In this chapter, we will expand on earlier topics discussed in this book. We introduce more advanced character operations, N-dimensional arrays, double precision and complex data types.

10.1 Character Operations

FORTRAN provides the capability of operating on character data. But what kinds of operations make sense on character strings? Certainly the arithmetic operators: +, -, *, / and logical operators: NOT, AND, OR do not make sense with respect to character data. In this section, we shall highlight the kinds of operations that we can apply on strings.

10.1.1 Character Assignment

Character constants can be assigned to character variables using an assignment statement. If the length of a character constant is shorter than the character variable length, blanks are added to the right of the constant. If the length of a character constant is longer than the character variable length, the excess characters on the right are ignored.

Example 2: What will be printed be the following program?

```fortran
CHARACTER *5 MSG1, MSG2
MSG1 = 'GOOD'
MSG2 = 'EXCELLENT'
PRINT*, MSG1, MSG2
END
```

Solution:

GOOD EXCEL

Notice that MSG1 contains the word GOOD followed by 1 blank; an equivalent statement would be

```fortran
MSG1 = 'GOOD ' 
```

while MSG2 contains 'EXCEL'.

Example 2: What will be printed be the following program?

```fortran
CHARACTER *5 MSG1, MSG2
MSG1 = 'GOOD1'
MSG2 = 'EXCELLENT'
PRINT*, MSG1, MSG2
END
```
Solution:

```plaintext
GOOD1EXCEL
```

Notice that there is no automatic blanks between the values of character variables.

A character variable can be used to initialize another character variable as follows:

```plaintext
CHARACTER BTYPE1*3 , BTYPE2*3
BTYPE1 = 'AB+
BTYPE2 = BTYPE1
```

Both variables, BTYPE1 and BTYPE2, contain the character string 'AB+'.

### 10.1.2 Comparison of Character Strings

To perform the comparison, the following points have to be considered:

1. A collating sequence includes all possible characters from lowest to the highest values. Two standard sequences are known: ASCII (American Standard Code for Information Interchange) and EBCDIC (Extended Binary Coded Decimal Interchange Code). In the following table the number that represent a character is equal to the sum of its row number and column number. b represents the space character. Gaps in the tables represent unprintable or control characters.

<table>
<thead>
<tr>
<th>ASCII Table</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
</tr>
<tr>
<td>16</td>
</tr>
<tr>
<td>32</td>
</tr>
<tr>
<td>48</td>
</tr>
<tr>
<td>64</td>
</tr>
<tr>
<td>80</td>
</tr>
<tr>
<td>96</td>
</tr>
<tr>
<td>112</td>
</tr>
</tbody>
</table>
These sequences are based on the numeric value used to represent a character in order to store that character in the computer memory. The ASCII and the EBCDIC sequences use different numeric values for each character. An important point to note here is that the numeric values associated with alphabetic characters do not appear in a continuous numeric sequence in either the ASCII or the EBCDIC character sets. But the numeric values of numeric characters ('0','1', etc.) appear in a continuous sequence in both character sets. Also note that the numeric characters appear after the alphabetic characters in the EBCDIC collating sequence while they appear before in the ASCII collating sequence.

2. All of the relational operators: .EQ., .NE., .LT., .LE., .GT. and .GE. can be used to compare character strings.

3. In order to compare two strings they must be equal in length. If one string is shorter than the other, FORTRAN adds blanks to the right of the shorter string so that they become of equal length.

4. The comparison of two strings starts from left to right character by character.

5. In order for two strings to be equal, they must be identical, character by character. For example, the string 'ICS' is not equal to 'ICS' because of different position of the blank character.

6. If a character string is less than another character string, it is implied that the first string precedes the second string in the order indicated in the collating sequence. Thus 'ABC' is less than 'BCD'.

7. For clarity, sometimes, we use b to represent a blank.

**Example:** What will be printed be the following program?
**10.1.3 Extraction of Substrings**

Each character in a string of size N can be referred to by a number called a character position. The first position in a string is character position 1, and the last character is character position N. By specifying a starting position and an ending position in a string, we can identify parts of a string called the substring. If TEXT is a character variable of size N, then TEXT(I:J) is a substring starting with the Ith character of TEXT and ending with the Jth character of TEXT, where I and J are integer values. J must be greater than or equal to I; otherwise an execution error would occur. In addition, both I and J must be in the range 1,2,3,...,n; otherwise they would not correspond to any character position within the variable. If I is omitted (i.e. TEXT(:J)), it is assumed to be 1. If J is omitted (i.e. TEXT(I:)), it is assumed to be N.

**Example 1:** What will be printed be the following program?

```fortran
CHARACTER A*10 , B
A = 'FORTRAN 77'
B = 'PASCAL'
PRINT 10, A(1:4) , A(9:) , B(:3)
10 FORMAT (' ', A4, 2X, A2, 2X, A3)
END
```

Solution:

```
 FORT 77 PAS
```

**Example 2:** Vowel Determination: Write a program that reads a character string of length 100. The program should print all the vowels in the string.

Solution:

```fortran
CHARACTER TEXT*100 , VOWELS(5)*1
READ*, (VOWELS(K), K = 1, 5)
READ*, TEXT
DO 10 I = 1, 100
   DO 20 J = 1, 5
      IF (TEXT(I:I) .EQ. VOWELS(J)) PRINT*, VOWELS(J)
20 CONTINUE
10 CONTINUE
END
```

**Example 3:** What will be printed be the above program if the input is:

```
'A' 'E' 'I' 'O' 'U'
```
'CAT + DOG = FIGHT'

Solution:

A
O
I

10.1.4 String Concatenation

New character strings may be formed by combining two or more character strings. This operation is known as concatenation and is denoted by a double slash placed between the character strings to be combined.

Example: What will be printed be the following program?

```
CHARACTER DAY*2, MONTH*3, YEAR*4
DAY = '03'
MONTH = 'MAY'
YEAR = '1993'
PRINT 55, MONTH//DAY//YEAR,MONTH//'-'//DAY//'-'//YEAR
55 FORMAT (' ', A9, 5X, A13)
END
```

Solution:

```
MAY031993 MAY-03-1993
```

10.1.5 Character Intrinsic Functions

Just as there are some intrinsic functions for numeric data such as INT, REAL, SQRT, and MOD, there are a number of intrinsic functions designed for use with character strings. These functions are:

10.1.6 Function INDEX(c1 , c2)

The function INDEX takes as arguments two character strings c1 and c2. The function returns an integer value giving the first occurrence of string c2 within string c1; otherwise zero is returned.

Example 1: What will be printed be the following program?

```
CHARACTER FRUIT*6
FRUIT = 'BANANA'
PRINT*, INDEX(FRUIT,'NA')
END
```

Solution:

```
3
```

Example 2: What will be printed be the following program?

```
CHARACTER STR*18
STR = 'TO BE OR NOT TO BE'
K = INDEX(STR, 'BE')
J = INDEX(STR(K+1:), 'BE') + K
PRINT*, K, J
END
```

Solution:

```
4 17
```
Notice that the value of J represent the location of the second occurrence of the string 'BE' in STR.

10.1.7 Function LEN(c)

The function LEN takes as an argument one character string c. It returns the integer length of the string c. The function is used primarily in functions and subroutines that have character string arguments.

Example 1: What will be printed the following program segment:

```
CHARACTER TEXT*10
PRINT*, LEN(TEXT)
```

Solution:

```
10
```

Example 2: Frequency of Blanks: Write a function that accepts a character string and returns the number of blanks in the string.

Solution:

```
INTEGER FUNCTION NB(X)
CHARACTER (*) X
NB = 0
DO 10 I = 1, LEN(X)
   IF (X(I:I) .EQ. ' ') NB = NB + 1
10 CONTINUE
RETURN
END
```

10.1.8 Function CHAR(i)

The function CHAR takes as an argument an integer value i and returns the ith character in the collating sequence.

Example: What is the output of the following program?

```
INTEGER N
N = 65
PRINT*, CHAR(N)
END
```

Solution: Assuming ASCII code representation the program will print

A

10.1.9 Function ICHAR(c)

ICHAR the function is the reverse of function CHAR. It takes as an argument a single character c and returns its position in the collating sequence. The first character in the collating sequence corresponds to position 0 and the last to n-1, where n is the number of characters in the collating sequence.

Example 1: What is the output of the following program?

```
INTEGER J
J = ICHAR('C') - ICHAR('A')
PRINT*, J
END
```

Solution: Assuming ASCII code representation the program will print

2
Example 2: Character Code Determination: What is the output of the following program?

```
CHARACTER CH(26)*1
INTEGER CODE(26)
READ*, CH
DO 10 I = 1, 26
   CODE(I) = ICHAR(CH(I))
10 CONTINUE
PRINT*, CODE
END
```

Assume the input is

'C' 'D' 'E' 'F' 'G' 'H' 'I' 'J' 'K' 'L' 'M' 'N' 'O' 'P' 'Q' 'R' 'S' 'T' 'U' 'V' 'W' 'X' 'Y' 'Z'

Solution:

```
193 194 195 196 197 198 199 200 201 202 203 204 205 206 207 208 209 210 211 212 213 214 215 216
217 218 219 220 221 222 223 224 225 226 227 228 229 230 231 232 233
```

10.1.10 Functions LGE, LGT, LLE, LLT

These functions allow comparisons to be made based on an ASCII collating sequence. They produce one of the two logical values: .TRUE., .FALSE.. Each function takes as arguments two character strings. The function LGE(STRG1, STRG2) is true if STRG1 is greater than or equal to STRG2. The LGT, LLE, LLT functions perform the comparisons greater than, less than or equal and less than respectively. For example, LLT('ABC', 'XYZ') would produce a .TRUE. value.

10.2 N-Dimensional Arrays

In chapter 5, one-dimensional and two-dimensional array data structures were introduced. FORTRAN provides for arrays of up to seven dimensions. A two dimensional array data structure is one that varies in two attributes, a three dimensional array data structure is one that varies in three attributes, a four dimensional array data structure is one that varies in four attributes, and an N dimensional array data structure is one that varies in N attributes. Because of similarities between two and higher dimensional arrays, this section presents three dimensional arrays only. Higher dimensional arrays are treated similarly. An example of three-dimensional arrays is the grades of students in several classes for several quizzes; such an array is declared in FORTRAN as

```
REAL GRADES (50 , 5 , 4)
```

Where we have 50 students, 5 quizzes and 4 classes. In three dimensional arrays, as in two-dimensional arrays, the elements are stored column-wise with the first subscript changing fastest, the second subscript changing more slowly, and the third subscript changing the slowest. For the array declaration

```
REAL A (2 , 2 , 2)
```

The elements are stored in the following order:

A(1,1,1)
A(2,1,1)
A(1,2,1)
A(2,2,1)
A(1,1,2)
A(2,1,2)
A(1,2,2)
A(2,2,2)

To access a three-dimensional array, a nesting of three DO loops is common. Also an implied DO loop can be used.

**Example**

*If we have the declaration:*

```fortran
INTEGER A (3, 4, 5)
```

*then the following three READ statements do the same job of storing data in the three dimensional array A:*

```fortran
READ*, A
READ*, ((A((I, J, K), I = 1, 3), J = 1, 4), K = 1, 5)
DO 10 K = 1, 5
   DO 10 J = 1, 4
      DO 10 I = 1, 3
         READ*, A (I, J, K)
10 CONTINUE
```

### 10.3 Double Precision Data Type

Some applications require that calculations are performed with more precision than is normally provided by the real data type. The real data type has only seven significant digits, while the double precision data type has fourteen digits of significance.

#### 10.3.1 Double Precision Definition

To declare variables of double precision type we use **DOUBLE PRECISION** statement as follows:

```fortran
DOUBLE PRECISION LIST OF VARIABLES
```

or

```fortran
REAL*8 LIST OF VARIABLES
```

#### 10.3.2 Double Precision Operations

The operations that are done on variables declared as double precision will be carried out internally with fourteen significant digits. All the operations that are done on real data type, can also be done on double precision data type such as addition, subtraction, multiplication, division, and exponentiation. Expressions that involve mixed types like double precision, real, and integer will be converted automatically to double precision.

Reading double precision variables is possible and up to fourteen digits to the right of the decimal point are taken from the input stream. Printing double precision values is also possible and the output will show fourteen digits to the right of the decimal point if no formatting is used. The **FORMAT** statement can be used to print double precision
values, the D specification may be used to print double precision numbers. Dw.d format specifier is used where w represents the total width and d represents the number of digits to the right of the decimal point.

10.3.3 Double Precision Intrinsic Functions
There is a large number of mathematical functions that has real arguments and/or real results. There exists an extension to these functions to work with double precision with only one simple change, which is prefixing the function name with the letter D like DSIN(DX), DLOG(DX), DEXP(DX), DABS(DX), etc. DX indicates that the argument to these functions is of the type double precision.

10.4 Complex Data Type
Some applications require that calculations are performed using complex numbers rather than real numbers. A complex number is represented by two real numbers where the first is the real part and the second is the imaginary part.

10.4.1 Complex Data Type Definition
To declare variables of complex type, the following declaration statement should be used in your program:

```
COMPLEX LIST OF VARIABLES
```

10.4.2 Complex Operations
The complex constants appear in the program as two real numbers separated by a comma and enclosed between a pair of parentheses as shown below:

**Example 1**

```
COMPLEX VALUE
VALUE = (2.0, 3.0)
```

The operations that are done on variables defined as complex will be carried out in the same way as defined mathematically. Here is the definition of some of these operations:

- **Addition**
  \[(a+ib) + (c+id) = (a+c) + i(b+d)\]

- **Subtraction**
  \[(a+ib) - (c+id) = (a-c) + i(b-d)\]

- **Multiplication**
  \[(a+ib) \times (c+id) = (ac-bd) + i(ad+bc)\]

- **Division**
  \[
  \frac{a+ib}{c+id} = \left(\frac{ac+bd}{c^2+d^2}\right) + i\left(\frac{cb-da}{c^2+d^2}\right) \\
  \text{where } i = \sqrt{-1}
  \]

When a complex variable is read, two real numbers are taken from the input stream; one for the real part and the other for the imaginary part. Printing a complex variable will result also in two real numbers representing the real part and the imaginary part. If formatting is to be used then two FORMAT specifiers are needed of type F.

10.4.3 Complex Intrinsic Functions
There is a large number of mathematical functions that has real arguments and/or real results. There exists an extension to these functions to work with complex type with only one simple change which is prefixing the function name with the letter C like
CSIN(CX), CLOG(CX), CEXP(CX), CABS(DX), etc. CX indicates that the argument to these functions is of the complex type. In addition there are four functions for complex type which are:

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>REAL(CX)</td>
<td>gives the real part of the argument</td>
</tr>
<tr>
<td>AIMAG(CX)</td>
<td>gives the imaginary part of the argument</td>
</tr>
<tr>
<td>CMPLX(X,Y)</td>
<td>gives the complex number X + i Y</td>
</tr>
<tr>
<td>CONJG(CX)</td>
<td>gives the conjugate of the argument</td>
</tr>
</tbody>
</table>

### 10.5 Exercises

1. What will be printed by the following programs?

```fortran
1. CHARACTER X(1:2)*2
   READ*, X
   PRINT 11, X
11 FORMAT (1X, 2X, I2, 2X, I2)
   END
```

Assume the input is:

'12' '34'

```fortran
2. CHARACTER INPUT*60, SPACE*1
   INTEGER KK, JJ
   INPUT = 'THIS IS A TEST.'
   SPACE = ' '
   KK = 1
   JJ = INDEX(INPUT(KK:), SPACE)
   KK = KK + JJ
   PRINT*, INPUT(:KK - 1)
   IF (KK.LT.INDEX(INPUT, '.')) GOTO 10
   END
```

```fortran
3. CHARACTER STR*10
   INTEGER LL, J, NUM
   STR = '1234'
   LL = INDEX(STR, ' ')
   NUM = 0
   DO 10 J = LL - 1, 1, -1
      NUM = NUM + (ICHAR(STR(J:J)) - ICHAR('0'))*10**J
10 CONTINUE
   PRINT*, NUM
   END
```

```fortran
4. CHARACTER*7 STR, SUB*6
   INTEGER L, K
   L = 3
   SUB = 'AA'
   STR = '++++++++'
   K = INDEX(SUB, ' ')
   IF (K.NE.0) L = LEN(STR) - K + 1
   STR (L/2+1:) = SUB(:K-1)
   PRINT*, STR, K, L
   END
```
5. CHARACTER*1 A, B
   A = 'B'
   B = 'C'
   PRINT 11, B
11   FORMAT(1X,'B=',A)
   END

6. CHARACTER*8 F, K, X
   F(K) = K(1:2)//'REF'//K(6:8)
   X = 'CANDEUULL'
   PRINT*, F(X)
   END

7. INTEGER FUNCTION LENGTH(A)
   CHARACTER *(*) A
   LENGTH = LEN(A)
   RETURN
   END
   CHARACTER*9 A, B, C*6
   INTEGER LENGTH
   READ*, A, B, C
   PRINT*, (LENGTH(A)+LENGTH(B)+LENGTH(C))/5
   END

Assume the input is:

   'AN' 'EASY' 'EXAM'

8. CHARACTER X*9, Y*4
   INTEGER L
   X = 'ABDABDA'
   Y = 'HIJK'
10   L = INDEX(X, 'A')
   IF (L.NE.0) THEN
      X(L:L) = '*'
      GOTO 10
   ENDIF
   PRINT*, LEN(X), X//Y
   END

9. CHARACTER*30 S1, S2
   S1 = 'TODAY IS SATURDAY'
   S2 = 'EXAM 201 + EXAM 101'
   PRINT 11, S1(10:)
   PRINT 22, S2(10:)
11   FORMAT(10X,A)
22   FORMAT(A)
   END

10. LOGICAL LEQ, X, Y, EQAL(4)
    CHARACTER*20 L(8)
    INTEGER K, L
    LEQ(X,Y) = .NOT.X.AND..NOT.Y
    READ*, L
    K = 1
    DO 10 J = 1,7,2
       EQAL(K) = LEQ(L(J),L(J+1)), LLT(L(J),L(J+1))
       K = K + 1
10   CONTINUE
    PRINT*, EQAL
    END

Assume the input is:
11. INTEGER WC, CC, J, K
    CHARACTER SENT*30, BLANK
    WC = 0
    SENT = 'I HAVE FORTRAN CLASSES.'
    J = 0
    BLANK = '.
    CC = INDEX(SENT(J+1:), '.') - 1
    K = INDEX(SENT(J+1:), BLANK)
    IF (K.NE.0 .AND. J.LT.CC) THEN
        WC = WC + 1
        J = K
        GOTO 10
    ENDIF
    IF (CC.NE.0) WC = WC + 1
    CC = CC - WC + 1
    PRINT*, WC, CC, J
END

12. CHARACTER*1 FUNCTION LCHAR(STR)
    CHARACTER*20 STR
    INTEGER LAST
    LAST = 20
    10 IF (STR(LAST:LAST).EQ. ' ') THEN
        LAST = LAST - 1
        GOTO 10
    ENDIF
    LCHAR = STR(LAST:LAST)
    RETURN
END
CHARACTER LCHAR*1, LINE*20
READ*, LINE
PRINT*, LCHAR(LINE)
END

Assume the input is:

'GOOD FINAL EXAM'
13. **SUBROUTINE** INSERT(STR, SUBSTR, AFTER, RESULT, FLAG)
   **CHARACTER** *(*) STR, SUBSTR, AFTER, RESULT
   **LOGICAL** FLAG
   **INTEGER** IPOS
   IPOS = INDEX(STR, AFTER)
   **IF** (IPOS.EQ.0) **THEN**
     FLAG = .FALSE.
   **RETURN**
   **ENDIF**
   FLAG = .TRUE.
   LENAFT = LEN(AFTER)
   LENWOR = LEN(SUBSTR)
   LENSTR = LEN(STR)
   INSPOS = IPOS+LENAFT
   RESULT = STR(:INSPOS) // SUBSTR // STR(INSPOS:)
   **RETURN**
   **END**

   **CHARACTER** STR*13, S1*7, S2*3, RES1*22, RES2*28
   **LOGICAL** FLAG
   **READ***, STR
   **READ***, S1, S2
   **CALL** INSERT(STR, S1, S2, RES1, FLAG)
   **READ***, S1, S2
   **CALL** INSERT(RES1, S1, S2, RES2, FLAG)
   **IF** (FLAG) **THEN**
     PRINT 5, RES2
   **ELSE**
     PRINT 6
   **ENDIF**

   5 **FORMAT** (' ', 'RESULT = "', A, '"')
   6 **FORMAT** (' ', 'NO MATCH')
   **END**

Assume the input is:

'ICS 101 EXAM'
'FORTRAN', '101'
'FINAL', '101'

14. **CHARACTER**+4 ONE, TWO, THREE, FOUR
   ONE = '+'
   TWO = ONE // ONE
   THREE = ONE // TWO
   FOUR = TWO // (ONE // ONE)
   **PRINT***, 'ONE =', ONE
   **PRINT***, 'TWO =', TWO
   **PRINT***, 'THREE =', THREE
   **PRINT***, 'FOUR =', FOUR
   **END**
15. CHARACTER CH*3
   INTEGER A(3),I, J, K, L, M, N
   READ*, (A(J),J=1,2)
   L = 1
   M = 2
   N = 1
   CH = 'ICS'
   DO 10 I = 1,2
      DO 20 J = L,M,N
      PRINT*, (CH(K:K),K=1,A(J))
   20 CONTINUE
   K = L
   L = M
   M = K
   N = -1
   10 CONTINUE
END

Assume the input is:
1 2

2. How many characters one can store in each variable in the following declaration?

   CHARACTER*10 A, B(-2:3), C(2,5:10)*5

3. Assume that the only declaration statements in a FORTRAN program are the following:

   INTEGER A(1:10),B(3,5)
   CHARACTER*7 NUM(50), NAME, CH, C

Which of the following statement(s) is (are) correct FORTRAN statement(s) ?

1. NUM(2)(2:2) = '2'
2. A(3:3) = 2
3. (A(K) = A(K)+2, K = 1,10)
4. NAME(:,3) = NAME(3:)
5. NUM(2) = B(2,2)

4. From the INPUT strings:
   'THIS' 'ASY' 'VERY'
   'EXAM'

generate the message
   \text{THIS IS EASY}

by completing the print statement in the following program

CHARACTER A(2,2)*4
READ*, A
PRINT*, ________________________
END

Hint (Use substring and concatenation of the INPUT strings)

5. Complete the missing parts to produce the expected output:

   CHARACTER*11 NAME, COURSE*6
   NAME = 'COMPUTER'
   COURSE = 'ICS101'
   NAME(__(1)__) = COURSE(__(2)__) 
   PRINT*, NAME
END

The expected output:

   COMPUTER101
Q6) A palindrome is a word of text that is spelled the same forward and backward. The string 'RADAR' is an example of palindrome. Write a FORTRAN program to tell whether an INPUT string of length 60 is a palindrome or not.

7. Write a FORTRAN program that will do the following:
   - Read N, the number of students.
   - Read N data lines, each line contains a student ID, major, course code and grade. The program stores the data into a two-dimensional character array (CLASS) of size 20×4 such that each element has a length of 7 characters.
   - Print all those students who have a major CE and a course code ICS101 and a grade A.

8. Write a FORTRAN program which reads a character string STR of length 7 characters, and an integer array LIST of 7 elements. Then the program should print the string in the order of the numbers stored in the array LIST.
   For example: If STR = 'RNFROTA' and LIST = 3 5 1 6 4 2
   Then your program outputs the 3rd, 5th, 1st,... characters from STR.
   The output should look like the following (Use FORMAT)

   assume the following data:

   'RNFROTA'
   3,5,1,6,4,2

9. Write a FORTRAN program that accepts a string INPUT (at most 60 characters long), and a string PAT (exactly one character long). Then it should find the number of times string PAT is found in the string INPUT and replace every occurrence of PAT by '*'.

10. Consider the following FORTRAN statements

    CHARACTER * 3 STR*5, X
    STR = 'APPLE'

    Which of the following statements will place the string APL in variable X?

    i. X = STR(1:1)//STR(3:3)//STR(4:4)
    ii. X = STR(1:1)//STR(3:4)
    iii. X = STR(1:2)//STR(3:4)
    iv. X = STR(:2)//STR(3:)

11. Write a FORTRAN program that:
   - a) Reads a sentence of up to 70 characters long.
   - b) Replaces each blank within the sentence by the character '$' and prints out the new sentence.
   - c) Places each vowel in the sentence into a new character string called NEW and prints out the string NEW.

    Note: The sentence is terminated by a full stop.
    Vowels are alphabets A, E, I, O and U.
10.6 Solutions to Exercises

Ans 1.

1. ERROR: TYPE MISMATCH IN FORMAT
2. THIS
   THIS IS
   THIS IS A
   THIS IS A TEST.
3. 43210
4. ++AA 3 5
5. B=C
6. CAREFULL
7. 4
8. 9*BD*BD* HIJK
9. EXAM 101 SATURDAY
10. F F F T
11. 1 -1 0
12. M
13. RESULT = 'ICS 101FINAL FORTRAN EXAM'
14. ONE +=
    TWO +=
    THREE=+
    FOUR =+
15. I
    IC
    IC
    IC

Ans 2.

A) 10
B) 60
C) 60

Ans 3

1 and 4

Ans 4.

\texttt{PRINT*, A(1,1)//' '//A(1,1)(3:4)//' E'//A(2,1)}

Ans 5.

(1) 9:10
(2) 4:6
Ans 6.

```fortran
CHARACTER INPUT*60
LOGICAL PALIN
INTEGER K
READ*, INPUT
PALIN = .TRUE.
K = 1
10 IF (PALIN .AND. K .LE. 30) THEN
   IF (INPUT(K:K) .NE. INPUT(61-K:61-K)) PALIN = .FALSE.
   K = K + 1
   GOTO 10
ENDIF
PRINT*, PALIN
END
```

Ans 7.

```fortran
CHARACTER*7 CLASS(20,4)
LOGICAL COND1, COND2, COND3
INTEGER K, N
READ*, N
DO 10 K = 1, N
   READ*, (CLASS(K,J), J = 1 , 4)
10 CONTINUE
DO 20 K = 1 , N
   COND1 = CLASS(K,2) .EQ. 'CE'
   COND2 = CLASS(K,3) .EQ. 'ICS101'
   COND3 = CLASS(K,4) .EQ. 'A'
   IF (COND1 .AND. COND2 .AND. COND3) PRINT*, CLASS(K,1)
20 CONTINUE
END
```

Ans 8.

```fortran
CHARACTER STR*7
INTEGER LIST(7)
INTEGER K
READ*, STR
READ*, (LIST(K), K = 1 , 7)
1 FORMAT(1X, 'DECODED STRING = ', 7A)
END
```

Ans 9.

```fortran
CHARACTER INPUT*60, PAT*1
READ*, INPUT
READ*, PAT
NT = 0
10 K = INDEX(INPUT, PAT)
   IF (K .NE. 0) THEN
      NT = NT + 1
      INPUT(K:K) = '*'
   ENDIF
   GOTO 10
PRINT*, 'THE NUMBER OF TIMES PAT OCCURRED = ', NT
END
```

Ans 10.

I am III
Ans 11.

```plaintext
CHARACTER SENT*70, NEW*70, VOWLS*5
INTEGER K, M
READ*, SENT
VOWLS = 'AEIOU'
NEW = '

10 K = INDEX(SENT, ' ')
   IF (K .NE. 0) THEN
       SENT(K:K) = '$'
       GOTO 10
   ENDIF
   PRINT*, SENT
   M = 0
   DO 20 K = 1, 70
       IF (INDEX(VOWLS, SENT(K:K)) .NE. 0) THEN
           M = M + 1
           NEW(M:M) = SENT(K:K)
       ENDIF
   20 CONTINUE
   PRINT*, NEW
END
```
Index

A specification, 167
ABS, 61
actual arguments, 56
Addition, 13
arguments, 56
arithmetic expression, 14
arithmetic operations, 13
Arithmetic Operators, 14
array declaration, 141
arrays, 117
ascending, 189
ASCII, 202
assembler, 3
assignment, 3
assignment statement, 20
binary operations, 14
binary system, 3
CALL, 64
carriage control, 159, 169
central processing unit, 2
CHAR, 206
CHARACTER, 13
Character Assignment, 201
c caracter constant, 10
character position, 204
character variables, 13
CLOSE, 172
column-wise, 142
comment, 6
comparison, 202
compiler, 3, 5
complex type, 210
c onstant, 9
c ontinuation, 5
CONTINUE, 93
COS, 61
D specification, 209
data, 9
Declaration of a character array, 118
Declaration of a logical array, 118
Declaration of a real array, 118
Declaration of an integer array, 117
declaration statement, 117
declaration statement., 117
decreasing, 189
digits, 10
DIMENSION, 118, 141
division, 13
DO, 91, 92
double precision, 209
double spacing, 160
dummy arguments, 56

—E—

EBCDIC, 202
editor, 5
END, 6, 56
evaluation, 14
EXP, 61
explicit definition, 11
exponentiation, 13

—F—

F specification, 163
FILE, 170
files, 169
FORMAT, 159, 209, 210
function, 56
function body, 56
functions, 55

—G—

GOTO, 97

—H—

Hardware, 2
header, 56
high level language, 3

—I—

I specification, 160
ICHAR, 207
IF, 36, 42
IF-ELSE, 35
IF-ELSEIF, 38
IF-THEN, 97
implicit definition, 11
Implied loops, 102
increment, 93
index, 93, 117, 205
initial, 93
inner loop, 95
input arguments, 63
input devices, 2
input statement, 22
INT, 61
INTEGER, 11
integer constant, 9
integer operator, 15
integer variable, 11
intrinsic function, 61
intrinsic functions, 205

—K—

keyboard, 2

—L—

L specification, 168
LEN, 206
LGT, 207
limit, 93
literal specification, 167
LLE, 207
LLT, 207
LOG, 61
LOG10, 61
LOGICAL, 12
logical constant, 10
logical expression, 19
Logical operations, 17
Logical variables, 12
loop, 91
loop body, 91

—M—

main program, 56, 94
maintainer, 1
memory, 2
microcomputers, 1
minicomputers, 1
mixed-mode operator, 15
MOD, 61
mouse, 2
multiplication, 13

—N—

N dimensional array, 208
natural language, 2
nested DO loops, 95
Nested implied loops, 103
Nested WHILE Loops, 99
new page, 160

—O—

one-dimensional array, 117
OPEN, 169, 171
order, 189
outer loop, 95
output arguments, 63
output buffer, 159
output devices, 2
output statements, 24

—P—

parameters, 56
parameters of DO loop, 93
Personal computers, 1
Index

power, 14
precedence. See priority
precedency, 14
PRINT, 24, 159
printer, 2
printing an array, 121
Printing Two-Dimensional Arrays, 145
priority, 14, 18, 19
program, 3, 5

—R—
READ, 22, 170
reading arrays, 119
REAL, 12, 61
real constant, 9
real operator, 15
real variable, 12
relational expression, 19
relational operators, 19
Repetition, 91
RETURN, 56, 63
REWIND, 172
right-justified, 160
row-wise, 142

—S—
scientific notation, 9
screen, 2
Searching, 189
Sequential search, 191
SIN, 61
single quote, 10
single spacing, 160
Software, 3
Sorting, 189
special characters, 11
SQRT, 61
statement, 5
statement function, 61
statement number, 105
step-wise refinement. See topdown design
STOP, 6
subprogram, 95, 125, 149
subprograms, 55, 103
subroutine, 63
subroutines, 55
substring, 204
subtraction, 13
successive refinement. See topdown design
swapping, 124

—T—
TAN, 61
termination condition, 91
three-dimensional array, 208
top down design, 55
top-down design, 4
triple spacing, 160
two-dimensional array, 141

—V—
variable name, 10
Variables, 10

—W—
WHILE, 91
WHILE loop, 96
WRITE, 171

—X—
X specification, 166

—Z—
zero-trip, 94