreparenteser]
3. $y \leftarrow 0$
4. $val \leftarrow \text{JA}$
5. **For** $i = 1$ **to** n **do**
 - 5.1. **If** $x_i = ($ **then**
 - 5.1.1. $y \leftarrow y + 1$
 - else**
 - 5.1.2. **If** $y = 0$ **then**
 - 5.1.2.1. $val \leftarrow \text{NEI}$
 - else**
 - 5.1.2.2. $y \leftarrow y - 1$
6. **If** $y > 0$ **then**

Kontrollstrukturer

1. Input n [Lengden av uttrykket, antall parenteser totalt]
2. Input $x_1 \cdots x_n$ [En liste av venstre og høyreparenteser]
3. $y \leftarrow 0$
4. $val \leftarrow \text{JA}$
5. **For** $i = 1$ **to** n **do**
 - 5.1. **If** $x_i = ($ **then**
 - 5.1.1. $y \leftarrow y + 1$
 - else**
 - 5.1.2. **If** $y = 0$ **then**
 - 5.1.2.1. $val \leftarrow \text{NEI}$
 - else**
 - 5.1.2.2. $y \leftarrow y - 1$
6. **If** $y > 0$ **then**
 - 6.1. $val \leftarrow \text{NEI}$

Kontrollstrukturer

1. Input n [Lengden av uttrykket, antall parenteser totalt]
2. Input $x_1 \cdots x_n$ [En liste av venstre og høyreparenteser]
3. $y \leftarrow 0$
4. $val \leftarrow \text{JA}$
5. **For** $i = 1$ **to** n **do**
 - 5.1. **If** $x_i = ($ **then**
 - 5.1.1. $y \leftarrow y + 1$
 - else**
 - 5.1.2. **If** $y = 0$ **then**
 - 5.1.2.1. $val \leftarrow \text{NEI}$
 - else**
 - 5.1.2.2. $y \leftarrow y - 1$
6. **If** $y > 0$ **then**
 - 6.1. $val \leftarrow \text{NEI}$
7. Output val

Hva skal dere kunne fra kapittel 1?

Hva skal dere kunne fra kapittel 1?

Forventede ferdigheter:

Hva skal dere kunne fra kapittel 1?

Forventede ferdigheter:

- Kunne uttrykke en algoritme i pseudokode, og kunne bruke de forskjellige kontrollstrukturene på riktig måte.

Hva skal dere kunne fra kapittel 1?

Forventede ferdigheter:

- Kunne uttrykke en algoritme i pseudokode, og kunne bruke de forskjellige kontrollstrukturene på riktig måte.
- Kunne følge en algoritme gitt ved en pseudokode og inputverdier på variablene skritt for skritt, og kunne holde orden på hvordan verdiene på variablene endrer seg under “utregningen”.

Hva skal dere kunne fra kapittel 1?

Forventede ferdigheter:

- Kunne uttrykke en algoritme i pseudokode, og kunne bruke de forskjellige kontrollstrukturene på riktig måte.
- Kunne følge en algoritme gitt ved en pseudokode og inputverdier på variablene skritt for skritt, og kunne holde orden på hvordan verdiene på variablene endrer seg under “utregningen”.
- Kunne forklare hvorfor en pseudokode løser den oppgaven den er satt til å utføre,

.

Hva skal dere kunne fra kapittel 1?

Forventede ferdigheter:

- Kunne uttrykke en algoritme i pseudokode, og kunne bruke de forskjellige kontrollstrukturene på riktig måte.
- Kunne følge en algoritme gitt ved en pseudokode og inputverdier på variablene skritt for skritt, og kunne holde orden på hvordan verdiene på variablene endrer seg under “utregningen”.
- Kunne forklare hvorfor en pseudokode løser den oppgaven den er satt til å utføre, det vil si, kunne gi en muntlig eller skriftlig *dokumentasjon*.

Kapittel 2: Representasjon av tall

Hva som gjennomgås i forelesningene

Hva som gjennomgås i forelesningene

- Hele kapittel 2 og 3 er pensum.

Hva som gjennomgås i forelesningene

- Hele kapittel 2 og 3 er pensum.
- Siden dette stoffet er kjent for mange, så overlates mesteparten av dette til den enkelte.

Hva som gjennomgås i forelesningene

- Hele kapittel 2 og 3 er pensum.
- Siden dette stoffet er kjent for mange, så overlates mesteparten av dette til den enkelte.
- Det er lite i de senere kapitlene som avhenger direkte av det som står i kapittel 2 og 3.

Hva som gjennomgås i forelesningene

- Hele kapittel 2 og 3 er pensum.
- Siden dette stoffet er kjent for mange, så overlates mesteparten av dette til den enkelte.
- Det er lite i de senere kapitlene som avhenger direkte av det som står i kapittel 2 og 3.
- I boken står det f.eks. godt forklart – ved hjelp av algoritmer beskrevet ved pseudokoder – hvordan man konverterer fra binære tall til desimaltall og vice versa.

Hva som gjennomgås i forelesningene

- Hele kapittel 2 og 3 er pensum.
- Siden dette stoffet er kjent for mange, så overlates mesteparten av dette til den enkelte.
- Det er lite i de senere kapitlene som avhenger direkte av det som står i kapittel 2 og 3.
- I boken står det f.eks. godt forklart – ved hjelp av algoritmer beskrevet ved pseudokoder – hvordan man konverterer fra binære tall til desimaltall og vice versa.
- Vi skal gå raskt gjennom det viktigste her.

Tallmengder

Tallmengder

Hvilke tall vi betrakter er avhengig av hva vi ønsker å bruke dem til.

Tallmengder

Hvilke tall vi betrakter er avhengig av hva vi ønsker å bruke dem til. I MAT1030 vil vi stort sett betrakte følgende typer tall:

Tallmengder

Hvilke tall vi betrakter er avhengig av hva vi ønsker å bruke dem til. I MAT1030 vil vi stort sett betrakte følgende typer tall:

- Naturlige tall \mathbb{N}

1, 2, 3, ...

Tallmengder

Hvilke tall vi betrakter er avhengig av hva vi ønsker å bruke dem til. I MAT1030 vil vi stort sett betrakte følgende typer tall:

- Naturlige tall \mathbb{N}

$$1, 2, 3, \dots$$

- Hele tall \mathbb{Z}

$$\dots, -3, -2, -1, 0, 1, 2, \dots$$

Tallmengder

Hvilke tall vi betrakter er avhengig av hva vi ønsker å bruke dem til. I MAT1030 vil vi stort sett betrakte følgende typer tall:

- Naturlige tall \mathbb{N}

$$1, 2, 3, \dots$$

- Hele tall \mathbb{Z}

$$\dots, -3, -2, -1, 0, 1, 2, \dots$$

- Rasjonale tall \mathbb{Q}

Tall som kan skrives som en brøk $\frac{p}{q}$

Tallmengder

Hvilke tall vi betrakter er avhengig av hva vi ønsker å bruke dem til. I MAT1030 vil vi stort sett betrakte følgende typer tall:

- Naturlige tall \mathbb{N}

$$1, 2, 3, \dots$$

- Hele tall \mathbb{Z}

$$\dots, -3, -2, -1, 0, 1, 2, \dots$$

- Rasjonale tall \mathbb{Q}

Tall som kan skrives som en brøk $\frac{p}{q}$

- Reelle tall \mathbb{R}

“alle tallene”

Tallmengder

Tallmengder

- Mange mener at tall er punkter på tall-linja, og at det ikke spiller noen rolle om vi betrakter 2 som et naturlig tall, et heltall, et rasjonalt tall eller et reelt tall.

Tallmengder

- Mange mener at tall er punkter på tall-linja, og at det ikke spiller noen rolle om vi betrakter 2 som et naturlig tall, et heltall, et rasjonalt tall eller et reelt tall.
- I programmeringsammenheng kan det spille en stor rolle hva slags verdier en variabel kan få lov til å ta, og representasjonen av et tall som et dataobjekt kan variere med hva slags type tall vi betrakter.

Tallmengder

Tallmengder

Det finnes andre tallmengder som også er av interesse i matematikk og informatikk

Tallmengder

Det finnes andre tallmengder som også er av interesse i matematikk og informatikk, eksempelvis

Tallmengder

Det finnes andre tallmengder som også er av interesse i matematikk og informatikk, eksempelvis

- Komplekse tall

Tallmengder

Det finnes andre tallmengder som også er av interesse i matematikk og informatikk, eksempelvis

- Komplekse tall
- Algebraiske tall

Tallmengder

Det finnes andre tallmengder som også er av interesse i matematikk og informatikk, eksempelvis

- Komplekse tall
- Algebraiske tall
- Kvaternioner

Tallmengder

Det finnes andre tallmengder som også er av interesse i matematikk og informatikk, eksempelvis

- Komplekse tall
- Algebraiske tall
- Kvaternioner
- Ordinaltall

Representasjon av tall

Representasjon av tall

Så langt tilbake vi har informasjon om, har mennesker og kulturer hatt muntlig og skriftlig språk for tall.

Representasjon av tall

Så langt tilbake vi har informasjon om, har mennesker og kulturer hatt muntlig og skriftlig språk for tall.

Romertallet

MCMXXVIII

er en alternativ måte å skrive

1928

på.

Representasjon av tall

Så langt tilbake vi har informasjon om, har mennesker og kulturer hatt muntlig og skriftlig språk for tall.

Romertallet

MCMXXVIII

er en alternativ måte å skrive

1928

på.

Hvis vi blir bedt om å skrive et program for addisjon av to tall, betyr det mye om vi bruker den romerske eller dagens måte å skrive tall på.

Representasjon av tall

Representasjon av tall

- De tallene vi bruker til daglig kalles **desimaltall**, eller tall i 10-tallsystemet.

Representasjon av tall

- De tallene vi bruker til daglig kalles **desimaltall**, eller tall i **10-tallsystemet**.
- Dette er et **plass-siffersystem** med **basis 10**.

Representasjon av tall

- De tallene vi bruker til daglig kalles **desimaltall**, eller tall i **10-tallsystemet**.
- Dette er et **plass-siffersystem** med **basis 10**.
- Det betyr igjen at hvert **siffer** angir et antall 10'er potenser, og sifferets posisjon forteller oss hvor stor potensen er.

Representasjon av tall

Representasjon av tall

Eksempel

Representasjon av tall

Eksempel

- 258 står for

$$2 \cdot 10^2 + 5 \cdot 10^1 + 8 \cdot 10^0.$$

Representasjon av tall

Eksempel

- 258 står for

$$2 \cdot 10^2 + 5 \cdot 10^1 + 8 \cdot 10^0.$$

- 3,14 står for

$$3 \cdot 10^0 + 1 \cdot 10^{-1} + 4 \cdot 10^{-2}$$

Tverrsumtesten (digresjon)

Tverrsumtesten (digresjon)

Tverrsummen til et desimaltall er summen av alle sifrene.

Tverrsumtesten (digresjon)

Tverrsummen til et desimaltall er summen av alle sifrene.

Eksempel

Tverrsumtesten (digresjon)

Tverrsummen til et desimaltall er summen av alle sifrene.

Eksempel

- Tverrsummen til 234 er $2 + 3 + 4 = 9$

Tverrsumtesten (digresjon)

Tverrsummen til et desimaltall er summen av alle sifrene.

Eksempel

- Tverrsummen til 234 er $2 + 3 + 4 = 9$
- Tverrsummen til 15987 er $1 + 5 + 9 + 8 + 7 = 30$

Tverrsumtesten (digresjon)

Tverrsummen til et desimaltall er summen av alle sifrene.

Eksempel

- Tverrsummen til 234 er $2 + 3 + 4 = 9$
- Tverrsummen til 15987 er $1 + 5 + 9 + 8 + 7 = 30$
- Tverrsummen til 2825 er $2 + 8 + 2 + 5 = 17$

Tverrsumtesten (digresjon)

Tverrsummen til et desimaltall er summen av alle sifrene.

Eksempel

- Tverrsummen til 234 er $2 + 3 + 4 = 9$
- Tverrsummen til 15987 er $1 + 5 + 9 + 8 + 7 = 30$
- Tverrsummen til 2825 er $2 + 8 + 2 + 5 = 17$

- Legg merke til at resten vi får når vi deler tallet på 9 er det samme som vi får når vi deler tverrsummen på 9.

Tverrsumtesten (digresjon)

Tverrsummen til et desimaltall er summen av alle sifrene.

Eksempel

- Tverrsummen til 234 er $2 + 3 + 4 = 9$
 - Tverrsummen til 15987 er $1 + 5 + 9 + 8 + 7 = 30$
 - Tverrsummen til 2825 er $2 + 8 + 2 + 5 = 17$
-
- Legg merke til at resten vi får når vi deler tallet på 9 er det samme som vi får når vi deler tverrsummen på 9.
 - Kan dette forklares matematisk?

Tverrsumtesten (digresjon)

Tverrsumtesten (digresjon)

Påstand (Tverrsumtesten)

Hvis vi skriver et tall n på desimalform og lar $T(n)$ være tverrsummen til n , så får vi samme rest når vi deler n på 9 som når vi deler $T(n)$ på 9.

Tverrsumtesten (digresjon)

Påstand (Tverrsumtesten)

Hvis vi skriver et tall n på desimalform og lar $T(n)$ være tverrsummen til n , så får vi samme rest når vi deler n på 9 som når vi deler $T(n)$ på 9.

Bevis

Tverrsumtesten (digresjon)

Påstand (Tverrsumtesten)

Hvis vi skriver et tall n på desimalform og lar $T(n)$ være tverrsummen til n , så får vi samme rest når vi deler n på 9 som når vi deler $T(n)$ på 9.

Bevis

La $a_k \dots a_0$ være desimalformen til n .

Tverrsumtesten (digresjon)

Påstand (Tverrsumtesten)

Hvis vi skriver et tall n på desimalform og lar $T(n)$ være tverrsummen til n , så får vi samme rest når vi deler n på 9 som når vi deler $T(n)$ på 9.

Bevis

La $a_k \dots a_0$ være desimalformen til n .

Da er

$$n = a_k 10^k + \dots + a_1 10 + a_0$$

Tverrsumtesten (digresjon)

Påstand (Tverrsumtesten)

Hvis vi skriver et tall n på desimalform og lar $T(n)$ være tverrsummen til n , så får vi samme rest når vi deler n på 9 som når vi deler $T(n)$ på 9.

Bevis

La $a_k \dots a_0$ være desimalformen til n .

Da er

$$n = a_k 10^k + \dots + a_1 10 + a_0$$

Når $1 \leq i \leq k$ kan $10^i - 1$ deles på 9, siden sifrene består av bare 9-tall.

Tverrsumtesten (digresjon)

Påstand (Tverrsumtesten)

Hvis vi skriver et tall n på desimalform og lar $T(n)$ være tverrsummen til n , så får vi samme rest når vi deler n på 9 som når vi deler $T(n)$ på 9.

Bevis

La $a_k \dots a_0$ være desimalformen til n .

Da er

$$n = a_k 10^k + \dots + a_1 10 + a_0$$

Når $1 \leq i \leq k$ kan $10^i - 1$ deles på 9, siden sifrene består av bare 9-tall.

Vi har at

Tverrsumtesten (digresjon)

Påstand (Tverrsumtesten)

Hvis vi skriver et tall n på desimalform og lar $T(n)$ være tverrsummen til n , så får vi samme rest når vi deler n på 9 som når vi deler $T(n)$ på 9.

Bevis

La $a_k \dots a_0$ være desimalformen til n .

Da er

$$n = a_k 10^k + \dots + a_1 10 + a_0$$

Når $1 \leq i \leq k$ kan $10^i - 1$ deles på 9, siden sifrene består av bare 9-tall.

Vi har at

$$n = T(n) + a_k(10^k - 1) + \dots + a_1(10 - 1)$$

Tverrsumtesten (digresjon)

Påstand (Tverrsumtesten)

Hvis vi skriver et tall n på desimalform og lar $T(n)$ være tverrsummen til n , så får vi samme rest når vi deler n på 9 som når vi deler $T(n)$ på 9.

Bevis

La $a_k \dots a_0$ være desimalformen til n .

Da er

$$n = a_k 10^k + \dots + a_1 10 + a_0$$

Når $1 \leq i \leq k$ kan $10^i - 1$ deles på 9, siden sifrene består av bare 9-tall.

Vi har at

$$n = T(n) + a_k(10^k - 1) + \dots + a_1(10 - 1)$$

(trenger litt ettertanke)

Tverrsumtesten (digresjon)

Påstand (Tverrsumtesten)

Hvis vi skriver et tall n på desimalform og lar $T(n)$ være tverrsummen til n , så får vi samme rest når vi deler n på 9 som når vi deler $T(n)$ på 9.

Bevis

La $a_k \dots a_0$ være desimalformen til n .

Da er

$$n = a_k 10^k + \dots + a_1 10 + a_0$$

Når $1 \leq i \leq k$ kan $10^i - 1$ deles på 9, siden sifrene består av bare 9-tall.

Vi har at

$$n = T(n) + a_k(10^k - 1) + \dots + a_1(10 - 1)$$

(trenger litt ettertanke), og påstanden følger

Tverrsumtesten (digresjon)

Påstand (Tverrsumtesten)

Hvis vi skriver et tall n på desimalform og lar $T(n)$ være tverrsummen til n , så får vi samme rest når vi deler n på 9 som når vi deler $T(n)$ på 9.

Bevis

La $a_k \dots a_0$ være desimalformen til n .

Da er

$$n = a_k 10^k + \dots + a_1 10 + a_0$$

Når $1 \leq i \leq k$ kan $10^i - 1$ deles på 9, siden sifrene består av bare 9-tall.

Vi har at

$$n = T(n) + a_k(10^k - 1) + \dots + a_1(10 - 1)$$

(trenger litt ettertanke), og påstanden følger (trenger litt ettertanke til).

Tverrsumtesten (digresjon)

Tverrsumtesten (digresjon)

Vi får også at vi får den samme resten om vi deler et tall på 3 som når vi deler tverrsummen på 3.

Tverrsumtesten (digresjon)

Vi får også at vi får den samme resten om vi deler et tall på 3 som når vi deler tverrsummen på 3.

Det betyr at vi kan kontrollere om et tall kan deles på 3 ved å se på tverrsummen.

Tverrsumtesten (digresjon)

Vi får også at vi får den samme resten om vi deler et tall på 3 som når vi deler tverrsummen på 3.

Det betyr at vi kan kontrollere om et tall kan deles på 3 ved å se på tverrsummen.

Er tallet veldig stort, kan vi se på tverrsummen av tverrsummen, osv.

Binære tall

Binære tall

- Det er kulturelt betinget at vi bruker 10 som basis i tallsystemet vårt.

Binære tall

- Det er kulturelt betinget at vi bruker 10 som basis i tallsystemet vårt.
- Alle tall > 1 kan i prinsippet brukes.

Binære tall

- Det er kulturelt betinget at vi bruker 10 som basis i tallsystemet vårt.
- Alle tall > 1 kan i prinsippet brukes.
- I informatikksammenheng er det like naturlig å bruke 2, 8 og 16 som basistall.

Binære tall

- Det er kulturelt betinget at vi bruker 10 som basis i tallsystemet vårt.
- Alle tall > 1 kan i prinsippet brukes.
- I informatikksammenheng er det like naturlig å bruke 2, 8 og 16 som basistall.
- Bruker vi 2 som basis, sier vi at tallet er på **binær form**.

Binære tall

Eksempel

Eksempel

Vi tolker en binær form (hvor alle sifrene er 0 eller 1) omtrent som om det var et desimaltall, bortsett fra at vi erstatter 10 med 2:

Binære tall

Eksempel

Vi tolker en binær form (hvor alle sifrene er 0 eller 1) omtrent som om det var et desimaltall, bortsett fra at vi erstatter 10 med 2:

- $1010_2 = 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0 = 8 + 2 = 10_{10}$

Binære tall

Eksempel

Vi tolker en binær form (hvor alle sifrene er 0 eller 1) omtrent som om det var et desimaltall, bortsett fra at vi erstatter 10 med 2:

- $1010_2 = 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0 = 8 + 2 = 10_{10}$
- $11011_2 = 16 + 8 + 2 + 1 = 27$

Eksempel

Vi tolker en binær form (hvor alle sifrene er 0 eller 1) omtrent som om det var et desimaltall, bortsett fra at vi erstatter 10 med 2:

- $1010_2 = 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0 = 8 + 2 = 10_{10}$
- $11011_2 = 16 + 8 + 2 + 1 = 27$
- $100111001_2 = 256 + 32 + 16 + 8 + 1 = 313$

Eksempel

Vi tolker en binær form (hvor alle sifrene er 0 eller 1) omtrent som om det var et desimaltall, bortsett fra at vi erstatter 10 med 2:

- $1010_2 = 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0 = 8 + 2 = 10_{10}$
- $11011_2 = 16 + 8 + 2 + 1 = 27$
- $100111001_2 = 256 + 32 + 16 + 8 + 1 = 313$

Binær representasjon kan selvfølgelig også brukes til tall som er mindre enn 1.

Eksempel

Vi tolker en binær form (hvor alle sifrene er 0 eller 1) omtrent som om det var et desimaltall, bortsett fra at vi erstatter 10 med 2:

- $1010_2 = 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0 = 8 + 2 = 10_{10}$
- $11011_2 = 16 + 8 + 2 + 1 = 27$
- $100111001_2 = 256 + 32 + 16 + 8 + 1 = 313$

Binær representasjon kan selvfølgelig også brukes til tall som er mindre enn 1.

- $0,100101_2 = \frac{1}{2} + \frac{1}{16} + \frac{1}{64} = \frac{32+4+1}{64} = \frac{100101_2}{64}$

Eksempel

Vi tolker en binær form (hvor alle sifrene er 0 eller 1) omtrent som om det var et desimaltall, bortsett fra at vi erstatter 10 med 2:

- $1010_2 = 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0 = 8 + 2 = 10_{10}$
- $11011_2 = 16 + 8 + 2 + 1 = 27$
- $100111001_2 = 256 + 32 + 16 + 8 + 1 = 313$

Binær representasjon kan selvfølgelig også brukes til tall som er mindre enn 1.

- $0,100101_2 = \frac{1}{2} + \frac{1}{16} + \frac{1}{64} = \frac{32+4+1}{64} = \frac{100101_2}{64}$
- $0,01101_2 = \frac{1}{4} + \frac{1}{8} + \frac{1}{32} = \frac{8+4+1}{32} = \frac{01101_2}{32}$

Binære tall

Binære tall

Det finnes en enkel prosedyre for å regne ut verdien av et binært tall:

Binære tall

Det finnes en enkel prosedyre for å regne ut verdien av et binært tall:

1. Input n [n er antall sifre i sekvensen]

Binære tall

Det finnes en enkel prosedyre for å regne ut verdien av et binært tall:

1. Input n [n er antall sifre i sekvensen]
2. Input $x_1 \dots x_n$ [en sekvens av 0'er og 1'ere]

Binære tall

Det finnes en enkel prosedyre for å regne ut verdien av et binært tall:

1. Input n [n er antall sifre i sekvensen]
2. Input $x_1 \dots x_n$ [en sekvens av 0'er og 1'ere]
3. $y \leftarrow 0$ [y skal bli verdien på sekvensen tolket som et binært tall]

Binære tall

Det finnes en enkel prosedyre for å regne ut verdien av et binært tall:

1. Input n [n er antall sifre i sekvensen]
2. Input $x_1 \dots x_n$ [en sekvens av 0'er og 1'ere]
3. $y \leftarrow 0$ [y skal bli verdien på sekvensen tolket som et binært tall]
4. **For** $i = 1$ **to** n **do**

Binære tall

Det finnes en enkel prosedyre for å regne ut verdien av et binært tall:

1. Input n [n er antall sifre i sekvensen]
2. Input $x_1 \dots x_n$ [en sekvens av 0'ere og 1'ere]
3. $y \leftarrow 0$ [y skal bli verdien på sekvensen tolket som et binært tall]
4. **For** $i = 1$ **to** n **do**
 - 4.1. $y \leftarrow 2y$

Binære tall

Det finnes en enkel prosedyre for å regne ut verdien av et binært tall:

1. Input n [n er antall sifre i sekvensen]
2. Input $x_1 \dots x_n$ [en sekvens av 0'ere og 1'ere]
3. $y \leftarrow 0$ [y skal bli verdien på sekvensen tolket som et binært tall]
4. **For** $i = 1$ **to** n **do**
 - 4.1. $y \leftarrow 2y$
 - 4.2. **If** $x_i = 1$ **then**

Binære tall

Det finnes en enkel prosedyre for å regne ut verdien av et binært tall:

1. Input n [n er antall sifre i sekvensen]
2. Input $x_1 \dots x_n$ [en sekvens av 0'ere og 1'ere]
3. $y \leftarrow 0$ [y skal bli verdien på sekvensen tolket som et binært tall]
4. **For** $i = 1$ **to** n **do**
 - 4.1. $y \leftarrow 2y$
 - 4.2. **If** $x_i = 1$ **then**
 - 4.2.1. $y \leftarrow y + 1$

Binære tall

Det finnes en enkel prosedyre for å regne ut verdien av et binært tall:

1. Input n [n er antall sifre i sekvensen]
2. Input $x_1 \dots x_n$ [en sekvens av 0'ere og 1'ere]
3. $y \leftarrow 0$ [y skal bli verdien på sekvensen tolket som et binært tall]
4. **For** $i = 1$ **to** n **do**
 - 4.1. $y \leftarrow 2y$
 - 4.2. **If** $x_i = 1$ **then**
 - 4.2.1. $y \leftarrow y + 1$
5. Output y

Binære tall

Det finnes en enkel prosedyre for å regne ut verdien av et binært tall:

1. Input n [n er antall sifre i sekvensen]
2. Input $x_1 \dots x_n$ [en sekvens av 0'ere og 1'ere]
3. $y \leftarrow 0$ [y skal bli verdien på sekvensen tolket som et binært tall]
4. **For** $i = 1$ **to** n **do**
 - 4.1. $y \leftarrow 2y$
 - 4.2. **If** $x_i = 1$ **then**
 - 4.2.1. $y \leftarrow y + 1$
5. Output y

Regn f.eks. ut hva som skjer med input $n = 4$ og $x_1x_2x_3x_4 = 1101$.

Aritmetikk

Aritmetikk

Vi utfører addisjon, subtraksjon, multiplikasjon og divisjon av tall på binær form omtrent som for tall i 10-tallsystemet, bortsett fra at alt i prinsippet blir mye enklere, den lille addisjonstabellen og den lille multiplikasjonstabellen blir så mye mindre.

Aritmetikk

Vi utfører addisjon, subtraksjon, multiplikasjon og divisjon av tall på binær form omtrent som for tall i 10-tallsystemet, bortsett fra at alt i prinsippet blir mye enklere, den lille addisjonstabellen og den lille multiplikasjonstabellen blir så mye mindre.

Som eksempler regner vi eventuelt følgende stykker på tavla (oppgaver for den som ikke er på forelesningen).

Aritmetikk

Vi utfører addisjon, subtraksjon, multiplikasjon og divisjon av tall på binær form omtrent som for tall i 10-tallsystemet, bortsett fra at alt i prinsippet blir mye enklere, den lille addisjonstabellen og den lille multiplikasjonstabellen blir så mye mindre.

Som eksempler regner vi eventuelt følgende stykker på tavla (oppgaver for den som ikke er på forelesningen).

- $17 + 14$

Aritmetikk

Vi utfører addisjon, subtraksjon, multiplikasjon og divisjon av tall på binær form omtrent som for tall i 10-tallsystemet, bortsett fra at alt i prinsippet blir mye enklere, den lille addisjonstabellen og den lille multiplikasjonstabellen blir så mye mindre.

Som eksempler regner vi eventuelt følgende stykker på tavla (oppgaver for den som ikke er på forelesningen).

- $17 + 14$
- $17 - 14$

Aritmetikk

Vi utfører addisjon, subtraksjon, multiplikasjon og divisjon av tall på binær form omtrent som for tall i 10-tallsystemet, bortsett fra at alt i prinsippet blir mye enklere, den lille addisjonstabellen og den lille multiplikasjonstabellen blir så mye mindre.

Som eksempler regner vi eventuelt følgende stykker på tavla (oppgaver for den som ikke er på forelesningen).

- $17 + 14$
- $17 - 14$
- $5 \cdot 11$

Aritmetikk

Vi utfører addisjon, subtraksjon, multiplikasjon og divisjon av tall på binær form omtrent som for tall i 10-tallsystemet, bortsett fra at alt i prinsippet blir mye enklere, den lille addisjonstabellen og den lille multiplikasjonstabellen blir så mye mindre.

Som eksempler regner vi eventuelt følgende stykker på tavla (oppgaver for den som ikke er på forelesningen).

- $17 + 14$
- $17 - 14$
- $5 \cdot 11$
- $11 : 5$ med fire siffer bak komma.

Aritmetikk

Vi utfører addisjon, subtraksjon, multiplikasjon og divisjon av tall på binær form omtrent som for tall i 10-tallsystemet, bortsett fra at alt i prinsippet blir mye enklere, den lille addisjonstabellen og den lille multiplikasjonstabellen blir så mye mindre.

Som eksempler regner vi eventuelt følgende stykker på tavla (oppgaver for den som ikke er på forelesningen).

- $17 + 14$
- $17 - 14$
- $5 \cdot 11$
- $11 : 5$ med fire siffer bak komma.

Det er selvfølgelig mulig å finne pseudokoder som uttrykker de algoritmene vi vil bruke, men som i skolematematikken er det her best å demonstrere algoritmene ved eksempler.

Oktal og heksadesimal form

Oktal og heksadesimal form

Hvis man bruker 8-tallsystemet arbeider man med tall på **oktal** form.

Oktal og heksadesimal form

Hvis man bruker 8-tallsystemet arbeider man med tall på **oktal** form.
Eksempelvis vil vi ha

Oktal og heksadesimal form

Hvis man bruker 8-tallsystemet arbeider man med tall på **oktal** form. Eksempelvis vil vi ha

- $443_8 = 4 \cdot 8^2 + 4 \cdot 8 + 3 = 256 + 32 + 3 = 291_{10}$

Oktal og heksadesimal form

Hvis man bruker 8-tallsystemet arbeider man med tall på **oktal** form.

Eksempelvis vil vi ha

- $443_8 = 4 \cdot 8^2 + 4 \cdot 8 + 3 = 256 + 32 + 3 = 291_{10}$
- $3,21_8 = 3 + 2 \cdot \frac{1}{8} + \frac{1}{64}$

Oktal og heksadesimal form

Hvis man bruker 8-tallsystemet arbeider man med tall på **oktal** form.

Eksempelvis vil vi ha

- $443_8 = 4 \cdot 8^2 + 4 \cdot 8 + 3 = 256 + 32 + 3 = 291_{10}$
- $3,21_8 = 3 + 2 \cdot \frac{1}{8} + \frac{1}{64}$

Hvis man bruker 16-tallsystemet arbeider man med tall på **heksadesimal** form.

Oktal og heksadesimal form

Hvis man bruker 8-tallsystemet arbeider man med tall på **oktal** form.

Eksempelvis vil vi ha

- $443_8 = 4 \cdot 8^2 + 4 \cdot 8 + 3 = 256 + 32 + 3 = 291_{10}$
- $3,21_8 = 3 + 2 \cdot \frac{1}{8} + \frac{1}{64}$

Hvis man bruker 16-tallsystemet arbeider man med tall på **heksadesimal** form.

Her må man supplere symbolene $0, 1, \dots, 9$ med sifre A, B, C, D, E og F.

Oktal og heksadesimal form

Hvis man bruker 8-tallsystemet arbeider man med tall på **oktal** form. Eksempelvis vil vi ha

- $443_8 = 4 \cdot 8^2 + 4 \cdot 8 + 3 = 256 + 32 + 3 = 291_{10}$
- $3,21_8 = 3 + 2 \cdot \frac{1}{8} + \frac{1}{64}$

Hvis man bruker 16-tallsystemet arbeider man med tall på **heksadesimal** form.

Her må man supplere symbolene 0, 1, ... 9 med sifre A, B, C, D, E og F. Eksempelvis vil

$$2C3_{16} = 2 \cdot 16^2 + 12 \cdot 16 + 3 = 512 + 192 + 3 = 707_{10}.$$

Oktal og heksadesimal form

Oktal og heksadesimal form

Fordelen med oktal og heksadesimal form er at regning med tall i disse tallsystemene representerer en rasjonalisering av regning med binære tall.

Oktal og heksadesimal form

Fordelen med oktal og heksadesimal form er at regning med tall i disse tallsystemene representerer en rasjonalisering av regning med binære tall.

Ved å gruppere tre og tre siffer kan en binær form omgjøres direkte til oktal form:

$$101\ 100\ 001\ 010_2 = 5412_8$$

Oktal og heksadesimal form

Fordelen med oktal og heksadesimal form er at regning med tall i disse tallsystemene representerer en rasjonalisering av regning med binære tall.

Ved å gruppere tre og tre siffer kan en binær form omgjøres direkte til oktal form:

$$101\ 100\ 001\ 010_2 = 5412_8$$

og ved å gruppere fire og fire sifre kan en binær form omgjøres til heksadesimal form:

$$1011\ 0000\ 1010_2 = B0A_{16}.$$

Oktal og heksadesimal form

Oktal og heksadesimal form

Oppgave (Tverrsumstest)

Gå tilbake til beviset for at tverrsumstesten for delelighet med 3 og 9 holder i 10-tallsystemet, og finn ut for hvilke tall vi har en tverrsumstest for tall på oktal og heksadesimal form.