Programming Languages: Imperative Program Construction
Practicals 9: Array Manipulation

Shin-Cheng Mu

Autumn Term, 2024

Typical Array Manipulation

1. Given $a : \textbf{array}\ [0..10)\ \textbf{of}\ Int$, compute $wp\ (a[i] := 0)\ (a[2] \neq 0)$.

Solution:

$$
\begin{array}{ll}
& wp\ (a[i] := 0)\ (a[2] \neq 0) \\
\equiv & 0 \leqslant i < 10 \wedge (a : i \mapsto 0)[2] \neq 0 \\
\equiv & \quad \{\ \text{function alteration}\ \} \\
& 0 \leqslant i < 10 \wedge (i = 2 \Rightarrow 0 \neq 0) \wedge (i \neq 2 \Rightarrow a[2] \neq 0) \\
\equiv & \quad \{\ 0 \neq 0 \equiv \mathit{False}\ \} \\
& 0 \leqslant i < 10 \wedge (i = 2 \Rightarrow \mathit{False}) \wedge (i \neq 2 \Rightarrow a[2] \neq 0) \\
\equiv & \quad \{\ P \Rightarrow \mathit{False} \equiv \neg\ P\ \} \\
& 0 \leqslant i < 10 \wedge i \neq 2 \wedge (i \neq 2 \Rightarrow a[2] \neq 0) \\
\equiv & \quad \{\ \text{proposition logic}\ \} \\
& 0 \leqslant i < 10 \wedge i \neq 2 \wedge a[2] \neq 0\ \ .
\end{array}
$$

2. Given constant $N, Y : Int$ with $0 \leqslant N$, and variables $b : \textbf{array}\ [0..N)\ \textbf{of}\ Int$, $x, i : Int$,
   (a) compute $wp\ (b[i-1] := x+1)\ \langle \forall j : i \leqslant j < N : b[j] = Y \rangle$.

Solution:

$$
\begin{array}{ll}
& wp\ (b[i-1] := x+1)\ \langle \forall j : i \leqslant j < N : b[j] = Y \rangle \\
\equiv & 0 \leqslant i - 1 < N \wedge \langle \forall j : i \leqslant j < N : (b : i - 1 \mapsto x + 1)[j] = Y \rangle \\
\equiv & \quad \{\ \text{since } i - 1 < j, \text{ function alteration}\ \} \\
& 1 \leqslant i \leqslant N \wedge \langle \forall j : i \leqslant j < N : b[j] = Y \rangle\ \ .
\end{array}
$$

   (b) Compute $wp\ (b[i-1] := x+1; i := i-1)\ \langle \forall j : i \leqslant j < N : b[j] = Y \rangle$.

Solution:

$$
\begin{array}{ll}
& wp\ (b[i-1] := x+1; i := i-1)\ \langle \forall j : i \leqslant j < N : b[j] = Y \rangle \\
\equiv & wp\ (b[i-1] := x+1)\ \langle \forall j : i - 1 \leqslant j < N : b[j] = Y \rangle \\
\equiv & 0 \leqslant i - 1 < N \wedge \langle \forall j : i - 1 \leqslant j < N : (b : i - 1 \mapsto x + 1)[j] = Y \rangle \\
\equiv & \quad \{\ \text{split off } j = i - 1\ \} \\
& 1 \leqslant i \leqslant N \wedge (b : i - 1 \mapsto x + 1)[i - 1] = Y \wedge \\
& \langle \forall j : i \leqslant j < N : (b : i - 1 \mapsto x + 1)[j] = Y \rangle \\
\equiv & \quad \{\ \text{function alteration}\ \} \\
& 1 \leqslant i \leqslant N \wedge x + 1 = Y \wedge \langle \forall j : i \leqslant j < N : b[j] = Y \rangle\ \ .
\end{array}
$$

3. Derive

con N : *Int* $\{1 \leqslant N\}$
con F : **array** $[0..N)$ **of** *Int*
var h : **array** $[0..N)$ **of** *Int*
running_sum
$\{\langle \forall k : 0 \leqslant k < N : h[k] = \langle \Sigma i : 0 \leqslant i \leqslant k : F[i] \rangle \rangle\}$.

Solution: This problem can be seen as a slightly varied instance of Simple Array Assignment mentioned in the handouts. We could have utilised the results. For practice, however, let's start from the basics.

Let $P\ n \equiv \langle \forall k : 0 \leqslant k < n : h[k] = \langle \Sigma i : 0 \leqslant i \leqslant k : F[i] \rangle \rangle$. Conjecture the following skeleton:

con N : *Int* $\{1 \leqslant N\}$
con F : **array** $[0..N)$ **of** *Int*
var h : **array** $[0..N)$ **of** *Int*
var n : *Int*
initialise
$\{P\ 1\}$
$n := 1$
$\{P\ n \wedge 1 \leqslant n \leqslant N, bnd : N - n\}$
do $n \neq N \rightarrow$ *step*
$n := n + 1$
od
$\{\langle \forall k : 0 \leqslant k < N : h[k] = \langle \Sigma i : 0 \leqslant i \leqslant k : F[i] \rangle \rangle\}$.

Note that $1 \leqslant N$, and we decided to start the loop with $n = 1$. The *initialise* statement thus has to be $h[0] := F[0]$. (Proof omitted – do it if it is not yet familiar to you!) The reason we start with $n = 1$ will be evident later.

We conjecture that *step* can be performed by a single array assignment $h[I] := E$. We then have to find I and E such that

$$P\ n \wedge 1 \leqslant n < N \Rightarrow (P\ (n+1))[h \backslash (h : I \rightarrow E)] \ .$$

Let us inspect $P\ (n+1)$, assuming $1 \leqslant n < N$:

$$\begin{array}{ll} & \langle \forall k : 0 \leqslant k < n+1 : h[k] = \langle \Sigma i : 0 \leqslant i \leqslant k : F[i] \rangle \rangle \\ \equiv & \quad \{\ 1 \leqslant n < N, \text{split off } k = n\ \} \\ & \langle \forall k : 0 \leqslant k < n : h[k] = \langle \Sigma i : 0 \leqslant i \leqslant k : F[i] \rangle \rangle \wedge \\ & h[n] = \langle \Sigma i : 0 \leqslant i \leqslant n : F[i] \rangle \ . \end{array}$$

(One could start with expanding $(P\ (n+1))[h \backslash (h : I \rightarrow E)]$ directly. I find it easier to take it slower.)

Now consider $(P\ (n+1))[h \backslash (h : I \rightarrow E)]$, assuming $P\ n \wedge 1 \leqslant n < N$

$$\begin{array}{ll} & \langle \forall k : 0 \leqslant k < n : (h : I \rightarrow E)[k] = \langle \Sigma i : 0 \leqslant i \leqslant k : F[i] \rangle \rangle \wedge \\ & (h : I \rightarrow E)[n] = \langle \Sigma i : 0 \leqslant i \leqslant n : F[i] \rangle \\ \equiv & \quad \{\ 1 \leqslant n < N, \text{split off } i = n. \text{ ((*) see the } \textbf{Think} \text{ remark in the end)}\ \} \\ & \langle \forall k : 0 \leqslant k < n : (h : I \rightarrow E)[k] = \langle \Sigma i : 0 \leqslant i \leqslant k : F[i] \rangle \rangle \wedge \\ & (h : I \rightarrow E)[n] = \langle \Sigma i : 0 \leqslant i \leqslant n-1 : F[i] \rangle + F[n] \\ \equiv & \quad \{\ P\ n\ \} \\ & \langle \forall k : 0 \leqslant k < n : (h : I \rightarrow E)[k] = h[k] \rangle \wedge \\ & (h : I \rightarrow E)[n] = h[n-1] + F[n] \\ \equiv & \quad \{\ \text{choose } I = n, E = h[n-1] + F[n]\ \} \\ & \langle \forall k : 0 \leqslant k < n : h[k] = h[k] \rangle \wedge \\ & h[n-1] + F[n] = h[n-1] + F[n] \\ \equiv & \ \mathit{True} \ . \end{array}$$

con $N : Int\ \{1 \leqslant N\}$
con F : **array** $[0..N)$ **of** Int
var h : **array** $[0..N)$ **of** Int
var $n : Int$
$h[0] := F[0]$
$\{P\ 1\}$
$n := 1$
$\{P\ n \wedge 1 \leqslant n \leqslant N, bnd : N - n\}$
do $n \neq N \rightarrow h[n] := h[n-1] + F[n]$
$n := n + 1$
od
$\{\langle \forall k : 0 \leqslant k < N : h[k] = \langle \Sigma i : 0 \leqslant i \leqslant k : F[i] \rangle \rangle\}$.

In retrospect, we need $1 \leqslant n < N$ to guarantee *def* $(h[n-1] + F[n])$ (that is, both array accesses are within bound). Therefore we have to start the loop with $n = 1$. Fortunately we can do so because $1 \leqslant N$.

Think: in the step labelled (*) above, why could we not do the following instead of the splitting?

$$
\begin{array}{ll}
 & \ldots \wedge (h : I \rightarrow E)[n] = \langle \Sigma i : 0 \leqslant i \leqslant n : F[i] \rangle \\
= & \quad \{\ P\ n\ \} \\
 & \ldots \wedge (h : I \rightarrow E)[n] = h[n] \ .
\end{array}
$$

Practice: try solving this problem using Simple Array Assignment.

4. Derive

con $N : Int\ \{1 \leqslant N\}$
var f : **array** $[0..N)$ **of** Int
con H : **array** $[0..N)$ **of** Int
decompose
$\{\langle \forall k : 0 \leqslant k < N : H[k] = \langle \Sigma i : 0 \leqslant i \leqslant k : f[i] \rangle \rangle\}$.

Solution: Similar to the previous exercise, we let $P\ n \equiv \langle \forall k : 0 \leqslant k < n : H[k] = \langle \Sigma i : 0 \leqslant i \leqslant k : f[i] \rangle \rangle$, and conjecture the following skeleton:

con $N : Int\ \{1 \leqslant N\}$
con H : **array** $[0..N)$ **of** Int
var f : **array** $[0..N)$ **of** Int
var $n : Int$
$f[0] := H[0]$
$\{P\ 1\}$
$n := 1$
$\{P\ n \wedge 1 \leqslant n \leqslant N, bnd : N - n\}$
do $n \neq N \rightarrow$ *step*
$n := n + 1$
od
$\{\langle \forall k : 0 \leqslant k < N : H[k] = \langle \Sigma i : 0 \leqslant i \leqslant k : f[i] \rangle \rangle\}$.

Conjecture that *step* can be performed by a single array assignment $f[I] := E$. We then have to find I and E such that

$P\ n \wedge 1 \leqslant n < N \Rightarrow (P\ (n+1))[f \backslash (f : I \rightarrow E)]$.

Inspect $P\ (n+1)$, assuming $1 \leqslant n < N$:

$$
\begin{array}{ll}
& \langle \forall k : 0 \leqslant k < n+1 : H[k] = \langle \Sigma i : 0 \leqslant i \leqslant k : f[i] \rangle \rangle \\
\equiv & \quad \{\ 1 \leqslant n < N, \text{split off } k = n\ \} \\
& \langle \forall k : 0 \leqslant k < n : H[k] = \langle \Sigma i : 0 \leqslant i \leqslant k : f[i] \rangle \rangle \wedge \\
& H[n] = \langle \Sigma i : 0 \leqslant i \leqslant n : f[i] \rangle \ .
\end{array}
$$

Now consider $(P\ (n+1))[f \backslash (f : I \rightarrow E)]$, assuming $P\ n \wedge 1 \leqslant n < N$

$$
\begin{array}{ll}
& \langle \forall k : 0 \leqslant k < n : H[k] = \langle \Sigma i : 0 \leqslant i \leqslant k : (f : I \rightarrow E)[i] \rangle \rangle \wedge \\
& H[n] = \langle \Sigma i : 0 \leqslant i \leqslant n : (f : I \rightarrow E)[i] \rangle \\
\equiv & \quad \{\ 1 \leqslant n < N, \text{split off } i = n\ \} \\
& \langle \forall k : 0 \leqslant k < n : H[k] = \langle \Sigma i : 0 \leqslant i \leqslant k : (f : I \rightarrow E)[i] \rangle \rangle \wedge \\
& H[n] = \langle \Sigma i : 0 \leqslant i \leqslant n-1 : (f : I \rightarrow E)[i] \rangle + (f : I \rightarrow E)[n] \\
\equiv & \quad \{\ \text{choose } I = n, \text{see below (*)}\ \} \\
& \langle \forall k : 0 \leqslant k < n : H[k] = \langle \Sigma i : 0 \leqslant i \leqslant k : f[i] \rangle \rangle \wedge \\
& H[n] = \langle \Sigma i : 0 \leqslant i \leqslant n-1 : f[i] \rangle + E \\
\equiv & \quad \{\ P\ n\ \} \\
& \langle \forall k : 0 \leqslant k < n : H[k] = \langle \Sigma i : 0 \leqslant i \leqslant k : f[i] \rangle \rangle \wedge \\
& H[n] = H[n-1] + E \\
\equiv & \quad \{\ \text{choose } E = H[n] - H[n-1]\ \} \\
& \langle \forall k : 0 \leqslant k < n : H[k] = \langle \Sigma i : 0 \leqslant i \leqslant k : f[i] \rangle \rangle \wedge \\
& H[n] = H[n-1] + (H[n] - H[n-1]) \\
\equiv & \quad \{\ P\ n\ \} \\
& \mathit{True}\ .
\end{array}
$$

In the step marked (*), since $0 \leqslant i \leqslant n-1$, by choosing $I = n$ both occurrences of $(f : I \rightarrow E)[i]$ reduce to $f[i]$. Meanwhile, $(f : I \rightarrow E)[n]$ reduces to E.

The derived program is

$$
\begin{array}{l}
\textbf{con}\ N : \mathit{Int}\ \{1 \leqslant N\} \\
\textbf{con}\ H : \textbf{array}\ [0..N)\ \textbf{of}\ \mathit{Int} \\
\textbf{var}\ f\ : \textbf{array}\ [0..N)\ \textbf{of}\ \mathit{Int} \\
\textbf{var}\ n\ : \mathit{Int} \\
f[0] := H[0] \\
\{P\ 1\} \\
n := 1 \\
\{P\ n \wedge 1 \leqslant n \leqslant N, \mathit{bnd} : N - n\} \\
\textbf{do}\ n \neq N \rightarrow f[n] := H[n] - H[n-1] \\
\qquad\qquad\quad n := n+1 \\
\textbf{od} \\
\{\langle \forall k : 0 \leqslant k < N : H[k] = \langle \Sigma i : 0 \leqslant i \leqslant k : f[i] \rangle \rangle\}\ .
\end{array}
$$

Swaps

5. Prove that

$\{h[0] = 0 \wedge h[1] = 1\}$ -- hence $h[h[0]] = 0$
swap $h\ (h[0])\ (h[1])$
$\{h[h[1]] = 1\}$

Solution: Assume $h[0] = 0 \wedge h[1] = 1$, we have

$$\begin{aligned} & (h : h[0], h[1] \rightarrow h[h[1]], h[h[0]]) \\ = & (h : 0, 1 \rightarrow h[1], h[0]) \\ = & (h : 0, 1 \rightarrow 1, 0) \ . \end{aligned}$$

Therefore, let $h' = (h : h[0], h[1] \rightarrow h[h[1]], h[h[0]])$,

$$\begin{aligned} & wp\ (swap\ h\ (h[0])\ (h[1]))\ (h[h[1]] = 1) \\ \equiv\ & h'[h'[1]] = 1 \\ \equiv\ & h'[0] = 1 \\ \equiv\ & 1 = 1 \\ \equiv\ & True \ . \end{aligned}$$

6. Given h : **array** $[0..N)$ **of** A, prove the rule that when h does not occur free in E and F,

$$\{\langle \forall i : 0 \leqslant i < N \wedge i \neq E \wedge i \neq F : h[i] = H\ i \rangle \wedge h[E] = X \wedge h[F] = Y\}$$
swap h E F
$$\{\langle \forall i : 0 \leqslant i < N \wedge i \neq E \wedge i \neq F : h[i] = H\ i \rangle \wedge h[E] = Y \wedge h[F] = X\} \ .$$

Notes:

- Recall that E and F are expressions, while X, Y, H are logical variables. It means that, for example, one can conclude immediately $X[z \backslash w] = X$ for $z \neq X$, while to determine whether $E[z \backslash w] = E$ we have to look into $E - E[z \backslash w] = E$ if z does not occur free in E.
- With $h[E] = X$, for example, we implicitly assume that $def\ (h[E])$ holds.

Solution: Abbreviate $(h : E, F \rightarrow h[F], h[E])$ to h'. We reason:

$$\begin{aligned} & wp\ (swap\ h\ E\ F)\ (\langle \forall i : 0 \leqslant i < N \wedge i \neq E \wedge i \neq F : h[i] = H\ i \rangle \wedge h[E] = Y \wedge h[F] = X) \\ \equiv\ & \quad \{ \text{ definition of } wp; N, H, X, Y \text{ are logical variables } \} \\ & \langle \forall i : 0 \leqslant i < N \wedge i \neq (E[h \backslash h']) \wedge i \neq (F[h \backslash h']) : h'[i] = H\ i \rangle \wedge h'[E[h \backslash h']] = Y \wedge h'[F[h \backslash h']] = X \\ \equiv\ & \quad \{ \ h \text{ does not occur free in } E \text{ and } F \ \} \\ & \langle \forall i : 0 \leqslant i < N \wedge i \neq E \wedge i \neq F : h'[i] = H\ i \rangle \wedge h'[E] = Y \wedge h'[F] = X \\ \equiv\ & \quad \{ \text{ function alteration: } h'[i] = h[i] \text{ for } i \neq E \wedge i \neq F \ \} \\ & \langle \forall i : 0 \leqslant i < N \wedge i \neq E \wedge i \neq F : h[i] = H\ i \rangle \wedge h'[E] = Y \wedge h'[F] = X \\ \equiv\ & \quad \{ \text{ function alternation } \} \\ & \langle \forall i : 0 \leqslant i < N \wedge i \neq E \wedge i \neq F : h[i] = H\ i \rangle \wedge h[F] = Y \wedge h[E] = X \ . \end{aligned}$$

7. Derive the following program, where arrays are manipulated only by swapping.

con N : *Int* $\{0 \leqslant N\}$
var h : **array** $[0..N)$ **of** *Int*
var p : *Int*
?
$$\{0 \leqslant p \leqslant N \wedge \langle \forall i : 0 \leqslant i < p : h[i] \leqslant 0 \rangle \wedge \langle \forall i : p \leqslant i < N : 0 \leqslant h[i] \rangle\} \ .$$

Solution: As the usual practice, we use an up-loop in which n is incremented in the end. Let $P\ n = \langle \forall i : 0 \leqslant i < p : h[i] \leqslant 0 \rangle \wedge \langle \forall i : p \leqslant i < n : 0 \leqslant h[i] \rangle$. The plan is:

con $N : Int\ \{0 \leqslant N\}$
var h : **array** $[0..N)$ **of** Int
var $p, n : Int$

$p, n := 0, 0$
$\{0 \leqslant p \leqslant n \leqslant N \wedge P\ n, bnd : N - n\}$
do $n \neq N \rightarrow ... n := n + 1$ **od**
$\{0 \leqslant p \leqslant N \wedge P\ N\}$.

Assuming $0 \leqslant p \leqslant n < N$, examine $P\ (n + 1)$:

$\langle \forall i : 0 \leqslant i < p : h[i] \leqslant 0 \rangle \wedge \langle \forall i : p \leqslant i < n + 1 : 0 \leqslant h[i] \rangle$
$\equiv$ { since $0 \leqslant n < N$, split off $i = n$ }
$\langle \forall i : 0 \leqslant i < p : h[i] \leqslant 0 \rangle \wedge \langle \forall i : p \leqslant i < n : 0 \leqslant h[i] \rangle \wedge 0 \leqslant h[n]$
$\equiv P\ n \wedge 0 \leqslant h[n]$.

Therefore, if $0 \leqslant h[n]$ there is nothing more we need to do before $n := n + 1$. We can introduce an **if** and put $n := n + 1$ under a guard $0 \leqslant h[n]$.

To make the **if** total we consider what to do when $h[n] \leqslant 0$. In this case we consider two cases.

Case: $p \neq n$. We aim to construct

$\{\langle \forall i : 0 \leqslant i < p : h[i] \leqslant 0 \rangle \wedge \langle \forall i : p \leqslant i < n : 0 \leqslant h[i] \rangle \wedge h[n] \leqslant 0 \wedge 0 \leqslant p < n < N\}$
???
$\{P\ (n + 1) \wedge 0 \leqslant p \leqslant n + 1 \leqslant N\}$
$n := n + 1$
$\{P\ n \wedge 0 \leqslant p \leqslant n \leqslant N\}$

Since $p \neq n$, we can split $i = p$ from $\langle \forall i : p \leqslant i < n : 0 \leqslant h[i] \rangle$, resulting in

$\{\langle \forall i : 0 \leqslant i < p : h[i] \leqslant 0 \rangle \wedge 0 \leqslant h[p] \wedge \langle \forall i : p + 1 \leqslant i < n : 0 \leqslant h[i] \rangle \wedge 0 \leqslant p < n < N \wedge h[n] \leqslant 0\}$
$swap\ h\ p\ n$
$\{\langle \forall i : 0 \leqslant i < p : h[i] \leqslant 0 \rangle \wedge h[p] \leqslant 0 \wedge \langle \forall i : p + 1 \leqslant i < n : 0 \leqslant h[i] \rangle \wedge 0 \leqslant p < n < N \wedge 0 \leqslant h[n]\}$
$p := p + 1$
$\{P\ (n + 1) \wedge 0 \leqslant p \leqslant n + 1 \leqslant N\}$
$n := n + 1$
$\{P\ n \wedge 0 \leqslant p \leqslant n \leqslant N\}$.

Case: $p = n$. In this case the range $p \leqslant i < n$ is empty and the precondition reduces as such:

$\{\langle \forall i : 0 \leqslant i < p : h[i] \leqslant 0 \rangle \wedge 0 \leqslant p = n < N \wedge h[n] \leqslant 0\}$
???
$\{P\ (n + 1) \wedge 0 \leqslant p \leqslant n + 1 \leqslant N\}$
$n := n + 1$
$\{P\ n \wedge 0 \leqslant p \leqslant n \leqslant N\}$.

It turns out that the same code still works:

$\{\langle \forall i : 0 \leqslant i < p : h[i] \leqslant 0 \rangle \wedge 0 \leqslant p = n < N \wedge h[n] \leqslant 0\}$
$swap\ h\ p\ n$
$p := p + 1$
$\{P\ (n + 1) \wedge 0 \leqslant p \leqslant n + 1 \leqslant N\}$
$n := n + 1$
$\{P\ n \wedge 0 \leqslant p \leqslant n \leqslant N\}$.

Therefore the code is:

con $N : Int\ \{0 \leqslant N\}$
var h : **array** $[0..N)$ **of** Int
var $p, n : Int$
$p, n := 0, 0$
$\{0 \leqslant p \leqslant n \leqslant N \wedge P\ n, bnd : N - n\}$
do $n \neq N \rightarrow$
  if $0 \leqslant h[n] \rightarrow n := n + 1$
  $\mid\ h[n] \leqslant 0 \rightarrow swap\ h\ p\ n$
    $p, n, := p + 1, n + 1$
  fi
od
$\{0 \leqslant p \leqslant N \wedge P\ N\}$.

8. The following is a specification of sorting:

con $N : Int\ \{0 \leqslant N\}$
var h : **array** $[0..N)$ **of** Int
sort
$\{\langle \forall i\ j : 0 \leqslant i \leqslant j < N : h[i] \leqslant h[j] \rangle\}$.

where *sort* mutates the array h only by swapping. Derive a $O(N^2)$ algorithm for sorting. The algorithm will contain a loop within a loop. The outer loop uses as invariant $P_0 \wedge P_1$, where

$P_0 \equiv \langle \forall i : 0 \leqslant i < n : \langle \forall j : i \leqslant j < N : h[i] \leqslant h[j] \rangle \rangle$,
$P_1 \equiv 0 \leqslant n \leqslant N$.

The inner loop uses Q as *part of* its invariant:

$Q \equiv \langle \forall j : k \leqslant j < N : h[n] \leqslant h[j] \rangle$.

Solution: The invariant is designed such that $n := 0$ establishes $P_0 \wedge P_1$, while $P_0 \wedge P_1 \wedge n = N$ meets the postcondition. Therefore, the outline of the program could be:

con $N : Int\ \{0 \leqslant N\}$
var h : **array** $[0..N)$ **of** Int
var $n : Int$
$n := 0$
$\{P_0 \wedge P_1, bnd : N - n\}$
do $n \neq N \rightarrow$
  inner_loop
  $\{P_0 \wedge P_1 \wedge \langle \forall j : n \leqslant j < N : h[n] \leqslant h[j] \rangle \wedge n \neq N\}$ -- (*)
  $n := n + 1$
od
$\{\langle \forall i\ j : 0 \leqslant i \leqslant j < N : h[i] \leqslant h[j] \rangle\}$.

The assertion (*) before $n := n + 1$ is calculated by:

$$
\begin{array}{ll}
 & (P_0 \wedge P_1)[n\backslash n+1] \\
\equiv & \langle \forall i : 0 \leqslant i < n+1 : \langle \forall j : i \leqslant j < N : h[i] \leqslant h[j] \rangle \rangle \wedge 0 \leqslant n+1 \leqslant N \\
\Leftarrow & \quad \{ \text{ with } 0 \leqslant n < N, \text{ split off } i = n \ \} \\
 & \langle \forall i : 0 \leqslant i < n : \langle \forall j : i \leqslant j < N : h[i] \leqslant h[j] \rangle \rangle \wedge \\
 & \langle \forall j : n \leqslant j < N : h[n] \leqslant h[j] \rangle \wedge 0 \leqslant n < N \\
\equiv & \quad \{ \text{ def. of } P_0 \text{ and } P_1 \ \} \\
 & P_0 \wedge P_1 \wedge \langle \forall j : n \leqslant j < N : h[n] \leqslant h[j] \rangle \wedge n \neq N \ .
\end{array}
$$

We now try to construct the *inner_loop*. Compare the hint Q and the assertion (*), we note that

- $P_0 \wedge P_1 \wedge Q \wedge k = n$ establishes (*), and
- letting $k := N - 1$ establishes Q, and
- being in the outer loop, we have $P_1 \wedge n \neq N$, which is $0 \leqslant n < N$, therefore by choosing $k := N - 1$ we still have $0 \leqslant n \leqslant k < N$.

Therefore we start with trying:

$\{P_0 \wedge P_1 \wedge n \neq N\}$
$k := N - 1$
$\{P_0 \wedge Q \wedge 0 \leqslant n \leqslant k < N\}$
do $k \neq n \rightarrow$
  ?
  $k := k - 1$
od
$\{P_0 \wedge P_1 \wedge \langle \forall j : n \leqslant j < N : h[n] \leqslant h[j] \rangle \wedge n \neq N\}$
$n := n + 1$

To construct ? we examine $Q[k\backslash k-1]$, assuming $0 \leqslant n < k < N$:

$$
\begin{array}{ll}
 & \langle \forall j : k \leqslant j < N : h[n] \leqslant h[j] \rangle [k\backslash k-1] \\
\equiv & \langle \forall j : k-1 \leqslant j < N : h[n] \leqslant h[j] \rangle \\
\equiv & \quad \{ \text{ with } 0 \leqslant n < k < N, \text{ split off } j = k-1 \ \} \\
 & \langle \forall j : k \leqslant j < N : h[n] \leqslant h[j] \rangle \wedge h[n] \leqslant h[k-1] \\
\equiv & Q \wedge h[n] \leqslant h[k-1] \ .
\end{array}
$$

If $h[n] \leqslant h[k-1]$ already holds, we need only a *skip*. If $h[n] \geqslant h[k-1]$ holds instead, we do a *swap h n* $(k-1)$, whose validity can be established by:

$$
\begin{array}{ll}
 & wp\ (swap\ h\ n\ (k-1))\ ((P_0 \wedge Q \wedge 0 \leqslant n \leqslant k < N)[k\backslash k-1]) \\
\equiv & \quad \{ \text{ calculation above, } k \text{ not occurring free in } P_0 \ \} \\
 & wp\ (swap\ h\ n\ (k-1))\ (P_0 \wedge Q \wedge h[n] \leqslant h[k-1] \wedge 0 \leqslant n \leqslant k-1 < N) \\
\equiv & \quad \{ \text{ let } h' = (h : n, k-1 \rightarrow h[k-1], h[n]) \ \} \\
 & P_0[h\backslash h'] \wedge Q[h\backslash h'] \wedge h'[n] \leqslant h'[k-1] \wedge 0 \leqslant n \leqslant k-1 < N \ .
\end{array}
$$

Consider the first three terms, $h'[n] \leqslant h'[k-1]$ equivals $h[k-1] \leqslant h[n]$ by the definition of function alteration (this is why we do *swap h n* $(k-1)$ in the first place). For the second term, we have $Q[h\backslash h'] \Leftarrow Q \wedge h[k-1] \leqslant h[n]$:

$$
\begin{array}{ll}
 & Q[h\backslash h'] \\
\equiv & \langle \forall j : k \leqslant j < N : h'[n] \leqslant h'[j] \rangle \\
\equiv & \quad \{ \ h' = (h : n, k-1 \rightarrow h[k-1], h[n]) \text{ and thus } h'\ j = h\ j \text{ for } k \leqslant j < N \ \} \\
 & \langle \forall j : k \leqslant j < N : h[k-1] \leqslant h[j] \rangle \\
\Leftarrow & \langle \forall j : k \leqslant j < N : h[n] \leqslant h[j] \rangle \wedge h[k-1] \leqslant h[n] \\
\equiv & Q \wedge h[k-1] \leqslant h[n] \ .
\end{array}
$$

Consider $P_0[h\backslash h']$, assuming $0 \leqslant n < k < N$:

$$
\begin{array}{ll}
 & P_0[h\backslash h'] \\
\equiv & \langle \forall i : 0 \leqslant i < n : \langle \forall j : i \leqslant j < N : h'[i] \leqslant h'[j] \rangle \rangle \\
\equiv & \quad \{\ h' = (h{:}n, k-1 \mathrel{\triangleright} h[k-1], h[n]) \text{ and } n < k, \text{ thus } h'\ i = h\ i \text{ for } 0 \leqslant i < n\ \} \\
 & \langle \forall i : 0 \leqslant i < n : \langle \forall j : i \leqslant j < N : h[i] \leqslant h'[j] \rangle \rangle \\
\equiv & \quad \{\ \text{with } 0 \leqslant n < k < N, \text{ split off } j = n \text{ and } j = k-1\ \} \\
 & \langle \forall i : 0 \leqslant i < n : \langle \forall j : i \leqslant j < N \wedge j \neq n \wedge j \neq k-1 : h[i] \leqslant h'[j] \rangle \wedge \\
 & \qquad\qquad h[i] \leqslant h'[n] \wedge h[i] \leqslant h'[k-1] \rangle \\
\equiv & \quad \{\ h'\ j = h\ j \text{ within } i \leqslant j < N \wedge j \neq n \wedge j \neq k-1\ \} \\
 & \langle \forall i : 0 \leqslant i < n : \langle \forall j : i \leqslant j < N \wedge j \neq n \wedge j \neq k-1 : h[i] \leqslant h[j] \rangle \wedge \\
 & \qquad\qquad h[i] \leqslant h[k-1] \wedge h[i] \leqslant h[n] \rangle \\
\equiv & \quad \{\ \text{split off } j = n \text{ and } j = k-1, \text{ reversed}\ \} \\
 & \langle \forall i : 0 \leqslant i < n : \langle \forall j : i \leqslant j < N : h[i] \leqslant h[j] \rangle \rangle \\
\equiv & P_0\ .
\end{array}
$$

Therefore we have

$$
\begin{array}{ll}
 & P_0[h\backslash h'] \wedge Q[h\backslash h'] \wedge h'[n] \leqslant h'[k-1] \wedge 0 \leqslant n \leqslant k-1 < N \\
\Leftarrow & \quad \{\ \text{calculation above, some of them assuming } 0 \leqslant n < k < N\ \} \\
 & P_0 \wedge Q \wedge h[k-1] \leqslant h[n] \wedge 0 \leqslant n < k < N\ .
\end{array}
$$

Note that, in deriving the inner loop we cannot forget about P_0 — one still has to prove that P_0 is preserved.

In conclusion, the program we derived is:

```
con N : Int {0 ⩽ N}
var h : array [0..N) of Int
var n, k : Int

n := 0
{P₀ ∧ 0 ⩽ n ⩽ N, bnd : N − n}
do n ≠ N →
   k := N − 1
   {P₀ ∧ Q ∧ 0 ⩽ n ⩽ k < N, bnd : k}
   do k ≠ n → if h[n] ⩽ h[k − 1] → skip
               | h[n] ⩾ h[k − 1] → swap h n (k − 1)
               fi
               k := k − 1
   od
   {P₀ ∧ ⟨∀j : n ⩽ j < N : h[n] ⩽ h[j]⟩ ∧ 0 ⩽ n < N}
   n := n + 1
od
{⟨∀i j : 0 ⩽ i ⩽ j < N : h[i] ⩽ h[j]⟩} .
```