INTERNATIONAL ATOMIC ENERGY AGENCY
UNITED NATIONS EDUCATIONAL, SCIENTIFIC AND CULTURAL ORGANIZATION
INTERNATIONAL CENTRE FOR THEORETICAL PHYSICS
I.C.T.P., P.O. BOX 586, 34100 TRIESTE, ITALY, CABLE: CENTRATOM TRIESTE

H4.SMR/854-3

College on Computational Physics

15 May - 9 June 1995

Fortran 90, Exercises

M. Metcalf

CN Division, CERN
Geneva, Switzerland

2.14 Summary

n this chapter we have introduced the elements of the Fortran 90 language. :he character set has been listed, and the manner in which sequences of char- ıcters form literal constants and names explained. In this context we have ncountered the five intrinsic data types defined in Fortran 90, and seen how ·ach data type has corresponding literal constants and named objects. We ıave seen how derived types may be constructed from the intrinsic types. We ıave introduced one method by which arrays may be declared, and seen how heir elements may be referenced by subscript expressions. The concepts of he array section, character substring, and pointer have been presented, and ›ome important terms defined. In the following chapter we shall see how hese elements may be combined into expressions and statements, Fortran's :quivalents of 'phrases' and 'sentences'.

With respect to Fortran 77 there are many changes in this area: the larger :haracter set; a new source form; the significance of blanks; the ›arameterization of the intrinsic types; the binary, octal, and hexadecimal con- .tants; the ability to obtain a desired precision and range; quotes as well as ıpostrophes as character constant delimiters; longer names; derived data types; ıew array subscript notations; array constructors; and last, but not least, ›ointers. Together they represent a substantial improvement in the ease of use ınd power of the language.

Exercises (Chapter 2)

ı. For each of the following assertions, state whether it is true, false or not determined, ıccording to the Fortran 90 collating sequences:

B is less than M
8 is less than 2
* is greater than T
$ is less than /
blank is greater than A
blank is less than 6

:. Which of the Fortran 90 lines in Figure 4 are correctly written according to the equirements of the Fortran 90 source form? Which ones contain commentary? Which lines are initial lines and which are continuation lines?

```
    X = Y
 3  A = B+C ! Add
    WORD = 'String'
    A = 1.0; B = 2.0
    A = 15. ! Initialize A; B = 22. ! and B
    SONG = "Life is just&
       & a bowl of cherries"
    CHIDE = 'Waste not,
       want not!'
  0  C(3:4) = 'UP"
```

Figure 4.

3. Classify the following literal constants according to the five intrinsic data types of Fortran 90. Which are not legal literal constants?

```
-43               'WORD'
4.39               1.9-4
0.0001E+20         'STUFF & NONSENSE'
4 9                (0.,1.)
(1.E3,2)           'I CAN''T'
'(4.3E9, 6.2)'     .TRUE._1
E5                 'SHOULDN' 'T'
1_2                 "O.K."
Z10                 Z'10'
```

4. Which of the following names are legal Fortran 90 names?

```
NAME       NAME32
QUOTIENT   123
A1B2C3     NO-GO
STOP!      BURN_
NO_GO      LONG__NAME
```

5. What are the first, tenth, eleventh and last elements of the following arrays?

```
REAL, DIMENSION(11)       :: A
REAL, DIMENSION(0:11)     :: B
REAL, DIMENSION(-11:0)    :: C
REAL, DIMENSION(10,10)    :: D
REAL, DIMENSION(5,9)      :: E
REAL, DIMENSION(5,0:1,4)  :: F
```

Write an array constructor of eleven integer elements.

6. Given the array declaration

```
CHARACTER(LEN=10), DIMENSION(0:5,3) :: C
```

```
C(2,3)         C(4:3)(2,1)
C(6,2)         C(5,3)(9:9)
C(0,3)         C(2,1)(4:8)
C(4,3)(:)      C(3,2)(0:9)
C(5)(2:3)      C(5:6)
C(5,3)(9)      C(,)
```

} which are valid?

7. Write derived type definitions appropriate for:

a) a vehicle registration;
b) a circle;
c) a book (title, author, and number of pages).

Give an example of a derived type constant for each one.

8. Given the declaration for T in Section 2.12, which of the following objects and subobjects are arrays?

```
T              T(4)%VERTEX(1)
T(10)          T(5:6)
T(1)%VERTEX    T(5:5)
```

9. Write specifications for these entities:

a) an integer variable inside the range -10^{20} to 10^{20};
b) a real variable with a minimum of 12 decimal digits of precision and a range of 10^{-100} to 10^{100};
c) a Kanji character variable on a processor that supports Kanji with KIND=2.

If the *target* in a pointer assignment statement is itself a pointer, then a straightforward copy of the pointer takes place. If it is undefined or disassociated, this state is copied; otherwise the targets become identical.

The type, type parameters, and rank of the *pointer* and *target* in a pointer assignment statement must each be the same. If the *pointer* is an array, it takes its shape and bounds from the *target*. The bounds are as would be returned by the functions LBOUND and UBOUND (Section 8.12.2) for the target, which means that an array section or array expression is always taken to have the value 1 for a lower bound and the extent for the corresponding upper bound.

3.13 Summary

In this chapter, we have seen how scalar and array expressions, of numeric, logical, character, and derived types may be formed; and how the corresponding assignments of the results may be made. The relational expressions and the use of pointers have also been presented. We now have the information required to write short sections of code forming a sequence of statements to be performed one after the other. In the following chapter we shall see how more complicated sequences, involving branching and iteration, may be built up.

Features described in this chapter which are new to Fortran are the use of the alternative representations <, <=,... for the relational operators; the ability of the two sides of a character assignment to overlap; structure constructors; defined operators and assignment; array expressions and assignment; and the use of pointers in expressions and assignment.

Exercises (Chapter 3)

1. If all the variables are numeric scalars, which of the following are valid numeric expressions?

```
A+B              -C
A+-C             D+(-F)
(A+C)**(P+Q)     (A+C)(P+Q)
-(X+Y)**I        4.((A-D)-(A+4.*X)+1)
```

2. In the following expressions, add the parentheses which correspond to Fortran 90's rules of precedence (assuming A, C—F are real scalars, I—N are logical scalars, and B is a logical array), for example

–3–

```
A+D**2/C      becomes      A+((D**2)/C)

C+4.*F
4.*G-A+D/2.
A**E**C**D
A*E-C**D/A+E
I .AND. J .OR. K
.NOT. L .OR  .NOT. I .AND. M .NEQV. N
B(3).AND.B(1).OR.B(6).OR..NOT.B(2)
```

3. What are the results of the following expressions?

```
3+4/2        6/4/2
3.*4**2      3.**3/2
-1.**2       (-1.)**3
```

4. A scalar character variable R has length 8. What are the contents of R after each of the following assignments?

```
R = 'ABCDEFGH'
R = 'ABCD'//'01234'
R(:7) = 'ABCDEFGH'
R(:6) = 'ABCD'
```

5. Which of the following logical expressions are valid, (B is a logical array)?

```
.NOT.B(1).AND.B(2)       .OR.B(1)
B(1).OR..NOT.B(4)        B(2)(.AND.B(3).OR.B(4))
```

6. If all the variables are real scalars, which of the following relational expressions are valid?

```
D .LE. C             P .LT. T > 0.
X-1 /= Y             X+Y < 3 .OR. > 4.
D.LT.C.AND.3.0       Q.EQ.R .AND. S>T
```

7. Write expressions to compute:

a) the perimeter of a square of side L;

b) the area of a triangle of base B and height H;

c) the volume of a sphere of radius R.

8. An item costs *n* cents. Write a declaration statement for suitable variables and assignment statements which compute the change to be given from a $1 bill for any value of *n* from 1 to 99, using coins of denomination 1, 5, 10, and 25 cents.

9. Given the type declaration for INTERVAL in Section 3.8, the definitions of + given

```
TYPE(INTERVAL) A,B,C,D
REAL R
```

which of the following statements are valid?

```
A = B + C
C = B + 1.0
D = B + 1
R = B + C
A = R + 2
```

10. Given the type declarations

```
REAL, DIMENSION(5,6) :: A, B
REAL, DIMENSION(5) :: C
```

which of the following statements are valid?

```
A = B                C = A(:,2) + B(5,:5)
A = C+1.0            C = A(2,:) + B(:,5)
A(:,3) = C           B(2:,3) = C + B(:5,3)
```

which will output a line such as

```
VAR1 = 1.0  VAR2 = 2.0
```

Similarly, input data can be read by statements like

```
READ *, VAL1, VAL2
```

This is sufficient to allow us to write simple programs like that in Figure 12, which outputs the converted values of a temperature scale between specified limits. Valid inputs are shown at the end of the example.

Exercises (Chapter 4)

1. Write a program which
   a) defines an array to have 100 elements;
   b) assigns to the elements the values 1,2,3,....100;
   c) reads two integer values in the range 1 to 100;
   d) reverses the order of the elements of the array in the range specified by the two values.

2. The first two terms of the Fibonacci series are both 1, and all subsequent terms are defined as the sum of the preceding two terms. Write a program which reads an integer value LIMIT and which computes and prints the first LIMIT terms of the series.

3. The coefficients of successive orders of the binomial expansion are shown in the normal Pascal triangle form as

```
        1
      1   1
    1   2   1
  1   3   3   1
1   4   6   4   1
       etc.
```

Write a program which reads an integer value LIMIT and prints the first LIMIT lines of this Pascal triangle.

4. Define a character variable of length 80. Write a program which reads a value for this variable. Assuming that each character in the variable is alphabetic, write code which sorts them into alphabetic order, and prints out the frequency of occurrence of each letter.

5. Write a program to read an integer value LIMIT and print the first LIMIT prime numbers, by any method.

6. Write a program which reads a value X, and calculates and prints the corresponding value X/(1.+X). The case X=-1. should produce an error message and be followed by an attempt to read a new value of X.

7. Given a chain of entries of the type ENTRY of Section 2.13, modify the code in Figure 9 (Section 4.5) so that it removes the entry with index 10, and makes the previous entry point to the following entry.

5.22 Summary

A program consists of a sequence of program units, in any order. It must contain exactly one main program but may contain any number of modules and external subprograms. We have described each kind of program unit. Modules contain data definitions, type definitions, namelist groups, interface blocks, and module subprograms, all of which may be accessed in other program units with the USE statement.

Subprograms define procedures, which may be functions or subroutines. They may also be defined intrinsically (Chapter 8) and external procedures may be defined by means other than Fortran. We have explained how information is passed between program units and to procedures through argument lists and through the use of modules. Procedures may be called recursively provided they are suitably labelled.

The interface to a procedure may be explicit or implicit. If it is explicit, keyword calls may be made, and the procedure may have optional arguments. Interface blocks permit procedures to be invoked as operations or assignments, or by a generic name. The character lengths of dummy arguments may be assumed.

We have also explained about the scope of labels and Fortran names, and introduced the concept of a scoping unit.

Many of the features are new to Fortran: the internal subprogram, modules, the interface block, optional and keyword arguments, argument intent, pointer dummy arguments and function results, recursion, and overloading. These are powerful additions to the language, particularly in the construction of large programs and libraries.

Exercises (Chapter 5)

1. A subroutine receives as arguments an array of values, *x*, and the number of elements in *x*, *n*. If the mean and variance of the values in *x* are estimated by

$$mean = (\sum_{i=1}^{n} x(i))/n$$

and

$$variance = (\sum_{i=1}^{n} (x(i) - mean)^2)/(n-1)$$

write a subroutine which returns these calculated values as arguments. The subroutine should check for invalid values of n (≤ 1).

2. A subroutine MATRIX_MULT multiplies together two matrices A and B, whose dimensions are I×J and J×K, respectively, returning the result in a matrix C, dimensioned I×K. Write MATRIX_MULT, given that each element of C is defined by

$$C(m,n) = \sum_{l=1}^{J} (A(m,l) \times B(l,n))$$

The matrices should appear as arguments to MATRIX_MULT.

3. The subroutine RANDOM_NUMBER (Section 8.15.2) returns a random number in the range 0.0 to 1.0, that is

```
CALL RANDOM_NUMBER(R)   ! 0≤R<1
```

Using this function, write the subroutine SHUFFLE of Figure 16

4. A character string consists of a sequence of letters. Write a function to return that letter of the string which occurs earliest in the alphabet, for example, the result of applying the function to 'DGUMVETLOIC' is 'C'.

5. Write an internal procedure to calculate the volume of a cylinder of radius r and length l, $\pi r^2 l$, using as the value of π the result of ACOS(−1.0), and reference it in a host procedure.

6. Choosing a simple card game of your own choice, and using the random number procedure (Section 8.15.2), write subroutines DEAL and PLAY of Section 5.4, using data in a module to communicate between them.

7. Objects of the intrinsic type CHARACTER are of a fixed length. Write a module containing a definition of a variable length character string type, of maximum length 80, and also the procedures necessary to:

i) assign a character variable to a string;

ii) assign a string to a character variable;

iii) return the length of a string;

iv) concatenate two strings.

Exercises (Chapter 6)

1. Given the array declaration

```
REAL, DIMENSION(50,20) :: A
```

write array sections representing

i) the first row of A;

ii) the last column of A;

iii) every second element in each row and column;

iv) as for (iii) in reverse order in both dimensions;

v) a zero-sized array.

2. Write a WHERE statement to double the value of all the positive elements of an array Z.

3. Write an array declaration for an array J which is to be completely defined by the statement

```
J = (/ (3, 5, I=1,5), 5,5,5, (I, I = 5,3,-1 ) /)
```

4. Classify the following arrays:

```
SUBROUTINE EXAMPLE(N, A, B)
   REAL, DIMENSION(N, 10) :: W
   REAL A(:), B(0:)
   REAL, POINTER          :: D(:, :)
```

5. Write a declaration and a pointer assignment statement suitable to reference as an array all the third elements of component DU in the elements of the array TAR having all three subscript values even (Section 6.9).

6. Given the array declarations

```
INTEGER, DIMENSION(100, 100)       :: L, M, N
INTEGER, DIMENSION(:, :), POINTER :: LL, MM, NN
```

rewrite the statements

```
L(J:K+1, J-1:K) = L(J:K+1, J-1:K) + L(J:K+1, J-1:K)
L(J:K+1, J-1:K) = M(J:K+1, J-1:K) + N(J:K+1, J-1:K) + N(J:K+1, J:K+1)
```

- 7 -

as they could appear following execution of the statements

```
LL => L(J:K+1, J-1:K)
MM => M(J:K+1, J-1:K)
NN => N(J:K+1, J-1:K)
```

7. Complete Exercise 1 of Chapter 4 using array syntax instead of DO constructs.

8. Write a module to maintain a data structure consisting of a linked list of integers, with the ability to add and delete members of the list, efficiently.

9. Write a module that contains Figure 31 (Section 6.6) as a module procedure and supports the defined operations and assignments that it contains.

has the same effect as a single statement containing all the variable names in the same order. A namelist group object may belong to more than one namelist group.

If the type, type parameters, or shape of a namelist variable is specified in a specification statement in the same scoping unit, the specification statement must either appear before the NAMELIST statement, or be a type declaration statement that confirms the implicit typing rule in force in the scoping unit for the initial letter of the variable. Also, if any variable has the PUBLIC attribute, no variable in the list may have the PRIVATE attribute.

7.16 Summary

In this chapter all of the specification statements of Fortran 90 have been described. The following concepts have been introduced: implicit typing and its attendant dangers, named constants, constant expressions, data initialization, control of the accessibility of entities in modules, saving data between procedure calls, selective access of entities in a module, renaming entities accessed from a module, specification expressions that may be used when specifying data objects and function results, and the formation of variables into namelist groups. In addition, we have explained alternative ways of specifying attributes.

The features described here that are new to Fortran are IMPLICIT NONE; initialization and specification expressions; a much extended type declaration statement; DATA statement extended to include derived types, subobjects, and binary, octal, and hexadecimal constants; new attributes and statements: PUBLIC, PRIVATE, POINTER, ALLOCATABLE, TARGET, INTENT, and OPTIONAL; and the USE and NAMELIST statements.

Exercises (Chapter 7)

1. Write suitable type statements for the following quantities:

i) an array to hold the number of counts in each of the 100 bins of a histogram numbered from 1 to 100;

ii) an array to hold the temperature to two significant decimal places at points, on a sheet of iron, equally spaced at 1cm intervals on a rectangular grid 20cm square, with points in each corner, (the melting point of iron is 1530°C);

iii) an array to describe the state of 20 on/off switches;

iv) an array to contain the information destined for a printed page of 44 lines each of 70 letters or digits.

-8-

2. Explain the difference between the following pair of declarations

REAL :: I = 3.1

and

```
IMPLICIT REAL (I)
PARAMETER (I = 3.1)
```

What is the value of I in each case?

3. Write type declaration statements which initialize:

i) all the elements of an integer array of length 100 to the value zero.

ii) all the odd elements of the same array to 0 and the even elements to 1.

iii) the elements of a real 10x10 square array to 1.0 .

iv) a character string to the digits 0 to 9.

4. In the following module, identify all the scoping units and list the mappings for implicit typing for all the letters in all of them:

```
MODULE MOD
   IMPLICIT CHARACTER(10, 2) (A-B)
   :
CONTAINS
   SUBROUTINE OUTER
      IMPLICIT NONE
      :
   CONTAINS
      SUBROUTINE INNER(FUN)
         IMPLICIT COMPLEX (Z)
         INTERFACE
            FUNCTION FUN(X)
               IMPLICIT REAL (F, X)
               :
            END FUNCTION FUN
         END INTERFACE
      END SUBROUTINE INNER
   END SUBROUTINE OUTER
END MODULE MOD
```

5.

i) Write a type declaration statement that declares and initializes a variable of derived type PERSON (Section 2.9).

ii) Either

a) write a type declaration statement that declares and initializes a variable of type ENTRY (Section 2.13), or

b) write a type declaration statement for such a variable and a DATA statement to initialize its nonpointer components.

i) KIND(x), for X of type REAL

ii) SELECTED_REAL_KIND(6, 20)

iii) 1.7**2

iv) 1.7**2.0

v) (1.7, 2.3)**(-2)

vi) (/ (7*I, I=1, 10) /)

vii) PERSON('REID', 25*2., 22**2)

viii) ENTRY(1.7, 1, NULL)

c) –

8.16 Summary

In this chapter, we introduced the four categories of intrinsic procedures, explained the INTRINSIC statement, and gave detailed descriptions of all the procedures. The procedures of Sections 8.2, 8.6.2, and 8.7 to 8.15 are all new to Fortran. Within Sections 8.3 to 8.5 the procedures CEILING, FLOOR, MODULO, ACHAR, IACHAR, ADJUSTL, ADJUSTR, LEN_TRIM, SCAN, VERIFY, and LOGICAL are new; the remaining functions were present in Fortran 77, have been generalized to handle all kind type parameters, have become elemental, and several have been given additional optional arguments. The function LEN has become an inquiry function.

Exercises (Chapter 8)

1. Write a program to calculate the real or imaginary roots of the quadratic equation $ax^2+bx+c=0$ for any values of a, b, and c. The program should read these three values and print the results. Use should be made of the appropriate intrinsic functions.

2. Repeat Exercise 1 of Chapter 5, avoiding the use of DO constructs.

3. Given the rules explained in Sections 3.12 and 8.2, what are the values printed by the following program?

```
      REAL, TARGET  :: A(3:10)
      REAL, POINTER :: P1(:), P2(:)
      P1 => A(3:9:2)
      P2 => A(9:3:-2)
      PRINT *, ASSOCIATED(P1, P2)
      PRINT *, ASSOCIATED(P1, P2(4:1:-1))
   END PROGRAM
```

4. In the following program, two pointer assignments, one to an array the other to an array section, are followed by a subroutine call. Bearing in mind the rules given in Sections 3.12, 6.3, and 8.12.2, what values does the program print?

```
      REAL, TARGET  :: A(5:10)
      REAL, POINTER :: P1(:), P2(:)
      P1 => A
      P2 => A(:)
      PRINT *, LBOUND (A), LBOUND (A(:))
      PRINT *, LBOUND (P1), LBOUND (P2)
      CALL WHAT (A, A(:))
   CONTAINS
      SUBROUTINE WHAT (X, Y)
         REAL, INTENT (IN) :: X(:), Y(:)
         PRINT *, LBOUND (X), LBOUND (Y)
      END SUBROUTINE WHAT
   END PROGRAM
```

[illegible] —