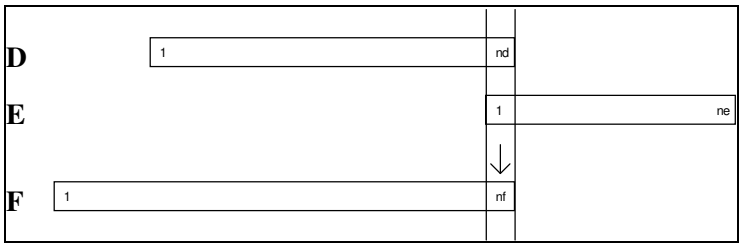
21. Übung zur
Konstruktion fehlerfreier Software

Frank Pilhofer

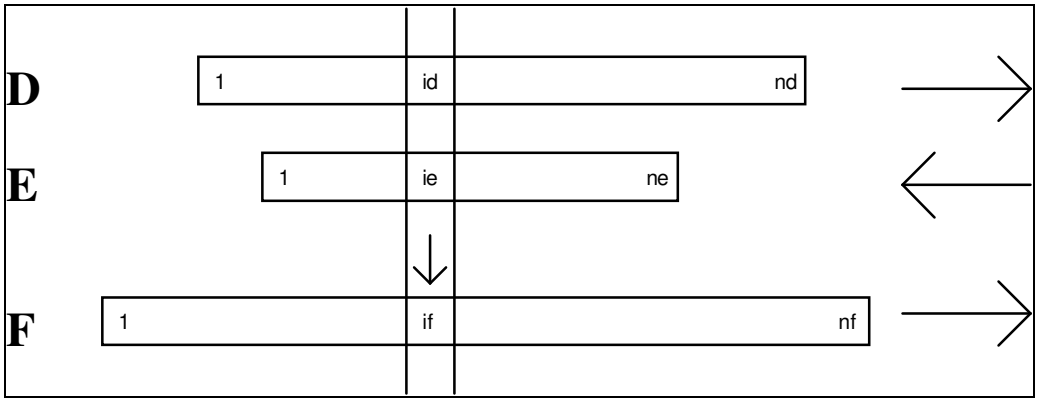
Montag 13-14 Uhr

Konstruktion eines Unterprogramms

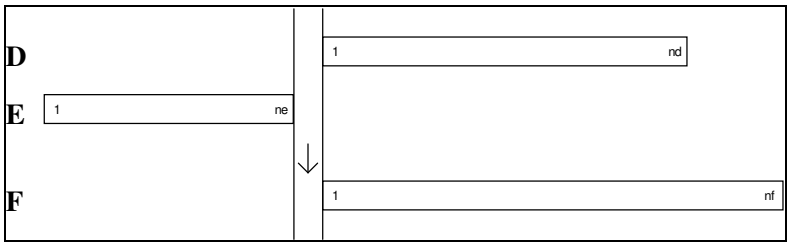
Zunächst wird die Schleifeninvariante festgelegt. Die Bedingungen der Schleifeninvariante überlegt man sich am besten grafisch:



Vor der Schleife



Innerhalb der Schleife



Nach Ende der Schleife

Die drei Felder werden so verschoben, daß ihre Felder am "Kopierpunkt" aufsteigend vorbeilaufen. Die Schleifeninvariante soll nun die Zustände vor Anfang und nach Ende des Schleifendurchlaufs zusammenfassen und all unsere Kenntnis der Zustände zu einem beliebigen Zeitpunkt innerhalb der Schleife beschreiben.
Wir wissen (zusätzlich zu den Bedingungen der Vorbedingung)

- Die Indizes id, ie und if zeigen auf Felder innerhalb des Arrays; am Ende der Schleife zeigen sie genau auf das erste Feld außerhalb.
- Das zuletzt nach F kopierte Feldelement ist kleiner als das nicht kopierte Feldelement von D bzw. E
- Der Feldindex in F läßt sich über die Indizes in E und D genau bestimmen

w Die beschriebenen Teile von F sind absteigend sortiert.

- Die beschriebenen Teile von F sind eine Permutation der gelesenen Teile der Felder D und E

Insgesamt folgt für die Schleifeninvariante:

$$I = V \wedge \begin{cases} 0 \le id \le nd \wedge 1 \le ie \le ne+1 \wedge 0 \le if \le nd+ne \wedge \\ \big((ie = ne+1 \vee F(if+1) \le E(ie)) \vee (nd = 0 \vee F(if+1) \le D(id))\big) \\ \wedge if = ne + id - ie + 1 \wedge \big(\wedge_{i=1}^{nd+ne-if-1} F(if+i) \ge F(if+i+1)\big) \wedge \\ \big[F(if+1),\ldots F(nd+ne)\big] \mathrm{Perm}\big[D(id+1),\ldots D(nd), E(1),\ldots E(ie-1)\big] \end{cases}$$

Die Initialisierung der Schleife wird so gewählt, daß der vorhin grafisch dargestellte Zustand erreicht wird:

```
declare (id, Z, nd), (ie, Z, 1), (if, Z, nd+ne);
```

Schleifenbedingung

Die Schleifenbedingung wird so gewählt, daß die Schleife erst terminiert, wenn der ebenfalls vorhin grafisch dargestellte Zustand erreicht ist, d.h.

$\overline{B} = id = 0 \wedge ie = ne+1$, d.h. $B = id > 0 \vee ie \le ne$

Bei der Negation der Bedingung wurde bereits die Schleifeninvariante berücksichtigt, obwohl das streng genommen nicht nötig wär.

Schleifenkern

Der Schleifenkern soll uns nun dem Ergebnis, dem sortierten Feld F, näher bringen. Deshalb muß im Schleifenkern die Variable if um 1 erniedrigt werden. Das zieht folgendes nach sich:

① Damit die Bedingung $if = ne + id - ie + 1$ immer noch gilt, muß entweder id erniedrigt oder ie erhöht werden.

② Der beschriebene Teil von F muß eine Permutation der gelesenen Teile von D und E bleiben, d.h. wenn id erniedrigt wurde, muß ein Element von D nach F kopiert werden; wenn ie erhöht wurden, muß ein Element von E nach F kopiert werden.

③ Falls ein Element von E kopiert wurde, d.h. $F(if+1) = E(ie-1)$, so muß $ie = ne+1 \vee F(if+1) \le E(ie)$ nicht unbedingt erfüllt sein. Analog gilt, falls ein Element von D kopiert wurde, d.h. $F(if+1) = D(id+1)$, so muß $id = 0 \vee F(if+1) \le D(id)$ nicht unbedingt erfüllt sein. Wenn wir jeweils das kleinere Element von $D(id+1)$ und $E(ie-1)$ kopieren, so wird eine dieser Bedingungen auf jeden Fall wahr. Eine Ausnahme ist, wenn nur noch Elemente in einem Feld zur Verfügung stehen, d.h. $id = 0 \vee ie = ne+1$. Wir schließen, daß, wenn beide Felder noch nicht "am Ende" sind, das kleinere Element kopiert werden muß. Das reicht aus, um Zeile 2 der Schleifeninvariante zu erfüllen.

④ Der vorige Punkt stellt sicher, daß das Feld F absteigend sortiert sein wird.

Zusammenfassung:

- Wenn die Felder noch nicht "am Ende" sind, so wird das kleinere Element von D bzw. E nach F kopiert.
- Der entsprechende Feldindex in D bzw. E wird erniedrigt bzw. erhöht.
- Anschließend wird der Feldindex in F erniedrigt.

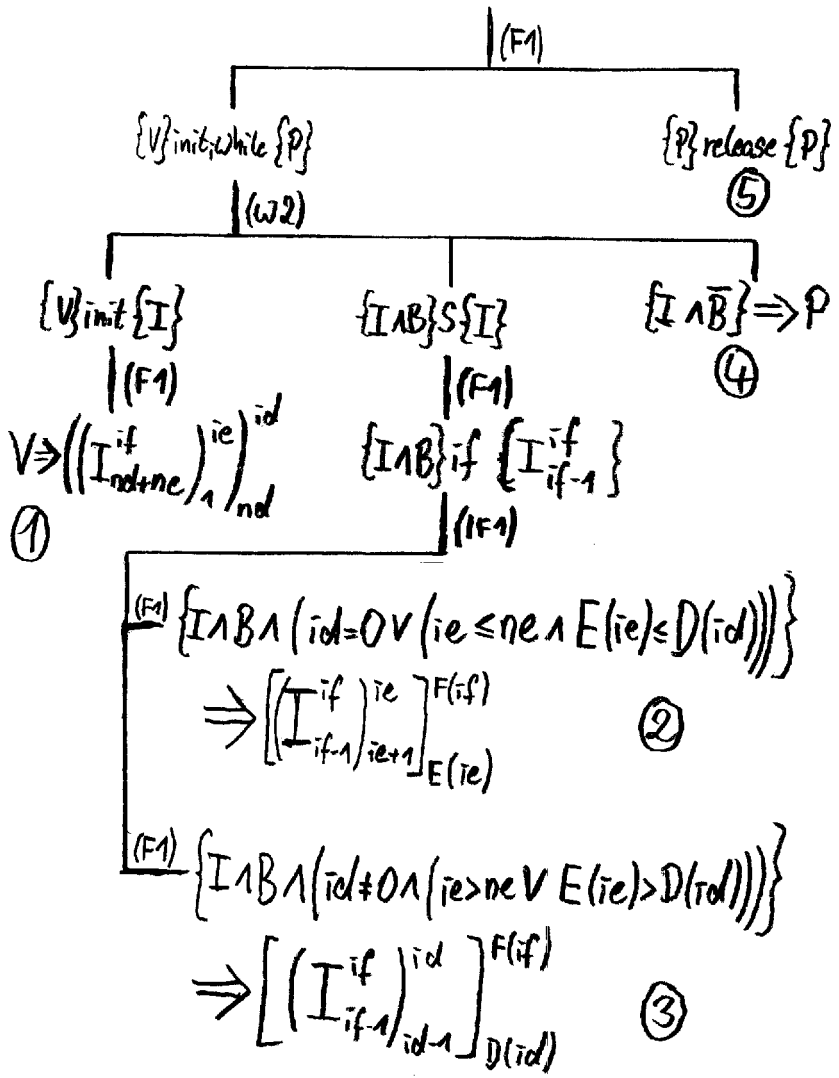
Beendigung der Schleife

Nach Beendigung der Schleife werden die Einzelindizes nicht mehr benötigt, und können verworfen werden.

Insgesamt

Insgesamt ergibt sich das folgende Unterprogramm:

```
declare (id, Z, nd), (ie, Z, 1), (if, nd+ne, 1);
while id>0 oder ie≤ne do
    if id=0 oder (ie≤ne und E(ie)≤D(id)) then
          F(if) := E(ie);  ie := ie+1;
    else
          F(if) := D(id);  id := id-1;
    end if;
    if := if - 1;
end while;
release id, ie, if;
```

Zwischenbedingungen

In der folgenden Grafik sind alle *zu beweisenden* Zwischenbedingungen aufgeführt. Triviale Implikationen, die durch Zuweisungen entstehen, sind der Übersichtlichkeit halber nicht einzeln aufgeführt.