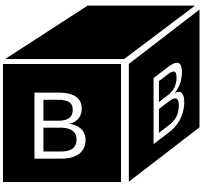


Chapter *12*

Proper Procedures

- Push the parameters.
- Push the return address.
- Push storage for the local variables.

Allocation for a proper procedure

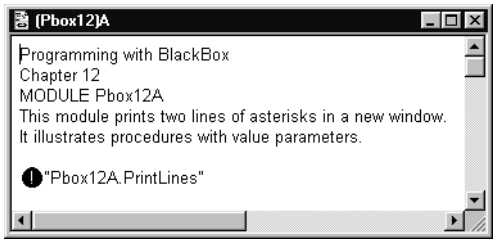


Figure 12.1
The output for the procedure of Figure 12.2.

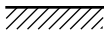
```
MODULE Pbox12A;
    IMPORT TextModels, TextViews, Views, PboxMappers;

    PROCEDURE PrintLine (n: INTEGER; IN f: PboxMappers.Formatter);
    (* Inserts a line of n asterisks to a text model with formatter f *)
        VAR
            i: INTEGER;
    BEGIN
        FOR i := 1 TO n DO
            f.WriteChar("*")
        END;
        f.WriteLn
    END PrintLine;
```

Figure 12.2
A procedure that prints a row of asterisks. The parameter specifies the number of asterisks in the row.

```
    PROCEDURE PrintLines*;
        VAR
            md: TextModels.Model;
            vw: TextViews.View;
            fm: PboxMappers.Formatter;
    BEGIN
        md := TextModels.dir.New();
        fm.ConnectTo(md);
        PrintLine(6, fm);
        (* ra1 *)
        PrintLine(5, fm);
        (* ra2 *)
        vw := TextViews.dir.New(md);
        Views.OpenView(vw)
    END PrintLines;

END Pbox12A.
```

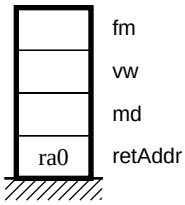
```
PROCEDURE PrintLine (n: INTEGER; IN f: PboxMappers.Formatter);
    VAR
        i: INTEGER;
BEGIN
    FOR i := 1 TO n DO
        f.WriteChar("*")
    END;
    f.WriteLn
END PrintLine;

PROCEDURE PrintLines*;
    VAR
        md: ...; vw: ...; fm: ... ;
BEGIN
    md := TextModels.dir.New();
    fm.ConnectTo(md);
    PrintLine(6, fm);
    (* ra1 *)
    PrintLine(5, fm);
    (* ra2 *)
    vw := TextViews.dir.New(md);
    Views.OpenView(vw)
END PrintLines;
```

Run-time stack **Text model**



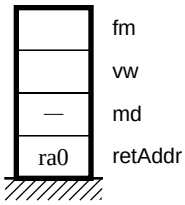
```
PROCEDURE PrintLine (n: INTEGER; IN f: PboxMappers.Formatter);
    VAR
        i: INTEGER;
BEGIN
    FOR i := 1 TO n DO
        f.WriteChar("*")
    END;
    f.WriteLn
END PrintLine;

PROCEDURE PrintLines*;
    VAR
        md: ...; vw: ...; fm: ... ;
BEGIN
    md := TextModels.dir.New();
    fm.ConnectTo(md);
    PrintLine(6, fm);
    (* ra1 *)
    PrintLine(5, fm);
    (* ra2 *)
    vw := TextViews.dir.New(md);
    Views.OpenView(vw)
END PrintLines;
```

Run-time stack **Text model**



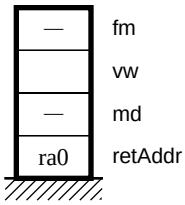
```
PROCEDURE PrintLine (n: INTEGER; IN f: PboxMappers.Formatter);
    VAR
        i: INTEGER;
BEGIN
    FOR i := 1 TO n DO
        f.WriteChar("*")
    END;
    f.WriteLn
END PrintLine;

PROCEDURE PrintLines*;
    VAR
        md: ...; vw: ...; fm: ... ;
BEGIN
    md := TextModels.dir.New();
    fm.ConnectTo(md);
    PrintLine(6, fm);
    (* ra1 *)
    PrintLine(5, fm);
    (* ra2 *)
    vw := TextViews.dir.New(md);
    Views.OpenView(vw)
END PrintLines;
```

Run-time stack **Text model**



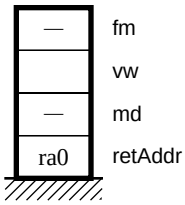
```
PROCEDURE PrintLine (n: INTEGER; IN f: PboxMappers.Formatter);
    VAR
        i: INTEGER;
BEGIN
    FOR i := 1 TO n DO
        f.WriteChar("*")
    END;
    f.WriteLn
END PrintLine;

PROCEDURE PrintLines*;
    VAR
        md: ...; vw: ...; fm: ... ;
BEGIN
    md := TextModels.dir.New();
    fm.ConnectTo(md);
    PrintLine(6, fm);
    (* ra1 *)
    PrintLine(5, fm);
    (* ra2 *)
    vw := TextViews.dir.New(md);
    Views.OpenView(vw)
END PrintLines;
```

Run-time stack **Text model**



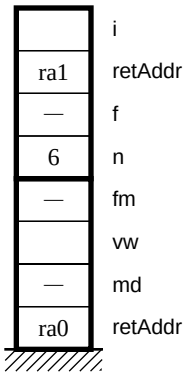
```
PROCEDURE PrintLine (n: INTEGER; IN f: PboxMappers.Formatter);
    VAR
        i: INTEGER;
BEGIN
    FOR i := 1 TO n DO
        f.WriteChar("*")
    END;
    f.WriteLn
END PrintLine;

PROCEDURE PrintLines*;
    VAR
        md: ...; vw: ...; fm: ... ;
BEGIN
    md := TextModels.dir.New();
    fm.ConnectTo(md);
    PrintLine(6, fm);
    (* ra1 *)
    PrintLine(5, fm);
    (* ra2 *)
    vw := TextViews.dir.New(md);
    Views.OpenView(vw)
END PrintLines;
```

Run-time stack **Text model**



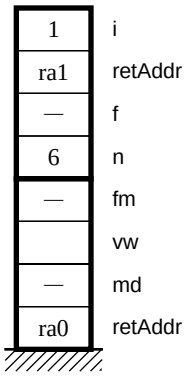
```
PROCEDURE PrintLine (n: INTEGER; IN f: PboxMappers.Formatter);
    VAR
        i: INTEGER;
BEGIN
    FOR i := 1 TO n DO
        f.WriteChar("*")
    END;
    f.WriteLn
END PrintLine;

PROCEDURE PrintLines*;
    VAR
        md: ...; vw: ...; fm: ... ;
BEGIN
    md := TextModels.dir.New();
    fm.ConnectTo(md);
    PrintLine(6, fm);
    (* ra1 *)
    PrintLine(5, fm);
    (* ra2 *)
    vw := TextViews.dir.New(md);
    Views.OpenView(vw)
END PrintLines;
```

Run-time stack **Text model**



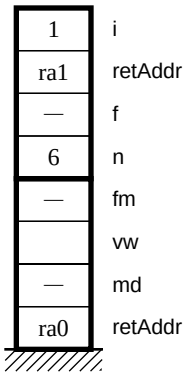
```
PROCEDURE PrintLine (n: INTEGER; IN f: PboxMappers.Formatter);
    VAR
        i: INTEGER;
BEGIN
    FOR i := 1 TO n DO
        f.WriteChar("*")
    END;
    f.WriteLn
END PrintLine;

PROCEDURE PrintLines*;
    VAR
        md: ...; vw: ...; fm: ... ;
BEGIN
    md := TextModels.dir.New();
    fm.ConnectTo(md);
    PrintLine(6, fm);
    (* ra1 *)
    PrintLine(5, fm);
    (* ra2 *)
    vw := TextViews.dir.New(md);
    Views.OpenView(vw)
END PrintLines;
```

Run-time stack **Text model**



```
PROCEDURE PrintLine (n: INTEGER; IN f: PboxMappers.Formatter);
    VAR
        i: INTEGER;
BEGIN
    FOR i := 1 TO n DO
        f.WriteChar("*")
    END;
    f.WriteLn
END PrintLine;

PROCEDURE PrintLines*;
    VAR
        md: ...; vw: ...; fm: ... ;
BEGIN
    md := TextModels.dir.New();
    fm.ConnectTo(md);
    PrintLine(6, fm);
    (* ra1 *)
    PrintLine(5, fm);
    (* ra2 *)
    vw := TextViews.dir.New(md);
    Views.OpenView(vw)
END PrintLines;
```

Run-time stack

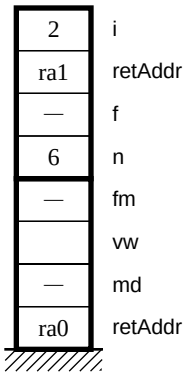


Text model

*

```
PROCEDURE PrintLine (n: INTEGER; IN f: PboxMappers.Formatter);
    VAR
        i: INTEGER;
BEGIN
    FOR i := 1 TO n DO
        f.WriteChar("*")
    END;
    f.WriteLn
END PrintLine;

PROCEDURE PrintLines*;
    VAR
        md: ...; vw: ...; fm: ... ;
BEGIN
    md := TextModels.dir.New();
    fm.ConnectTo(md);
    PrintLine(6, fm);
    (* ra1 *)
    PrintLine(5, fm);
    (* ra2 *)
    vw := TextViews.dir.New(md);
    Views.OpenView(vw)
END PrintLines;
```

Run-time stack

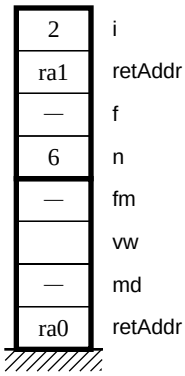


Text model

*

```
PROCEDURE PrintLine (n: INTEGER; IN f: PboxMappers.Formatter);
    VAR
        i: INTEGER;
BEGIN
    FOR i := 1 TO n DO
        f.WriteChar("*")
    END;
    f.WriteLn
END PrintLine;

PROCEDURE PrintLines*;
    VAR
        md: ...; vw: ...; fm: ... ;
BEGIN
    md := TextModels.dir.New();
    fm.ConnectTo(md);
    PrintLine(6, fm);
    (* ra1 *)
    PrintLine(5, fm);
    (* ra2 *)
    vw := TextViews.dir.New(md);
    Views.OpenView(vw)
END PrintLines;
```

Run-time stack

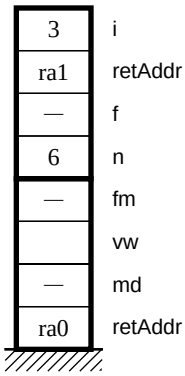


Text model

******

```
PROCEDURE PrintLine (n: INTEGER; IN f: PboxMappers.Formatter);
    VAR
        i: INTEGER;
BEGIN
    FOR i := 1 TO n DO
        f.WriteChar("*")
    END;
    f.WriteLn
END PrintLine;

PROCEDURE PrintLines*;
    VAR
        md: ...; vw: ...; fm: ... ;
BEGIN
    md := TextModels.dir.New();
    fm.ConnectTo(md);
    PrintLine(6, fm);
    (* ra1 *)
    PrintLine(5, fm);
    (* ra2 *)
    vw := TextViews.dir.New(md);
    Views.OpenView(vw)
END PrintLines;
```

Run-time stack

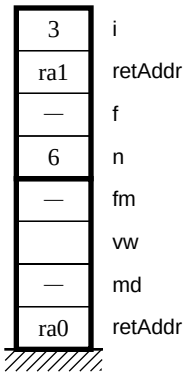


Text model

* *

```
PROCEDURE PrintLine (n: INTEGER; IN f: PboxMappers.Formatter);
    VAR
        i: INTEGER;
BEGIN
    FOR i := 1 TO n DO
        f.WriteChar("*")
    END;
    f.WriteLn
END PrintLine;

PROCEDURE PrintLines*;
    VAR
        md: ...; vw: ...; fm: ... ;
BEGIN
    md := TextModels.dir.New();
    fm.ConnectTo(md);
    PrintLine(6, fm);
    (* ra1 *)
    PrintLine(5, fm);
    (* ra2 *)
    vw := TextViews.dir.New(md);
    Views.OpenView(vw)
END PrintLines;
```

Run-time stack

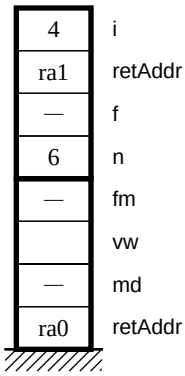


Text model

* * *

```
PROCEDURE PrintLine (n: INTEGER; IN f: PboxMappers.Formatter);
    VAR
        i: INTEGER;
BEGIN
    FOR i := 1 TO n DO
        f.WriteChar("*")
    END;
    f.WriteLn
END PrintLine;

PROCEDURE PrintLines*;
    VAR
        md: ...; vw: ...; fm: ... ;
BEGIN
    md := TextModels.dir.New();
    fm.ConnectTo(md);
    PrintLine(6, fm);
    (* ra1 *)
    PrintLine(5, fm);
    (* ra2 *)
    vw := TextViews.dir.New(md);
    Views.OpenView(vw)
END PrintLines;
```

Run-time stack

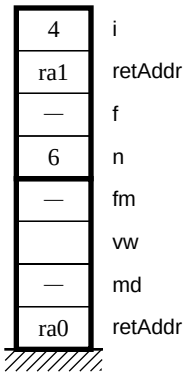


Text model

* * *

```
PROCEDURE PrintLine (n: INTEGER; IN f: PboxMappers.Formatter);
    VAR
        i: INTEGER;
BEGIN
    FOR i := 1 TO n DO
        f.WriteChar("*")
    END;
    f.WriteLn
END PrintLine;

PROCEDURE PrintLines*;
    VAR
        md: ...; vw: ...; fm: ... ;
BEGIN
    md := TextModels.dir.New();
    fm.ConnectTo(md);
    PrintLine(6, fm);
    (* ra1 *)
    PrintLine(5, fm);
    (* ra2 *)
    vw := TextViews.dir.New(md);
    Views.OpenView(vw)
END PrintLines;
```

Run-time stack

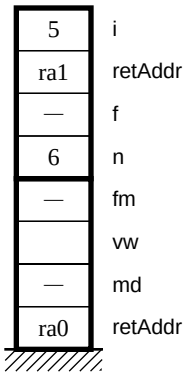


Text model

* * * *

```
PROCEDURE PrintLine (n: INTEGER; IN f: PboxMappers.Formatter);
    VAR
        i: INTEGER;
BEGIN
    FOR i := 1 TO n DO
        f.WriteChar("*")
    END;
    f.WriteLn
END PrintLine;

PROCEDURE PrintLines*;
    VAR
        md: ...; vw: ...; fm: ... ;
BEGIN
    md := TextModels.dir.New();
    fm.ConnectTo(md);
    PrintLine(6, fm);
    (* ra1 *)
    PrintLine(5, fm);
    (* ra2 *)
    vw := TextViews.dir.New(md);
    Views.OpenView(vw)
END PrintLines;
```

Run-time stack

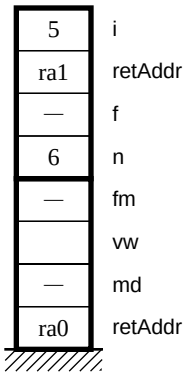


Text model

* * * *

```
PROCEDURE PrintLine (n: INTEGER; IN f: PboxMappers.Formatter);
    VAR
        i: INTEGER;
BEGIN
    FOR i := 1 TO n DO
        f.WriteChar("*")
    END;
    f.WriteLn
END PrintLine;

PROCEDURE PrintLines*;
    VAR
        md: ...; vw: ...; fm: ... ;
BEGIN
    md := TextModels.dir.New();
    fm.ConnectTo(md);
    PrintLine(6, fm);
    (* ra1 *)
    PrintLine(5, fm);
    (* ra2 *)
    vw := TextViews.dir.New(md);
    Views.OpenView(vw)
END PrintLines;
```

Run-time stack

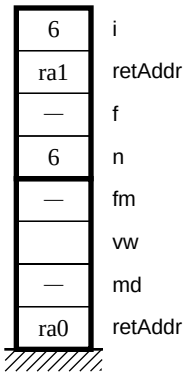


Text model

* * * * *

```
PROCEDURE PrintLine (n: INTEGER; IN f: PboxMappers.Formatter);
    VAR
        i: INTEGER;
BEGIN
    FOR i := 1 TO n DO
        f.WriteChar("*")
    END;
    f.WriteLn
END PrintLine;

PROCEDURE PrintLines*;
    VAR
        md: ...; vw: ...; fm: ... ;
BEGIN
    md := TextModels.dir.New();
    fm.ConnectTo(md);
    PrintLine(6, fm);
    (* ra1 *)
    PrintLine(5, fm);
    (* ra2 *)
    vw := TextViews.dir.New(md);
    Views.OpenView(vw)
END PrintLines;
```

Run-time stack

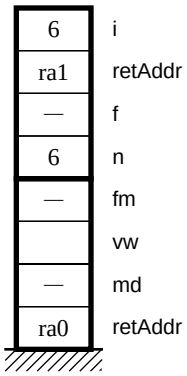


Text model

* * * * *

```
PROCEDURE PrintLine (n: INTEGER; IN f: PboxMappers.Formatter);
    VAR
        i: INTEGER;
BEGIN
    FOR i := 1 TO n DO
        f.WriteChar("*")
    END;
    f.WriteLn
END PrintLine;

PROCEDURE PrintLines*;
    VAR
        md: ...; vw: ...; fm: ... ;
BEGIN
    md := TextModels.dir.New();
    fm.ConnectTo(md);
    PrintLine(6, fm);
    (* ra1 *)
    PrintLine(5, fm);
    (* ra2 *)
    vw := TextViews.dir.New(md);
    Views.OpenView(vw)
END PrintLines;
```

Run-time stack

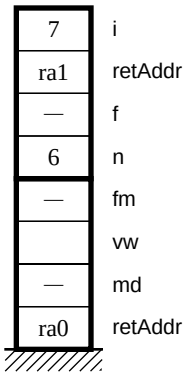


Text model

* * * * * *

```
PROCEDURE PrintLine (n: INTEGER; IN f: PboxMappers.Formatter);
    VAR
        i: INTEGER;
BEGIN
    FOR i := 1 TO n DO
        f.WriteChar("*")
    END;
    f.WriteLn
END PrintLine;

PROCEDURE PrintLines*;
    VAR
        md: ...; vw: ...; fm: ... ;
BEGIN
    md := TextModels.dir.New();
    fm.ConnectTo(md);
    PrintLine(6, fm);
    (* ra1 *)
    PrintLine(5, fm);
    (* ra2 *)
    vw := TextViews.dir.New(md);
    Views.OpenView(vw)
END PrintLines;
```

Run-time stack

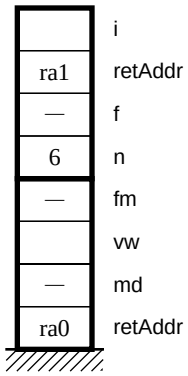


Text model

* * * * * *

```
PROCEDURE PrintLine (n: INTEGER; IN f: PboxMappers.Formatter);
    VAR
        i: INTEGER;
BEGIN
    FOR i := 1 TO n DO
        f.WriteChar("*")
    END;
    f.WriteLn
END PrintLine;

PROCEDURE PrintLines*;
    VAR
        md: ...; vw: ...; fm: ... ;
BEGIN
    md := TextModels.dir.New();
    fm.ConnectTo(md);
    PrintLine(6, fm);
    (* ra1 *)
    PrintLine(5, fm);
    (* ra2 *)
    vw := TextViews.dir.New(md);
    Views.OpenView(vw)
END PrintLines;
```

Run-time stack

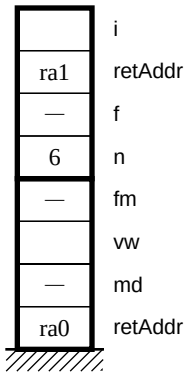


Text model

* * * * * *

```
PROCEDURE PrintLine (n: INTEGER; IN f: PboxMappers.Formatter);
    VAR
        i: INTEGER;
BEGIN
    FOR i := 1 TO n DO
        f.WriteChar("*")
    END;
    f.WriteLn
END PrintLine;

PROCEDURE PrintLines*;
    VAR
        md: ...; vw: ...; fm: ... ;
BEGIN
    md := TextModels.dir.New();
    fm.ConnectTo(md);
    PrintLine(6, fm);
    (* ra1 *)
    PrintLine(5, fm);
    (* ra2 *)
    vw := TextViews.dir.New(md);
    Views.OpenView(vw)
END PrintLines;
```

Run-time stack

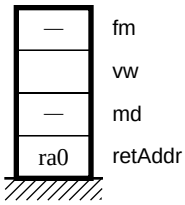


Text model

* * * * * *

```
PROCEDURE PrintLine (n: INTEGER; IN f: PboxMappers.Formatter);
    VAR
        i: INTEGER;
BEGIN
    FOR i := 1 TO n DO
        f.WriteChar("*")
    END;
    f.WriteLn
END PrintLine;

PROCEDURE PrintLines*;
    VAR
        md: ...; vw: ...; fm: ... ;
BEGIN
    md := TextModels.dir.New();
    fm.ConnectTo(md);
    PrintLine(6, fm);
    (* ra1 *)
    PrintLine(5, fm);
    (* ra2 *)
    vw := TextViews.dir.New(md);
    Views.OpenView(vw)
END PrintLines;
```

Run-time stack

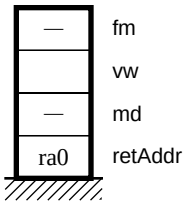


Text model

******

```
PROCEDURE PrintLine (n: INTEGER; IN f: PboxMappers.Formatter);
    VAR
        i: INTEGER;
BEGIN
    FOR i := 1 TO n DO
        f.WriteChar("*")
    END;
    f.WriteLn
END PrintLine;

PROCEDURE PrintLines*;
    VAR
        md: ...; vw: ...; fm: ... ;
BEGIN
    md := TextModels.dir.New();
    fm.ConnectTo(md);
    PrintLine(6, fm);
    (* ra1 *)
    PrintLine(5, fm);
    (* ra2 *)
    vw := TextViews.dir.New(md);
    Views.OpenView(vw)
END PrintLines;
```

Run-time stack

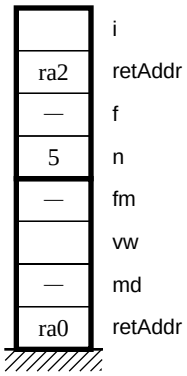


Text model

* * * * * *

```
PROCEDURE PrintLine (n: INTEGER; IN f: PboxMappers.Formatter);
    VAR
        i: INTEGER;
BEGIN
    FOR i := 1 TO n DO
        f.WriteChar("*")
    END;
    f.WriteLn
END PrintLine;

PROCEDURE PrintLines*;
    VAR
        md: ...; vw: ...; fm: ... ;
BEGIN
    md := TextModels.dir.New();
    fm.ConnectTo(md);
    PrintLine(6, fm);
    (* ra1 *)
    PrintLine(5, fm);
    (* ra2 *)
    vw := TextViews.dir.New(md);
    Views.OpenView(vw)
END PrintLines;
```

Run-time stack

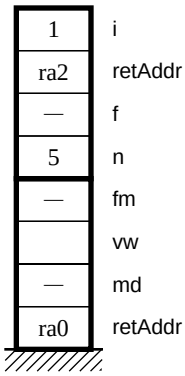


Text model

******

```
PROCEDURE PrintLine (n: INTEGER; IN f: PboxMappers.Formatter);
    VAR
        i: INTEGER;
BEGIN
    FOR i := 1 TO n DO
        f.WriteChar("*")
    END;
    f.WriteLn
END PrintLine;

PROCEDURE PrintLines*;
    VAR
        md: ...; vw: ...; fm: ... ;
BEGIN
    md := TextModels.dir.New();
    fm.ConnectTo(md);
    PrintLine(6, fm);
    (* ra1 *)
    PrintLine(5, fm);
    (* ra2 *)
    vw := TextViews.dir.New(md);
    Views.OpenView(vw)
END PrintLines;
```

Run-time stack

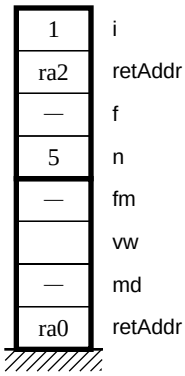


Text model

* * * * * *

```
PROCEDURE PrintLine (n: INTEGER; IN f: PboxMappers.Formatter);
    VAR
        i: INTEGER;
BEGIN
    FOR i := 1 TO n DO
        f.WriteChar("*")
    END;
    f.WriteLn
END PrintLine;

PROCEDURE PrintLines*;
    VAR
        md: ...; vw: ...; fm: ... ;
BEGIN
    md := TextModels.dir.New();
    fm.ConnectTo(md);
    PrintLine(6, fm);
    (* ra1 *)
    PrintLine(5, fm);
    (* ra2 *)
    vw := TextViews.dir.New(md);
    Views.OpenView(vw)
END PrintLines;
```

Run-time stack

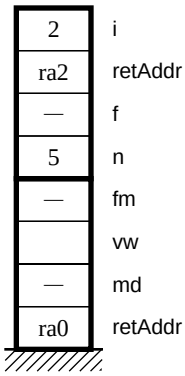


Text model

******
*

```
PROCEDURE PrintLine (n: INTEGER; IN f: PboxMappers.Formatter);
    VAR
        i: INTEGER;
BEGIN
    FOR i := 1 TO n DO
        f.WriteChar("*")
    END;
    f.WriteLn
END PrintLine;

PROCEDURE PrintLines*;
    VAR
        md: ...; vw: ...; fm: ... ;
BEGIN
    md := TextModels.dir.New();
    fm.ConnectTo(md);
    PrintLine(6, fm);
    (* ra1 *)
    PrintLine(5, fm);
    (* ra2 *)
    vw := TextViews.dir.New(md);
    Views.OpenView(vw)
END PrintLines;
```

Run-time stack

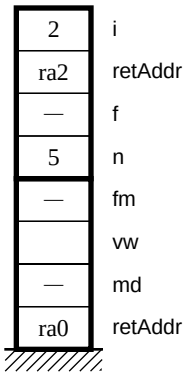


Text model

******
*

```
PROCEDURE PrintLine (n: INTEGER; IN f: PboxMappers.Formatter);
    VAR
        i: INTEGER;
BEGIN
    FOR i := 1 TO n DO
        f.WriteChar("*")
    END;
    f.WriteLn
END PrintLine;

PROCEDURE PrintLines*;
    VAR
        md: ...; vw: ...; fm: ... ;
BEGIN
    md := TextModels.dir.New();
    fm.ConnectTo(md);
    PrintLine(6, fm);
    (* ra1 *)
    PrintLine(5, fm);
    (* ra2 *)
    vw := TextViews.dir.New(md);
    Views.OpenView(vw)
END PrintLines;
```

Run-time stack

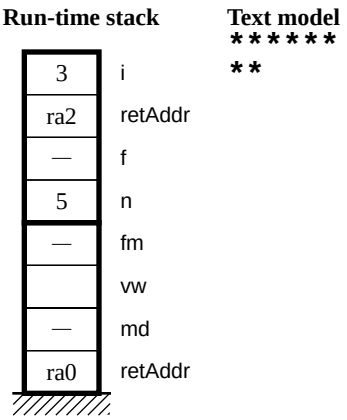


Text model

* * * * * *

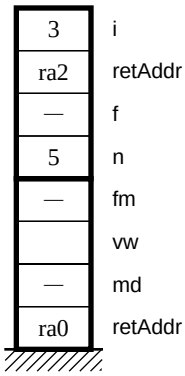
* *

```
PROCEDURE PrintLine (n: INTEGER; IN f: PboxMappers.Formatter);
    VAR
        i: INTEGER;
BEGIN
    FOR i := 1 TO n DO
        f.WriteChar("*")
    END;
    f.WriteLn
END PrintLine;

PROCEDURE PrintLines*;
    VAR
        md: ...; vw: ...; fm: ... ;
BEGIN
    md := TextModels.dir.New();
    fm.ConnectTo(md);
    PrintLine(6, fm);
    (* ra1 *)
    PrintLine(5, fm);
    (* ra2 *)
    vw := TextViews.dir.New(md);
    Views.OpenView(vw)
END PrintLines;
```



```
PROCEDURE PrintLine (n: INTEGER; IN f: PboxMappers.Formatter);
    VAR
        i: INTEGER;
BEGIN
    FOR i := 1 TO n DO
        f.WriteChar("*")
    END;
    f.WriteLn
END PrintLine;

PROCEDURE PrintLines*;
    VAR
        md: ...; vw: ...; fm: ... ;
BEGIN
    md := TextModels.dir.New();
    fm.ConnectTo(md);
    PrintLine(6, fm);
    (* ra1 *)
    PrintLine(5, fm);
    (* ra2 *)
    vw := TextViews.dir.New(md);
    Views.OpenView(vw)
END PrintLines;
```

Run-time stack

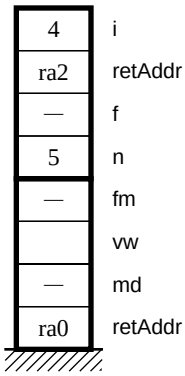


Text model

******
***

```
PROCEDURE PrintLine (n: INTEGER; IN f: PboxMappers.Formatter);
    VAR
        i: INTEGER;
BEGIN
    FOR i := 1 TO n DO
        f.WriteChar("*")
    END;
    f.WriteLn
END PrintLine;

PROCEDURE PrintLines*;
    VAR
        md: ...; vw: ...; fm: ... ;
BEGIN
    md := TextModels.dir.New();
    fm.ConnectTo(md);
    PrintLine(6, fm);
    (* ra1 *)
    PrintLine(5, fm);
    (* ra2 *)
    vw := TextViews.dir.New(md);
    Views.OpenView(vw)
END PrintLines;
```

Run-time stack

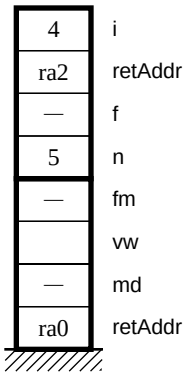


Text model

******
***

```
PROCEDURE PrintLine (n: INTEGER; IN f: PboxMappers.Formatter);
    VAR
        i: INTEGER;
BEGIN
    FOR i := 1 TO n DO
        f.WriteChar("*")
    END;
    f.WriteLn
END PrintLine;

PROCEDURE PrintLines*;
    VAR
        md: ...; vw: ...; fm: ... ;
BEGIN
    md := TextModels.dir.New();
    fm.ConnectTo(md);
    PrintLine(6, fm);
    (* ra1 *)
    PrintLine(5, fm);
    (* ra2 *)
    vw := TextViews.dir.New(md);
    Views.OpenView(vw)
END PrintLines;
```

Run-time stack

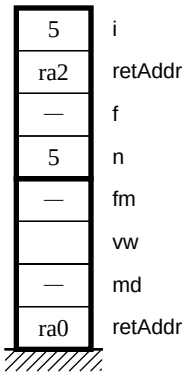


Text model

******
****

```
PROCEDURE PrintLine (n: INTEGER; IN f: PboxMappers.Formatter);
    VAR
        i: INTEGER;
BEGIN
    FOR i := 1 TO n DO
        f.WriteChar("*")
    END;
    f.WriteLn
END PrintLine;

PROCEDURE PrintLines*;
    VAR
        md: ...; vw: ...; fm: ... ;
BEGIN
    md := TextModels.dir.New();
    fm.ConnectTo(md);
    PrintLine(6, fm);
    (* ra1 *)
    PrintLine(5, fm);
    (* ra2 *)
    vw := TextViews.dir.New(md);
    Views.OpenView(vw)
END PrintLines;
```

Run-time stack

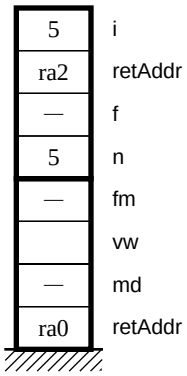


Text model

******
****

```
PROCEDURE PrintLine (n: INTEGER; IN f: PboxMappers.Formatter);
    VAR
        i: INTEGER;
BEGIN
    FOR i := 1 TO n DO
        f.WriteChar("*")
    END;
    f.WriteLn
END PrintLine;

PROCEDURE PrintLines*;
    VAR
        md: ...; vw: ...; fm: ... ;
BEGIN
    md := TextModels.dir.New();
    fm.ConnectTo(md);
    PrintLine(6, fm);
    (* ra1 *)
    PrintLine(5, fm);
    (* ra2 *)
    vw := TextViews.dir.New(md);
    Views.OpenView(vw)
END PrintLines;
```

Run-time stack

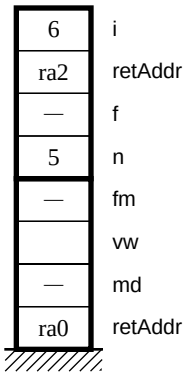


Text model

******
*****

```
PROCEDURE PrintLine (n: INTEGER; IN f: PboxMappers.Formatter);
    VAR
        i: INTEGER;
BEGIN
    FOR i := 1 TO n DO
        f.WriteChar("*")
    END;
    f.WriteLn
END PrintLine;

PROCEDURE PrintLines*;
    VAR
        md: ...; vw: ...; fm: ... ;
BEGIN
    md := TextModels.dir.New();
    fm.ConnectTo(md);
    PrintLine(6, fm);
    (* ra1 *)
    PrintLine(5, fm);
    (* ra2 *)
    vw := TextViews.dir.New(md);
    Views.OpenView(vw)
END PrintLines;
```

Run-time stack

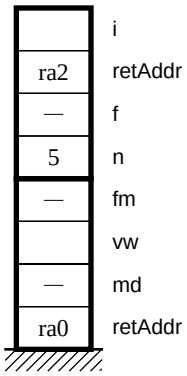


Text model

* * * * * *
* * * * *

```
PROCEDURE PrintLine (n: INTEGER; IN f: PboxMappers.Formatter);
    VAR
        i: INTEGER;
BEGIN
    FOR i := 1 TO n DO
        f.WriteChar("*")
    END;
    f.WriteLn
END PrintLine;

PROCEDURE PrintLines*;
    VAR
        md: ...; vw: ...; fm: ... ;
BEGIN
    md := TextModels.dir.New();
    fm.ConnectTo(md);
    PrintLine(6, fm);
    (* ra1 *)
    PrintLine(5, fm);
    (* ra2 *)
    vw := TextViews.dir.New(md);
    Views.OpenView(vw)
END PrintLines;
```

Run-time stack

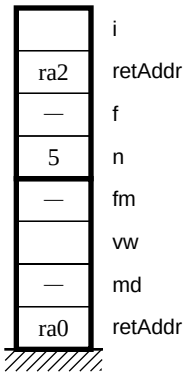


Text model

******
*****

```
PROCEDURE PrintLine (n: INTEGER; IN f: PboxMappers.Formatter);
    VAR
        i: INTEGER;
BEGIN
    FOR i := 1 TO n DO
        f.WriteChar("*")
    END;
    f.WriteLn
END PrintLine;

PROCEDURE PrintLines*;
    VAR
        md: ...; vw: ...; fm: ... ;
BEGIN
    md := TextModels.dir.New();
    fm.ConnectTo(md);
    PrintLine(6, fm);
    (* ra1 *)
    PrintLine(5, fm);
    (* ra2 *)
    vw := TextViews.dir.New(md);
    Views.OpenView(vw)
END PrintLines;
```

Run-time stack

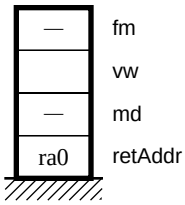


Text model

******
*****

```
PROCEDURE PrintLine (n: INTEGER; IN f: PboxMappers.Formatter);
    VAR
        i: INTEGER;
BEGIN
    FOR i := 1 TO n DO
        f.WriteChar("*")
    END;
    f.WriteLn
END PrintLine;

PROCEDURE PrintLines*;
    VAR
        md: ...; vw: ...; fm: ... ;
BEGIN
    md := TextModels.dir.New();
    fm.ConnectTo(md);
    PrintLine(6, fm);
    (* ra1 *)
    PrintLine(5, fm);
    (* ra2 *)
    vw := TextViews.dir.New(md);
    Views.OpenView(vw)
END PrintLines;
```

Run-time stack

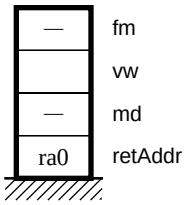


Text model

******
*****

```
PROCEDURE PrintLine (n: INTEGER; IN f: PboxMappers.Formatter);
    VAR
        i: INTEGER;
BEGIN
    FOR i := 1 TO n DO
        f.WriteChar("*")
    END;
    f.WriteLn
END PrintLine;

PROCEDURE PrintLines*;
    VAR
        md: ...; vw: ...; fm: ... ;
BEGIN
    md := TextModels.dir.New();
    fm.ConnectTo(md);
    PrintLine(6, fm);
    (* ra1 *)
    PrintLine(5, fm);
    (* ra2 *)
    vw := TextViews.dir.New(md);
    Views.OpenView(vw)
END PrintLines;
```

Run-time stack

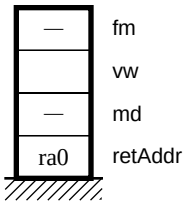


Text model

******
*****

```
PROCEDURE PrintLine (n: INTEGER; IN f: PboxMappers.Formatter);
    VAR
        i: INTEGER;
BEGIN
    FOR i := 1 TO n DO
        f.WriteChar("*")
    END;
    f.WriteLn
END PrintLine;

PROCEDURE PrintLines*;
    VAR
        md: ...; vw: ...; fm: ... ;
BEGIN
    md := TextModels.dir.New();
    fm.ConnectTo(md);
    PrintLine(6, fm);
    (* ra1 *)
    PrintLine(5, fm);
    (* ra2 *)
    vw := TextViews.dir.New(md);
    Views.OpenView(vw)
END PrintLines;
```

Run-time stack

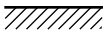


Text model

******
*****

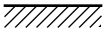
```
PROCEDURE PrintLine (n: INTEGER; IN f: PboxMappers.Formatter);
    VAR
        i: INTEGER;
BEGIN
    FOR i := 1 TO n DO
        f.WriteChar("*")
    END;
    f.WriteLn
END PrintLine;

PROCEDURE PrintLines*;
    VAR
        md: ...; vw: ...; fm: ... ;
BEGIN
    md := TextModels.dir.New();
    fm.ConnectTo(md);
    PrintLine(6, fm);
    (* ra1 *)
    PrintLine(5, fm);
    (* ra2 *)
    vw := TextViews.dir.New(md);
    Views.OpenView(vw)
END PrintLines;
```

Run-time stack

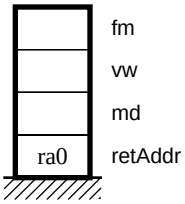


Text model

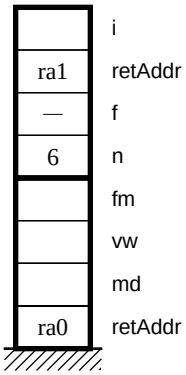
******
*****



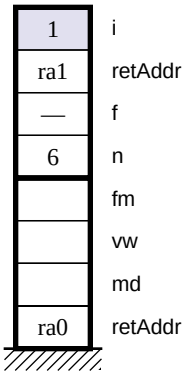
(a) User clicks button.



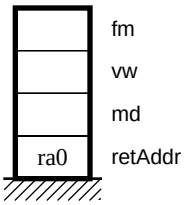
(b) Call PrintLines



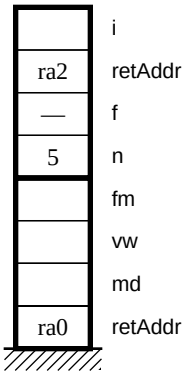
(c) Call PrintLine



(d) FOR i := 1 TO n DO

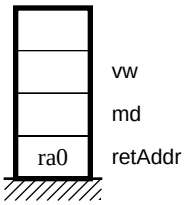


(e) Return to PrintLines

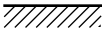


(f) Call PrintLine

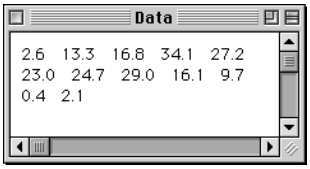
Figure 12.3
The run-time stack for module Pbox12A in Figure 12.2.



(g) Return to PrintLines



(h) Return to StdInterpreter.CallProc



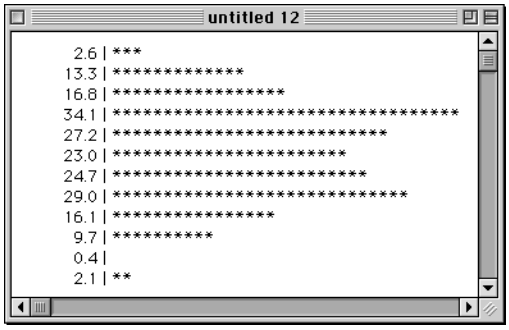


Figure 12.4
The input and output for the program of Listing 12.2.

```
MODULE Pbox12B;
    IMPORT TextModels, TextViews, Views, TextControllers, PboxMappers;

    PROCEDURE PrintLine (x: REAL; IN f: PboxMappers.Formatter);
        VAR
            i, n: INTEGER;
    BEGIN
        ASSERT(x >= 0.0, 20);
        f.WriteReal(x, 8, 1);
        f.WriteString(" | ");
        n := SHORT(ENTIER(x + 0.5));
        FOR i := 1 TO n DO
            f.WriteChar("*")
        END;
        f.WriteLn
    END PrintLine;
```

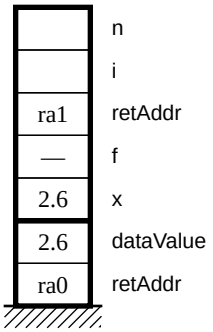
Figure 12.5
A program that prints a bar chart from data values in a file. The procedure prints a single bar.

```
    PROCEDURE PrintHistogram*;
        VAR
            cn: TextControllers.Controller;
            sc: PboxMappers.Scanner;
            dataValue: REAL;
            md: TextModels.Model;
            vw: TextViews.View;
            fm: PboxMappers.Formatter;
    BEGIN
        cn := TextControllers.Focus();
        IF cn # NIL THEN
            md := TextModels.dir.New();
            fm.ConnectTo(md);
            sc.ConnectTo(cn.text);
            sc.ScanReal(dataValue);
            WHILE ~sc.eot DO
                PrintLine(dataValue, fm);
                (* ra1 *)
                sc.ScanReal(dataValue)
            END;
            vw := TextViews.dir.New(md);
            Views.OpenView(vw)
        END
    END PrintHistogram;

END Pbox12B.
```

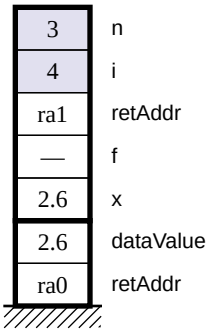
Figure 12.6
Memory allocation for Listing 12.5. Many MVC variables are not shown.

(a) After call to PrintHistogram

(b) Before call to PrintLine



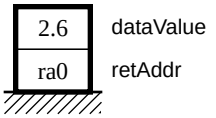
(c) After call to PrintLine



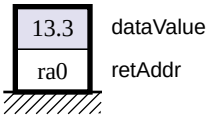
(d) Before return to PrintHistogram

Figure 12.6
Memory allocation for Listing 12.5. Many MVC variables are not shown.

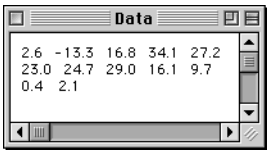
Figure 12.6
Memory allocation for Listing 12.5. Many MVC variables are not shown.



(e) After return to PrintHistogram



(f) Before call to PrintLine



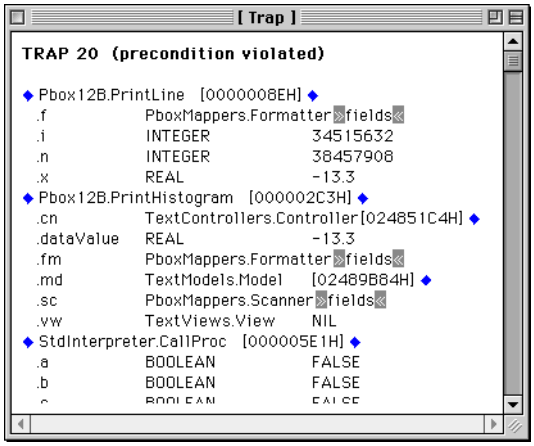


Figure 12.7
The trap generated by violation of the precondition of procedure PrintLine in Figure 12.5.

```
MODULE Pbox12C;
    IMPORT StdLog;

    PROCEDURE PassVal (j: INTEGER);
    BEGIN
        INC(j);
        StdLog.String("j = "); StdLog.Int(j); StdLog.Ln;
    END PassVal;

    PROCEDURE CallByValue*;
        VAR
            i: INTEGER;
    BEGIN
        i := 6;
        StdLog.String("i = "); StdLog.Int(i); StdLog.Ln;
        PassVal(i);
        (* ra1 *)
        StdLog.String("i = "); StdLog.Int(i); StdLog.Ln
    END CallByValue;

END Pbox12C.
```

Figure 12.8
A procedure with a parameter called by value.

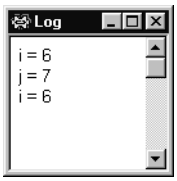
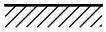


Figure 12.9
The output for Figure 12.8.

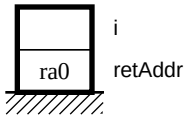
```
PROCEDURE PassVal (j: INTEGER);
BEGIN
    INC(j);
    StdLog.String("j = "); StdLog.Int(j); StdLog.Ln;
END PassVal;

PROCEDURE CallByValue*;
    VAR
        i: INTEGER;
BEGIN
    i := 6;
    StdLog.String("i = "); StdLog.Int(i); StdLog.Ln;
    PassVal(i);
    (* ra1 *)
    StdLog.String("i = "); StdLog.Int(i); StdLog.Ln
END CallByValue;
```

Run-time stack **Log**



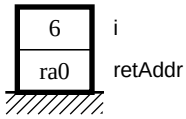
```
PROCEDURE PassVal (j: INTEGER);
BEGIN
    INC(j);
    StdLog.String("j = "); StdLog.Int(j); StdLog.Ln;
END PassVal;

PROCEDURE CallByValue*;
    VAR
        i: INTEGER;
BEGIN
    i := 6;
    StdLog.String("i = "); StdLog.Int(i); StdLog.Ln;
    PassVal(i);
    (* ra1 *)
    StdLog.String("i = "); StdLog.Int(i); StdLog.Ln
END CallByValue;
```

Run-time stack **Log**



```
PROCEDURE PassVal (j: INTEGER);
BEGIN
    INC(j);
    StdLog.String("j = "); StdLog.Int(j); StdLog.Ln;
END PassVal;

PROCEDURE CallByValue*;
    VAR
        i: INTEGER;
BEGIN
    i := 6;
    StdLog.String("i = "); StdLog.Int(i); StdLog.Ln;
    PassVal(i);
    (* ra1 *)
    StdLog.String("i = "); StdLog.Int(i); StdLog.Ln
END CallByValue;
```

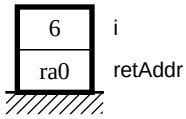
Run-time stack **Log**



```
PROCEDURE PassVal (j: INTEGER);
BEGIN
    INC(j);
    StdLog.String("j = "); StdLog.Int(j); StdLog.Ln;
END PassVal;

PROCEDURE CallByValue*;
    VAR
        i: INTEGER;
BEGIN
    i := 6;
    StdLog.String("i = "); StdLog.Int(i); StdLog.Ln;
    PassVal(i);
    (* ra1 *)
    StdLog.String("i = "); StdLog.Int(i); StdLog.Ln
END CallByValue;
```

Run-time stack **Log**

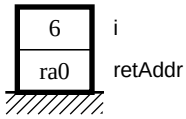
i = 6



```
PROCEDURE PassVal (j: INTEGER);
BEGIN
    INC(j);
    StdLog.String("j = "); StdLog.Int(j); StdLog.Ln;
END PassVal;

PROCEDURE CallByValue*;
    VAR
        i: INTEGER;
BEGIN
    i := 6;
    StdLog.String("i = "); StdLog.Int(i); StdLog.Ln;
    PassVal(i);
    (* ra1 *)
    StdLog.String("i = "); StdLog.Int(i); StdLog.Ln
END CallByValue;
```

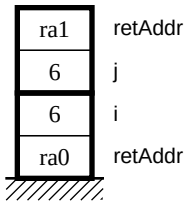
Run-time stack **Log**

i = 6



```
PROCEDURE PassVal (j: INTEGER);
BEGIN
    INC(j);
    StdLog.String("j = "); StdLog.Int(j); StdLog.Ln;
END PassVal;

PROCEDURE CallByValue*;
    VAR
        i: INTEGER;
BEGIN
    i := 6;
    StdLog.String("i = "); StdLog.Int(i); StdLog.Ln;
    PassVal(i);
    (* ra1 *)
    StdLog.String("i = "); StdLog.Int(i); StdLog.Ln
END CallByValue;
```

Run-time stack

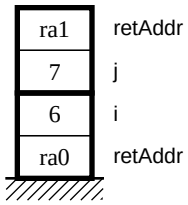


Log

```
i = 6
```

```
PROCEDURE PassVal (j: INTEGER);
BEGIN
    INC(j);
    StdLog.String("j = "); StdLog.Int(j); StdLog.Ln;
END PassVal;

PROCEDURE CallByValue*;
    VAR
        i: INTEGER;
BEGIN
    i := 6;
    StdLog.String("i = "); StdLog.Int(i); StdLog.Ln;
    PassVal(i);
    (* ra1 *)
    StdLog.String("i = "); StdLog.Int(i); StdLog.Ln
END CallByValue;
```

Run-time stack

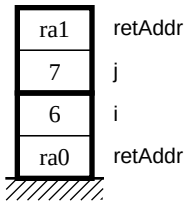


Log

i = 6

```
PROCEDURE PassVal (j: INTEGER);
BEGIN
    INC(j);
    StdLog.String("j = "); StdLog.Int(j); StdLog.Ln;
END PassVal;

PROCEDURE CallByValue*;
    VAR
        i: INTEGER;
BEGIN
    i := 6;
    StdLog.String("i = "); StdLog.Int(i); StdLog.Ln;
    PassVal(i);
    (* ra1 *)
    StdLog.String("i = "); StdLog.Int(i); StdLog.Ln
END CallByValue;
```

Run-time stack

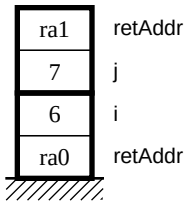


Log

```
i = 6
j = 7
```

```
PROCEDURE PassVal (j: INTEGER);
BEGIN
    INC(j);
    StdLog.String("j = "); StdLog.Int(j); StdLog.Ln;
END PassVal;

PROCEDURE CallByValue*;
    VAR
        i: INTEGER;
BEGIN
    i := 6;
    StdLog.String("i = "); StdLog.Int(i); StdLog.Ln;
    PassVal(i);
    (* ra1 *)
    StdLog.String("i = "); StdLog.Int(i); StdLog.Ln
END CallByValue;
```

Run-time stack

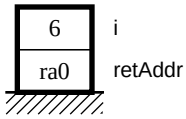


Log

```
i = 6
j = 7
```

```
PROCEDURE PassVal (j: INTEGER);
BEGIN
    INC(j);
    StdLog.String("j = "); StdLog.Int(j); StdLog.Ln;
END PassVal;

PROCEDURE CallByValue*;
    VAR
        i: INTEGER;
BEGIN
    i := 6;
    StdLog.String("i = "); StdLog.Int(i); StdLog.Ln;
    PassVal(i);
    (* ra1 *)
    StdLog.String("i = "); StdLog.Int(i); StdLog.Ln
END CallByValue;
```

Run-time stack

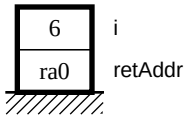


Log

```
i = 6
j = 7
```

```
PROCEDURE PassVal (j: INTEGER);
BEGIN
    INC(j);
    StdLog.String("j = "); StdLog.Int(j); StdLog.Ln;
END PassVal;

PROCEDURE CallByValue*;
    VAR
        i: INTEGER;
BEGIN
    i := 6;
    StdLog.String("i = "); StdLog.Int(i); StdLog.Ln;
    PassVal(i);
    (* ra1 *)
    StdLog.String("i = "); StdLog.Int(i); StdLog.Ln
END CallByValue;
```

Run-time stack

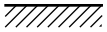


Log

```
i = 6
j = 7
i = 6
```

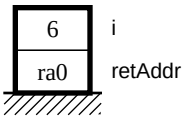
```
PROCEDURE PassVal (j: INTEGER);
BEGIN
    INC(j);
    StdLog.String("j = "); StdLog.Int(j); StdLog.Ln;
END PassVal;

PROCEDURE CallByValue*;
    VAR
        i: INTEGER;
BEGIN
    i := 6;
    StdLog.String("i = "); StdLog.Int(i); StdLog.Ln;
    PassVal(i);
    (* ra1 *)
    StdLog.String("i = "); StdLog.Int(i); StdLog.Ln
END CallByValue;
```

Run-time stack

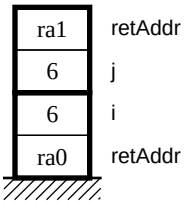


Log

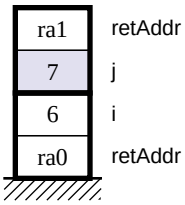
```
i = 6
j = 7
i = 6
```



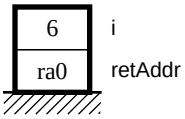
(a) Before call to PassVal



(b) After call to PassVal



(c) After INC(j)



(d) After return from PassVal

Figure 12.10
Memory allocation for Figure 12.8.

```
MODULE Pbox12D;
   IMPORT StdLog;

   PROCEDURE PassRef (VAR j: INTEGER);
   BEGIN
      INC(j);
      StdLog.String("j = "); StdLog.Int(j); StdLog.Ln;
   END PassRef;

   PROCEDURE CallByReference*;
      VAR
         i: INTEGER;
   BEGIN
      i := 6;
      StdLog.String("i = "); StdLog.Int(i); StdLog.Ln;
      PassRef(i);
      (* ra1 *)
      StdLog.String("i = "); StdLog.Int(i); StdLog.Ln
   END CallByReference;

END Pbox12D.
```

Figure 12.11
A procedure with a parameter called by reference.

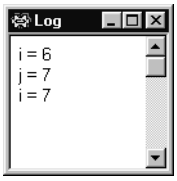
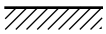


Figure 12.12
The output for Figure 12.11.

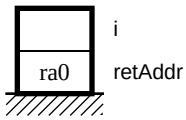
```
PROCEDURE PassRef (VAR j: INTEGER);
BEGIN
    INC(j);
    StdLog.String("j = "); StdLog.Int(j); StdLog.Ln;
END PassRef;

PROCEDURE CallByReference*;
    VAR
        i: INTEGER;
BEGIN
    i := 6;
    StdLog.String("i = "); StdLog.Int(i); StdLog.Ln;
    PassRef(i);
    (* ra1 *)
    StdLog.String("i = "); StdLog.Int(i); StdLog.Ln
END CallByReference;
```

Run-time stack **Log**



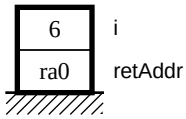
```
PROCEDURE PassRef (VAR j: INTEGER);
BEGIN
    INC(j);
    StdLog.String("j = "); StdLog.Int(j); StdLog.Ln;
END PassRef;

PROCEDURE CallByReference*;
    VAR
        i: INTEGER;
BEGIN
    i := 6;
    StdLog.String("i = "); StdLog.Int(i); StdLog.Ln;
    PassRef(i);
    (* ra1 *)
    StdLog.String("i = "); StdLog.Int(i); StdLog.Ln
END CallByReference;
```

Run-time stack **Log**



```
PROCEDURE PassRef (VAR j: INTEGER);
BEGIN
    INC(j);
    StdLog.String("j = "); StdLog.Int(j); StdLog.Ln;
END PassRef;

PROCEDURE CallByReference*;
    VAR
        i: INTEGER;
BEGIN
    i := 6;
    StdLog.String("i = "); StdLog.Int(i); StdLog.Ln;
    PassRef(i);
    (* ra1 *)
    StdLog.String("i = "); StdLog.Int(i); StdLog.Ln
END CallByReference;
```

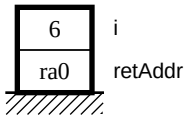
Run-time stack **Log**



```
PROCEDURE PassRef (VAR j: INTEGER);
BEGIN
    INC(j);
    StdLog.String("j = "); StdLog.Int(j); StdLog.Ln;
END PassRef;

PROCEDURE CallByReference*;
    VAR
        i: INTEGER;
BEGIN
    i := 6;
    StdLog.String("i = "); StdLog.Int(i); StdLog.Ln;
    PassRef(i);
    (* ra1 *)
    StdLog.String("i = "); StdLog.Int(i); StdLog.Ln
END CallByReference;
```

Run-time stack **Log**

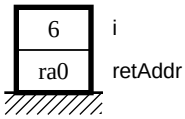
i = 6



```
PROCEDURE PassRef (VAR j: INTEGER);
BEGIN
    INC(j);
    StdLog.String("j = "); StdLog.Int(j); StdLog.Ln;
END PassRef;

PROCEDURE CallByReference*;
    VAR
        i: INTEGER;
BEGIN
    i := 6;
    StdLog.String("i = "); StdLog.Int(i); StdLog.Ln;
    PassRef(i);
    (* ra1 *)
    StdLog.String("i = "); StdLog.Int(i); StdLog.Ln
END CallByReference;
```

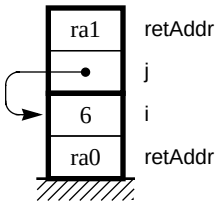
Run-time stack **Log**

i = 6



```
PROCEDURE PassRef (VAR j: INTEGER);
BEGIN
    INC(j);
    StdLog.String("j = "); StdLog.Int(j); StdLog.Ln;
END PassRef;

PROCEDURE CallByReference*;
    VAR
        i: INTEGER;
BEGIN
    i := 6;
    StdLog.String("i = "); StdLog.Int(i); StdLog.Ln;
    PassRef(i);
    (* ra1 *)
    StdLog.String("i = "); StdLog.Int(i); StdLog.Ln
END CallByReference;
```

Run-time stack

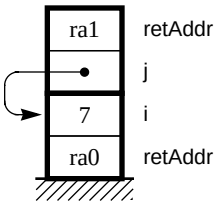


Log

```
i = 6
```

```
PROCEDURE PassRef (VAR j: INTEGER);
BEGIN
    INC(j);
    StdLog.String("j = "); StdLog.Int(j); StdLog.Ln;
END PassRef;

PROCEDURE CallByReference*;
    VAR
        i: INTEGER;
BEGIN
    i := 6;
    StdLog.String("i = "); StdLog.Int(i); StdLog.Ln;
    PassRef(i);
    (* ra1 *)
    StdLog.String("i = "); StdLog.Int(i); StdLog.Ln
END CallByReference;
```

Run-time stack

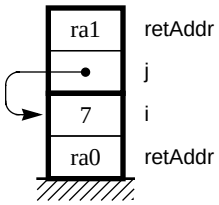


Log

```
i = 6
```

```
PROCEDURE PassRef (VAR j: INTEGER);
BEGIN
    INC(j);
    StdLog.String("j = "); StdLog.Int(j); StdLog.Ln;
END PassRef;

PROCEDURE CallByReference*;
    VAR
        i: INTEGER;
BEGIN
    i := 6;
    StdLog.String("i = "); StdLog.Int(i); StdLog.Ln;
    PassRef(i);
    (* ra1 *)
    StdLog.String("i = "); StdLog.Int(i); StdLog.Ln
END CallByReference;
```

Run-time stack

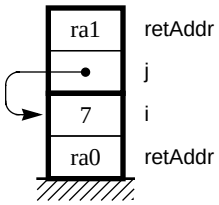


Log

```
i = 6
j = 7
```

```
PROCEDURE PassRef (VAR j: INTEGER);
BEGIN
    INC(j);
    StdLog.String("j = "); StdLog.Int(j); StdLog.Ln;
END PassRef;

PROCEDURE CallByReference*;
    VAR
        i: INTEGER;
BEGIN
    i := 6;
    StdLog.String("i = "); StdLog.Int(i); StdLog.Ln;
    PassRef(i);
    (* ra1 *)
    StdLog.String("i = "); StdLog.Int(i); StdLog.Ln
END CallByReference;
```

Run-time stack



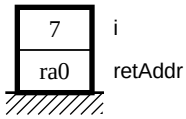
Log

```
i = 6
j = 7
```

```
PROCEDURE PassRef (VAR j: INTEGER);
BEGIN
    INC(j);
    StdLog.String("j = "); StdLog.Int(j); StdLog.Ln;
END PassRef;

PROCEDURE CallByReference*;
    VAR
        i: INTEGER;
BEGIN
    i := 6;
    StdLog.String("i = "); StdLog.Int(i); StdLog.Ln;
    PassRef(i);
    (* ra1 *)
    StdLog.String("i = "); StdLog.Int(i); StdLog.Ln
END CallByReference;
```

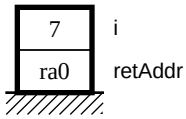
Run-time stack **Log**

```
i = 6
j = 7
```



```
PROCEDURE PassRef (VAR j: INTEGER);
BEGIN
    INC(j);
    StdLog.String("j = "); StdLog.Int(j); StdLog.Ln;
END PassRef;

PROCEDURE CallByReference*;
    VAR
        i: INTEGER;
BEGIN
    i := 6;
    StdLog.String("i = "); StdLog.Int(i); StdLog.Ln;
    PassRef(i);
    (* ra1 *)
    StdLog.String("i = "); StdLog.Int(i); StdLog.Ln
END CallByReference;
```

Run-time stack

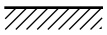


Log

```
i = 6
j = 7
i = 7
```

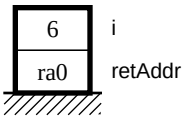
```
PROCEDURE PassRef (VAR j: INTEGER);
BEGIN
    INC(j);
    StdLog.String("j = "); StdLog.Int(j); StdLog.Ln;
END PassRef;

PROCEDURE CallByReference*;
    VAR
        i: INTEGER;
BEGIN
    i := 6;
    StdLog.String("i = "); StdLog.Int(i); StdLog.Ln;
    PassRef(i);
    (* ra1 *)
    StdLog.String("i = "); StdLog.Int(i); StdLog.Ln
END CallByReference;
```

Run-time stack

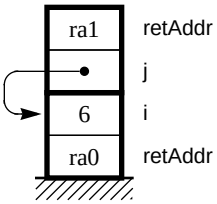


Log

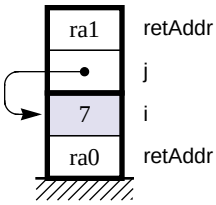
```
i = 6
j = 7
i = 7
```



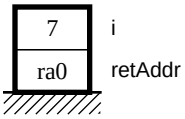
(a) Before call to PassRef



(b) After call to PassRef



(c) After INC(j)



(d) After return from PassRef

Figure 12.13
Memory allocation for Figure 12.11.

```
MODULE Pbox12E;
    IMPORT Dialog;
    VAR
        d*: RECORD
            width*, height*: REAL;
            area-, perim-: REAL
        END;

    PROCEDURE CalcRectSize (wid, ht: REAL; OUT ar, per: REAL);
    BEGIN
        ASSERT(wid > 0.0, 20);
        ASSERT(ht > 0.0, 21);
        ar := wid * ht;
        per := 2.0 * (wid + ht)
    END CalcRectSize;
```

Figure 12.14
Computing the area and perimeter of a rectangle with a procedure.

```
    PROCEDURE Rectangle*;
    BEGIN
        IF (d.width > 0.0) & (d.height > 0.0) THEN
            CalcRectSize(d.width, d.height, d.area, d.perim);
            (* ra1 *)
        ELSE
            d.width := 0.0; d.height := 0.0;
            d.area := 0.0; d.perim := 0.0
        END;
        Dialog.Update(d)
    END Rectangle;

BEGIN
    d.width := 0.0; d.height := 0.0;
    d.area := 0.0; d.perim := 0.0
END Pbox12E.
```

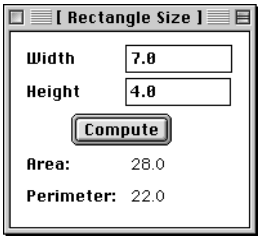
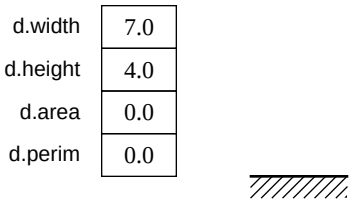
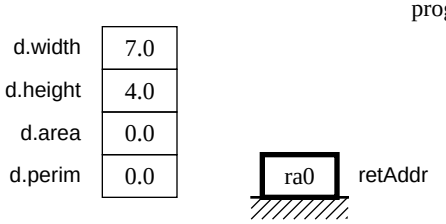


Figure 12.15
The dialog box for the procedure of Figure 12.14.

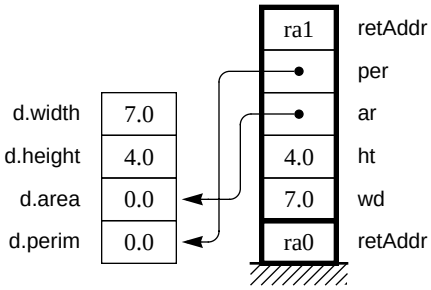


(a) Before call to Rectangle

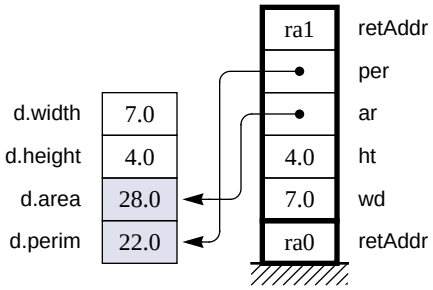


(b) Before call to CalcRectSize

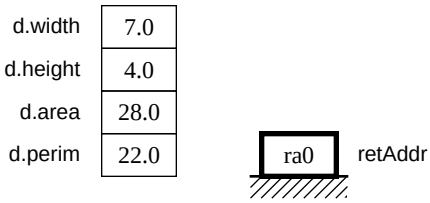
Figure 12.16
Memory allocation for the program of Figure 12.14.



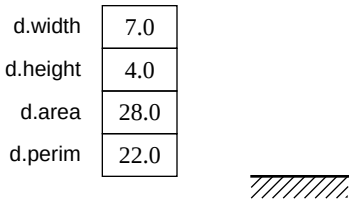
(c) After call to CalcRectSize



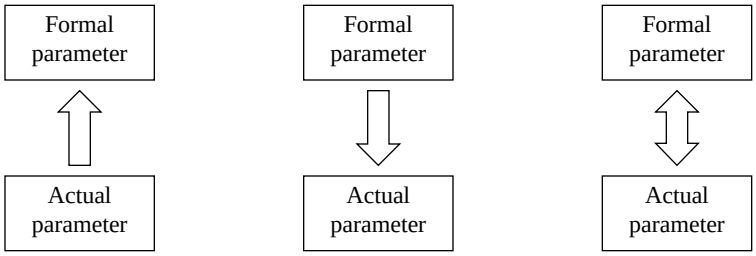
(d) Before return to Rectangle



(e) After return to Rectangle



(f) After return to framework



(a) Call by value. **(b)** Call by result. **(c)** Call by reference.

Figure 12.17
The flow of information for three different calling mechanisms.

Summary of call by value, call by result, and call by reference

- Call by value
  - Default
  - Use when the actual parameter should not change.
  - The actual parameter can be an expression.

Summary of call by value, call by result, and call by reference

- Call by value
  - Default
  - Use when the actual parameter should not change.
  - The actual parameter can be an expression.
- Call by result
  - OUT
  - Use when the actual parameter should change and its initial value is undefined.
  - The actual parameter must be a variable.

Summary of call by value, call by result, and call by reference

- Call by value
  - Default
  - Use when the actual parameter should not change.
  - The actual parameter can be an expression.
- Call by result
  - OUT
  - Use when the actual parameter should change and its initial value is undefined.
  - The actual parameter must be a variable.
- Call by reference
  - VAR
  - Use when the actual parameter should change and the called procedure uses its initial value.
  - The actual parameter must be a variable.

```
MODULE Pbox12F;
    IMPORT TextModels, TextViews, Views, TextControllers, PboxMappers;

    VAR (* WARNING-Unnecessary global variables. Bad design. *)
        cn: TextControllers.Controller;
        sc: PboxMappers.Scanner;
        dataValue: REAL;
        md: TextModels.Model;
        vw: TextViews.View;
        fm: PboxMappers.Formatter;

    PROCEDURE PrintLine;
        VAR
            i, n: INTEGER;
    BEGIN
        ASSERT(dataValue >= 0.0, 20);
        fm.WriteReal(dataValue, 8, 1);
        fm.WriteString(" | ");
        n := SHORT(ENTIER(dataValue + 0.5));
        FOR i := 1 TO n DO
            fm.WriteChar("*")
        END;
        fm.WriteLn
    END PrintLine;
```

Figure 12.18
A procedure whose computation is the same as that in Figure 12.5, but without parameters for PrintLine.

```
    PROCEDURE PrintHistogram*;
    BEGIN
        cn := TextControllers.Focus();
        IF cn # NIL THEN
            md := TextModels.dir.New();
            fm.ConnectTo(md);
            sc.ConnectTo(cn.text);
            sc.ScanReal(dataValue);
            WHILE ~sc.eot DO
                PrintLine;
                sc.ScanReal(dataValue)
            END;
            vw := TextViews.dir.New(md);
            Views.OpenView(vw)
        END
    END PrintHistogram;

END Pbox12F.
```

- Easier to read
- Easier to modify
- Easier to use in other programs

When must you resort to a global variable? When its value must be persistent between invocations of the module's exported procedures. *When to use a global variable*

```
PROCEDURE PrintLine (x: REAL; IN f: PboxMappers.Formatter);
    VAR
    i, n: INTEGER;
BEGIN
    ASSERT(x >= 0.0, 20);
    f.WriteReal(x, 8, 1);
    f.WriteString(" | ");
    n := SHORT(ENTIER(x + 0.5));
    FOR i := 1 TO n DO
        f.WriteChar("*")
    END;
    f.WriteLn
END PrintLine;
```

```
PROCEDURE PrintLine;
    VAR
        i, n: INTEGER;
BEGIN
    ASSERT(dataValue >= 0.0, 20);
    fm.WriteReal(dataValue, 8, 1);
    fm.WriteString(" | ");
    n := SHORT(ENTIER(dataValue + 0.5));
    FOR i := 1 TO n DO
        fm.WriteChar("*")
    END;
    fm.WriteLn
END PrintLine;
```