ForTran program

A ForTran (**For**mula **Tran**slation) program consists of a number of statements, each written in a separate line and represents a command to be performed by the computer.

It may have comment lines (written with a C) and blank lines. A statement must skip the first six columns of the line (cannot start before the 7th column), except for statement numbers. The program is not case sensitive (i.e., it can be written in upper or lower cases or a mixture of both) but has to finish with a statement END.

The following is a sample ForTran program written in upper case and mixed cases.

```
C******MAIN PROGRAM******
      REAL SUM

      READ(*,*)N
      SUM=0.
      DO 10 I=1,N
       IF(I.GE.N/2)THEN
        SUM=SUM+I**2
       ELSE
        SUM=SUM+I**4
       ENDIF
10    CONTINUE
      WRITE(*,*)SUM
      END
```

```
c******Main Program******
      Real sum

      read(*,*)n
      sum=0.
      do i=1,N
       If(i.ge.n/2)then
        Sum=sum+I**2
       else
        sum=sum+i**4
       endif
10    continue
      write(*,*)SUM
      end
```

Constants (Data whose values cannot be changed during the program)

Numeric Type (A string of digits, preceded by a single algebraic sign)

Integer: 12   0   –102   +127   (Not   12.0   –2.5   5,234   5–)

Real: 12.4   .1   –102.   +127.2   (Not   12   5,234.5)

3.27E-5   3.27E+10   3.269E8   -3.0E8

Character Type (A string of characters enclosed within apostrophes)

'CE-WRE (ECJ), Univ. of TX @ Austin'   'U. A. P.'   'Tom's cabin'

Logical Type (.TRUE.   .FALSE.)

Variables (Data whose values can be changed during the program)

Can be: SUM1XY   INSIDE   SUM OF SOME

(Not   1SUMXY   IN-SIDE $SUM.OF+SOME   READ   END)

Type Specification

```
INTEGER IX,JY,SUM
REAL DIV1,SECT2,IMULT
CHARACTER DAT,INS,OUTS        or CHARACTER*10 DAT,INS,OUTS*12,NAMS*3
LOGICAL TRUE,FALSE
```

IMPLICIT Statement

```
IMPLICIT REAL(A-H,O-Z)
IMPLICIT INTEGER*3(I-N)
```

Unless specified, variable names beginning with I-N are assumed integers and others real

Selected Functions

Intrinsic Functions	Meanings	Examples
IABS, ABS	Absolute value	IABS(–152) =152, ABS(–1.52) =1.52
MAX	Maximum	MAX(1.,3.3,–2.5) = 3.3
MIN	Minimum	MIN(1.,3.3,–2.5) =-2.5
SQRT	Square Root	SQRT(4.41) = $\sqrt{4.41}$ = 2.1
EXP	Exponential	EXP(1.35) = $e^{1.35}$= 3.857
LOG, LOG10 (ALOG, ALOG10)	Log, Log_{10}	LOG(3.857) = 1.35, LOG10(3.857) = 0.586
SIN, ASIN, SINH	Sin, Sin^{-1}, Sinh	SIN(1.) = 0.841, ASIN(1.) = $\pi/2$
COS, ACOS, COSH	Cos, Cos^{-1}, Cosh	COS(1.) = 0.540, ACOS(1.) = 0.
TAN, ATAN, TANH	Tan, Tan^{-1}, Tanh	TAN (1.) = 1.557, ATAN (1.) = $\pi/4$

Besides .EQ. , .NE., .GT., .GE., .LT., .LE. are used for $=,\neq,>,\geq,<,\leq$ respectively

Mathematical Operations

Mathematical Expression	ForTran Equivalent
$a + b/c - d$	A + B/C – D
$(a + b)/(c + d)$	(a + b)/(c + d)
$a^2 - b^2$	A**2 – B**2
$a/cd - b^2$	A/(C*D) – B**2
$\sqrt{(3a^2 - b/c^2)}$	SQRT(3*A**2 – B/C**2)
$\cos(2x+y) + \lvert a^2 - b^2 \rvert + e^{xy}$	COS(2*X+Y) + ABS(A**2 – B**2) + EXP(X*Y)
$\log_{10}(a + 3y)^3 + \log_e(x + y)$	ALOG10((A + 3*Y)**3)+LOG (X + Y)

Examples

ForTran Operation	Result
9 – 6 + 3	6
3**2 + 4/5	9
3**2 + 4.0/5	9.8
3 + 2.**3	11.0
(3 + 2)**3	125
(3 + 2**3)/5	2
(3 + 2.**3)/5	2.2
3**2**3	6561
(3**2)**3	729

Sequential Structure

Assignment Statement

It is used to assign values and is of the form

Variable = Expression

where the Variable may be integer, real, character or logical type while the Expression may be

(i) A constant (e.g., I = 2, or X = 2. 53, or INSIDE = 'PROG.IN')

(ii) Another variable to which a value has already been assigned

(e.g., I1 = 2….. I = I1, or X0 = 2.53……..X = X0, or INS = 'PROG.IN'……. INSIDE = INS)

(iii) A formula that the computer can evaluate (e.g., M = 1.0……..C = 3.0E7…… E = M*C**2)

It is essential that the Variable and the Expression should be of consistent type (e.g., both are numeric, character or logical type). Moreover, the Variable to be assigned a value should appear on the left side of the equal sign and a legal Expression should appear on the right side. The following are not valid assignment statements

102 = I

A*2 + 4.35 = 13.22

X = y = 3.5

Z = '5' + '4'

The symbol '=' is not to be interpreted as an 'equal to' sign, but rather 'is assigned'. It is not a statement of algebraic equality but is a replacement statement. For example, the statement

SUM = SUM + 100.

is algebraically incorrect if read as 'SUM is equal to SUM + 100.'. However, it is entirely consistent if read as 'SUM is assigned the value of SUM + 100.' or 'The value of SUM is to be replaced by the value of SUM + 100.'

List-directed Input Statement

READ*, v1, v2, v3,………………vn

where v1, v2, v3,………………vn are the n number of input data that has to be read by the program once entered in the screen. The variable names must be separated by commas in the program while the input data may be entered in the same line or in different lines in the screen.

List-directed Output Statement

PRINT*, v1, v2, v3,………………vn

where v1, v2, v3,………………vn are the n number of output data that has to be printed by the program in the screen. The variable names must be separated by commas in the program while the output data will appear in the same line in the screen.

Selective Structure

Logical Expressions and Operations

A logical expression can be a logical constant, logical variable, a relation or a combination of these. The logical operators used are .NOT., .AND., .OR., .EQV. and .NEQV. meaning the following

Logical Operators	Meanings	Examples
.NOT.	Negation	.NOT.TRUE. is .FALSE.
.AND.	Addition	.TRUE.AND.FALSE. is .FALSE.
.OR.	Alternative	.TRUE.OR.FALSE. is .TRUE.
. EQV.	Equivalence	FALSE.EQV.FALSE. is .TRUE.
.NEQV.	Non-Equivalence	FALSE.NEQV.FALSE. is .FALSE.

The logical operations have equal precedence and hence are operated from left to right, but parentheses may be used to override this order. For example, if A = 1.0, B = 2.5 and C = 6.5

(i) A.GT.B.AND.A*B.LE.C is FALSE.AND.TRUE; i.e., .FALSE.

(ii) A.GT.B.OR.A*B.LE.C is FALSE.OR.TRUE; i.e., .TRUE.

(iii) NOT.A.GT.B.AND.A*B.LE.C is TRUE.AND.TRUE; i.e., .TRUE.

(iv) A.GT.B.AND.(A.LE.C.OR.B.NE.C).OR.NOT.(A.EQ.B) is
FALSE.AND.(TRUE.OR.TRUE).OR.NOT.FALSE; or FALSE.AND.TRUE.OR.TRUE or
FALSE.OR.TRUE.; i.e., .TRUE.

GOTO Statement

The GOTO (or GO TO) statement is used to direct the computer program to a specific statement by branching around one or more statements. The most common GOTO statement has the general form

GOTO s [For example, GOTO 10]

where s is the statement number of an executable statement. In the computer program, the next statement to be executed after the above is statement number s. In the example, the statement 'GOTO 10' transfers the program to statement number 10.

Among other GOTO statements, the computed GOTO statement has the following form.

GOTO (n1, n2……….nk), integer expression [For example, GOTO (10, 32, 15, 20),I*J+2]

which shifts the program to statement number ni if the 'integer expression' is equal to i. If the 'integer expression' is not equal to any integer between 1 and k, the statement is not executed. In the example, the program shifts to statements 10, 32, 15 or 20 if (I*J+2) is equal to 1, 2, 3 or 4 respectively.

IF Statement

The IF statement shifts control conditionally in a program. The three types of IF statements are

1. Logical IF Statement

IF(logical expression) statement [For example, IF(I.GT.2.AND.I.LT.10) SUM=SUM+1]

The logical IF statement conditionally executes a ‘statement’ if a particular ‘logical expression’ is true. In the example, the value of SUM is replaced by SUM+1 if I is greater than 2 and less than 10.

2. Arithmetic IF Statement

IF(arithmetic expression) s1, s2, s3 [For example, IF(3*x+2) 10, 30, 25]

The arithmetic IF statement transfers control to s1, or s2 or s3 if the ‘arithmetic expression’ is less than, or equal to or greater than zero. The arithmetic IF statement is comparatively less used.

3. Block IF Statement

```
IF(logical expression) THEN          [For example, IF(I.GT.2.AND.I.LT.10) THEN
………………………………                                  SUM=SUM+1
……………………………..                                  PRINT*,SUM
ELSE                                                ELSE
………………………………                                  SUM=SUM+I
ENDIF                                               ENDIF                    ]
```

The block IF statement conditionally executes a number of statements if a particular ‘logical expression’ is true and another set of statements if it is false. In the example, if I is greater than 2 and less than 10, the value of the variable SUM is replaced by the value of SUM+1 and the value of SUM printed on screen. If the condition is not satisfied (i.e., I ≤ 2 or I ≥ 10) the value of SUM is replaced by SUM+I.

The following are some significant features of the block IF statement.

(i) The ELSE statement is not essential in the block IF statement; i.e., when there is no operation under ELSE, it may contain IF-THEN-ENDIF only; e.g., the following programs are equivalent.

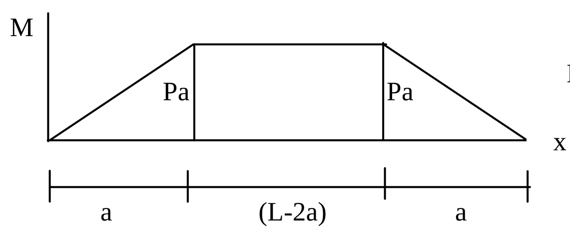
```
IF(I.GT.2.AND.I.LT.10) THEN                  IF(I.GT.2.AND.I.LT.10) THEN
 SUM=SUM+1                                    SUM=SUM+1
 PRINT*,SUM                                   PRINT*,SUM
ENDIF                                        ELSE
                                             ENDIF
```

(ii) The block IF statement may contain any executable statement, including several other block IF statements (used in the forms of nested block IF or ELSEIF).

```
IF(I.GT.2.AND.I.LT.10) THEN                  IF(I.GT.2.AND.I.LT.10) THEN
 SUM=SUM+1                                    SUM=SUM+1
 PRINT*,SUM                                   PRINT*,SUM
ELSE                                         ELSEIF(I.LE.2.OR.I.GE.10) THEN
 IF(I.LE.2.OR.I.GE.10) THEN                   SUM=SUM+I
  SUM=SUM+I                                  ELSE
 ENDIF                                       ENDIF
ENDIF
```

Class Assignments

1. Write a program that reads a number and writes on the screen if it is odd or even.
2. Use the block IF and GOTO statements to write a program that calculates the summations
   (i) $1 + x/1! + x^2/2! + x^3/3! + \ldots\ldots\ldots\ldots\ldots.. + x^n/n!$ (ii) $1 - x^2/2! + x^4/4! - \ldots\ldots\ldots\ldots\ldots.. + (-x^2)^n/2n!$
   (iii) $x - x^3/3! + x^5/5! - \ldots\ldots\ldots\ldots\ldots.. + (-1)^n x^{2n+1}/(2n+1)!$
   for given values of x and n.
3. Write a program that asks the user 4 basic mathematical questions (on screen), reads the answers (through screen), gives 25 points for each correct answer and prints the total points scored after 4 questions. The questions are as follows:
   (i) What is 119+87 ?, (ii) What is 83–65 ?, (iii) What is 13×9 ?, (iv) What is 133÷7 ?
4. Write a program that calculates the real roots of any quadratic equation $ax^2 + bx + c = 0$ for given values of a, b and c. The program should print a message on screen if the roots are imaginary and should also be able to solve the equation if a = 0.
5. The x and y components of a force are given by X and Y. Write a program that reads X and Y, calculates the magnitude $R = \sqrt{(X^2 + Y^2)}$ and angle $\theta = \tan^{-1}(Y/X)$ of the force. The program should be able to calculate the angle when $X \leq 0$.
6. The bending moment diagram of a beam is shown below. Write a program that reads real constants x, P, a, L, c, I, calculates and prints the bending moment and maximum bending stress for any value of x.



Maximum Bending stress = Mc/I

7. The preliminary selection criteria for different positions in a cricket team are as follows
   (i) Batsman: [25 ≤ Age ≤ 40 and Batting average ≥ 40 and Catching Reliability ≥ 70%]
   or [Batting average ≥ 50].
   (ii) Bowler: [20 ≤ Age ≤ 35 and Height ≥ 5.75 and Bowling average ≤ 30]
   or [Bowling average ≤ 25]
   (iii) Wicketkeeper: [25 ≤ Age ≤ 35 and Batting average ≥ 20 and Catching Reliability ≥ 80%].
   Write a program to read a player's qualifications and print if he qualifies for any position in the team.
8. The marks distribution of a course is: Attendance - 10%, Class Tests - 20%, Midterm Exam - 20% and Final Exam - 50%. The grades given are: A for ≥ 90%, B for ≥ 80% and < 90%, C for ≥ 70% and < 80%, D for ≥ 60% and < 70%, F for < 60%. If the teacher takes 40 classes and if Class Tests, Midterm and Final Exam have full marks of 100 each, write a program to calculate a student's grade.

Writing a ForTran90 program in Microsoft ForTran

Path to Microsoft ForTran

Start → Programs →

ForTran Power Station 4.0 → Microsoft Developer Studio

(Or just double-click on the Microsoft Developer Studio icon on Desktop)

To create a Workspace

1. Close ‘Tip of the Day’
2. File → New
3. Project Workspace → OK
4. Name (***) → Create

To create, type and save a file

1. File → New
2. Text File → OK
3. Tab (Adjust to 7th Column)
4. Type the program
5. Save → Name (***. for) → Save

To run a program

1. Build → Compile ***.for
2. Build → Build ***.exe
3. Build → Execute ***.exe

To close Microsoft ForTran

1. File → Close Workspace
2. File → Exit

Solution of Class Assignments

Problem #4

```
      PRINT*,'ENTER A,B,C'
      READ*,A,B,C
```

Using Logical IF

```
      IF(A.EQ.0)GOTO 3

      DET=B*B–4*A*C
      IF(DET<0)GOTO 6

      X1=(–B+SQRT(DET))/(2*A)
      X2=(–B–SQRT(DET))/(2*A)
      PRINT*,X1,X2
      GOTO 7
6     PRINT*,'ROOTS ARE IMAGINARY'
      GOTO 7
3     X=–C/B
      PRINT*,X
7     END
```

Using Block IF

```
IF(A.EQ.0)THEN
 X=–C/B
 PRINT*,X
ELSE
 DET=B*B–4*A*C
 IF(DET<0)THEN
  PRINT*,'ROOTS ARE IMAGINARY'
 ELSE
  X1=(–B+SQRT(DET))/(2*A)
  X2=(–B–SQRT(DET))/(2*A)
  PRINT*,X1,X2
 ENDIF
ENDIF
END
```

Problem #8

```
      REAL MT
      PRINT*,'ENTER AT,CT,MT,FE'
      READ*,AT,CT,MT,FE
      TOT=AT*10/40+CT*20/100+MT*20/100+FE*50/100
      PRINT*,'TOTAL IS',TOT
```

Using Logical IF

```
      IF(TOT>=90) PRINT*,'GRADE IS A'
      IF(TOT>=80.AND.TOT<90)PRINT*,'GRADE IS B'
      IF(TOT>=70.AND.TOT<80)PRINT*,'GRADE IS C'
      IF(TOT>=60.AND.TOT<70)PRINT*,'GRADE IS D'
      IF(TOT<60)PRINT*,'GRADE IS F'

      END
```

Using Block IF

```
IF(TOT>=90) THEN
 PRINT*,'GRADE IS A'
ELSE
 IF(TOT>=80)THEN
  PRINT*,'GRADE IS B'
 ELSE
  IF(TOT>=70)THEN
   PRINT*,'GRADE IS C'
  ELSE
   IF(TOT>=60)THEN
    PRINT*,'GRADE IS D'
   ELSE
    PRINT*,'GRADE IS F'
   ENDIF
  ENDIF
 ENDIF
 ENDIF
 END
```

Practice Problems

For the computer programs shown below, write the outputs in the screen.

```
(i)   X=1.
      Y=2.
      Z1=ASIN((2*X+Y)/20) + ABS(SQRT(X**2 –B**2))
      Z2=LOG (A+Y)–EXP(X*Y)
      PRINT*,Z1,Z2,A,B
      END
(ii)  X=2.
      CS=COS(2*X)
      SN=SIN(2*X)
      X1=MAX(ABS(SN),CS,–CS,0.5)
      X2=MIN(ABS(SN),CS,–CS,0.5)
      PRINT*,X1,X2
      END
(iii) INTEGER I,J
      I=2
      IE=2**I*3.
      C1=MOD(IE,2)+I/4
      C2=I/4.
      IF(C1*C2.GT.0)J=C1–C2
      IF(J)10,12,15
10    PRINT*,C1,C2,J
12    PRINT*,C2,C1,J
15    PRINT*,J
      END
(iv)  I=1
      N=3
      PROD=15.E–1
      PRINT*, I, N, PROD
      IF(I.GT.2.AND.I.LT.N.OR.I.EQ.4)THEN
       PROD=PROD*I
       I=I+2
       PRINT*, I, N, PROD
      ENDIF
      END
(v)   I=1
      N=4
10    IF(I.GE.1.AND.I.LT.N)THEN
       I=I+2
       GOTO 10
       SUM = SUM+ (–2)**I/I
      ENDIF
       PRINT*, I, N, SUM
      END
```

Repetitive Structure

DO loops

Although IF loops (combination of block IF and GOTO) can also perform repetitive operations, DO loops are used more often for this purpose.

1. DO loop or DO-CONTINUE loop

This statement is used to perform repetitive works. It has a standard form like

```
    DO s u= us,ut,ui
    ………………………………
    ……………………………..
  s CONTINUE
```

For example,

```
DO 11 X=4.,10.,0.2
      Y= X**2+ 20.*X
      PRINT*,X,Y
11 CONTINUE
```

where s (= 11) is a statement reference number, u (= X) is the DO control variable and

u_s,u_t,u_i (= 4.,10.,0.2) are values of u that control the DO loop.

u_s is the starting value, u_t the terminal value and u_i the incremental value of u.

The example mentioned above assigns values of X ranging between 4. and 10. (increment 0.2) and defines variable Y= X^2+20X. The PRINT statement in the next line writes both X and Y to the screen.

The DO-CONTINUE loop ensures that this is done for each value of X.

The following are some significant features of the DO-CONTINUE loop.

(i) The values of u_s,u_t and u_i can be integer, real, zero, positive, negative or assigned numbers. Moreover if u_i =1, it does not need to be written. The following are all valid forms of DO statement.

```
DO 5 I=1, 15, 2
DO 10 I=1, 5 (meaning DO 10 I=1, 5, 1)
DO 12 X = –10., –20., –5. (But DO 12 X = –10., –20., 5. is not valid because it never ends)
DO 10 X = A, A+20*B, B/2
```

(ii) It is not necessary to write CONTINUE at the end of a DO-loop; i.e., the example above can also be written as below. However, the CONTINUE statement helps to locate the end of the loop.

```
    DO 11 X=4.,10.,0.2
     Y= X**2+ 20.*X
 11 PRINT*,X,Y
```

(iii) The control variable cannot be changed within the loop; i.e., the following loop is not valid

```
    DO 10 I=0, 5
 10 I=25
```

(iv) Statements within the loop can be any valid operations, including more DO-CONTINUE loops.

2. DO-ENDDO loop

The DO-loop can also be written in the following general form, called DO-ENDDO loop. It avoids writing statement numbers, but otherwise its basic properties are similar to the DO-CONTINUE loop.

```
    DO u= us,ut,ui
    ………………………………
    ……………………………..
    ENDDO
```

For example,

```
DO X=4.,10.,0.2
     Y= X**2+ 20.*X
     PRINT*,X,Y
ENDDO
```

Class Assignments

1. Write a program that reads a number and writes on the screen if it is a prime number or not.
2. Write a program to obtain the greatest and the smallest of n integer numbers and print them on screen.
3. Write a program to generate the first n terms of the Fibonacci series (1, 2, 3, 5, 8, 13…….).
4. Use the DO loop to calculate the summations of the following series for given values of x and n

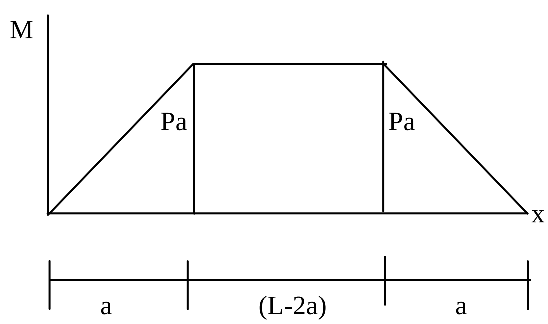
   (i) $4 – 4/3 + 4/5 – 4/7$…………….. n^{th} term,

   (ii) $1 + x/1! + x^2/2! + x^3/3! +$…………….. $+ x^n/n!$

   (iii) $1 – x^2/2! + x^4/4! –$…………….. $+ (–x^2)^n/2n!$

   (iv) $x – x^3/3! + x^5/5! –$…………….. $+ (–1)^n x^{2n+1}/(2n+1)!$

   Also compare the result with (i) the constant π, (ii) e^x, (iii) cos(x) and (iv) sin(x) respectively.
5. Write a program that reads a set of numbers x_1, x_2, x_3,………, x_n and calculates their arithmetic mean, geometric mean, standard deviation, skewness and kurtosis.
6. The Bending Moment Diagram of a beam is shown below. Write a program to read P, a, L and calculate bending moments at the interval of L/20.



Maximum Bending stress = Mc/I

7. Write a program to evaluate the integral $\int f(x)dx = \int[5 \tan^{-1}(x) – e^{–2 \cos x}]dx$ between x = 0 and x =1 using (i) the Trapezoidal rule and (ii) Simpson’s Rule.
8. Write a program to calculate the total score and percentage of grades obtained by each of n number of students in m class tests.

Practice Problems

For the computer programs shown below, write the outputs in the screen.

(i)
```
NITEM=3
DO I=1,NITEM
 COSTI=I*3.+20.
 CSUMI=CSUMI+COSTI
 PRINT*,I, CSUMI
 IF(I.EQ.NITEM)PRINT*, 'TOTAL COST = TK.', CSUMI
ENDDO
END
```

(ii)
```
    X=2.
    DO 10 I=1,4,2
     I0=I/2
     SUM=SUM+(–1)**I0*X**I/I
     PRINT*, I,I0,SUM
10  CONTINUE
    IF(I.EQ.4)PRINT*, I, SUM
    END
```

(iii)
```
H=0.5
DO X=1.,0.2,–H
 FUN=3*ATAN(X)+EXP(–X)
 IF(X.EQ.0.OR.X.EQ.1)THEN
  SUM=SUM+FUN/2
 ELSE
  SUM=SUM+FUN
 ENDIF
 PRINT*,FUN
ENDDO
AR=SUM*H
PRINT*,SUM,AR
END
```

(iv)
```
    DO 10 K=1,2
     SUM=0.
     DO 10 J=K,2
      SUM=SUM+K+J
      PRINT*, SUM
10  CONTINUE
     END
```

Lab Assignment 4

Write a program to calculate the height reached by a cricket ball after rebounding from ground. The concepts of impulse and momentum are used.

1. Read the height of fall h_1 and height of rebound h_2 [e.g., $h_1 = 30''$, $h_2 = 12''$]
2. Calculate coefficient of restitution $e = \sqrt{(h_2/h_1)}$
3. Read the height of delivery y_1, velocity u, angle of delivery α and gravitational acceleration g [e.g., $y_1 = 10'$, $u = 120'/\text{sec}$, $\alpha = 15°$, $g = 32.2'/\text{sec}^2$]
4. Calculate $S = u^2 \sin\alpha \cos\alpha/g$, and length of first bounce, $x_1 = -S + \sqrt{(S^2 + 2u^2 y_1 \cos^2\alpha / g)}$
5. Read the total length of pitch x [e.g., $x = 60'$]
6. (i) If $x_1 > x$, print 'Full-tossed'

   (ii) Otherwise calculate distance $x_2 = x - x_1$, vertical velocity of impact $v_y = \sqrt{(u^2 \sin^2\alpha + 2 g y_1)}$, horizontal velocity of impact $v_x = u \cos\alpha$, slope of impact $s_1 = v_y/v_x$

   Calculate height $y_2 = e\, x_2\, s_1 - g\, x_2^2/(2u^2 \cos^2\alpha)$, velocity $w = \sqrt{(v_x^2 + (ev_y)^2 - 2 g y_2)}$

   Print x_1, x_2, y_2 and w. Also, if (a) $y_2 < 1'$, print 'Yorker', (b) $1' < y_2 < 2'$, print 'Over-pitched', (c) $2' < y_2 < 3.5'$, print 'Good-length', (d) $3.5' < y_2$, print 'Short-pitched'

Home Assignment 4

1. Write a program to calculate the rebound velocities v_{A2}, v_{B2} and angles θ_{A2}, θ_{B2} of two smooth spheres (weighing W_A, W_B) colliding at velocities v_{A1}, v_{B1} and angles θ_{A1}, θ_{B1} with the line of impact, if the coefficient of restitution is e [Refer to Example 315 of your Analytic Mechanics book].
2. Write a program to perform the shear design of rectangular RC sections by the Working Stress Method using the following procedure. The units used for the calculations are kips, ksi, inch etc.
   1. Read material properties f'_c, f_s [e.g., $f'_c = 2.5$ ksi, $f_s = 18$ ksi] and replace f'_c by $f'_c/1000$
   2. Read sectional properties b, d, A_s [e.g., $b = 10''$, $d = 12.5''$, $A_s = 0.22$ in^2]
   3. Calculate $V_{c\,(max)} = 5\sqrt{f'_c}\, bd$
   4. Read design shear force V [e.g. V = 15 kips]
   5. (i) If $V > V_{c\,(max)}$, print 'Change the cross-section'

      (ii) Otherwise, calculate $V_c = 1.1\sqrt{f'_c}\, bd$, design spacing, $S_{(req)} = A_s f_s d/(V - V_c)$, $V_{c0} = 3\sqrt{f'_c}\, bd$

      (a) If $V > V_{c0}$, the stirrup spacing (S) should be the minimum of $S_{(req)}$, d/4, 12″, $A_s/0.0015b$

      (b) If $V < V_{c0}$, but $V > V_c$, S should be the minimum of $S_{(req)}$, d/2, 24″ $A_s/0.0015b$

      (c) If $V < V_c$, S should be the minimum of d/2, 24″ and $A_s/0.0015b$

      Print S, A_s

Practice Problems

1. For the computer programs shown below, write the outputs in the screen.

(i)

```
X1=1.
DO X=2,3
 DY=FUN(X)–FUN(X1)
 DX=X–X1
 SL=DY/DX
 PRINT*,X,DY,DX,SL
 X1=X
ENDDO
END

FUNCTION FUN(Z)
FUN=TAN(Z)–LOG(Z)+EXP(–Z)
END
```

(ii)

```
   DIMENSION F(10),X(10),Y(10),Z(10),RL(10)
   N=2
   DO I=2,N
    F(I)=10.
    Y(I)=2.+I
     Z(I)=1.
     RL(I)=SQRT(X(I)**2+Y(I)**2+Z(I)**2)
     FX=FX+F(I)*X(I)/RL(I)
     FY=FY+F(I)*Y(I)/RL(I)
     FZ=FZ+F(I)*Z(I)/RL(I)
   ENDDO
   R=SQRT(FX**2+FY**2+FZ**2)
   PRINT 10, R,FX,FY,FZ
10 FORMAT(2X,F6.2,3(2X,F6.3))
   END
```

2. The horizontal (x) and vertical (y) distances traveled by a cricket ball are given by

   $x = ut \cos \alpha$ ……….(1) $y = ut \sin \alpha - gt^2/2$ ……….(2)

   Write a FORTRAN program to do the following,

   (i) Read u, α, g and x from input file.

   (ii) Calculate t from equation (1), y from equation (2) and write t and y to the screen.

   (iii) Write (to output file) 'Big Six' if $y \geq 8$, 'Maybe Six' if $0 \leq y \leq 8$, 'Not Six' otherwise.

3. The following FORTRAN program is written to calculate the cumulative total price of 4 items, which cost Tk. 60.5, 13.25, 87.0 and 55.5 respectively. Complete the program with OPEN statements and also write the input and output files.

```
……………………….
……………………….
READ(2,*)NITEM

DO I=1,NITEM
 READ(2,*)COSTI

 CSUMI=CSUMI+COSTI

 IF(I.EQ.NITEM)THEN
  WRITE(1,*)'TOTAL COST IS = TK.', CSUMI
 ELSE
  WRITE(1,*)I, CSUMI
 ENDIF
ENDDO

END
```

File Processing

OPEN Statement

This statement is used to open files for input/output purposes, which helps to read inputs from files or write outputs to files rather than to screens. This allows better control and preservation of the data or results.

For example

(i) OPEN(1,FILE='FRAME2.IN',STATUS='OLD')

defines the file ‘FRAME2.IN’ as an ‘OLD’ (existing) file (to be used for data input), referred to by ‘1’ in the program.

(ii) OPEN(2,FILE='OUT',STATUS='NEW')

defines the file ‘OUT’ as a ‘NEW’ (to be newly created) file (for output of results), referred to by ‘2’ in the program.

Actually, the OPEN statements may have several other items, all of which are optional and assume default values if not mentioned.

CLOSE Statement

This statement is used to close the files that have been opened.

For example

```
CLOSE (10)
```

closes the file on unit 10 to be closed. Unit 10 can then be used for other purposes like reading other inputs or writing outputs. However, it is not always necessary to close a file that had been opened before. Therefore CLOSE statements do not necessarily follow OPEN statements.

File Input and Output Statements

The READ and PRINT statements mentioned before need to be modified in order to use the opened files for the purposes of input and output. There are several ways to read from input files and write to output files, but the following are the most common.

For example, to read the variable X from input file, the READ (input) statement may look like

```
READ(1,*) X
```

rather than READ*, X (for screen input). Here the number ‘1’ is the input file number mentioned in the OPEN statement before while* means that the input would be unformatted (not using the FORMAT specifications mentioned later).

To print the variables I and J to an output file, the WRITE (output) statement may be

```
WRITE(2,*) I, J
```

rather than PRINT*, I, J (for screen input). The number ‘2’ is the output file number mentioned in the OPEN statement before, and * means that the output would be unformatted.

Format-directed Input and Output

In many applications of ForTran , it is desirable for the user to have complete control over the way the input data is read or the results are printed out. This may be applicable when reading organized data from input files rather than from screens and are particularly important when writing to output files where an organized output is frequently required.

Input/Output Statements for Format-directed Input and Output

The format-directed input and output statements are similar to the file input and output statements mentioned before but have a statement number instead of the * sign mentioned before.

For example, to read the variable X from input file, the READ (input) statement may look like

```
READ(1,70) X
```

Here the number '1' is the input file number mentioned in the OPEN statement before while the number '70' refers to the FORMAT specification number to be used for the input.

To print the variables I and J to an output file, the WRITE (output) statement would be

```
WRITE(2,50) I, J
```

The number '2' is the output file number mentioned before and '50' refers to the FORMAT specification number to be used for the output.

As an alternative to their direct use in the READ and WRITE statements, file numbers can be integer variables whose values are assigned/calculated within the program before the corresponding statements.

Format Statements

The format specification statements referred from the READ and WRITE statements mentioned before can have several forms; e.g., I-format for integers, F, E-format for real data, A-format for characters, L-format for logical data, X-format for blanks, etc. Besides, there are format statements with G, H, apostrophe, slash and reusable formats.

All of them will have the following general form

```
s    FORMAT (Specification)
```

where s is the format statement number and 'Specification' is the way the input/output is to be arranged. The following specifications are shown for illustration.

(i) I-format (Iw) specification

Specification	Data	Output
I3	123	123
I5	123	øø123
I5	–1234	–1234
I3	–1234	***

(ii) F-format (Fw.d) specification

Specification	Data	Output
F10.3	123456.789	123456.789
F10.2	–56.789	øøøø–56.79
F10.7	56.789	56.7890000
F10.5	123456.789	**********

(iii) E-format (Ew.d) specification

Specification	Data	Output
E14.8	123456.789	0.12345679E+06
E10.2	–56.789	ø–0.57E+02
E10.5	56.789	**********
E15.5	0.00001234	øøøø0.12340E–04

(iv) Examples of some other specifications

10 FORMAT(1X,I3) keeps 1 blank column, 3 columns for an integer data

30 FORMAT(2X,F10.3) keeps 2 blank columns, 10 columns for a real data (with 3 after decimal)

20 FORMAT(10(2X,I5)) keeps 10 sets, each with 2 blank columns and 5 columns for an integer data

15 FORMAT(‘TOTAL=’,I4) writes TOTAL= then keeps 4 columns for an integer data

15 FORMAT(I4//I5) keeps 4 columns for an integer, skips 2 lines then keeps 5 columns for an integer

Subscripted Variables

Subscripted variables are used to represent/store/print groups of related data under common names. In ForTran, as in many other languages, such variables are represented by arrays (groups). Depending on the type of data, there are 1-dimensional or multi-dimensional arrays.

DIMENSION Statement

This statement is used to define variables as arrays of values. Depending on the type of statement, it can define vectors by 1-dimensional arrays and matrices by two-dimensional or multi-dimensional arrays. The following is the general form of DIMENSION statement

DIMENSION array name (m_1:n_1, m_2:n_2,……, m_k:n_k)

where each pair m_i:n_i is a pair of integer constants, specifying the range of values for the i^{th} dimension to be between m_i and n_i. If m_i is equal to 1, it need not be mentioned. The maximum allowable value of k is seven; i.e., an array can have a maximum of seven dimensions. For example

(i) DIMENSION X(–3:10) defines X as a 1-dimensional array of 14 subscripts between –3 and 10.

(ii) DIMENSION X(1:100) defines X as a 1-dimensional array of 100 subscripts between 1 and 100.

(iii) DIMENSION X(100) is the same as (ii); i.e., it defines X as a 1-dimensional array of 100 subscripts between 1 and 100.

(iv) DIMENSION X1(60),Y1(40) defines X1 and Y1 as 1-dimensional arrays of 60 and 40 subscripts (maximum) respectively.

(v) DIMENSION FFOR(60,6), RFOR(60,6) defines FFOR and RFOR as 2-dimensional arrays; i.e., matrices of size 60×6 (maximum).

Arrays can also be declared by type specifications; e.g., REAL IFR(60,6), JFR(60,6)

Input and Output of Arrays

(i) Using array name: For example

```
INTEGER x(2,3,2)
READ(8,*)x
```

requires the 3-dimensional array X(2,3,2) to be read from input file '8' in the following sequence

x(1,1,1) x(2,1,1) x(1,2,1) x(2,2,1) x(1,3,1) x(2,3,1) x(1,1,2) x(2,1,2) x(1,2,2) x(2,2,2) x(1,3,2) x(2,3,2)

(ii) Using DO-loop: It requires the data to be stored/printed one item per line.

(iii) Using Implied DO-loop: For example,

```
WRITE(5,*) ((X(I,J) , J=1,3), I=1,2)
```

requires the 2-dimensional array X(2,3) to be printed to output file '5' in the following sequence

X(1,1) X(1,2) X(1,3) X(2,1) X(2,2) X(2,3)

Class Assignments

1. The following are the Final Numbers obtained in various courses by different students in a class.

   (i) Read the Roll Numbers and the Final Numbers from an Input File,

   (ii) Add them to obtain the Total number for each student. If the full mark in each exam is 100, calculate the Percentage of Marks obtained by each student.

   (iii) Print the Roll Number, Final Numbers, Total Number and Percentage of Mark of each student to an Output File.

Roll No.	Number 1	Number 2	Number 3	Number 4	Number 5
1	88.5	80.0	72.3	88.5	82.8
2	76.2	61.7	72.4	89.1	47.2
3	32.0	43.4	50.4	70.5	35.4
4	90.5	87.0	70.7	100.0	77.3
5	100.0	90.3	75.6	97.3	87.6
6	55.9	57.8	43.0	75.2	55.2
7	60.7	67.4	46.3	70.3	64.3
8	40.0	50.7	41.20	60.0	48.0
9	36.5	40.0	23.9	60.1	53.5
10	56.7	65.0	45.4	66.7	57.2

[Hint: Use OPEN statements to open input and output files.

You can define 2 arrays IROLL(10) for Roll Numbers (integer) and FTN(10,7) (real numbers) for the Final and Total Numbers as well as Percentages. Use a DO-CONTINUE loop (for I=1,10) to add the Final Numbers for each student and print them to an output file.]

2. (i) Read the following matrices (A and B) from an Input File.

   (ii) Multiply them

   (iii) Write the product matrix C to an Output File.

$$A = \begin{pmatrix} 1.0 & 2.5 & 3.2 \\ 3.1 & -2.4 & 1.9 \end{pmatrix} \qquad B = \begin{pmatrix} 3.0 & 2.1 \\ 0.0 & -1.2 \\ 1.2 & 0.9 \end{pmatrix}$$

[Hint: If A is a ($L\times M$) matrix and B is a ($M\times N$) matrix, C will be a ($L\times N$) matrix.

$C(i,j) = \sum A(i,k)\, B(k,j)$; where $\sum$ is a summation for $k = 1$ to M]

Subprograms

FUNCTION and SUBROUTINE

These statements are used to perform certain operations outside the main program and are therefore called ‘Sub-programs’. They are useful in performing often repeated or widely used operations in large programs and in maintaining clarity and continuity of the ‘Main-programs’.

The FUNCTION statement defines a ‘function’ that can be used in the main program.

The following is an example of the FUNCTION statement. It defines a function PLUSSQ to calculate (A+B)**2 for various values of A and B and return the results to the main program.

```
C******MAIN PROGRAM************************************
      READ*,X,Y
      SUM1=PLUSSQ(X,Y)
      SUM2=PLUSSQ(X,–2.*Y)
      SUM3=PLUSSQ(SIN(X),Y**2)
      SUM4=PLUSSQ(X,EXP(–Y))
      SUM5=SUM1+SUM2+SUM3+SUM4
      PRINT*,SUM1,SUM2,SUM3,SUM4,SUM5
      END
C******FUNCTION PLUSSQ********************************
      FUNCTION PLUSSQ(A,B)
       PLUSSQ=(A+B)**2
      END
```

The SUBROUTINE statement does not represent a function, but performs operations that can help to define/redefine certain variables. It is invoked from the main program by the CALL statement.

An example of the SUBROUTINE statement is shown below. Here, matrix SK and vector P are read in the main program from input file FUNSUB.IN, the set of equations [SK]{X}= {P} are solved using the SUBROUTINE GAUSS and the solution vector {X} is written as {P} in the screen.

```
C******MAIN PROGRAM******************************************
      DIMENSION SK(990,990),P(990)
      OPEN(1,FILE='FUNSUB.IN',STATUS='OLD')

      READ(1,*)NDF
      DO I=1,NDF
        READ(1,*)(SK(I,J),J=1,NDF),P(I)
      ENDDO
      CALL GAUSS(SK,P,NDF)

      DO I=1,NDF
        PRINT*,P(I)
      ENDDO

      END
C******GAUSS ELIMINATION************************************
      SUBROUTINE GAUSS(AG,BG,N)
      DIMENSION AG(990,990),BG(990)
```

```
      N1=N–1
      DO 10 I=1,N1
       I1=I+1
        CG=1./AG(I,I)
       DO 11 KS=I1,N
        DG=AG(KS,I)*CG
        DO 12 J=I1,N
12       AG(KS,J)=AG(KS,J)– DG*AG(I,J)
11      BG(KS)=BG(KS)–DG*BG(I)
10    CONTINUE

      BG(N)=BG(N)/AG(N,N)
      DO 13 II=1,N1
       I=N–II
       I1=I+1
      SUM=0.
       DO 14 J=I1,N
14      SUM=SUM+AG(I,J)*BG(J)
13    BG(I)=(BG(I)–SUM)/AG(I,I)

      END
```

COMMON Statement

The COMMON statement is used to allocate common memories to the main program and one or more subprograms. Rather than writing the variables to be transferred by the SUBROUTINE in each CALL statement, the COMMON statement can allow such transfers without writing them in the CALL statements.

For example, SUBROUTINE GAUSS can also be written in the following form (blank COMMON)

```
………………………………..
COMMON SK(990,990),P(990),NDF
………………………………..
CALL GAUSS
………………………………..
SUBROUTINE GAUSS
COMMON AG(990,990),BG(990),N
………………………………..
```

or in the following form (named COMMON)

```
………………………………..
COMMON/SOLVER/SK(990,990),P(990),NDF
………………………………..
CALL GAUSS
………………………………..
SUBROUTINE GAUSS
COMMON/SOLVER/AG(990,990),BG(990),N
………………………………..
```

Lab Assignment 5

Write a program to implement corrections in the measured lengths and bearings of the sides of a polygon. Use the following algorithm to carry out the computations.

1. Read (from input file) the number of sides (N) of the polygon [e.g., 5].
2. Read (from input file) the measured lengths (L_i) and whole circle bearings in degrees (ϕ_{Di}) for the sides of the polygon

   [e.g., use (L_i, ϕ_{Di}) as (25, 0), (125, 60), (90, 105), (175, 180) and (225, 300)].
3. Calculate the x and y components L_{xi} (= L_i cosd ϕ_{Di}) and L_{yi} (= L_i sind ϕ_{Di}) of the sides.
4. Add the x and y components of the sides of the polygon; i.e., $S_x = \sum L_{xi}$, $S_y = \sum L_{yi}$. Also calculate the perimeter of the polygon $P = \sum L_i$.
5. Distribute the corrections $-S_x$ and $-S_y$ among the sides of the polygon in proportion to their lengths; i.e., $L_{xi} = L_{xi} - S_x (L_i/P)$, $L_{yi} = L_{yi} - S_y (L_i/P)$.
6. Calculate and print (in output file) the corrected x and y components (L_{xi}, L_{yi}) as well as corrected length and whole circle bearing is degrees (L_i, ϕ_{Di}) of each side of the polygon

   [Use $L_i = \sqrt{}(L_{xi}^2 + L_{yi}^2)$ and $\theta_i = \tan^{-1}(L_{yi}/L_{xi})$ with appropriate quadrant corrections].
7. Verify (manually) if the conditions $\sum L_{xi} = 0$, $\sum L_{yi} = 0$ are satisfied.

<u>Home Assignment</u>

1. Write a program to arrange N real numbers in descending order.
2. Write a subroutine to arrange the highest M numbers out of a given set of N numbers in descending order (note that $N \geq M$). Call the subroutine from main program and then add the best M out of N class test numbers for NS students. Call the subroutine again to arrange the total numbers (after adding) obtained by the students in descending order.

   [Use the class test numbers of the 10 students from the previous assignment. Choose the best 3 out of 5 class test numbers obtained by each student; i.e., NS = 10, N = 5, M = 3, and arrange the total numbers in descending order]

```
      DIMENSION RL(50),WCBD(50)
      OPEN(1,FILE='ASSGN51.IN',STATUS='OLD')
      OPEN(2,FILE='ASSGN51.OUT',STATUS='NEW')

      PI=4*ATAN(1.)
      READ(1,*)N
      DO I=1,N
       READ(1,*)RL(I),WCBD(I)
       TL=TL+RL(I)
       SX=SX+RL(I)*COSD(WCBD(I))
       SY=SY+RL(I)*SIND(WCBD(I))
      ENDDO

      DO I=1,N
       FXI=RL(I)*COSD(WCBD(I))–SX*RL(I)/TL
       FYI=RL(I)*SIND(WCBD(I))–SY*RL(I)/TL
       TLI=SQRT(FXI**2+FYI**2)
       ANID=ASIN(FYI/TLI)*180/PI
       IF(FXI.LT.0)THEN
        IF(FYI.GE.0)ANID=180–ANID
        IF(FYI.LT.0)ANID=180+ANID
       ENDIF
       WRITE(2,10)I,FXI,FYI,TLI,ANID
      ENDDO
 10   FORMAT(2X,I3,4(2X,F8.3))

      END
```

INPUT FILE:

```
5
25 0
125 60
90 105
175 180
225 300
```

****

```
      DIMENSION F(50),X(50),Y(50),Z(50),RL(50)
      OPEN(1,FILE='ASSGN52.IN',STATUS='OLD')
      OPEN(2,FILE='ASSGN52.OUT',STATUS='NEW')

      READ(1,*)N
      DO I=1,N
       READ(1,*)F(I),X(I),Y(I),Z(I)
       RL(I)=SQRT(X(I)**2+Y(I)**2+Z(I)**2)
       FX=FX+F(I)*X(I)/RL(I)
       FY=FY+F(I)*Y(I)/RL(I)
       FZ=FZ+F(I)*Z(I)/RL(I)
      ENDDO

      R=SQRT(FX**2+FY**2+FZ**2)
      CX=FX/R
      CY=FY/R
      CZ=FZ/R
      WRITE(2,10)R,CX,CY,CZ

      DO I=1,N
       FI=(CX*X(I)+CY*Y(I)+CZ*Z(I))*F(I)/RL(I)
       WRITE(2,11)I,FI
      ENDDO
 10   FORMAT(2X,F6.2,3(2X,F6.4))
 11   FORMAT(2X,I3,2X,F6.2)

      END
```

INPUT FILE:

```
4
10 –1 2 3
8 0 5 1
12 2 -2 4
7 5 3 0
```

Problem: Arrangement of the minimum NC of a set of NT numbers RNUMS in ascending order

```
      DIMENSION RNUMS(100),N(100)

      OPEN(1, FILE='ASCEND1.IN', STATUS='OLD')
      OPEN(2, FILE='ASCEND1.OUT', STATUS='OLD')

      READ(1,*)NT,NC
      READ(1,*)(RNUMS(I),I=1,NT)

      DO 20 J=1,NC
       WRST=100.
       DO I=1,NT
        IF(RNUMS(I).LT.WRST.AND.N(I).NE.1)THEN
         WRST=RNUMS(I)
         I0=I
        ENDIF
       ENDDO
       N(I0)=1
20    WRITE(2,6)WRST

6     FORMAT(1X,F10.2)

      END
```

INPUT FILE:

```
10    5
8.12    10.23    0.87    –5.75    12.64    –6.16    1.89    7.27    –3.00    4.45
```

```
      DIMENSION IROLL(50),FTN(50,10)
      OPEN(1,FILE='STUD.IN',STATUS='OLD')
      OPEN(2,FILE='STUD.OUT',STATUS='NEW')

      READ(1,*)NST,NCT

      DO 10 I=1,NST
       READ(1,*)IROLL(I),(FTN(I,J),J=1,NCT)
       DO 20 K=1,NCT
        FTN(I,NCT+1)=FTN(I,NCT+1)+FTN(I,K)
 20    CONTINUE
       FTN(I,NCT+2)=FTN(I,NCT+1)/5.
       WRITE(2,30)IROLL(I),FTN(I,NCT+1),FTN(I,NCT+2)
 10   CONTINUE

 30   FORMAT(2X,I3,2(2X,F6.2))
      END
```

****

```
      DIMENSION A(10,10),B(10,10),C(10,10)
      OPEN(1,FILE='MATMUL.IN',STATUS='OLD')
      OPEN(2,FILE='MATMUL.OT',STATUS='NEW')

      READ(1,*)L,M1,M2,N
      IF(M1.NE.M2)THEN
       PRINT*,'MATRICES CANNOT BE MULTIPLIED'
       GOTO 20
      ENDIF

      M=M1
      DO I=1,L
       READ(1,*)(A(I,J),J=1,M)
      ENDDO

      DO I=1,M
       READ(1,*)(B(I,J),J=1,N)
      ENDDO

      DO 10 I=1,L
       DO 10 J=1,N
        DO 10 K=1,M
         C(I,J)=C(I,J)+A(I,K)*B(K,J)
10    CONTINUE

      DO I=1,L
       WRITE(2,*)(C(I,J),J=1,N)
      ENDDO
20    END
```

INPUT FILE:

```
2 3 3 2
1.0   2.5  3.2
3.1  –2.4  1.9
3.0   2.1
0.0  –1.2
1.2   0.9
```

```
      DIMENSION RN(100,20),RTS(20),SUM(100)
      OPEN(1,FILE='DESCEND.IN',STATUS='OLD')
      OPEN(2,FILE='DESCEND.OUT',STATUS='NEW')

      READ(1,*)NS,N,M

      DO 10 I=1,NS
   10 READ(1,*)(RN(I,J),J=1,N)

      DO I=1,NS
       DO 15 K=1,N
   15  RTS(K)=RN(I,K)

       CALL DESCEND(RTS,N,M)

       PRINT *,(RTS(J),J=1,M)

       DO 20 J=1,M
   20   SUM(I)=SUM(I)+RTS(J)
      ENDDO

      CALL DESCEND(SUM,NS,NS)

      DO 30 I=1,NS
   30 PRINT *,SUM(I)

      END
C******SUBROUTINE DESCEND****************************
      SUBROUTINE DESCEND(RNUMS,NT,NC)
      DIMENSION RNUM(100),RNUMS(100),N(100)

      DO 10 I=1,NT
   10 N(I)=0

      DO 20 J=1,NC
       BST=0.
       DO I=1,NT
        IF(RNUMS(I).GT.BST.AND.N(I).NE.1)THEN
         BST=RNUMS(I)
         I0=I
        ENDIF
       ENDDO
       N(I0)=1
   20 RNUM(J)=BST

      DO 30 I=1,NC
   30 RNUMS(I)=RNUM(I)

      END

      INPUT FILE:
      4   5   3
      10.   5.   7.   3.   4.
      7.   8.   4.   6.   7.
      9.   5.   10.   3.   3.
      7.   6.   0.   6.   7.
```

Lab Assignment 6

Write a program to implement the multiplication of the matrices shown below using the following steps, which is the algorithm to convert member stiffness matrix and mass matrix of a two-dimensional truss from local axes to global axes ($\mathbf{K}^L$ to $\mathbf{K}^G$ and $\mathbf{M}^L$ to $\mathbf{M}^G$).

1. Read (from input file) the values of E, A, L, m and θ.
2. Calculate the matrices **T**, $\mathbf{K}^L$ and $\mathbf{M}^L$ shown below.

$$\mathbf{T} = \begin{pmatrix} C & S & 0 & 0 \\ -S & C & 0 & 0 \\ 0 & 0 & C & S \\ 0 & 0 & -S & C \end{pmatrix} \qquad \mathbf{K}^L = \begin{pmatrix} S_x & 0 & -S_x & 0 \\ 0 & 0 & 0 & 0 \\ -S_x & 0 & S_x & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \qquad \mathbf{M}^L = (mL/6)\begin{pmatrix} 2 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

where $C = \cos\theta$, and $S = \sin\theta$

$S_x = EA/L$

1. Call a subroutine for matrix multiplication to calculate the matrix $\mathbf{K}' = \mathbf{K}^L\, \mathbf{T}$.
2. Call the subroutine again to calculate and print (in the output file) the matrix $\mathbf{K}^G = \mathbf{T}^T\, \mathbf{K}'$, where $\mathbf{T}^T$ is the transpose of matrix **T**.
3. Call the subroutine again to calculate the matrix $\mathbf{M}' = \mathbf{T}^T\, \mathbf{M}^L$.
6. Call the subroutine again to calculate and print (in the output file) the matrix $\mathbf{M}^G = \mathbf{M}'\, \mathbf{T}$.

Home Assignment 6

1. Write a FORTRAN program and an input file to

   (i) Read N and the matrices **A**, **B** and **C** (of size N×N, N×N and N) using the implied DO-loop

   (ii) Add the diagonal elements of matrices **A** and **B**

   (iii) Perform the summations, $S1 = \sum A(i,i)\, B(i,i)$, $S2 = \sum A(i,i)\, C(i)$ and $S3 = \sum B(i,i)\, C(i)$

   (iv) Form the matrices $\mathbf{D} = \mathbf{A}^T$, $\mathbf{E} = \mathbf{B}^T$, $\mathbf{F} = \mathbf{C}^T$ and calculate $\mathbf{F}\times\mathbf{D}$ and $\mathbf{F}\times\mathbf{E}$

   (v) Calculate the product of matrices $\mathbf{A}\times\mathbf{B}\times\mathbf{C}$ and $\mathbf{B}\times\mathbf{A}\times\mathbf{C}$

   Use the following matrices as **A**, **B**, **C**

$$\mathbf{A} = \begin{pmatrix} 2.0 & -3.3 & 1.8 \\ 1.2 & 2.1 & -1.9 \\ 0.5 & 3.5 & 2.6 \end{pmatrix} \qquad \mathbf{B} = \begin{pmatrix} 0.2 & -3.3 & 8.1 \\ 2.1 & 1.2 & -9.1 \\ 5.0 & 5.3 & 6.2 \end{pmatrix} \qquad \mathbf{C} = \begin{Bmatrix} 4.5 \\ 0.0 \\ 3.2 \end{Bmatrix}$$

```
      DIMENSION T(4,4),TT(4,4)
      DIMENSION KL(4,4),K1(4,4),KG(4,4),ML(4,4),M1(4,4),MG(4,4)
      REAL M,L,M0,KL,K1,KG,ML,M1,MG

      OPEN(1,FILE='MATPROD.INP',STATUS='OLD')
      OPEN(2,FILE='MATPROD.OUT',STATUS='OLD')

      READ(1,*)E,A,L,M,TH

      C=COSD(TH)
      S=SIND(TH)
      DO 15 I=1,3,2
       T(I,I)=C
       T(I,I+1)=S
       T(I+1,I)= –S
 15   T(I+1,I+1)=C

      SX=E*A/L
      KL(1,1)=SX
      KL(1,3)= –SX
      KL(3,1)= –SX
      KL(3,3)=SX

      M0=M*L/6
      ML(1,1)=2*M0
      ML(1,3)=M0
      ML(3,1)=M0
      ML(3,3)=2*M0

      DO 10 I=1,4
       DO 10 J=1,4
 10   TT(I,J)=T(J,I)

      CALL MATMULP(KL,T,K1,4,4,4)
      CALL MATMULP(TT,K1,KG,4,4,4)

      CALL MATMULP(ML,T,M1,4,4,4)
      CALL MATMULP(TT,M1,MG,4,4,4)

      DO 20 I=1,4
 20   WRITE(2,6)(KG(I,J),J=1,4),(MG(I,J),J=1,4)

 6    FORMAT(4(1X,F10.3),4X,4(1X,F10.3))

      END
C*********************************************
      SUBROUTINE MATMULP(A,B,C,L,M,N)
      DIMENSION A(L,M),B(M,N),C(L,N)

      DO 10 I=1,L
       DO 10 J=1,M
        C(I,J)=0.
        DO 10 K=1,N
 10   C(I,J)=C(I,J)+A(I,K)*B(K,J)

      END
```