Names, \textit{l-values, r-values}

\begin{verbatim}
var I, J : integer;
    A : array[1..100] of integer; begin

    J := 3;  I := J + 1;
    A[I] := I;

    ...
\end{verbatim}

\begin{tabular}{|l|l|l|}
\hline
\textbf{name} & \textbf{l-value} & \textbf{r-value} \\
\hline
I & 05834726 & \\
J & 05834727 & \\
A ([1]) & 05834728 & \\
\hline
A ([4]) & 05834731 & \\
\hline
3 & 00000001 & \\
\hline
\end{tabular}
PASS BY VALUE

- One-way communication
- Provides input to the called procedure
- Protects the actual parameters from change

When a procedure is called:

1. An environment and a store are allocated for the called procedure, including the formal parameters and locally declared variables.
2. The \( r \)-values of the actual parameters are copied into locations specified by the \( l \)-values of the formal parameters.

When control returns from a called procedure:

1. The environment and store for the called procedure is deallocated.
PASS BY VALUE

program
  I : integer;
  A : array[1..100] of integer;

procedure VAL_SWAP(value X, Y : integer);
  var temp : integer;
begin
  temp := X; X := Y; Y := temp;
end;

begin  {main program}
  I := 3;
  A[I] := 6;
  write(’I =’, I);
  VAL_SWAP(I, A[I]);  \[→→→ temp:=X; X:=Y; Y:=temp;\]
  write(’I =’, I);
end.

OUTPUT

PASS BY RESULT

- One-way communication
- Provides output to the calling procedure
- Protects the formal parameters from initialization by the calling routine

When a procedure is called:

1. An environment and a store are allocated for the called procedure, including the formal parameters and locally declared variables.
2. The \( l \)-values of the actual parameters are saved, one for each of the formal parameters.

When control returns from a called procedure:

1. The \( r \)-values of the formal parameters are copied into locations specified by the saved \( l \)-values of the actual parameters.
2. The environment and store for the called procedure is deallocated.
PASS BY RESULT

program
  I : integer;
  A : array[1..100] of integer;

procedure RES_SWAP(result X, Y : integer);
  var temp : integer;
begin
  temp := X; X := Y; Y := temp;
end;

begin {main program}
  I := 3;
  A[I] := 6;
  write('I =', I);
  RES_SWAP(I, A[I]); → → → temp := X; X := Y; Y := temp;
  write('I =', I);
end.

OUTPUT

*** ERROR : UNDEFINED VARIABLE X
PASS BY VALUE-RESULT

- Two-way communication
- Provides both input to, and output from the called procedure
- No protection for formal or actual parameters

When a procedure is called:

1. An environment and a store are allocated for the called procedure, including the formal parameters and locally declared variables.
2. The \( l \)-values of the actual parameters are saved, one for each of the formal parameters.
3. The \( r \)-values of the actual parameters are copied into locations specified by the \( l \)-values of the formal parameters.

When control returns from a called procedure:

1. The \( r \)-values of the formal parameters are copied into locations specified by the saved \( l \)-values of the actual parameters.
2. The environment and store for the called procedure is deallocated.
PASS BY VALUE-RESULT

program
  I : integer;
  A : array[1..100] of integer;

procedure V_R_SWAP(val_res X, Y : integer);
  var temp : integer;
begin
  temp := X;  X := Y;  Y := temp;
end;

begin  {main program}
  I := 3;
  A[I] := 6;
  write(’I =’, I);
  V-R_SWAP(I, A[I]);  → → →  temp:=X;  X:=Y;  Y:=temp;
  write(’I =’, I);
end.

OUTPUT

**PASS BY REFERENCE**

- Two-way communication
- Provides both input to, and output from the called procedure
- No protection for formal or actual parameters
- Almost equivalent to value-result parameters; discrepancies show up in the presence of aliasing.

When a procedure is called:

1. An environment and a store are allocated for the called procedure, including the formal parameters and locally declared variables.
2. The \( l \)-values of the formal parameters are set equal to the \( l \)-values of the actual parameters.

When control returns from a called procedure:

1. The environment and store for the called procedure is deallocated.
PASS BY REFERENCE

program
  I : integer;
  A : array[1..100] of integer;

procedure REF_SWAP(var X, Y : integer);
  var temp : integer;
begin
  temp := X; X := Y; Y := temp;
end;

begin  {main program}
  I := 3;
  A[I] := 6;
  write(’I =’, I);
  REF_SWAP(I, A[I]); →→→ temp:=X; X:=Y; Y:=temp;
  write(’I =’, I);
end.

OUTPUT

PASS BY NAME

- One or two-way communication. If an actual parameter has no l-value, then assignment to the corresponding formal parameter is disallowed.
- Delayed evaluation of actual parameters – by need only.
- No protection for formal or actual parameters

When a procedure is called:

1. An environment and a store are allocated for the called procedure, including the locally declared variables.
2. If necessary, local variables are given new names not appearing elsewhere in the program.
2. The literal text of the actual parameters replaces each occurrence of the corresponding formal parameters. Variables mentioned in these expressions are global to the called procedure.

When control returns from a called procedure:

1. The environment and store for the called procedure is deallocated.
PASS BY NAME

program
    I : integer;
    A : array[1..100] of integer;

procedure NAM_SWAP(name X, Y : integer);
    var temp : integer;
begin
    temp := X;  X := Y;  Y := temp;
end;

begin  {main program}
    I := 3;
    A[I] := 6;
    write(’I =’, I);
    write(’I =’, I);
end.

OUTPUT