
SOURCE CODE EXAMPLES: FORTRAN

This appendix contains the code examples referred to in the main text. There is a version of the appendix for each language. This is the Fortran version.

#1 EXAMPLE FOR THE CALLING DIAGRAM

```

subroutine main()
  call function_one()
  call function_two()
end
C
subroutine function_one()
  call function_three()
end
C
subroutine function_two()
  call function_four()
  call function_one()
  call function_two()
end
C
subroutine function_three()
end

subroutine function_four()
  call function_three()
end

```

#2 SMALL EXAMPLE FOR HAND SIMLUATION

```

integer v1, v2, excess

v1 = 10
v2 = v1 * 100
100 if (v2.le.500) then
      goto 200
endif
excess = v2 - 500
print *, 'Excess over 500 = ', v2 - 500
v1 = v1 - 1
v2 = v1 * 100
goto 100
200 ...

```

#3 EXAMPLE FOR SCOPE REGIONS

Fortran has only two levels of scope. Variables can be local to a subroutine or function (this applies to parameters as well) or they can be accessible in a common block to any subprogram. Even then variables in a common block are only accessible if the declaration is repeated in the body of the subprogram.

```

subroutine A()
  integer B
end
C
subroutine E()
  integer C
end

```

#4 LARGE EXAMPLE FOR HAND SIMULATION

```

block data
  integer g1, g2
  common /global/ g1, g2
  data g1, g2 /100, 20/
end
C
subroutine f1(xx)
  integer xx
  integer w, res
  integer g1, g2
  common /global/ g1, g2
  w = 10
  w = w + g1
  res = w / xx
  print *, res
end
C
integer function f2()
  integer w
  integer g1, g2
  common /global/ g1, g2
  w = 5
  g1 = g2 + w
  f2 = w + 2
end
C
C main program
integer r, g1, g2
common /global/ g1, g2
integer f2
r = f2()
call f1(r)
print *, g1
end

```

#5 TOP-DOWN DESIGN EXAMPLE

```

C   The paint estimator
    block data
      real paintc, height
      common /consts/ paintc, height
      data paintc, height/12.5, 8.0/
      real totalc, length, width
      common /globals/ totalc, length, width, cover, area
    end

C
    subroutine invalc()
      real paintc, height
      common /consts/ paintc, height
      real totalc, length, width, cover, area
      common /globals/ totalc, length, width, cover, area
      print *, 'Type in length, width, coverage:'
      read *, length, width, cover
    end

C
    subroutine calca()
      real paintc, height
      common /consts/ paintc, height
      real totalc, length, width, cover, area
      common /globals/ totalc, length, width, cover, area
      area = 2 * (length + width) * height + length * width
      print *, area
    end

C
    subroutine calcca()
      real paintc, height
      common /consts/ paintc, height
      real totalc, length, width, cover, area
      common /globals/ totalc, length, width, cover, area
      print *, area, cover, paintc
      totalc = area / cover * paintc;
    end

C
    subroutine calcc()
      call calca()
      call calcca()
    end

C
    subroutine dispc()
      real totalc, length, width, cover, area
      common /globals/ totalc, length, width, cover, area
      print *, 'Cost of painting a room is ', totalc
    end

```

```

C
C   main program
C   call invals()
C   call calcc()
C   call dispc()
C   end

```

C#6 TOP-DOWN DESIGN EXAMPLE, REWORKED

```

C The Paint Estimator
C
C   block data
C   real paintc, height
C   common /consts/ paintc, height
C   data paintc, height /12.5, 8.0/
C   end
C
C   subroutine invals(l, w, c)
C   real l, w, c
C
C   print *, 'Type in length, width, coverage:'
C   read *, l, w, c
C   end
C
C   real function calca(l, w)
C   real l, w
C
C   real paintc, height
C   common /consts/ paintc, height
C   calca = 2 * (l + w) * height + l * w
C   end
C
C   real function calcca(a, c)
C   real a, c
C
C   real paintc, height
C   common /consts/ paintc, height
C   calcca = a / c * paintc;
C   end
C
C   real function calcc(l, w, c)
C   real l, w, c, area
C
C   real calca
C   area = calca(l, w)
C   calcc = calcca(area, c)
C   end
C

```

```

subroutine dispc(l, w, c, cp)
  real l, w, c, cp
  print *, 'Cost of painting a room ', l, ' by ', w, ' with'
  print *, ' paint of coverage ', c, ' is ', cp
end
C
C  main program
  real costp, length, width, cover
  call invalc(length, width, cover)
  costp = calcc(length, width, cover)
  call dispc(length, width, cover, costp)
endC

```

#7 THE MAIN PROGRAM FOR THE TOP-DOWN DESIGN EXAMPLE

```

C main program
  float costp, length, width, coverp;
C
  length = inputl()
  width = inputw()
  coverp = inputc()
  costp = calcc(length, width, coverage);
  call dispc(length, width, coverage, costp);
end

```

#8 THE PAINT ESTIMATOR DESIGN IN CLASSES

Since Fortran does not have classes, we must simulate the encapsulation of data with named common areas within a block data subprogram, and the encapsulation of subprograms through naming conventions. The common areas are:

```

block data RoomClass
  real width, length, height
  common /Room/ width, length, height
  save /Room/
  data height /8.0/
end
C
block data PaintClass
  real pcost, cover
  common /Paint/ pcost, coverp
  save /Paint/
  data pcost /12.5/
end
C
block data EstClass
  real costp
  common /Est/ costp
end

```

#9 THE PAINT ESTIMATOR DESIGN WITH CLASS METHODS

The methods will be functions whose name begin with the first letter of the “class” name. There is no need for accessor methods or the constructor in this case, since we can use the variables in the common blocks directly. Initialization is done in the block data subprograms.

```

C Room methods
  subroutine RInput()
    real width, length, height
    common /Room/ width, length, height
    print *, 'Type width and length: '
    read *, width, length
  end

C Paint methods
  subroutine PInput()
    real pcost, coverp
    common /Paint/ pcost, coverp
    print *, 'Type coverage: '
    read *, coverp
  end

C
  subroutine EDisp()
    real costp
    common /Est/ costp
    print *, 'Cost is ', costp
  end

C
  subroutine ERun()
C   defined below
  end

```

#10 THE PAINT ESTIMATOR TOP LEVEL METHODS

```

  subroutine EEstm()
    real warea, carea, width, length, height
    real costp, coverp
    common /Room/ width, length, height
    common /Paint/ pcost, coverp
    common /Est/ costp
    warea = 2 * (width + length) * height
    carea = width * length
    print *, warea, carea
    costp = (warea + carea) / coverp * pcost
  end

C
  subroutine ERun()
    call RInput()
    call PInput()
    call EEstm()
    call EDisp()
  end

```

```
C   main program
      call ERun()
      end
```

#11 SIMPLE EXAMPLE FOR DEBUGGING

```
100  do 100 n = 1, 10
      total = total + num
      read *, num
      continue
```

#12 DEBUGGING WITH TRACE ADDED

```
100  do 100 n = 1, 10
      print *, 'Trace total: ', total
      total = total + num
      read *, num
      continue
```

#13 ADDING AN ASSERTION

Fortran does not have a built-in assertion mechanism. See Example #14.

#14 AN ASSERTION IN STRAIGHT CODE

```
if (x .lt. 0 .or. x .gt. 9) then
  print *, 'Error: x is out of range ', x
  stop
endif
```

#15 A DEBUGGING EXAMPLE

```
      character*1 c, lastc
      character*80 line
      integer totalies, totaleis
*
      totalies = 0
      totaleis = 0
      lastc = char(0)
*
      print *, 'Type any number of lines followed by EOF'
*
100  read (unit = *, fmt = 101, end = 1000) line
101  format (A80)
      index = 1
200  c = line(index:index)
      index = index + 1
      if (index .gt. 80) goto 100
      if (c .ge. 'A' .and. c .le. 'Z') then
          c = char(ichar(c) + 32)
      endif
      if (c .eq. 'i' .and. lastc .eq. 'e') then
          totaleis = totaleis + 1
      else if (c .eq. 'e' .and. lastc .eq. 'i') then
```

```

        totalies = totalies + 1
    else
        lastc = c
    endif
    goto 200
*
1000  print *, 'the number of IEs is ', totalies
      print *, 'and the number of EIs is ', totaleis
end

```

#16 THE EXAMPLE WITH TRACE ADDED

```

character*1 c, lastc
character*80 line
integer totalies, totaleis
*
totalies = 0
totaleis = 0
lastc = char(0)
*
print *, 'Type any number of lines followed by EOF'
*
100  read (unit = *, fmt = 101, end = 1000) line
101  format (A80)
      index = 1
200  c = line(index:index)
      index = index + 1
      if (index .gt. 80) goto 100
      if (c .ge. 'A' .and. c .le. 'Z') then
          c = char(ichar(c) + 32)
      endif
      if (c .eq. 'i' .and. lastc .eq. 'e') then
          totaleis = totaleis + 1
      else if (c .eq. 'e' .and. lastc .eq. 'i') then
          totalies = totalies + 1
      else
          lastc = c
      endif
      print *, 'c = ', c, 'lastc = ', lastc, 'totalIEs = ', totalies,
c      'totalEIS = ', totaleis
      goto 200
*
1000  print *, 'the number of IEs is ', totalies
      print *, 'and the number of EIs is ', totaleis
end

```

#17 THE EXAMPLE WITH ASSERTION ADDED

```

else
    lastc = c
*****
    if (lastc .ne. c) then
        print *, 'lastc not equal to c'
        stop
    endif
*****

```


#18 THE EXAMPLE ASSERTION AND TRACE

```

...
    else
        lastc = c
C ***** atrace statement *****
    print *, 'lastc = ', lastc, 'c = ', c
C *****
C *****
C if (lastc .ne. c) then
C     print *, 'lastc not equal to c'
C     stop
C endif
C *****

```

#19 THE EXAMPLE DEBUGGED

```

character*1 c, lastc
character*80 line
integer index
integer totalies, totaleis

C
totalies = 0
totaleis = 0
lastc = char(0)

C
print *, 'Type any number of lines followed by EOF'

C
100 read (unit = *, fmt = 101, end = 1000) line
101 format (A80)
    index = 1
200 c = line(index:index)
    index = index + 1
    if (index .gt. 80) goto 100
    if (c .ge. 'A' .and. c .le. 'Z') then
        c = char(ichar(c) + 32)
    endif
    if (c .eq. 'i' .and. lastc .eq. 'e') then
        totaleis = totaleis + 1
    else if (c .eq. 'e' .and. lastc .eq. 'i') then
        totalies = totalies + 1
    endif
C     else **** not necessary ****
    lastc = c
C     endif **** moved up ****
    goto 200

C
1000 print *, 'the number of IEs is ', totalies
    print *, 'and the number of EIs is ', totaleis
end

```

