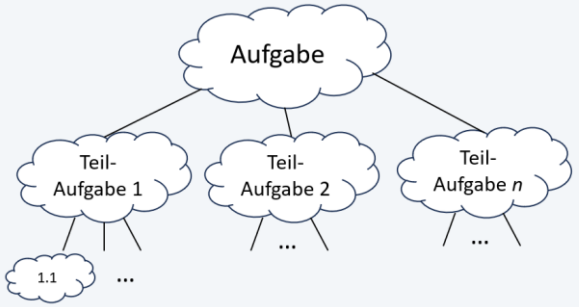
4 Entwurf von Algorithmen



4.1 Vorgehensmodell

4.2 Schrittweise Verfeinerung

4.3 Anwendungsbeispiel

4.1 Vorgehensmodell

Wir zerlegen im Sinne eines allgemeinen Vorgehensmodells den Entwurfs- und Implementierungsprozess für Algorithmen(systeme) in folgende Schritte:

1. Verschaffe dir Klarheit über die Aufgabenstellung
   (Was ist gegeben, was ist gesucht?
   Neben-, Ausnahme-, Umgebungsbedingungen, Beispiel-Szenarien)
   z.B. was sind die größten, was die kleinsten Werte
2. Fälle eine Entscheidung über die Entwurfsstrategie
   (z.B. schrittweise Verfeinerung)
3. Finde den grundsätzlichen Aufbau der Eingabeobjekte bzw. des Eingabestroms (Beispiele aus Schritt 1 verallgemeinern)
4. Definiere die Algorithmenschnittstelle(n)
   (passender Name, Eingangs –und Ausgangsparameter)
5. Entwickle eine Lösungsidee für die jeweilige (Teil-)Aufgabe

Vorgehensmodell

6. Transformiere die Lösungsidee in eine algorithmische Form (z.B. in Pseudocode)
7. Überprüfe die Korrektheit der algorithmischen Form der Lösung (z.B. "Handsimulation", typische Prüfungen: Wurde allen Datenobjekten ein Wert zugewiesen, bevor sie benutzt wurden? Werden alle Ergebnisobjekte ermittelt?)
8. Beurteile die Qualität des Algorithmus und überarbeite ihn gegebenenfalls (Korrespondenz zur Lösungsidee, Schnittstelle, Datenobjekte, Ablaufstruktur; Strukturqualität, Eleganz und Verständlichkeit)
9. Transformiere den Algorithmus in ein Programm und überprüfe dessen syntaktische Korrektheit (Beachtung programmiersprachenspezifischer Abweichungen vom Entwurf)

Vorgehensmodell

10. Teste das den Algorithmus repräsentierende Programm
    (statischer und dynamischer Test)
11. Vervollständige die Dokumentation des entworfenen Algorithmen-/Programmsystems

4.2 Das Prinzip der schrittweisen Verfeinerung

Für den systematischen Algorithmenentwurf hat der Turing-Award-Preisträger (sowie Pascal-Designer) Niklaus Wirth das so genannte Prinzip der schrittweisen Verfeinerung (*stepwise refinement*) vorgeschlagen:

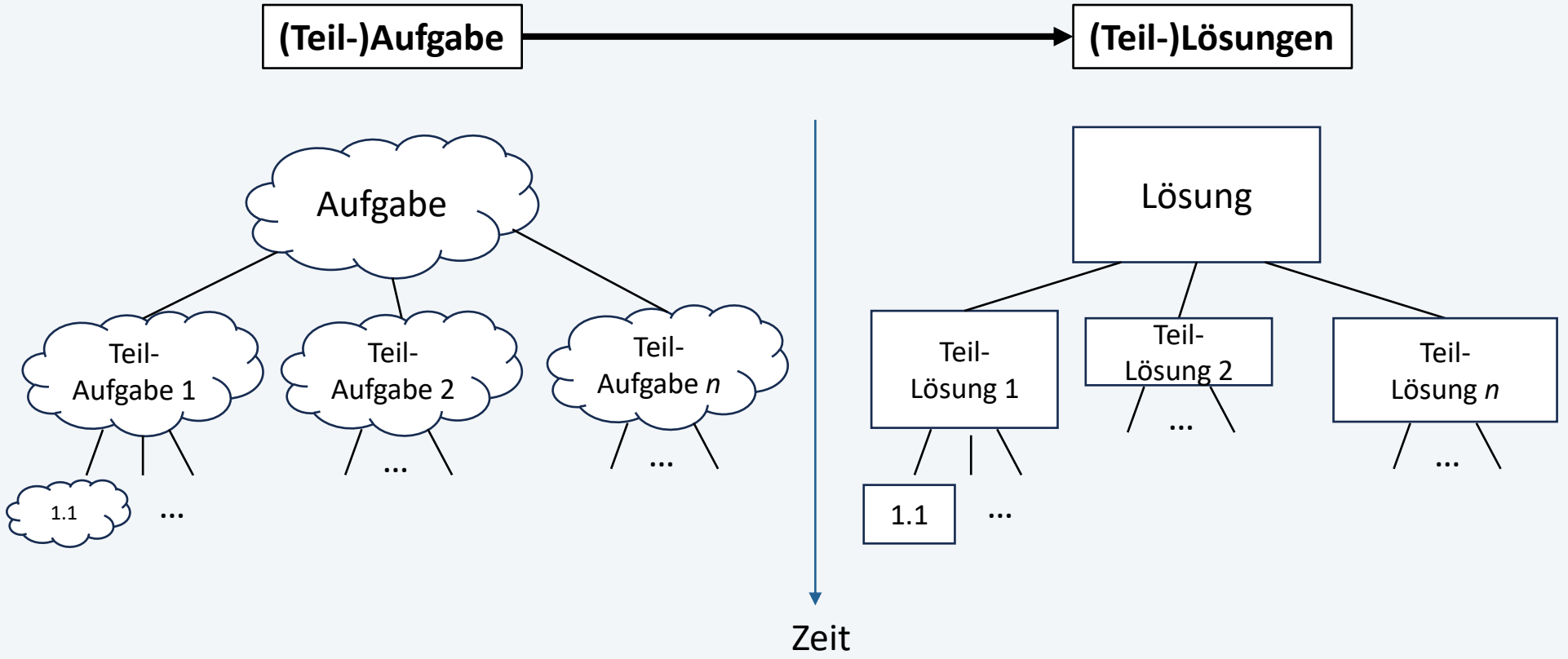
> Zerlege eine Aufgabe in Teilaufgaben. Betrachte jede Teilaufgabe möglichst losgelöst von den anderen Teil-aufgaben; zerlege sie weiter in Teilaufgaben, bis diese so einfach geworden sind, dass man dafür einen Algorithmus angeben kann.

N. Wirth

Die Zerlegung bzw. Abstraktion steht im Vordergrund, Unterscheidung von Wesentlichen und Unwesentlichen, Zurückstellung von Details

Das Prinzip der schrittweisen Verfeinerung

(Teil-)Aufgaben und ihre (Teil-)Lösungen



4.3 Anwendungsbeispiel

Aufgabenstellung (informelle Spezifikation)

- Ein Strom von einlaufenden Mails[1] ist vor Verteilung zu statistischen und dokumentarischen Zwecken zu verarbeiten.
- Jede Mail wird durch die Zeichenfolge ++++ abgeschlossen. Der Mail-Strom ist beendet, wenn eine leere Mail, gefolgt von der Zeichenfolge ++++, eintrifft.
- Die Mails sollen ohne das abschließende ++++ zeilenweise mit max. 120 Zeichen pro Zeile auf einem Ausgabemedium ausgegeben werden.
- Außerdem sollen die Wörter jeder Mail gezählt werden; Wörter mit mehr als n Zeichen sollen zudem extra gezählt werden.
- Nach der Ausgabe einer Mail sollen auf einer neuen Zeile die Anzahlen der Wörter insgesamt und der Wörter mit mehr als n Zeichen ausgegeben werden.
- Das längste zugelassene Wort hat y Zeichen; längere Wörter sollen zum Abbruch des Dokumentationsalgorithmus führen.

[1] Die allgemeine Bezeichnung *Mail* steht z.B. für einen Tweet oder einen Werbetext, der wortweise abgerechnet wird.

Beispiel: Mailverarbeitung (Schritt 1)

Schritt 1: Verschaffe dir Klarheit über die Aufgabenstellung

Beispielszenario:

Gegeben:

- Als Inputparameter: n, y z. B. `n = 10, y = 50`
- Als sequentieller Inputstrom auf einem Inputmedium: z.B.

```
Die drei Bundeslaender mit den meisten Neuinfektionen sind uebrigens die drei
mit vorgezogener Sperrstunde. ++++ Software ist der Geist in der Maschine.
Sie steckt zum Beispiel in der Kaffeemaschine, im Auto und natuerlich in
jedem Rechner: im Smartphone ebenso wie im Supercomputer. ++++ Das für
Dienstag angesetzt gewesene Nachtragsspiel der ICE Hockey League zwischen den
Bratislava Capitals und dem HC Bozen Suedtirol ist erneut verschoben worden.
++++ ++++
```

Beispiel: Mailverarbeitung (Schritt 1)

Schritt 1 (Fortsetzung): Verschaffe dir Klarheit über die Aufgabenstellung

Gesucht:

```
Die drei Bundeslaender mit den meisten Neuinfektionen sind uebrigens die drei mit vorgezoge

14 Worte 4 Worte ueberlang

Software ist der Geist in der Maschine. Sie steckt zum Beispiel in der Kaffeemaschine, im A
Rechner: im Smartphone ebenso wie im Supercomputer.

27 Worte 2 Worte ueberlang

Das für Dienstag angesetzt gewesene Nachtragsspiel der ICE Hockey League zwischen den Brati
Bozen Suedtirol ist erneut verschoben worden.

23 Worte 1 Worte ueberlang
```

Beispiel: Mailverarbeitung (Schritte 2 und 3)

Schritt 2: Fälle eine Entscheidung über die Entwurfsstrategie
Entwurfsstrategie: Schrittweise Verfeinerung

Schritt 3: Finde den grundsätzlichen Aufbau der Eingabeobjekte bzw. des Eingabestroms (Beispiele aus Schritt 1 verallgemeinern)

Annahme: *Mail*-Strom auf Input-Gerät (sequentiell) und Ausgabe auf Output-Gerät
Aufbau (*abstrakte Form*) des zu verarbeitenden Datenstromes:

m m m l
m m l
m l
l

anders geschrieben: { m } l

{} heißt: beliebig oft

m für Mail, *l* für leere Mail

Beispiel: Mailverarbeitung (Schritte 4, 5, 6)

Schritt 4: Definiere die Algorithmenschnittstelle(n)
(passender Name, Eingangs- und Ausgangsparameter)

Text selbst wird einfach per read eingelesen

```
ProcessMails(↓n: int ↓y: int)
```

Schritt 5: Entwickle eine Lösungsidee für die jeweilige (Teil-)Aufgabe

Lösungsidee für Gesamtaufgabe:

Lies (vom Input-Gerät) und wenn *m* gelesen, dann verarbeite *m* (zählen und Output produzieren), setze solange fort bis ein *l* (Leer-Mail) erkannt ist.

Schritt 6: Transformiere die Lösungsidee in eine algorithmische Form

```
ProcessMails(↓n: int ↓y: int)
  var empty: bool
begin
  repeat
    ReadAndProcessMail(↓n ↓y ↑empty)
  until empty
end ProcessMails
```

bricht ab, wenn Wort mit mehr als y Zeichen vorkommt

Nur eine Teilaufgabe wurde identifiziert (Zufall!). Sie ist einfacher, als die Gesamtaufgabe (nur mehr eine Mail ist zu bearbeiten).

Beispiel: Mailverarbeitung

Lösungsidee für Teilaufgabe `ReadAndProcessMail` (erste Verfeinerung):

- Aufbau (abstrakte Form) der zu verarbeitenden Mail:

W W W ... "++++"	wenn *Mail* nicht leer	{ W } "++++"
"++++"	wenn *Mail* leer	

w für Wort

- Fortlaufend jeweils ein Wort w lesen und verarbeiten (ausgeben, zählen) uzw. so lange, bis ein Wort mit dem Wert "++++" identifiziert wird. Danach Ausgabe der Zählergebnisse und Feststellung ob das bearbeitete Mail leer war.
- Lösungsidee für verarbeite Wort: Wort w ist gelesen. Prüfe ob relevantes Wort oder "++++". Wenn relevantes Wort, dann Wort w drucken, zählen und wenn mehr als n Zeichen extra zählen.

Beispiel: Mailverarbeitung

Algorithmus für Teilaufgabe `ReadAndProcessMail` (erste Verfeinerung)

```
ReadAndProcessMail(↓n: int ↓y: int ↑empty: bool)
  type
    Word
  var
    word: Word
begin
  repeat
    ReadWord(↓y ↑word)
    ProcessWord(↓word)
  until word = "++++"
  „gib Ergebnisse aus“
  „stelle Ausgangsparameter empty ein“
end ReadAndProcessMail
```

abstrakter Typ, der erst später konkretisiert wird

bricht ab, wenn Wort mit mehr als y Zeichen vorkommt

Beispiel: Mailverarbeitung

Algorithmus für Teilaufgabe `ReadAndProcessMail` (zweite Verfeinerung)

```
ReadAndProcessMail(↓n: int ↓y: int ↑empty: bool)
  type Word
  var word: Word; wCount, lwCount: int
begin
  wCount := 0; lwCount := 0
  repeat
    ReadWord(↓y ↑word)
    if word ≠ "++++" then
      PrintWord(↓word)
      wCount := wCount + 1
      if Length(↓word) > n then
        lwCount := lwCount + 1
      end
    end
  until word = "++++"
  if wCount > 0 then
    PrintResults(↓wCount ↓lwCount)
  end
  empty := (wCount = 0)
end ReadAndProcessMail
```

Verfeinerung für ProcessWord
Lösungsidee : Wort ist gelesen. Prüfe ob relevantes Wort oder "++++". Wenn relevantes Wort, dann Wort drucken, zählen und wenn mehr als n Zeichen extra zählen.

Drei neue Teilaufgaben wurden identifiziert und die Schnittstellen ihrer algorithm. Lösungen wurden festgelegt.

Beispiel: Mailverarbeitung

Lösungsidee für Teilaufgabe `ReadWord`

- Aufbau (abstrakte Form) der zu verarbeitenden Wörter:

  ```
  ЬЬЬ...CCC...Ь...
  CCC...Ь...
  ```
  } `{Ь}C{C}Ь{Ь}`

  c für beliebiges Zeichen, `ь` für Blank

- Eventuell vor einem Wort vorhandene (führende) Blanks überlesen und dann Wort zeichenweise aufbauen, d.h. Zeichen für Zeichen vom Inputmedium lesen, nach word übertragen, bis ein Wortende gefunden (= Blank tritt auf).

- Nebenbedingung: sind y Zeichen nach word übertragen und ist das vom Inputmedium zuletzt gelesene Zeichen kein Blank, so wird die Aktion beendet (Fehlermeldung und Stopp).

Beispiel: Mailverarbeitung

Algorithmus für Teilaufgabe `ReadWord`

```
ReadWord(↓y: int ↑word: Word)
  var
    firstChar: char
begin
  SkipBlanks(↑firstChar)
  FillWord(↓firstChar ↓y ↑word)
end ReadWord
```

firstChar enthält das erste Zeichen eines Wortes

bricht ab wenn wort zu lang wird

Verfeinerung von `SkipBlanks`

(trivial, daher keine verbal formul. Lösungsidee)

```
SkipBlanks(↑c: char)
begin
  repeat
    Read(↑c)
  until c ≠ ' '
end SkipBlanks
```

Wir gehen zunächst davon aus, dass die Mail-Struktur richtig ist, d. h. beim Lesen des Inputstromes kein Fehler auftritt.

Keine neuen abstrakten Operationen mehr eingeführt (Verfeinerungsende).

Beispiel: Mailverarbeitung

Lösungsidee für Teilaufgabe `FillWord`

- Das erste Zeichen des aufzubauenden Wortes wird vom aktivierenden Algorithmus übergeben und nach `word` übertragen.
- Danach soll jeweils ein Zeichen vom sequentiellen Inputstrom gelesen und geprüft werden, ob es relevant ist (≠ Blank). Wenn ja, wird das Zeichen, nachdem geprüft wurde, ob das Wort `word` nicht bereits die maximal zulässige Länge erreicht hat (wenn ja Fehlerstopp!), nach `word` übertragen. Dies wird so lange wiederholt, bis ein Wortende (Blank) gelesen wird.

Beispiel: Mailverarbeitung

Algorithmus für Teilaufgabe `FillWord`

```
FillWord(↓c: char ↓y: int ↑word: Word)
  var
    i: int -- aktuelle Anzahl der Zeichen von word
begin
  word := c; i := 1
  repeat
    Read(↑c)
    if c ≠ ' ' then
      if i = y then „Fehlerstopp“ return end
      word := word + c
      i := i + 1
    end
  until c = ' '
end FillWord
```

Konkatenationsoperator

Bemerkung: Diese Lösung setzt voraus, dass der Mail-Strom mit mindestens einem Blank abgeschlossen wird, d. h. hinter dem letzten "++++" mindestens ein Blank folgt.

Keine neuen abstrakten Operationen mehr eingeführt (Verfeinerungsende).

Beispiel: Mailverarbeitung

Lösungsidee Verfeinerung von `PrintWord`

Mit jedem Aufruf von `PrintWord` soll ein Wort (`word`) in der aktuellen Druckzeile ausgegeben werden, aber nur dann, wenn dadurch die Druckzeile nicht länger als 120 Zeichen wird.

Die aktuelle Länge der Druckzeile muss bei jedem Aufruf bekannt sein.

- (S1) Prüfe, ob eine neue Druckzeile begonnen werden muss. Wenn ja, veranlasse einen Zeilenvorschub, setze die aktuelle Zeilenlänge auf 0.
- (S2) Prüfe ob die aktuelle Zeilenlänge > 0 ist. Wenn ja, gib ein Worttrennzeichen (Blank) aus.
- (S3) Gib `word` aus und berechne die neue aktuelle Zeilenlänge.

Da wir bisher kein Konzept für persistente Variable eingeführt haben, müssen wir die aktuelle Zeilenlänge in Form eines Parameters realisieren.

Korrektur der Aufrufstelle erforderlich (großer Nachteil), siehe Seite 14

Beispiel: Mailverarbeitung

Algorithmus für Teilaufgabe `ReadAndProcessMail` (dritte Verfeinerung)

```
ReadAndProcessMail(↓n: int ↓y: int ↑empty: bool)
  type Word
  var word: Word; wCount, lwCount: int; lineLen: int
begin
  wCount := 0; lwCount := 0; lineLen := 0
  repeat
    ReadWord(↓y ↑word)
    if word ≠ "++++" then
      PrintWord(↓word ↓↑lineLen)
      wCount := wCount + 1
      if Length(↓word) > n then
        lwCount := lwCount + 1
      end
    end
  until word = "++++"
  if wCount > 0 then
    PrintResults(↓wCount ↓lwCount)
  end – if
  empty := (wCount = 0)
end ReadAndProcessMail
```

Die hier durchgeführte Korrektur der Aufrufstelle sowie Deklaration und Initialisierung des Datenobjekts `lineLen` (für Zeilenlänge) wurde durch die Verfeinerung `PrintWord` erforderlich.

Beispiel: Mailverarbeitung

Algorithmus Verfeinerung `PrintWord`

```
PrintWord(↓word: Word ↓↑lineLen: int)
begin
  assert 0 ≤ lineLen and lineLen ≤ 120
  if lineLen + 1 + Length(↓word) > 120 then
    WriteLn()
    lineLen := 0
  end

  assert 0 ≤ lineLen and lineLen ≤ (120 - 1 – Length(↓wort))
  if lineLen > 0 then
    Write(↓' ')
    lineLen := lineLen + 1
  end
  Write(↓ word)
  lineLen := lineLen + Length(↓ word)
end PrintWord
```

In der aktuellen Druckzeile sind `lineLen` Zeichen ausgegeben Position. `lineLen` enthält das letzte Zeichen des zuletzt ausgegebenen Wortes.

Wenn nicht erstes Wort einer Zeile, dann Worttrennung bilden (Blank ausgeben).

Beispiel: Mailverarbeitung

Verfeinerung von `PrintResults`

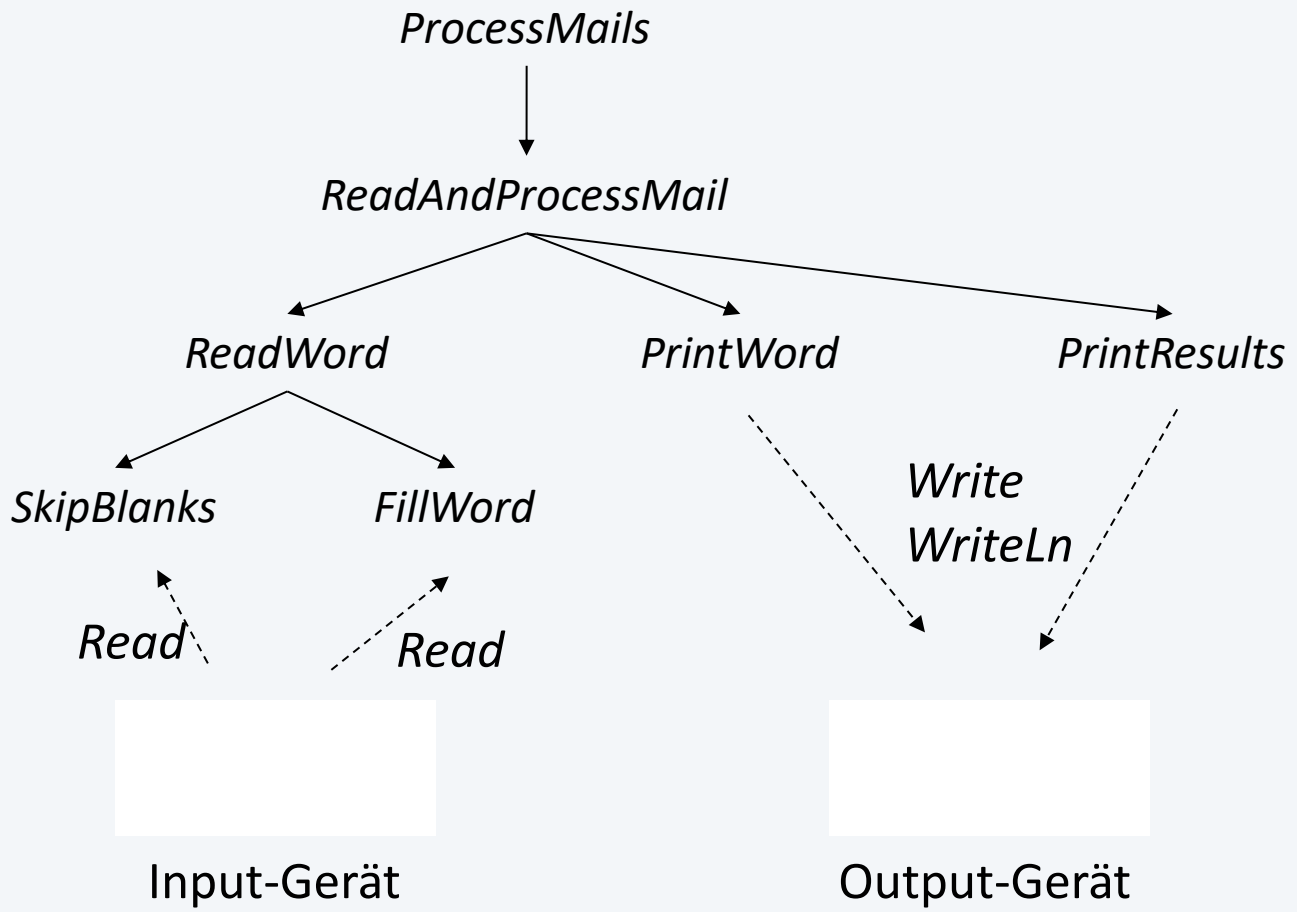
(trivial, daher keine verbal formulierte Lösungsidee)

```
PrintResults(↓wCount : int ↓lwCount : int)
begin
  WriteLn()
  Write(↓wCount ↓" Worte " ↓lwCount ↓" Worte überlang")
  WriteLn()
end PrintResults
```

Keine neuen abstrakten Operationen mehr eingeführt (Verfeinerungsende).

Beispiel: Mailverarbeitung

Übersicht Entwurfsergebnis: Algorithmensystem



Beispiel: Mailverarbeitung

Nun ist das Verfeinerungsende erreicht!

- Nur mehr die elementaren Aktionen `Read`, `Write`, `WriteLn`, `Length` und der Typ `Word` sind noch nicht verfeinert.
- Es wird angenommen, dass der vorgesehene Prozessor die entsprechenden Aktionen interpretieren und ausführen kann.
- Dem Typ `Word` ordnen wir zweckmäßigerweise den Standardtyp `string` (mit expliziter Längenbeschränkung auf 119 Zeichen) zu.

Auffälligkeit:

Die aktuelle Zeilenlänge ist ein Übergangsparameter (unschön!!!) und muss auch von `ReadAndProcessMail` behandelt werden, obwohl dort nicht wesentlich (Abhilfe siehe Kapitel 5)

Beispiel: Mailverarbeitung

Schritt 9: Transformiere den Algorithmus in ein Programm und überprüfe dessen syntaktische Korrektheit

WORD ist auch ein 2-Byte Integer Datentyp

Variable `word` wird in `wd` umbenannt, da `WORD` ein vordefinierter Datentyp ist.

Mail.pas U

Pro1xWS25 > vorlesung > Mail.pas

```
7   PROGRAM Mail;
20
21    PROCEDURE ProcessMails(n, y: INTEGER);
22      VAR
23        empty: BOOLEAN;
24    BEGIN
25      REPEAT
26        ReadAndProcessMail(n, y, empty);
27      UNTIL empty;
28    END; (* ProcessMails *)
29
30    PROCEDURE ReadAndProcessMail(n, y: INTEGER; VAR empty: BOOLEAN);
31      VAR
32        wd: STRING;
33        wCount, lwCount, lineLen: INTEGER;
34    BEGIN
35      wCount := 0;
36      lwCount := 0;
37      lineLen := 0;
38      REPEAT
39        ReadWord(y, wd);
40        IF wd <> endOfMail THEN BEGIN
41          PrintWord(wd, lineLen);
42          wCount := wCount + 1;
```

Auch ein Zeilenumbruch zählt als Zeichen

Command Prompt

```
C:\public\PascalWorkspaceWS20\Vorlesung\Kapitel 4>fpc Mail.pas
Free Pascal Compiler version 3.2.0 [2020/06/04] for i386
Copyright (c) 1993-2020 by Florian Klaempfl and others
Target OS: Win32 for i386
Compiling Mail.pas
Linking Mail.exe
114 lines compiled, 0.0 sec, 29568 bytes code, 1300 bytes data

C:\public\PascalWorkspaceWS20\Vorlesung\Kapitel 4>
```

```
24.10.2020  18:26            67 589 Mail.exe
               1 File(s)         67 589 bytes
               0 Dir(s)  401 582 874 624 bytes free
```

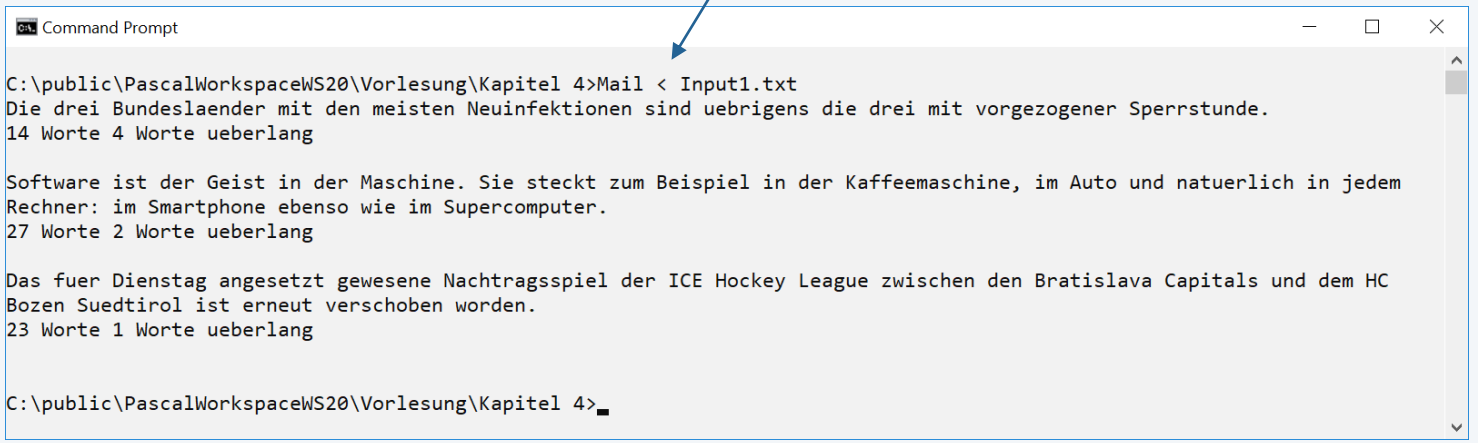
Beispiel: Mailverarbeitung

Schritt 10: Teste das den Algorithmus repräsentierende Programm (statischer und dynamischer Test)

Input1.txt

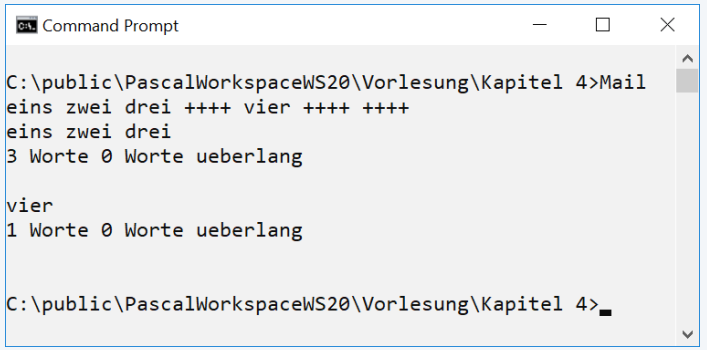
```
1  Die drei Bundeslaender mit den meisten
   Neuinfektionen sind uebrigens die drei mit
   vorgezogener Sperrstunde. ++++ Software ist der
   Geist in der Maschine. Sie steckt zum Beispiel in
   der Kaffeemaschine, im Auto und natuerlich in
   jedem Rechner: im Smartphone ebenso wie im
   Supercomputer. ++++ Das fuer Dienstag angesetzt
   gewesene Nachtragsspiel der ICE Hockey League
   zwischen den Bratislava Capitals und dem HC Bozen
   Suedtirol ist erneut verschoben worden. ++++ ++++
2
```

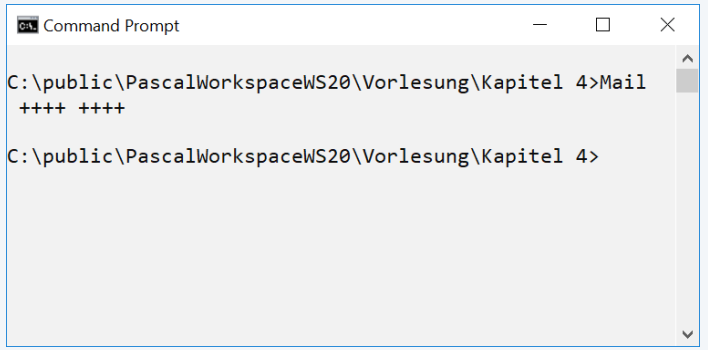
Leitet Text aus der Datei `Input1.txt` auf die Standardeingabe des Mail-Programms um.

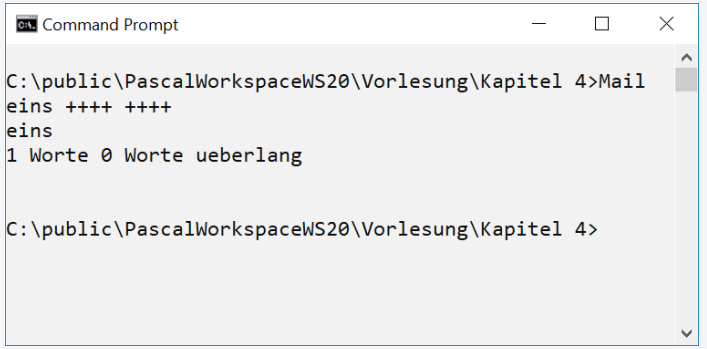


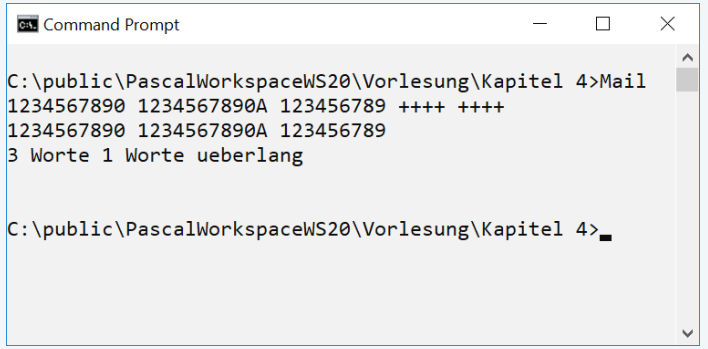
Beispiel: Mailverarbeitung

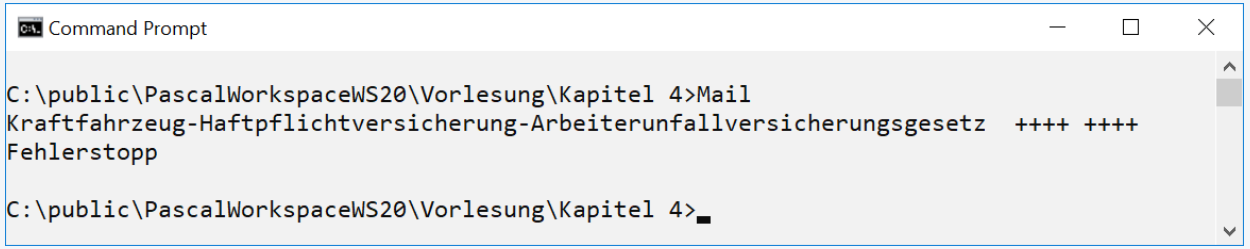
Schritt 10: Teste das den Algorithmus repräsentierende Programm (statischer und dynamischer Test)











Beispiel: Mailverarbeitung

Schritt 11: Vervollständige die Dokumentation des entworfenen Algorithmen-/Programmsystems

- Annahmen: Es wird angenommen, dass der Inputstrom s syntaktisch korrekt ist und daher folgendes gilt:

```
s  = { m } | ъ.                {beliebig oft}
m  = { w } "++++".
w  = { ъ } c { c } ъ { ъ }.
|  = "++++".
```

c ein Zeichen aus dem zu verwendenden Zeichensatz

- Andernfalls müsste der Inputstrom vor der Verarbeitung auf syntaktische Korrektheit geprüft und ggf. zurückgewiesen oder korrigiert werden.