This note has appeared as THI-TW-Memorandum 249, March 1979, by Haarten M. Fokkinga.
This note is a corrected version of TW-Memo 249.

A SIMPLER CORRECTNESS PROOF OF AN IN-PLACE PERMUTATION ALGORITHM.

Abstract. In 1972 Duijvestijn gave a correctness proof of a particular permutation algorithm using an invariant relation. We present another proof based on this relation. It uses ghost variables and consequently can be split up into easily comprehensible parts. This might be of interest to the reader. The verification itself is hardly of interest: the application of the predicate transformation rules is straightforward and involves nearly no mathematics.

The program. The program to be proved correct rearranges a variable
v: array [0..n-1] of T with initial value V, according to a given permutation
F of 0..n-1, using only auxiliary variables which do not depend on
n or on T. Thus the program establishes
R: A i: 0..n-1. v(i) = V(F(i))

Using here and in the sequel the notation of (Dijkstra 76), the program reads
j := 0;
do j ≠ n-1 +
    q:=F(j); do q<j + q:=F(q) od;
    v:=swap(j,q); j:=j+1
od.

We will give the correctness proof together with a construction of the program. We want to stress again that the verification of the invariant relations is merely a boring formula manipulation, involving no interesting mathematics. We present it only to contrast it with (Duijvestijn 72).

A "stepping stone" program: using a ghost variable

We try to establish R by means of a repetition with the invariant relation (found by standard techniques)
0 ≤ j < n and A i: 0..j-1. v(i) = V(F(i))

In order to know how the remaining elements of v need to be arranged yet, we introduce an array variable f, representing a permutation of
j..n-1, such that
A i: j..n-1. v(f(i)) = V(F(i))
Letting f(i) = i for i: 0..j-1, we can express the complete invariant relation as V0 and P1:
P0: $f$ denotes a permutation of $0..n-1$.

P1: $0 \leq j < n$ and $A i: 0..n-1. v(f(i)) = V(F(i))$ and $A i: 0..j-1. f(i) = i$.

The program, then, has the structure

$$f := F; j := 0 \{v = V; hence P0 and P1 established\};$$
$$\text{do "maintain P0 and P1, decrease n-j" od \{R\}}.$$ 

There are several ways to derive or invent the refinement

"maintain P0 and P1, decrease n-j":

$$j \neq n-1 \rightarrow v : \text{swap}(j,f(j)); f : \text{swap}(f^{-1}(j),j); j := j+1.$$ 

We will give the verification of the invariance only.

Recall the semantics of assignment and swapping.

$$wp(x := e, P) = P[x \leftarrow e].$$ 

$$wp(a : \text{swap}(x,y), P) = P[a \leftarrow a\prime],$$ where $a\prime = a[x \leftarrow a(y), y \leftarrow a(x)]$.

In general, the array value $a\prime = a[x \leftarrow el, y \leftarrow e2]$ is defined

for any $a, x, y, el, e2$ with $x \neq y$ or $el = e2$, as follows

$$a\prime(i) = \begin{cases} a(i) & \text{for } i \text{ different from } x \text{ and } y \\
 el & \text{for } i = x \\
 e2 & \text{for } i = y.\end{cases}$$ 

Now we prove the invariance; first of P0 and then of P1.

Because $f$ is subject to swap only, P0 is kept invariant. Formally this is shown as follows.

$$wp(v : \text{swap}(j, f(j)); f : \text{swap}(f^{-1}(j),j); j := j+1, P0) =$$

$$= ((P0[j + j+1])\{f + f\prime\}[v + v\prime])$$

where $f\prime = f[f^{-1}(j) + f(j), j + f(f^{-1}(j))]$, $v\prime = \ldots.$

$= f \circ p$ is a permutation, where $p$ is the pair exchange "$f^{-1}(j) \leftrightarrow j$"

and this holds true, because $f$ being a permutation on account of P0,

and $p$ being a permutation, so is the composition $f \circ p$.

(Note that, $f$ being a permutation, the inverse $f^{-1}$ is well defined!)

$$wp(v : \text{swap}(j, f(j)); f\prime : \text{swap}(f^{-1}(j),j); j := j+1, P1) =$$

$$= ((P1[j + j+1])\{f + f\prime\}[v + v\prime])$$

where $f\prime = f[f^{-1}(j) + f(j), j + f(f^{-1}(j))]$

$$v\prime = v[j + v(f(j)), f(j) + v(j)]$$

$= 0 \leq j+1 < n$ and $A i: 0..n-1. v\prime(f\prime(i)) = V(F(i))$ and $A i: 0..n-1. f\prime(i) = i.$
The first term is implied by \( P_1 \) and \( j \neq n-1 \); we prove

a: \( v'(f'(i)) = V(F(i)) \), and

b: \( i > j \text{ or } f'(i) = i \)

from \( P_1 \) by cases on \( i \):

For \( f^{-1}(j) \neq i \neq j \):

a. \( v'(f'(i)) = (\text{def } f') v'(f(i)) = (\text{def } v') V(f(i)) = (\text{from } P_1:) V(F(i)) \),

b. \( f'(i) = (\text{def } f') f(i) = (\text{from } P_1:) i, \text{ if } i \leq j \).

For \( i = j \):

a. \( v'(f'(j)) = (\text{def } f') v'(j) = (\text{def } v') V(f(j)) = (\text{from } P_1:) V(F(j)) \),

b. \( f'(j) = (\text{def } f') j \).

For \( i = f^{-1}(j) \):

a. \( v'(f'(i)) = (\text{def } f') v'(f(j)) = (\text{def } v') V(j) = V(f(f^{-1}(j))) = (\text{from } P_1:) V(F(i)) \),

b. (from \( P_1 \)): \( A \ i: 0 \ldots j-1. f(j) = i, \text{ hence } i = f^{-1}(j) \geq j \). Now either \( i = f^{-1}(j) > j \), or \( i = f^{-1}(j) = j \) and \( f'(i) = f'(j) = i \).

Final program: the ghost variable eliminated

There is an additional invariant relation, which enables us to eliminate variable \( f \):

P2: \( A \ i: j \ldots n-1. f(i) = \text{first elt in the seq } F(i), F^2(i), F^3(i) \ldots \)

which is \( \geq j \).

Here follows the proof of the invariance of P2.

\[
\begin{align*}
wp(v : \text{swap}(j, f(j)); f : \text{swap}(f^{-1}(j), j); j := j+1, P2) &= \\
&= ((P2[j + j+1])[f + f'][v + v']) \\
&= \text{where } f' = f[f^{-1}(j) + f(j), j + f(f^{-1}(j)]) \\
&\quad \text{and } v' = v[j + v(f(j)), f(j) + v(j)] \\
&= A \ i: j+1 \ldots n-1. f'(i) = \text{first elt in the seq } F(i), F^2(i), F^3(i) \ldots \)
\]

which is \( \geq j+1 \).

We prove the requirement for \( f'(i) \) from \( P2 \) by cases on \( i \).

For \( j+1 \leq i \leq n-1 \) and \( i \neq f^{-1}(j) \):

\( f'(i) = (\text{def } f') f(i) \) [and this is \( > j \) on account of \( P_0 \) and \( P_1 \)]

\[ = (\text{from } P2) \text{ the first elt in } F(i), F^2(i) \ldots \text{ which is } \geq j, \]

so \( f'(i) = \text{the first elt in the seq } F(i), F^2(i) \ldots \text{ which is } \geq j+1 \).

For \( j+1 \leq i \leq n-1 \) and \( i = f^{-1}(j) \):
f'(i) = (def f') f(j) \{\text{and this is } j \text{ on account of P0, P1 and } f^{-1}(j) \neq j\}
= (from P2:) the first elt in F(j), F^2(j) ... which is \geq j,
so f'(i) = the first elt in the seq F(j), F^2(j) ... which is \geq j+1 \quad \ldots \quad (\ast)

Also \quad j = f(f^{-1}(j)) = f(i)
= (from P2:) the first elt in F(i), F^2(i) ... which is \geq j \quad \ldots \quad (\ast\ast)

Combining (\ast) and (\ast\ast) yields
f'(i) = the first elt in the seq F(i), F^2(i) ... which is \geq j+1.

This, by the way, is the most non-trivial step of all verifications.

Hence, just before \quad v:swap(j, f(j)) \text{ we may compute } f(j) \text{ as follows:}
q := F(j); \text{ do } q < j \rightarrow q := F(q) \text{ od } \{q = f(j)\}.

The invariant relation of the repetition reads:
\quad f(j) = \text{the first elt in the seq } q, F(q), F^2(q) ... \text{ which is } \geq j.

The verification is easy, and is left to the reader. In addition P0, P1, P2, and j \neq n are invariant as well, because they do not contain q.

Once the above line has been inserted, and \quad v:swap(j, f(j)) \text{ has been replaced by } v:swap(j, q), \text{ it appears that } f \text{ is not used at all --}
except in updateings of itself -- and may therefore be deleted. So we have
proved the correctness of the given program.

In conclusion. The ghost variable f has enabled us to split the program
construction and the invariant relation in two easily comprehensible and
separately verifiable parts. The preliminary mathematical properties proved
by (Duijvestijn 72) have, more or less, been verified during the straightforward
and, indeed, rather boring verification of the invariants. Thus the only
interesting feature of the correctness proof is the formulation of an elegant
invariant.

References
Duijvestijn, A.J.W.: Correctness proof of an in-place permutation, BIT 12 (1972),
318-324.
CORRECTNESS PROOF OF AN IN-PLACE PERMUTATION

A. J. W. DUIJVESTIJN

Abstract.

The correctness of an in-place permutation algorithm is proved. The algorithm exchanges elements belonging to a permutation cycle. A suitable assertion is constructed from which the correctness can be deduced after completion of the algorithm.

An in-place rectangular matrix transposition algorithm is given as an example.

Key words and phrases: Proof of programs, algorithm, program correctness, theory of programming.

Introduction.

The in-place permutation problem deals with the rearrangement of the elements of a given vector $VEC[i]$, $i = 1(1)G$, $G \geq 1$, using an arbitrary permutation $f(i)$ of the integers $1, \ldots, G$.

The problem that has to be solved is: write an algorithm that permutes the elements of $VEC$ without using extra storage. That means if $VEC[i] = \alpha_i$ before the permutation then $VEC[i] = \alpha_{f(i)}$ after the permutation.

The solution of the permutation problem is given by the following algorithm:

procedure permute ($VEC, f, G$); value $G$; integer $G$; array $VEC$;
integer procedure $f$;
comment $f(x)$ is the index of $VEC$ where the element can be found that has to be moved to $VEC[x]$;
begin integer $k$, $ko$, $kn$, $wr$;
for $k := 1$ step 1 until $G$ do
begin
$kn := f(k)$;
for $ko := kn$ while $kn < k$ do $kn := f(ko)$;
if $kn = k$ then begin comment exchange ($VEC[kn]$, $VEC[k]$);

$wr := VEC[kn]$;
$VEC[kn] := VEC[k]$;
$VEC[k] := wr$
end
end

end

A special case of the permutation problem arises in the transposition of a rectangular matrix without using extra storage [2, 3]. In case the matrix $A[i,j]$, $i = 1(1)m$ and $j = 1(1)n$ is columnwise mapped onto a vector $VEC[k]$, $k = 1(1)m*n$, $G = m*n$, the function $f$ is defined as follows in ALGOL 60:

integer procedure $f(x)$; value $x$; integer $x$;
comment $f(x)$ is the index of $VEC$ where the element can be found that has to be moved to $VEC[x]$;
begin integer $w$;
$w := (x-1)+n$;
$f := (x-w+n-1)*m + w + 1$
end

The algorithm for which a correctness proof is given in this note is essentially that of R. F. Windley [1].

Correctness of the algorithm.

It has to be proved that the algorithm performs the following:

$\forall i (1 \leq i \leq G \rightarrow VEC[i] = \alpha_{f(i)})$.

First we introduce a function $\psi_k(i)$ that is defined for $k \leq i \leq G$ with $1 \leq k \leq G$:

$\psi_k(i) = \text{the first } f^{(t)}(i) \text{ with } f^{(t)}(i) \geq k, s \geq 1$.

The expression $f^s$ means: $f$ if $s = 1$, otherwise $f^{(s-1)}$. Consequently $\psi_k(i) = f^{(s)}(i) \geq k$, and $f^{(t)}(i) < k$ with $1 \leq t < s, s \geq 1$.

We prove certain properties of the function $\psi$.

PROPERTY 1 is a property of the permutation $f$:

$\forall i (1 \leq i \leq G \rightarrow \exists e(1 \leq e \leq G \land i = f(e)))$.

and

$\forall i (1 \leq i \leq G \rightarrow \exists e(2 \leq e \leq G \land e2 = f(i)))$.
Property 2.

(2) \( \forall i (k \leq i \leq G \rightarrow \exists e_l (k \leq e_l \leq G \land i = \psi_k(e_l))) \) and

(3) \( \forall i (k \leq i \leq G \rightarrow \exists e_2 (k \leq e_2 \leq G \land e_2 = \psi_k(i))) \).

Proof. Let \( V_{k,0} \) be the set of integers: \( V_{k,0} = \{ i : k \leq i \leq G \} \), then property 2 says that \( \psi_k(i) \) is a permutation on \( V_{k,0} \).

Apparently property 2 is true for \( k = 1 \) since \( \psi_1(i) = f(i) \) (property 1).

Assuming property 2 is true for \( k \) (induction assumption), we prove that property 2 is also true for \( k + 1 \).

According to the induction assumption there exists exactly one element \( e_1 \in V_{k,0} \) such that \( k = \psi_k(e_1) \) and exactly one element \( e_2 \in V_{k,0} \) such that \( e_2 = \psi_k(k) \). (A direct consequence of (2) and (3)).

We consider two cases:

Case 1. \( e_1 < k \). Then clearly \( e_2 > k \). Consider the sets \( V^{**}_{k+1,0} = V_{k+1,0} \setminus e_1 \) and \( V^*_{k+1,0} = V_{k+1,0} \setminus e_2 \).

According to the induction assumption we have:

(4) \( \forall a (a \in V^{**}_{k+1,0} \rightarrow \exists b (b \in V^{**}_{k+1,0} \land b = \psi_{k+1}(a))) \)

and

(5) \( \forall b (b \in V^{**}_{k+1,0} \rightarrow \exists a (a \in V^{**}_{k+1,0} \land b = \psi_{k+1}(a))) \)

Since \( b = \psi_k(a) > k \) it follows from the definition of \( \psi \):

\( b = f^t(a), s \geq 1 \) and \( f^s(a) < k \) for \( 1 \leq t < s \).

That

\( b = f^t(a) \geq k + 1, s \geq 1 \) and \( f^s(a) < k < k + 1 \) for \( 1 \leq t < s \);

(6) we conclude \( b = \psi_{k+1}(a) \).

Hence it follows that:

(7) \( \forall a (a \in V^{**}_{k+1,0} \rightarrow \psi_k(a) = \psi_{k+1}(a)) \).

Furthermore we prove \( e_2 = \psi_{k+1}(e_1) \).

From the definition of \( \psi \) and the induction assumption it follows:

\( \exists s (s \geq 1 \land k = f^t(e_1) \land \forall l (1 \leq l < s \rightarrow f^l(e_1) < k)) \)

and

\( \exists r (r \geq 1 \land e_2 = f^r(k) \land \forall u (1 \leq u < r \rightarrow f^u(k) < k)) \).

Clearly \( e_2 = f^{s+r}(e_1) \geq k + 1, s + r \geq 2 \) and \( f^p(e_1) < k + 1 \) with \( 1 \leq p < s + r \).

Hence

(8) \( e_2 = \psi_{k+1}(e_1) \).

Using (4), (5), (6), (7) and (8) we conclude:

(9) \( \forall a (a \in V_{k+1,0} \rightarrow \exists b (b \in V_{k+1,0} \land b = \psi_{k+1}(a))) \)

and

(10) \( \forall b (b \in V_{k+1,0} \rightarrow \exists a (a \in V_{k+1,0} \land b = \psi_{k+1}(a))) \).

Case 2. \( e_1 = k \). In this case \( e_1 = e_2 = k \). Furthermore \( V^*_{k+1,0} = V^**_{k+1,0} = V_{k+1,0} \) and according to (4), (5), (6) and (7) we have:

(11) \( \forall a (a \in V_{k+1,0} \rightarrow \exists b (b \in V_{k+1,0} \land b = \psi_{k+1}(a))) \)

and

(12) \( \forall b (b \in V_{k+1,0} \rightarrow \exists a (a \in V_{k+1,0} \land b = \psi_{k+1}(a))) \).

Using (9), (10), (11) and (12) then by induction property 2 is true for all \( k \leq G \).

We can now formulate property 3 and 4.

Property 3. If \( \psi_k(e_1) = k \) and \( \psi_k(k) = e_2 \), while \( e_1 > k \) and \( e_2 > k \) then according to (8) \( e_2 = \psi_{k+1}(e_1) \).

Remark. In case \( e_1 = e_2 = k \), \( \psi_{k+1}(e_1) \) is not defined.

Property 4. \( \psi_k(i) = \psi_{k+1}(i) \) for all \( i > k \) except that \( i \) for which \( \psi_k(i) = k \) (see (6) and (7)).

We prove the truth of the assertion \( E1 \land E2 \) on a certain label in the program. The definition of \( E1 \) and \( E2 \) is as follows:

(\( E1 \)) \( \forall i (1 \leq i < k \rightarrow VEC[i] = \alpha_{f(0)} \))

and

(\( E2 \)) \( \forall i (k \leq i \leq G \rightarrow VEC[\psi_k(i)] = \alpha_{f(0)} \)).

The structure of the program is:

for \( k := 1 \) step 1 until \( G \) do
begin ... end;

This program is equivalent with the program:

\( k := 1 \);

\( L: \) if \( k > G \) then goto \( \text{Ezh} \);
begin ... end;
\( k := k + 1 \); goto \( L \); \( \text{Ezh} \);
We prove $\vdash E_1 \land E_2$ on label $L$ for all $k, 1 \leq k \leq G + 1$.

**Proof.** If $k = 1$ then $\vdash E_1 \land E_2$ since $E_1$ is true (1 ≤ $i$ < 1 is false so the implication is true) and since $\psi_{\lambda}(i) = f(i)$ the assertion $E_2$ reads:

$$\forall i(1 \leq i \leq G \rightarrow VEC[\psi_{\lambda}(i)] = VEC[f(i)] = \alpha_{f(i)}$$

which is clearly true.

Assuming that $\vdash E_1 \land E_2$ on $L$ for a certain $k = k_1$ (1 ≤ $k_1$ ≤ $G$) the following statements are executed before returning to label $L$.

$L:$ $kn = f(k);$

for $ko := kn$ while $kn < k$ do $kn := f(ko);$  
$L1:$ if $kn \neq k$ then exchange $(VEC[kn], VEC[k]);$

$L2:$ $k := k + 1$; goto $L$;

The labels $L1$ and $L2$ are merely introduced as a reference. At label $L1$ we have $kn = \psi_{\lambda}(k)$. Consequently $kn \geq k$. In case $kn \neq k$, $VEC[kn]$ and $1 E[(k)]$ are exchanged. Since $\vdash E_1 \land E_2$ on $L$ it follows $\vdash E_1 \land E_2$ on $L1$.

We consider two cases:

**Case 1.** $kn > k$. From $\vdash E_1 \land E_2$ on $L1$ we have $VEC[\psi_{\lambda}(k)] = VEC[kn] = \alpha_{f(k)}$. After exchanging $VEC[kn]$ and $VEC[k]$, $VEC[k] = \alpha_{f(k)}$ at label $L2$.

Therefore the following assertion holds at $L2$:

$$\forall i(1 \leq i \leq k \rightarrow VEC[i] = \alpha_{f(k)}).$$

Hence

$$\forall i(1 \leq i < k + 1 \rightarrow VEC[i] = \alpha_{f(k)}).$$

Finally $\vdash E_1$ at $L$ for $k = k_1 + 1$.

Since $kn = \psi_{\lambda}(k) > k$ then according to property 2 there exist elements $e_1$ and $e_2$, $e_1 > k$, $e_2 > k$ such that:

$$e_2 = \psi_{\lambda}(k) \text{ and } k = \psi_{\lambda}(e_1)$$

and according to property 3:

$$e_2 = \psi_{\lambda}(e_1).$$

Applying $e_2 = kn$.

At label $L1$ we have

$$VEC[k] = VEC[\psi_{\lambda}(e_1)] = \alpha_{f(e_1)}, \text{ since } e_1 > k.$$ 

At label $L2$

$$VEC[kn] = VEC[\psi_{\lambda}(k)] = VEC[e_2] = \alpha_{f(e_2)}.$$
REMARK 2. Looking at the invariant $\vdash E_1 \land E_2$ we observe that $E_2$ describes the initial state of the program for $k = 1$. $E_1$ is then "empty". $E_1$ describes the final state for $k = O + 1$. $E_2$ is then "empty".

Acknowledgements.

The author wishes to thank F. Göbel and J. Englert for their discussions and remarks.

REFERENCES