Array Assignment

A is an array of n integers A[1], A[2], ..., A[n].

if x = y, ??? should be 1 = 0.

A[x] = 1;  if x ≠ y, ??? should be A[y] = 0.

A[1] = 0 D

When we use variables as indices into arrays, we need to account for multiple cases for many possible values that the variables can take.

Solutions: Write down the sequence of changes first and resolve them when we need to prove any implied conditions.


\( e_2 \) \( A[e_1] = e_2 \);  
\( e_3 \) \( D \) array assignment.

For an assignment to an array value \( A[e_1] = e_2 \), assume that the assignment produced a new array \( A' \), where:

- **Input:** array \( A \), index \( e_1 \), value \( e_2 \)
- **Output:** array \( A' \), except the \( e_1 \)th element is changed to have the value \( e_2 \).


We apply assignments from left to right.

004
Array assignment

Prove that the following program satisfies the given triple under partial correctness.

\[ (A[x] = \pi_0) \land (A[y] = \pi_0) \triangleright\]
\[ (A[y] = \pi_0) \land (A[y] = \pi_0) \triangleright\]

Assignment: 

\[ t = A[x] ; \]

Array assignment 

\[ A[x] = \text{AC}[y] ; \text{Q[A[x] A]} \]

Array assignment 

\[ A[y] = \pi_3 ; \text{Q[A[y] A]} \]

Array assignment 

\[ A[x] = \pi_0 \land (A[y] = \pi_0) \triangleright\]

To prove the implied condition, we need to prove the following:


Proof of 2: The first assignment \( x = A[y] \) does not matter because the second assignment changes the 4th element of \( A \) to \( A[x] \). This is what we want to show. QED

Proof of 0: Consider 2 cases:

1. \( x = y \). The second assignment can be rewritten as \( x = A[y] \), which is the same as the first assignment. Thus, the \( x \)-th element of \( A \) is \( A[y] \) after both assignments.
2. \( x \neq y \). The second assignment does not change the \( x \)-th element of \( A \). Therefore, the \( x \)-th element of \( A \) is \( A[y] \) after both assignments. QED
Array assignment

Prove that the following program satisfies the given triple under partial correctness.

\[ q \left( A[x] = x_0 \right) \land (A[y] = y_0) \]
\[ \downarrow \left( A[x] \leftarrow A[y] ; A[y] \leftarrow A[x], y \leftarrow A[y], x = x_0 \right) \]
\[ t = A[x] ; \]
\[ \downarrow \left( A[x] \leftarrow A[y] ; A[y] \leftarrow t \right) \land (A[y] = y_0) \]
\[ A[x] = A[y] ; \]
\[ \downarrow \left( A[y] \leftarrow t \right) \land (A[x] = x_0) \]
\[ A[y] = t ; \]
\[ \downarrow \left( A[x] = y_0 \right) \land (A[y] = x_0) \]

To prove the "implied" condition, we need to prove the following:
1. \( A[x] \leftarrow A[y] ; A[y] \leftarrow A[x], y \leftarrow A[y], x = x_0 \)
   - and

Proof of 1: The first assignment \( x \leftarrow A[y] \) assigns \( A[y] \) to the \( x \)th element of \( A \). Consider 2 cases for \( y \).
1. If \( y \neq x \), then the second assignment does not change the \( x \)th element of \( A \). Thus, the \( x \)th element of \( A \) is \( A[y] \) after the assignments.
2. If \( y = x \), the second assignment can be rewritten as \( x \leftarrow A[y] \), which is the same as the first assignment.
   Thus, the \( x \)th element of \( A \) is \( A[y] \) after the assignments.

Proof of 2: The first assignment does not matter. The second assignment assigns \( A[x] \) to the \( y \)th element of \( A \), and this is the desired result.