# Løkke-invarianter

Generelt: Egenskaber for variable og arrays i begyndelsen af hver iteration af en løkke, som bruges til at argumentere for korrekthed.

# Løkke-invarianter

Generelt: Egenskaber for variable og arrays i begyndelsen af hver iteration af en løkke, som bruges til at argumentere for korrekthed.

```
Insertion-Sort(A, n)
  for j = 1 to n − 1
    key = A[j]
    i = j − 1
    while i ≥ 0 and A[i] > key
      A[i + 1] = A[i]
      i = i − 1
    A[i + 1] = key
```

*Løkke-invariant (LI):* I begyndelsen af hver iteration af for-løkken indeholder $A[0 \ldots j-1]$ de samme tal som oprindeligt var i $A[0 \ldots j-1]$, men nu i sorteret rækkefølge.

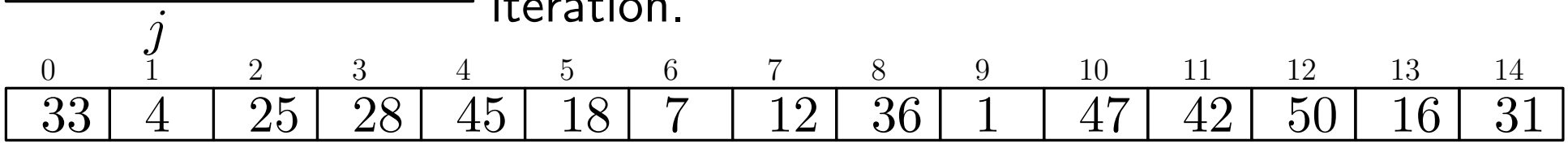
# Løkke-invarianter

Generelt: Egenskaber for variable og arrays i begyndelsen af hver iteration af en løkke, som bruges til at argumentere for korrekthed.

```
Insertion-Sort(A, n)
  for j = 1 to n − 1
    key = A[j]
    i = j − 1
    while i ≥ 0 and A[i] > key
      A[i + 1] = A[i]
      i = i − 1
    A[i + 1] = key
```

*Løkke-invariant (LI):* I begyndelsen af hver iteration af for-løkken indeholder $A[0 \ldots j-1]$ de samme tal som oprindeligt var i $A[0 \ldots j-1]$, men nu i sorteret rækkefølge.

*Initialisering:* LI passer i begyndelsen af første iteration.



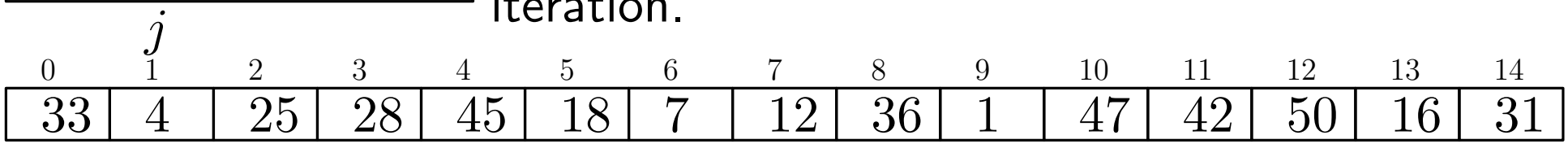
# Løkke-invarianter

Generelt: Egenskaber for variable og arrays i begyndelsen af hver iteration af en løkke, som bruges til at argumentere for korrekthed.

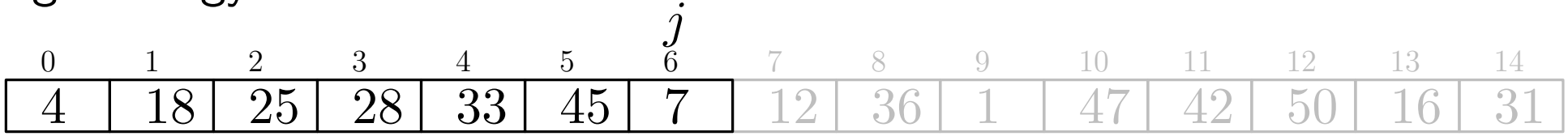
```
Insertion-Sort(A, n)
  for j = 1 to n − 1
    key = A[j]
    i = j − 1
    while i ≥ 0 and A[i] > key
      A[i + 1] = A[i]
      i = i − 1
    A[i + 1] = key
```

*Løkke-invariant (LI):* I begyndelsen af hver iteration af for-løkken indeholder $A[0 \ldots j-1]$ de samme tal som oprindeligt var i $A[0 \ldots j-1]$, men nu i sorteret rækkefølge.

*Initialisering:* LI passer i begyndelsen af første iteration.




*Vedligeholdelse:* LI passer i begyndelsen af én iteration $\Longrightarrow$ LI passer også i begyndelsen af den næste.



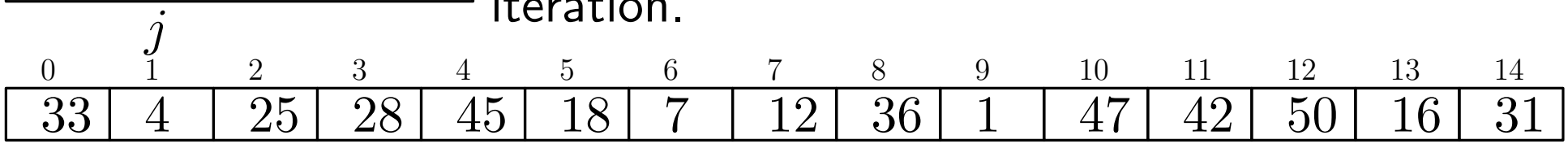
# Løkke-invarianter

Generelt: Egenskaber for variable og arrays i begyndelsen af hver iteration af en løkke, som bruges til at argumentere for korrekthed.

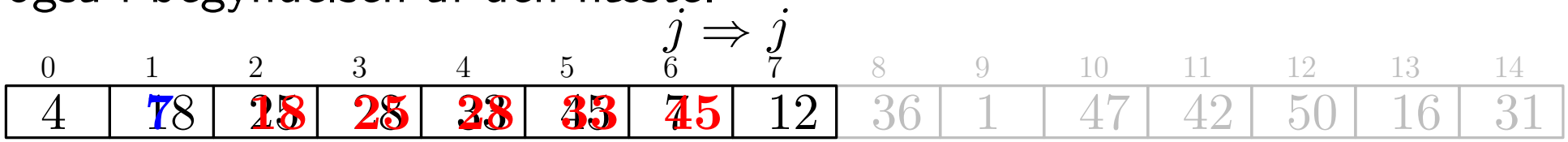
```
Insertion-Sort(A, n)
  for j = 1 to n − 1
    key = A[j]
    i = j − 1
    while i ≥ 0 and A[i] > key
      A[i + 1] = A[i]
      i = i − 1
    A[i + 1] = key
```

*Løkke-invariant (LI):* I begyndelsen af hver iteration af for-løkken indeholder $A[0 \ldots j-1]$ de samme tal som oprindeligt var i $A[0 \ldots j-1]$, men nu i sorteret rækkefølge.

*Initialisering:* LI passer i begyndelsen af første iteration.




*Vedligeholdelse:* LI passer i begyndelsen af én iteration $\Longrightarrow$ LI passer også i begyndelsen af den næste.



# Løkke-invarianter

Generelt: Egenskaber for variable og arrays i begyndelsen af hver iteration af en løkke, som bruges til at argumentere for korrekthed.

```
Insertion-Sort(A, n)
  for j = 1 to n − 1
    key = A[j]
    i = j − 1
    while i ≥ 0 and A[i] > key
      A[i + 1] = A[i]
      i = i − 1
    A[i + 1] = key
```

*Løkke-invariant (LI):* I begyndelsen af hver iteration af for-løkken indeholder $A[0 \ldots j-1]$ de samme tal som oprindeligt var i $A[0 \ldots j-1]$, men nu i sorteret rækkefølge.

*Initialisering:* LI passer i begyndelsen af første iteration.

$j$

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 33 | 4 | 25 | 28 | 45 | 18 | 7 | 12 | 36 | 1 | 47 | 42 | 50 | 16 | 31 |

*Vedligeholdelse:* LI passer i begyndelsen af én iteration $\Longrightarrow$ LI passer også i begyndelsen af den næste.

$j \Rightarrow j$

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 4 | 7 | 25 | 28 | 33 | 18 | 7 | 12 | 36 | 1 | 47 | 42 | 50 | 16 | 31 |
| 4 | 7 | 25 | 28 | 33 | 18 | 45 | 12 | 36 | 1 | 47 | 42 | 50 | 16 | 31 |

*Terminering:* LI passer når løkken terminerer $\Longrightarrow$ Algoritmen virker

$j$

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 4 | 7 | 12 | 16 | 18 | 25 | 28 | 31 | 33 | 36 | 42 | 45 | 47 | 50 |

# Løkke-invarianter

Generelt: Egenskaber for variable og arrays i begyndelsen af hver iteration af en løkke, som bruges til at argumentere for korrekthed.

```
Insertion-Sort(A, n)
  for j = 1 to n − 1
    key = A[j]
    i = j − 1
    while i ≥ 0 and A[i] > key
      A[i + 1] = A[i]
      i = i − 1
    A[i + 1] = key
```

*Løkke-invariant (LI):* I begyndelsen af hver iteration af for-løkken indeholder $A[0 \ldots j-1]$ de samme tal som oprindeligt var i $A[0 \ldots j-1]$, men nu i sorteret rækkefølge.

*Initialisering:* LI passer i begyndelsen af første iteration.

$j$

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 33 | 4 | 25 | 28 | 45 | 18 | 7 | 12 | 36 | 1 | 47 | 42 | 50 | 16 | 31 |

*Vedligeholdelse:* LI passer i begyndelsen af én iteration $\Longrightarrow$ LI passer også i begyndelsen af den næste.

$j \Rightarrow j$

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 4 | 7 | 25 | 28 | 33 | 18 | 45 | 12 | 36 | 1 | 47 | 42 | 50 | 16 | 31 |

*Terminering:* LI passer når løkken terminerer $\Longrightarrow$ Algoritmen virker

$j$

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 4 | 7 | 12 | 16 | 18 | 25 | 28 | 31 | 33 | 36 | 42 | 45 | 47 | 50 |

$j$

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | | | | |

# Løkke-invarianter

Generelt: Egenskaber for variable og arrays i begyndelsen af hver iteration af en løkke, som bruges til at argumentere for korrekthed.

```
Insertion-Sort(A, n)
  for j = 1 to n − 1
    key = A[j]
    i = j − 1
    while i ≥ 0 and A[i] > key
      A[i + 1] = A[i]
      i = i − 1
    A[i + 1] = key
```

*Løkke-invariant (LI):* I begyndelsen af hver iteration af for-løkken indeholder $A[0 \ldots j-1]$ de samme tal som oprindeligt var i $A[0 \ldots j-1]$, men nu i sorteret rækkefølge.

*Initialisering:* LI passer i begyndelsen af første iteration.

$j$

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 33 | 4 | 25 | 28 | 45 | 18 | 7 | 12 | 36 | 1 | 47 | 42 | 50 | 16 | 31 |

*Vedligeholdelse:* LI passer i begyndelsen af én iteration $\Longrightarrow$ LI passer også i begyndelsen af den næste.

$j \Rightarrow j$

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 4 | 7 8 | 25 | 28 | 45 | 18 | 7 | 12 | 36 | 1 | 47 | 42 | 50 | 16 | 31 |

*Terminering:* LI passer når løkken terminerer $\Longrightarrow$ Algoritmen virker

$j$

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 4 | 7 | 12 | 16 | 18 | 25 | 28 | 31 | 33 | 36 | 42 | 45 | 47 | 50 |

$j \Rightarrow j$

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | | | | |

# Løkke-invarianter

Generelt: Egenskaber for variable og arrays i begyndelsen af hver iteration af en løkke, som bruges til at argumentere for korrekthed.

```
Insertion-Sort(A, n)
  for j = 1 to n − 1
    key = A[j]
    i = j − 1
    while i ≥ 0 and A[i] > key
      A[i + 1] = A[i]
      i = i − 1
    A[i + 1] = key
```

*Løkke-invariant (LI):* I begyndelsen af hver iteration af for-løkken indeholder $A[0 \ldots j-1]$ de samme tal som oprindeligt var i $A[0 \ldots j-1]$, men nu i sorteret rækkefølge.

*Initialisering:* LI passer i begyndelsen af første iteration.

$j$

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 33 | 4 | 25 | 28 | 45 | 18 | 7 | 12 | 36 | 1 | 47 | 42 | 50 | 16 | 31 |

*Vedligeholdelse:* LI passer i begyndelsen af én iteration $\Longrightarrow$ LI passer også i begyndelsen af den næste.

$j \Rightarrow j$

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 4 | 7 | 25 | 28 | 33 | 18 | 7 | 12 | 36 | 1 | 47 | 42 | 50 | 16 | 31 |
| 4 | 7 | 25 | 28 | 33 | 18 | 33 | 12 | 36 | 1 | 47 | 42 | 50 | 16 | 31 |
| 4 | 7 | 25 | 28 | 33 | 25 | 33 | 12 | 36 | 1 | 47 | 42 | 50 | 16 | 31 |
| 4 | 7 | 25 | 28 | 28 | 25 | 33 | 12 | 36 | 1 | 47 | 42 | 50 | 16 | 31 |
| 4 | 7 | 18 | 28 | 28 | 25 | 33 | 12 | 36 | 1 | 47 | 42 | 50 | 16 | 31 |
| 4 | 7 | 18 | 25 | 28 | 25 | 33 | 12 | 36 | 1 | 47 | 42 | 50 | 16 | 31 |
| 4 | 7 | 18 | 25 | 28 | 25 | 33 | 45 | 36 | 1 | 47 | 42 | 50 | 16 | 31 |

*Terminering:* LI passer når løkken terminerer $\Longrightarrow$ Algoritmen virker

$j$

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 4 | 7 | 12 | 16 | 18 | 25 | 28 | 31 | 33 | 36 | 42 | 45 | 47 | 50 |

$j \Rightarrow j \Rightarrow j$

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | | | | |

# Løkke-invarianter

Generelt: Egenskaber for variable og arrays i begyndelsen af hver iteration af en løkke, som bruges til at argumentere for korrekthed.

```
Insertion-Sort(A, n)
  for j = 1 to n − 1
    key = A[j]
    i = j − 1
    while i ≥ 0 and A[i] > key
      A[i + 1] = A[i]
      i = i − 1
    A[i + 1] = key
```

*Løkke-invariant (LI):* I begyndelsen af hver iteration af for-løkken indeholder $A[0 \ldots j-1]$ de samme tal som oprindeligt var i $A[0 \ldots j-1]$, men nu i sorteret rækkefølge.

*Initialisering:* LI passer i begyndelsen af første iteration.

$j$

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 33 | 4 | 25 | 28 | 45 | 18 | 7 | 12 | 36 | 1 | 47 | 42 | 50 | 16 | 31 |

*Vedligeholdelse:* LI passer i begyndelsen af én iteration $\Longrightarrow$ LI passer også i begyndelsen af den næste.

$j \Rightarrow j$

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 4 | 7 | 25 | 28 | 45 | 18 | 7 | 12 | 36 | 1 | 47 | 42 | 50 | 16 | 31 |

*Terminering:* LI passer når løkken terminerer $\Longrightarrow$ Algoritmen virker

$j$

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 4 | 7 | 12 | 16 | 18 | 25 | 28 | 31 | 33 | 36 | 42 | 45 | 47 | 50 |

$j \Rightarrow j \Rightarrow j \Rightarrow j$

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | | | | |

# Løkke-invarianter

Generelt: Egenskaber for variable og arrays i begyndelsen af hver iteration af en løkke, som bruges til at argumentere for korrekthed.

```
Insertion-Sort(A, n)
  for j = 1 to n − 1
    key = A[j]
    i = j − 1
    while i ≥ 0 and A[i] > key
      A[i + 1] = A[i]
      i = i − 1
    A[i + 1] = key
```

*Løkke-invariant (LI):* I begyndelsen af hver iteration af for-løkken indeholder $A[0 \ldots j-1]$ de samme tal som oprindeligt var i $A[0 \ldots j-1]$, men nu i sorteret rækkefølge.

*Initialisering:* LI passer i begyndelsen af første iteration.

$j$

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 33 | 4 | 25 | 28 | 45 | 18 | 7 | 12 | 36 | 1 | 47 | 42 | 50 | 16 | 31 |

*Vedligeholdelse:* LI passer i begyndelsen af én iteration $\Longrightarrow$ LI passer også i begyndelsen af den næste.

$j \Rightarrow j$

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 4 | 7 | 25 | 28 | 33 | 18 | 45 | 12 | 36 | 1 | 47 | 42 | 50 | 16 | 31 |

*Terminering:* LI passer når løkken terminerer $\Longrightarrow$ Algoritmen virker

$j$

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 4 | 7 | 12 | 16 | 18 | 25 | 28 | 31 | 33 | 36 | 42 | 45 | 47 | 50 |

$j \Rightarrow j \Rightarrow j \Rightarrow j \Rightarrow j \Rightarrow j \Rightarrow j \Rightarrow j \Rightarrow j \Rightarrow j \Rightarrow j \Rightarrow j \Rightarrow j \Rightarrow j$

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | | | | |

# Bonus-info

Brugen af løkke-invarianter til at bevise korrekthed for algoritmer blev først beskrevet af Peter Naur (1928–2016) i en artikel fra 1966.

## PROOF OF ALGORITHMS BY GENERAL SNAPSHOTS

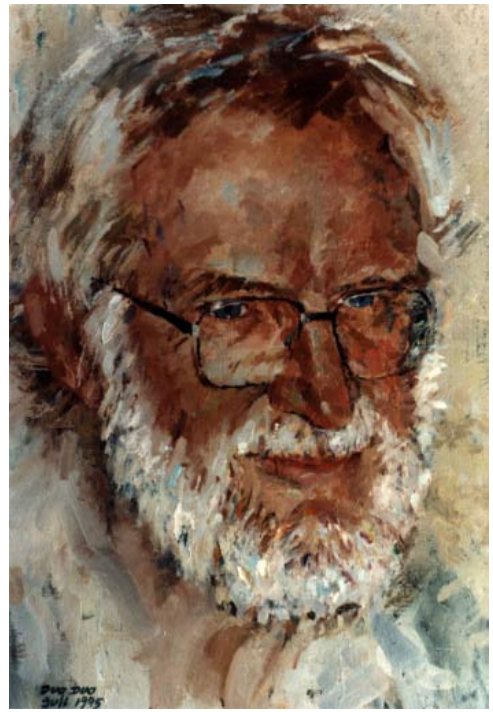
PETER NAUR

**Abstract.**

A constructive approach to the question of proofs of algorithms is to consider proofs that an object resulting from the execution of an algorithm possesses certain static characteristics. It is shown by an elementary example how this possibility may be used to prove the correctness of an algorithm written in ALGOL 60. The stepping stone of the approach is what is called General Snapshots, i.e. expressions of static conditions existing whenever the execution of the algorithm reaches particular points. General Snapshots are further shown to be useful for constructing algorithms.

BIBLIOGRAPHY

The basic question of proof has so far been ignored in data processing to an incredible degree.



Desuden opfandt han ordet *datalogi*, grundlagde DIKU i 1970 og modtog Turing-medaljen i 2005 (datalogiens "Nobel-pris") for sit arbejde med at udvikle de første moderne/strukturerede programmingssprog.