Ideen und Konzepte der Informatik

Programme und Algorithmen

Kurt Mehlhorn

max planck institut informatik

Saarland Informatics Campus

Algorithmen und Programme

- **Algorithmen und Programme**
  - Wichtige Begriffe: Algorithmen, Programme, Programmiersprache, Code, Pseudocode, Variable, Wertzuweisung, bedingte Anweisung, Schleife, Korrektheit, Effizienz.
  - Sie werden lernen einfache Programme zu schreiben. Sie werden nicht lernen umfangreiche Programme zu schreiben.
- **Unsere ersten Algorithmen**
  - Lösen einer quadratischen Gleichung
  - Addition von Dezimalzahlen.
  - Test, ob ein gegebenes Wort in einem Text vorkommt.
  - Das zweite Beispiel zeigt uns, dass man nicht nur mit Zahlen rechnen kann.

Algorithmen und Programme

Algorithmus – Schritt-für-Schritt Vorschrift zur Lösung eines Problems. Formuliert man umgangssprachlich, aber trotzdem präzise (**Pseudocode**). Für Menschen gedacht.

Programm – bis in alle Details spezifierte Vorschrift zur Lösung eines Problems. Maschinenausführbar. Formuliert man in einer Programmiersprache (**Code**).

Programmiersprache – Kunstsprache zur Formulierung von Programmen mit genau definierter Syntax und Semantik.

Syntax	=	was ist ein zulässiger Satz
Semantik	=	was bedeutet ein Satz

Ursprung des Wortes Algorithmus


Muhammad ibn Musa al-Khwarizmi

Persischer Mathematiker, 780 – 850

„Das kurzgefasste Buch über die Rechenverfahren durch Ergänzen und Ausgleichen"

- Enthält – unter anderem – Algorithmus zum Lösen von quadratischen Gleichungen.

Beispiel: Quadratische Gleichung

Algorithmus | **Ausführungsbeispiel**

- *Schreibe die Gleichung als* $x^2 + bx + c = 0$ $\quad x^2 + 8x - 9 = 0$

Beispiel: Quadratische Gleichung

Algorithmus **Ausführungsbeispiel**

- *Schreibe die Gleichung als* $x^2 + bx + c = 0$ $\quad x^2 + 8x - 9 = 0$
- *Bring das konstante Glied auf die andere Seite* $\quad x^2 + 8x = 9$

Beispiel: Quadratische Gleichung

Algorithmus	**Ausführungsbeispiel**
- *Schreibe die Gleichung als* $x^2 + bx + c = 0$	$x^2 + 8x - 9 = 0$
- *Bring das konstante Glied auf die andere Seite*	$x^2 + 8x = 9$
- *Addiere* $(b/2)^2$ *auf beiden Seiten*	$x^2 + 8x + 4^2 = 9 + 4^2$

Beispiel: Quadratische Gleichung

Algorithmus	**Ausführungsbeispiel**
- *Schreibe die Gleichung als* $x^2 + bx + c = 0$	$x^2 + 8x - 9 = 0$
- *Bring das konstante Glied auf die andere Seite*	$x^2 + 8x = 9$
- *Addiere* $(b/2)^2$ *auf beiden Seiten*	$x^2 + 8x + 4^2 = 9 + 4^2$
- *Schreibe LS als* $(x + b/2)^2$, *vereinfache RS*	$(x + 4)^2 = 25$

Beispiel: Quadratische Gleichung

Algorithmus	**Ausführungsbeispiel**
- *Schreibe die Gleichung als* $x^2 + bx + c = 0$	$x^2 + 8x - 9 = 0$
- *Bring das konstante Glied auf die andere Seite*	$x^2 + 8x = 9$
- *Addiere* $(b/2)^2$ *auf beiden Seiten*	$x^2 + 8x + 4^2 = 9 + 4^2$
- *Schreibe LS als* $(x + b/2)^2$*, vereinfache RS*	$(x + 4)^2 = 25$
- *Falls RS negativ, STOP (keine Lösung)*	

Beispiel: Quadratische Gleichung

Algorithmus	**Ausführungsbeispiel**
- *Schreibe die Gleichung als* $x^2 + bx + c = 0$	$x^2 + 8x - 9 = 0$
- *Bring das konstante Glied auf die andere Seite*	$x^2 + 8x = 9$
- *Addiere* $(b/2)^2$ *auf beiden Seiten*	$x^2 + 8x + 4^2 = 9 + 4^2$
- *Schreibe LS als* $(x + b/2)^2$*, vereinfache RS*	$(x + 4)^2 = 25$
- *Falls RS negativ, STOP (keine Lösung)*	
- *Entferne* ^{2} *auf LS, ersetze RS durch* $\pm\sqrt{RS}$	$x + 4 = \pm\sqrt{25}$

Beispiel: Quadratische Gleichung

Algorithmus	**Ausführungsbeispiel**
- *Schreibe die Gleichung als* $x^2 + bx + c = 0$	$x^2 + 8x - 9 = 0$
- *Bring das konstante Glied auf die andere Seite*	$x^2 + 8x = 9$
- *Addiere* $(b/2)^2$ *auf beiden Seiten*	$x^2 + 8x + 4^2 = 9 + 4^2$
- *Schreibe LS als* $(x + b/2)^2$, *vereinfache RS*	$(x + 4)^2 = 25$
- *Falls RS negativ, STOP (keine Lösung)*	
- *Entferne* ^{2} *auf LS, ersetze RS durch* $\pm\sqrt{RS}$	$x + 4 = \pm\sqrt{25}$
- *Bewege konstantes Glied von LS nach RS*	$x = -4 \pm 5$

Beispiel: Quadratische Gleichung

Algorithmus	Ausführungsbeispiel
- *Schreibe die Gleichung als* $x^2 + bx + c = 0$	$x^2 + 8x - 9 = 0$
- *Bring das konstante Glied auf die andere Seite*	$x^2 + 8x = 9$
- *Addiere* $(b/2)^2$ *auf beiden Seiten*	$x^2 + 8x + 4^2 = 9 + 4^2$
- *Schreibe LS als* $(x + b/2)^2$*, vereinfache RS*	$(x + 4)^2 = 25$
- *Falls RS negativ, STOP (keine Lösung)*	
- *Entferne* ^{2} *auf LS, ersetze RS durch* $\pm\sqrt{RS}$	$x + 4 = \pm\sqrt{25}$
- *Bewege konstantes Glied von LS nach RS*	$x = -4 \pm 5$
	$x = 1$ oder $x = -9$.

Beispiel: Quadratische Gleichung

Algorithmus | **Ausführungsbeispiel**

- *Schreibe die Gleichung als* $x^2 + bx + c = 0$ — $x^2 + 8x - 9 = 0$
- *Bring das konstante Glied auf die andere Seite* — $x^2 + 8x = 9$
- *Addiere* $(b/2)^2$ *auf beiden Seiten* — $x^2 + 8x + 4^2 = 9 + 4^2$
- *Schreibe LS als* $(x + b/2)^2$, *vereinfache RS* — $(x+4)^2 = 25$
- *Falls RS negativ, STOP (keine Lösung)*
- *Entferne* ^{2} *auf LS, ersetze RS durch* $\pm\sqrt{RS}$ — $x + 4 = \pm\sqrt{25}$
- *Bewege konstantes Glied von LS nach RS* — $x = -4 \pm 5$

$x = 1$ oder $x = -9$.

Algorithmus ist im Buch von **Al-Khwarizmi** enthalten.

Beispiel: Quadratische Gleichung

Algorithmus	**Ausführungsbeispiel**
- *Schreibe die Gleichung als* $x^2 + bx + c = 0$	$x^2 + 8x - 9 = 0$
- *Bring das konstante Glied auf die andere Seite*	$x^2 + 8x = 9$
- *Addiere* $(b/2)^2$ *auf beiden Seiten*	$x^2 + 8x + 4^2 = 9 + 4^2$
- *Schreibe LS als* $(x + b/2)^2$, *vereinfache RS*	$(x + 4)^2 = 25$
- *Falls RS negativ, STOP (keine Lösung)*	
- *Entferne* ^{2} *auf LS, ersetze RS durch* $\pm\sqrt{RS}$	$x + 4 = \pm\sqrt{25}$
- *Bewege konstantes Glied von LS nach RS*	$x = -4 \pm 5$
	$x = 1$ oder $x = -9$.

Algorithmus ist im Buch von **Al-Khwarizmi** enthalten.

Algorithmus ist intendiert für einen **menschlichen Computer**,
Programme für reale Computer sind viel detaillierter.

Korrektheit und Effizienz

- Erste Algorithmen wurden schon vor mehreren Jahrhunderten entwickelt – lange vor dem ersten Computer.

- **Korrektheit:** Wie können wir uns sicher sein, dass ein Algorithmus auch immer die versprochene Lösung liefert?

  Konkret: Gibt der "Quadratische Gleichung"–Algorithmus bei jeder Gleichung die richtige Lösung?

- **Effizienz:** Wieviele Rechenschritte braucht der Algorithmus für die Bestimmung der Lösung?

Grundbegriffe der Programmierung: Struktur von Programmen

- Programme operieren auf Daten. Programme formen Eingabedaten nach gewissen Regeln in Ausgabedaten um.
- Die Daten sind in einem Speicher abgelegt.
- Der Speicher besteht aus Speicherzellen. Jede Speicherzelle enthält einen Wert und hat einen Namen.
- Zuweisungen weisen Speicherzellen neue Werte zu.
- Die Kontrollstruktur des Programs legt fest, welche Zuweisungen ausgeführt werden.

Grundbegriffe der Programmierung: Struktur von Programmen

Zuweisungen weisen Speicherzellen Werte zu.

Um sich bequem auf Speicherzellen beziehen zu können, gibt man ihnen Namen.

Speicherzellen mit Namen heißen **Variablen**.

Grundbegriffe der Programmierung: Struktur von Programmen

Zuweisungen weisen Speicherzellen Werte zu.

Um sich bequem auf Speicherzellen beziehen zu können, gibt man ihnen Namen.

Speicherzellen mit Namen heißen **Variablen**.

Kontrollstrukturen

legen fest, welche Zuweisungen ausgeführt werden.

Beispiele: Falls **Bedingung** mache **dies**, sonst **das**.

Solange **Bedingung** mache **etwas**.

Variable, Ausdrücke, Zuweisungen

Variable (Speicherzellen mit Namen)

- haben einen Namen, z.B. x, y, $Gehalt$, i, x_0, x_1, x_2, ...
- und zu jedem Zeitpunkt einen Wert, z.B. x hat den Wert 5.
- Der Wert kann durch eine *Wertzuweisung* geändert werden, z.B. $x \leftarrow 7$ lies: x bekommt den Wert 7.

Wertzuweisung: Variable $\leftarrow$ Ausdruck

- Beispiele: $x \leftarrow 5$; $y \leftarrow 7$; $x \leftarrow x + y$;
- Vor der Zuweisung $x \leftarrow x + y$ haben x und y die Werte 5 und 7.
- Zur Bestimmung des Wertes des Ausdrucks $x + y$ werden die Variablen durch ihre augenblicklichen Werte ersetzt und dann gerechnet $x + y \rightarrow 5 + 7 = 12$.
- Der so bestimmte Wert wird der neue Wert von x.

Ein erstes Programm

$n \leftarrow 3;$
$s \leftarrow 0;$
$i \leftarrow 1;$
while $i \leq n$
    $s \leftarrow s + i;$
    $i \leftarrow i + 1;$
drucke s;

Das Obige nennt sich eine *While-Schleife*.

Solange die Bedingung $i \leq n$ zutrifft, führe den Rumpf der Schleife aus

Die Ausführung

Ein erstes Programm

$n \leftarrow 3;$
$s \leftarrow 0;$
$i \leftarrow 1;$
while $i \leq n$
    $s \leftarrow s + i;$
    $i \leftarrow i + 1;$
drucke s;

Das Obige nennt sich eine *While-Schleife*.

Solange die Bedingung $i \leq n$ zutrifft, führe den Rumpf der Schleife aus

Die Ausführung

$n \leftarrow 3;$
$s \leftarrow 0;$
$i \leftarrow 1;$
$i \leq n$ ist wahr (da $1 \leq 3$ wahr ist)
$s \leftarrow s + i = 0 + 1 = 1;$
$i \leftarrow i + 1 = 1 + 1 = 2;$
$i \leq n$ ist wahr;
$\vdots$

Ein erstes Programm

$n \leftarrow 3;$
$s \leftarrow 0;$
$i \leftarrow 1;$
while $i \leq n$
    $s \leftarrow s + i;$
    $i \leftarrow i + 1;$
drucke s;

Das Obige nennt sich eine *While-Schleife*.

Solange die Bedingung $i \leq n$ zutrifft, führe den Rumpf der Schleife aus

Die Ausführung

$n \leftarrow 3;$
$s \leftarrow 0;$
$i \leftarrow 1;$
$i \leq n$ ist wahr (da $1 \leq 3$ wahr ist)
$s \leftarrow s + i = 0 + 1 = 1;$
$i \leftarrow i + 1 = 1 + 1 = 2;$
$i \leq n$ ist wahr;
$\vdots$

"drucke s" gibt 6 aus.

Die Ausgabe der Rechnung ist die Summe $1 + 2 + 3$.

Ein erstes interessantes Programm

$n \leftarrow$ Eingabe;
$s \leftarrow 0$;
$i \leftarrow 1$;
while $i \leq n$
  $s \leftarrow s + i$;
  $i \leftarrow i + 1$;
drucke s;

Wir weisen n keinen festen Wert mehr zu, sondern lesen ihn ein.

Bei Eingabe 3 berechnet das Programm die Summe $1 + 2 + 3 = 6$.

Ein erstes interessantes Programm

$n \leftarrow$ Eingabe;
$s \leftarrow 0$;
$i \leftarrow 1$;
while $i \leq n$
  $s \leftarrow s + i$;
  $i \leftarrow i + 1$;
drucke s;

Wir weisen n keinen festen Wert mehr zu, sondern lesen ihn ein.

Bei Eingabe 3 berechnet das Programm die Summe $1 + 2 + 3 = 6$.

Bei Eingabe 4 berechnet das Programm die Summe .

Ein erstes interessantes Programm

$n \leftarrow$ Eingabe;
$s \leftarrow 0$;
$i \leftarrow 1$;
while $i \leq n$
  $s \leftarrow s + i$;
  $i \leftarrow i + 1$;
drucke s;

Wir weisen n keinen festen Wert mehr zu, sondern lesen ihn ein.

Bei Eingabe 3 berechnet das Programm die Summe $1 + 2 + 3 = 6$.

Bei Eingabe 100 berechnet das Programm die Summe .

Ein erstes interessantes Programm

$n \leftarrow$ Eingabe;
$s \leftarrow 0$;
$i \leftarrow 1$;
while $i \leq n$
  $s \leftarrow s + i$;
  $i \leftarrow i + 1$;
drucke s;

Wir weisen n keinen festen Wert mehr zu, sondern lesen ihn ein.

Bei Eingabe 3 berechnet das Programm die Summe $1 + 2 + 3 = 6$.

Bei Eingabe 100 berechnet das Programm die Summe .

Das Flussdiagramm zur Schleife

und als For-Schleife

Bedingte Anweisungen

if Bedingung
    dann-Fall
else
    sonst-Fall

Werte die Bedingung aus; die Bedingung ist ein logischer Ausdruck, der sich zu *wahr* oder *falsch* auswertet.

Falls wahr, dann führe den dann-Fall aus.

Falls falsch, dann führe den sonst-Fall aus.

Bedingte Anweisungen

if Bedingung
  dann-Fall
else
  sonst-Fall

Werte die Bedingung aus; die Bedingung ist ein logischer Ausdruck, der sich zu *wahr* oder *falsch* auswertet.

Falls wahr, dann führe den dann-Fall aus.

Falls falsch, dann führe den sonst-Fall aus.

$i \leftarrow 1;$
if i ist ungerade
  $i \leftarrow i + 1;$
else
  $i \leftarrow i + 2;$

Ausführung

$i \leftarrow 1;$
(i ist ungerade) ist wahr;
daher wird der dann-Fall ausgeführt;
$i \leftarrow i + 1 = 1 + 1 = 2;$

und nun mit Anfangswert 2

$i \leftarrow 2;$

Ein etwas kompliziertes Programm

Ausführung

$s \leftarrow 0;$
$i \leftarrow 1;$
while $i \leq 4$
    $s \leftarrow s + i;$
    $i \leftarrow i + 1;$
    if i ist ungerade
        drucke s
    else
        $i \leftarrow i + 1$
drucke s;

Ein etwas kompliziertes Programm

```
s ← 0;
i ← 1;
while  i ≤ 4
   s ← s + i;
   i ← i + 1;
   if  i ist ungerade
      drucke s
   else
      i ← i + 1
drucke s;
```

Ausführung

$s \leftarrow 0;$

$i \leftarrow 1;$

$i \leq 4$ ist wahr

$s \leftarrow s + i = 0 + 1 = 1;$

$i \leftarrow i + 1 = 1 + 1 = 2;$

i ist ungerade ist falsch

$i \leftarrow i + 1 = 2 + 1 = 3;$

$i \leq 4$ ist wahr;

$s \leftarrow s + i = 1 + 3 = 4;$

$i \leftarrow i + 1 = 3 + 1 = 4;$

i ist ungerade ist falsch

$i \leftarrow i + 1 = 4 + 1 = 5;$

$i \leq 4$ ist falsch;

“drucke s” gibt 4 aus.

Auch kurze Programme können knifflig sein (Lothar Collatz)

$n \leftarrow$ eine natürliche Zahl
while $n > 1$
  if n ist gerade
    $n \leftarrow n/2$;
  else
    $n \leftarrow 3n + 1$;

Ausführungen

$16 \rightarrow 8 \rightarrow 4 \rightarrow 2 \rightarrow 1$

$6 \rightarrow 3 \rightarrow 10 \rightarrow 5 \rightarrow 16 \rightarrow \ldots$

$17 \rightarrow 52 \rightarrow 26 \rightarrow 13 \rightarrow 40 \rightarrow 20 \ldots$

Auch kurze Programme können knifflig sein (Lothar Collatz)

$n \leftarrow$ eine natürliche Zahl
while $n > 1$
  if n ist gerade
    $n \leftarrow n/2$;
  else
    $n \leftarrow 3n + 1$;

Es ist nicht bekannt, ob dieses Programm für jede Eingabe hält.

Probieren sie den Startwert 27.

Ausführungen

$16 \rightarrow 8 \rightarrow 4 \rightarrow 2 \rightarrow 1$

$6 \rightarrow 3 \rightarrow 10 \rightarrow 5 \rightarrow 16 \rightarrow \ldots$

$17 \rightarrow 52 \rightarrow 26 \rightarrow 13 \rightarrow 40 \rightarrow 20 \ldots$

Auch kurze Programme können knifflig sein (Lothar Collatz)

$n \leftarrow$ eine natürliche Zahl
while $n > 1$
  if n ist gerade
    $n \leftarrow n/2;$
  else
    $n \leftarrow 3n + 1;$

Es ist nicht bekannt, ob dieses Programm für jede Eingabe hält.

Probieren sie den Startwert 27.

27, 82, 41, 124, 62, 31, 94, 47, 142, 71, 214, 107, 322, 161, 484, 242, 121, 364, 182, 91, 274, 137, 412, 206, 103, 310, 155, 466, 233, 700, 350, 175, 526, 263, 790, 395, 1186, 593, 1780, 890, 445, 1336, 668, 334, 167, 502, 251, 754, 377, 1132, 566, 283, 850, 425, 1276, 638, 319, 958, 479, 1438, 719, 2158, 1079, 3238, 1619, 4858, 2429, 7288, 3644, 1822, 911, 2734, 1367, 4102, 2051, 6154, 3077, 9232, 4616, 2308, 1154, 577, 1732, 866, 433, 1300, 650, 325, 976, 488, 244, 122, 61, 184, 92, 46, 23, 70, 35, 106, 53, 160, 80, 40, 20, 10, 5, 16, 8, 4, 2, 1

Ausführungen

$16 \rightarrow 8 \rightarrow 4 \rightarrow 2 \rightarrow 1$

$6 \rightarrow 3 \rightarrow 10 \rightarrow 5 \rightarrow 16 \rightarrow \ldots$

$17 \rightarrow 52 \rightarrow 26 \rightarrow 13 \rightarrow 40 \rightarrow 20 \ldots$

Summand	4	7	2	3
Summand	5	4	4	8
Überträge				0
Summe				

Summand	4	7	2	3
Summand	5	4	4	8
Überträge				0
Summe				

Der Übertrag in die letzte Spalte ist 0.

Wir addieren die drei Ziffern in einer Spalte. Nenne die Summe S.

$S \geq 10$: Übertrag ist 1, und Summenziffer ist $S - 10$.

$S \leq 9$: Übertrag ist 0, und Summenziffer ist S.

Summand	4	7	2	3
Summand	5	4	4	8
Überträge				0
Summe				

Der Übertrag in die letzte Spalte ist 0.

Wir addieren die drei Ziffern in einer Spalte. Nenne die Summe S.

$S \geq 10$: Übertrag ist 1, und Summenziffer ist $S - 10$.

$S \leq 9$: Übertrag ist 0, und Summenziffer ist S.

Zahl 1 hat Ziffern $a_3, \ldots, a_0$.

Zahl 2 hat Ziffern $b_3, \ldots, b_0$.

Summe hat Ziffern $s_4, s_3, \ldots, s_0$.

Wir haben auch noch einen Übertrag c.

$c \leftarrow 0$;
for i von 0 bis 3
  $S \leftarrow a_i + b_i + c$;
  if $S \leq 9$
    $s_i \leftarrow S$; $c \leftarrow 0$;
  else
    $s_i \leftarrow S - 10$; $c \leftarrow 1$;
$s_4 \leftarrow c$;

Addition von Dezimalzahlen

Und nun mit beliebig vielen Stellen.

Zahl 1 hat Ziffern $a_{n-1}, \ldots, a_0$.

Zahl 2 hat Ziffern $b_{n-1}, \ldots, b_0$.

Summe hat Ziffern $s_n, s_{n-1}, \ldots, s_0$.

Wir haben auch noch einen Übertrag c.

$c \leftarrow 0$;
for i von 0 bis $n-1$
  $S \leftarrow a_i + b_i + c$;
  if $S \leq 9$
    $s_i \leftarrow S$; $c \leftarrow 0$;
  else
    $s_i \leftarrow S - 10$; $c \leftarrow 1$;
$s_n \leftarrow c$;

Man kann nicht nur mit Zahlen rechnen.

Ein Wort ist eine Folge von Buchstaben, z.B., „Hoffnung“. Wir wollen feststellen, ob ein Wort (das *Muster, Pattern*) in einem anderen Wort (dem *Text*) vorkommt.

Text =aabaababab

Muster = abab

Dazu legen wir das Muster an jeder Stelle des Textes an und vergleichen Buchstabe für Buchstabe. Wenn wir dabei bis zum Ende des Musters kommen, dann haben wir ein Vorkommen des Musters gefunden. Wenn wir vorher auf ungleiche Buchstaben kommen, passt das Muster an dieser Stelle nicht.

Man kann nicht nur mit Zahlen rechnen.

Ein Wort ist eine Folge von Buchstaben, z.B., „Hoffnung“. Wir wollen feststellen, ob ein Wort (das *Muster, Pattern*) in einem anderen Wort (dem *Text*) vorkommt.

Text =aabaababab

Muster = abab

Dazu legen wir das Muster an jeder Stelle des Textes an und vergleichen Buchstabe für Buchstabe. Wenn wir dabei bis zum Ende des Musters kommen, dann haben wir ein Vorkommen des Musters gefunden. Wenn wir vorher auf ungleiche Buchstaben kommen, passt das Muster an dieser Stelle nicht.

An jeder Stelle anlegen = wir legen das Muster zuerst am Anfang des Textes an und schieben es nach jedem Probieren (erfolgreich oder nocht erfolgreich) um eine Stelle nach hinten.

Probieren, ob das Muster an der Stelle i passt

Der Text ist eine Folge von Buchstaben. Sei n die Länge des Textes (= Anzahl der Buchstaben). Wir nummerieren die Buchstaben durch. Also

$$\text{Text} = t_0 t_1 \dots t_{n-1} \quad \text{oder} \quad \text{Text} = t_1 t_2 \dots t_n.$$

Ob man man mit Null oder mit Eins beginnt, ist Geschmackssache. Mit Null zu beginnen ist oft eleganter.

Beispiel: Text = abab. Dann $t_0 = a$, $t_1 = b$, $t_2 = a$, $t_3 = b$.

Probieren, ob das Muster an der Stelle i passt

Der Text ist eine Folge von Buchstaben. Sei n die Länge des Textes (= Anzahl der Buchstaben). Wir nummerieren die Buchstaben durch. Also

$$\text{Text} = t_0 t_1 \dots t_{n-1} \quad \text{oder} \quad \text{Text} = t_1 t_2 \dots t_n.$$

Ob man man mit Null oder mit Eins beginnt, ist Geschmackssache. Mit Null zu beginnen ist oft eleganter.

Beispiel: Text = abab. Dann $t_0 = a$, $t_1 = b$, $t_2 = a$, $t_3 = b$.

Die Länge des Musters sei k. Dann Muster = $p_0 p_1 \dots p_{k-1}$.

Wir legen das Muster an der Stelle i des Textes an und überprüfen, ob es passt.

for j von 0 bis $k-1$
  if $t_{i+j} = p_j$
    mach weiter
  else
    breche den Versuch ab

for j von 0 bis $k-1$
  if $t_{i+j} \neq p_j$
    breche den Versuch ab
  else
    mach weiter

Probieren, ob das Muster an der Stelle i passt

Der Text ist eine Folge von Buchstaben. Sei n die Länge des Textes (= Anzahl der Buchstaben). Wir nummerieren die Buchstaben durch. Also

$$\text{Text} = t_0 t_1 \dots t_{n-1} \quad \text{oder} \quad \text{Text} = t_1 t_2 \dots t_n.$$

Ob man man mit Null oder mit Eins beginnt, ist Geschmackssache. Mit Null zu beginnen ist oft eleganter.

Beispiel: Text = abab. Dann $t_0 = a$, $t_1 = b$, $t_2 = a$, $t_3 = b$.

Die Länge des Musters sei k. Dann Muster = $p_0 p_1 \dots p_{k-1}$.

Wir legen das Muster an der Stelle i des Textes an und überprüfen, ob es passt.

for j von 0 bis $k-1$
  if $t_{i+j} = p_j$
    mach weiter
  else
    breche den Versuch ab

for j von 0 bis $k-1$
  if $t_{i+j} \neq p_j$
    breche den Versuch ab

Probieren, ob das Muster an der Stelle i passst

$$\text{Text} = t_0 t_1 \ldots t_{n-1} \quad \text{und} \quad \text{Muster} = p_0 p_1 \ldots p_{k-1}.$$

Wir legen das Muster an der Stelle i des Textes an und überprüfen, ob es passt. Wenn wir bis zum Ende kommen, dann melden wir Erfolg und drucken i. Andernfalls brechen wir den Versuch ab.

for j von 0 bis $k-1$
    if $t_{i+j} \neq p_j$
        breche den Versuch ab

drucke i; /* der Druckbefehl ist nicht mehr Teil der Schleife */

Probieren, ob das Muster an der Stelle i passst

$$\text{Text} = t_0 t_1 \dots t_{n-1} \quad \text{und} \quad \text{Muster} = p_0 p_1 \dots p_{k-1}.$$

Wir legen das Muster an der Stelle i des Textes an und überprüfen, ob es passt. Wenn wir bis zum Ende kommen, dann melden wir Erfolg und drucken i. Andernfalls brechen wir den Versuch ab.

for j von 0 bis $k-1$
  if $t_{i+j} \neq p_j$
    goto Abbruch /* Wir verlassen die Schleife */

drucke i; /* der Druckbefehl ist nicht mehr Teil der Schleife */
Abbruch: /* Ziel des Sprungs aus der Schleife */

Probieren, ob das Muster an der Stelle i passst

$$\text{Text} = t_0 t_1 \ldots t_{n-1} \quad \text{und} \quad \text{Muster} = p_0 p_1 \ldots p_{k-1}.$$

Wir legen das Muster an der Stelle i des Textes an und überprüfen, ob es passt. Wenn wir bis zum Ende kommen, dann melden wir Erfolg und drucken i. Andernfalls brechen wir den Versuch ab.

```
for j von 0 bis k − 1
  if t_{i+j} ≠ p_j
    goto Abbruch                       /* Wir verlassen die Schleife */

drucke i;        /* der Druckbefehl ist nicht mehr Teil der Schleife */
Abbruch:                        /* Ziel des Sprungs aus der Schleife */
```

Jetzt muessen wir nur noch das Muster an jeder Stelle anlegen.

Mit Buchstaben rechnen.

Dazu legen wir das Muster an jeder Stelle des Textes an und vergleichen Buchstabe für Buchstabe.

/* Text hat Buchstaben $t_0, \ldots, t_{n-1}$. */
/* Muster hat Buchstaben $p_0, \ldots, p_{k-1}$. */
for i von 0 bis $n - k$
  /* Wir legen das Muster an der Stelle i an. */
  for j von 0 bis $k - 1$
    if $t_{i+j} \neq p_j$
      goto Abbruch

  drucke i; /* außerhalb der Schleife **for** j */
  Abbruch: /* Ziel des Sprungs aus der inneren Scheife */
  /* Ende des Rumpfs der Schleife **for** i */

Und nochmals anders

```
/* Text hat Buchstaben t_0, ..., t_{n−1}. */
/* Muster hat Buchstaben p_0, ..., p_{k−1}. */
for i von 0 bis n − k
   /* Wir legen das Muster an der Stelle i an. */
   j ← 0;
   while j < k und t_{i+j} = p_j
      j ← j + 1
   if j = k
      drucke i;
/* Ende des Rumpfs der Schleife for i */
```

Ich finde diese Version verständlicher als die vorherige Version. Aber das ist Geschmachssache.

Und nun das Orginal.

Dazu legen wir das Muster an jeder Stelle des Textes an und vergleichen Buchstabe für Buchstabe.

/* Text = $t_0 \dots t_{n-1}$. */
/* Muster = $p_0 \dots p_{k-1}$. */
for i von 0 bis $n-k$
  passt $\leftarrow$ wahr;
  /* Probiere die Stelle i. */
  for j von 0 bis $k-1$
    if $t_{i+j} \neq p_j$
      passt $\leftarrow$ falsch;
  if *passt* $=$ wahr
    drucke i;

- In **passt** merken wir uns, ob es schon ein Ungleich gegeben hat.
- Unelegant.
- Weniger effizient, da wir die innere Schleife immer ganz ausführen.
- In diesem Text ist **Variable** das einzige Wort, das mit **Va** anfängt. Außer bei den Vorkommen von **Variable** können wir die innere Schleife immer bei $j = 2$ verlassen.
- Aber **Text = aaaa...aaaaaaaa, Muster = aaaaaaaab**.

Zusammenfassung

- Der Wert von Variablen kann durch Wertzuweisungen geändert werden.
- Programme werden in Programmiersprachen (C, C++, Java, Python, usw) formuliert.
- Unsere Beispielprogramme würden in den genannten Programmiersprachen ähnlich aussehen,
  - allerdings mit historisch bedingten verwirrenden Schreibweisen (kleiner Zeichensatz): $x = 5$ statt $x \leftarrow 5$ und „Ist $x == y$?“ statt „Ist $x = y$?”.
- Algorithmen werden in Pseudocode formuliert. Detaillierungsgrad hängt vom Leserkreis ab.
- Falls Sie programmieren lernen möchten:
  - Python ist eine leicht zu lernende und ausdrucksstarke Sprache.
  - Calliope ist ein Kleinstcomputer für Schüler ab 8 Jahren.