Practical Exercise 1: The Basics

Familliarisation with Editor and Operating System

Practical Exercise 2: Elements of Fortran 90

Question 1: Reformatting Code

The following program (which is available by anonymous ftp from `ftp.liv.ac.uk` in the directory `/pub/f90courses/progs`, filename `BasicReformatQuestion.f90`) has been badly laid out, Reformat it so its is neat and readable but performs exactly the same function,

```
PROGRAM MAIN;INTEGER::degreesfahrenheit&
  ,degreescentigrade;READ*,&
  degreesfahrenheit;degreescentigrade&
  =5*(degreesfahrenheit-32)/9;PRINT*,&
  degreesCENtiGrAde;END
```

Question 2: Declaration Format

Which of the following are incorrect declarations and why? (If you think a declaration may be correct in a given situation then say what the situation would be.)

1. `ReAl :: x`
2. `CHARACTER :: name`
3. `CHARACTER(LEN=10) :: name`
4. `REAL :: var-1`
5. `INTEGER :: 1a`
6. `BOOLEAN :: loji`
7. `DOUBLE :: X`
8. `CHARACTER(LEN=5) :: town = "Glasgow"`
9. `CHARACTER(LEN=*) :: town = "Glasgow"`
10. `CHARACTER(LEN=*), PARAMETER :: city = "Glasgow"`
11. `INTEGER :: pi = +22/7`
12. `LOGICAL :: wibble = .TRUE.`
13. `CHARACTER(LEN=*), PARAMETER :: "Bognor"`
14. `REAL, PARAMETER :: pye = 22.0/7.0`
15. `REAL :: two_pie = pye*2`
16. `REAL :: a = 1., b = 2`
17. `LOGICAL(LEN=12) :: frisnet`

18. `CHARACTER(LEN=6) :: you_know = 'y'know"`
19. `CHARACTER(LEN=6) :: you_know = "y'know"`
20. `INTEGER :: ia ib ic id`
    (in free format source form)
21. `DOUBLE PRECISION :: pattie = +1.0D0`
22. `DOUBLE PRECISION :: pattie = -1.0E-0`
23. `LOGICAL, DIMENSION(2) bool`
24. `REAL :: poie = 4.*atan(1.)`
25. declare the following objects,

Name	**Status**	**Type**	**Initial Value**
feet	variable	integer	-
miles	variable	real	-
town	variable	character ($\leq$ 20 letters)	-
home_town	constant	character	< *your home town* >
in_uk	constant	logical	< *is your home town in UK?* >
sin_half	constant	real	$\sin(0.5) = 0.47942554$

Question 3: Operator Precedence

Rewrite the following expression so that it contains the equivalent symbolic relational operators and then add parenthesis to indicate the order of evaluation,

```
.NOT.A.AND.B.EQV.C.OR.D.AND.E.OR.x.GT.y.AND.y.eq.z
```

Add parenthesis to this expression to indicate the order of evaluation,

```
-a*b-c/d**e/f+g**h+1-j/k
```

Question 4: Area Of a Circle

Write a simple program to read in the radius and calculate the area of the corresponding circle and volume of the sphere. Demonstrate correctness by calculating the area and volume using radii of 2, 5, 10 and -1.

Area of a circle,

$$\text{area} = \pi r^2 \tag{1}$$

Volume of a sphere,

$$\text{volume} = \frac{4\pi r^3}{3} \tag{2}$$

Hint 1: place the `READ`, the area calculation and the `PRINT` statement in a loop as follows. A program template (which is available by anonymous ftp from `ftp.liv.ac.uk` in the directory `/pub/f90courses/progs`, filename `BasicAreaOfCircleQuestion.f90`) is given below.

```
 PROGRAM Area
  DO
   PRINT*, "Type in the radius, a negative value will terminate"
   READ*, radius
   IF (radius .LT. 0) EXIT
    ... area calculation
    PRINT*, "Area of circle with radius ",&
            radius, " is ", area
    PRINT*, "Volume of sphere with radius ",&
            radius, " is ", volume
  END DO
  END PROGRAM Area
```

In this way when a negative radius is supplied the program will terminate.

Hint 2: use the value 3.14159 for π.

Practical Exercise 3: Control Structure and Intrinsics

Question 1: The 'Triangle Program'

Write a program to accept three (`INTEGER`) lengths and report back on whether these lengths could define an equilateral, isosoles or scalene triangle (3, 2 or 0 equal length sides) or whether they cannot form a triangle.

Demonstrate that the program works by classifying the following:

1. (1, 1, 1)
2. (2, 2, 1)
3. (1, 1, 0)
4. (3, 4, 5)
5. (3, 2, 1)
6. (1, 2, 4)

[Hint: If three lengths form a triangle then 2 times the longest side must be less than the sum of all three sides. In Fortran 90 terms, the following must be true:

```
(2*MAX(side1,side2,side3) < side1+side2+side3)
```

]

Question 2: Mathematical Magic

If you take a positive integer, halve it if it is even or triple it and add one if it is odd, and repeat, then the number will eventually become one.

Set up a loop containing a statement to read in a number (input terminated by zero) and to print out the sequence obtained from each input. The number 13 is considered to be very unlucky and if it is obtained *as part of the sequence* then execution should *immediately* terminate with an appropriate message.

Demonstrate that your program works by outputting the sequences generated by the following sets of numbers:

1. 7
2. 106, 46, 3, 0

Question 3: Decimal to Roman Numerals Conversion

Using a `SELECT CASE` block and integer division write a program that reads in a decimal number between 0 and 999 and prints out the equivalent in Roman Numerals.

Demonstrate that your program works with the numbers:

1. 888
2. 0
3. 222
4. 536

The output should contain no embedded spaces.

0					
1	*i*	1·	*x*	1··	*c*
2	*ii*	2·	*xx*	2··	*cc*
3	*iii*	3·	*xxx*	3··	*ccc*
4	*iv*	4·	*xl*	4··	*cd*
5	*v*	5·	*l*	5··	*d*
6	*vi*	6·	*lx*	6··	*dc*
7	*vii*	7·	*lxx*	7··	*dcc*
8	*viii*	8·	*lxxx*	8··	*dccc*
9	*ix*	9·	*xc*	9··	*cm*

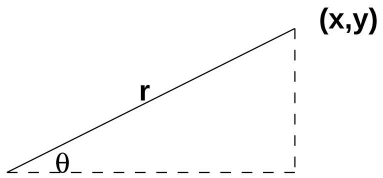
Hint: Use a `CHARACTER` string (or `CHARACTER` strings) to store the number before output. The 'longest' number is 888, *dccclxxxviii* (12 characters).

Question 4: Point on a circle

Write a program to read in a vector defined by a length, $\mathbf{r}$ and an angle, θ, in degrees which prints out the corresponding $(\mathbf{x}, \mathbf{y})$ co-ordinates. Recall that arguments to trigonometric functions are in radians.

Demonstrate correctness by giving the $(\mathbf{x}, \mathbf{y})$ co-ordinates for the following vectors

1. $\mathbf{r} = 12$, $\theta = 77°$
2. $\mathbf{r} = 1000$, $\theta = 0°$
3. $\mathbf{r} = 1000$, $\theta = 90°$
4. $\mathbf{r} = 20$, $\theta = 100°$
5. $\mathbf{r} = 12$, $\theta = 437°$



Hint: remember that

$$\sin\theta = \frac{y}{r}$$

and

$$\cos\theta = \frac{x}{r}$$

Extra Questions

Question 5: Concatenate Names

Write a program which accepts two names (Christian name and Family name, a maximum of 10 characters each) and outputs a single string containing the full name separated by one space with the first letter of each name in upper case and the rest of the name in lower case. You may assume that all inputs are valid names.

Hint a comes before `A` in the ASCII collating sequence.

(This sequence is given in the notes.)

Question 6: Quadratic equation solver

Write a program to read in values of a, b and c and calculate the real roots of the corresponding quadratic equation:

$$\mathbf{y} = a^2\mathbf{x} + b\mathbf{x} + c$$

Point out if the equation only has one or no real roots.

The program should repeatedly expect input; $a = 0$, $b = 0$ and $c = 0$ should be used to terminate.

Hint 1: recall that the solution of a general quadratic equation equation is:

$$\mathbf{x} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Hint 2: The program has a single root if

$$b^2 - 4ac = 0$$

two real roots if

$$b^2 - 4ac > 0$$

and imaginary roots if

$$b^2 - 4ac < 0$$

Practical Exercise 4: Array Processing

Question 1: Rank, Extents etc.

Give the rank, bounds, size and shape of the arrays defined as follows:

```
REAL, DIMENSION(1:10)  :: ONE
REAL, DIMENSION(2,0:2) :: TWO
INTEGER, DIMENSION(-1:1,3,2) :: THREE
REAL, DIMENSION(0:1,3) :: FOUR
```

Question 2: Conformance

Given

```
REAL, DIMENSION(1:10)  :: ONE
REAL, DIMENSION(2,0:2) :: TWO
INTEGER, DIMENSION(-1:1,3,2) :: THREE
REAL, DIMENSION(0:1,3) :: FOUR
```

Which two of the arrays are conformable?

Question 3: Array Element Ordering

Given

```
REAL, DIMENSION(1:10)  :: ONE
REAL, DIMENSION(2,0:2) :: TWO
INTEGER, DIMENSION(-1:1,3,2) :: THREE
REAL, DIMENSION(0:1,3) :: FOUR
```

Write down the array element order of each array.

Question 4: Array References

Given,

```
INTEGER :: i = 3, j = 7
REAL, DIMENSION(1:20) :: A
```

which of the following are valid array references for the array:

- ☐ `A(12)`

- ☐ `A(21)`
- ☐ `A(I)`
- ☐ `A(3.0)`
- ☐ `A(I*J)`
- ☐ `A(1+INT(4.0*ATAN(1.0)))`

[Hint: `4.0*ATAN(1.0)` is π]

Question 5: Hotel Array

Declare an array of rank 3 which might be suitable for representing a hotel with 8 floors and 16 rooms on each floor and two beds in each room. How would the second bed in the 5th room on floor 7 be referenced?

Question 6: Inquiry intrinsics etc.

Given,

```
INTEGER, DIMENSION(-1:1,3,2) :: A
```

Write a small program which contains intrinsic function calls to show:

1. the total number of elements in `A`,
2. the shape of `A`
3. the lower bound in dimension 2
4. the upper bound in dimension 3

Question 7: Array Constructor

Write an array constructor for the 5 element rank 1 array `BOXES` containing the values `1, 4, 6, 12, 23`.

Question 8: Array Sections

Declare an array which would be suitable for representing draughts board. Write a program to set all the white squares to zero and the black squares to unity. (A draughts board is 8×8 with alternate black and white squares)

Question 9: `MAXLOC`

What is the value of:

- ☐ `MAXLOC((/2,7,3,5,9,1/))`?
- ☐ `MAXVAL((/2,7,3,5,9,1/))`?
- ☐ `MINLOC((/2,7,3,5,9,1/))`?
- ☐ `MINVAL((/2,7,3,5,9,1/))`?

If

$$A = \begin{pmatrix} 0 & -5 & 8 & 3 \\ 3 & 4 & -1 & 2 \\ 1 & 5 & 6 & -4 \end{pmatrix}$$

what is

- ☐ `MAXLOC(A, MASK = A .LT. 5)`?
- ☐ `MAXVAL(A, MASK = A .LT. 5)`?
- ☐ `MAXLOC(A, MASK = A .LT. 4)`?
- ☐ `MAXVAL(A, MASK = A .LT. 4)`?

Question 10: `WHERE` Statement

Write a `WHERE` statement that will take a 2D `INTEGER` array and negate all odd-valued positive numbers.

Question 11: Array Masked Array Assignment

Using an array constructor and the `WHERE` statement, implement the following algorithm for finding prime numbers:

1. define a vector, `Prime`, of size n,
2. initialise `Prime` such that `Prime(i) = i` for `i = 1,n`
3. set `i = 2`
4. for all $j >$ `i`, ($j \in$ `(i+1:n)`) if `Prime(j)` is exactly divisible by `i` then set `Prime(j) = 0`, [hint: use the `MOD` (remainder) intrinsic in conjunction with a `WHERE` statement.]
5. increment `i`,
6. if `i` equals n then exit
7. if `Prime(i)` is zero then goto step 5
8. goto step 4

Print out all non-zero entries of the vector (the prime numbers).

Hint: the `WHERE` statement is an *array assignment statement* and **not** a control construct therefore it cannot contain a `PRINT` statement. The `PACK` intrinsic can accept an array argument and a conformable `MASK` and will return a 1D vector of all the elements of the array where the corresponding mask elements are `.TRUE.`.

```
Print*, PACK(Array,Mask)
```

Question 12: Random Number Generation

Using the intrinsic subroutine `RANDOM_NUMBER`, write a program to simulate the throw of a die.

Question 13: `MATMUL` Intrinsic

For the declarations

```
REAL, DIMENSION(100,100) :: A, B, C
```

what is the difference between `C=MATMUL(A,B)` and `C=A*B`.

Question 14: Operations on arrays and array intrinsics

Declare constant arrays `A` and `X` where:

$$\mathtt{A} = \begin{pmatrix} -4 & 5 & 9 \\ 6 & -7 & 8 \end{pmatrix}$$

$$\mathtt{X} = \begin{pmatrix} 1.5 \\ -1.9 \\ 1.7 \\ -1.2 \\ 0.3 \end{pmatrix}$$

1. Using the relevant intrinsics set `M` and `N` to be the extents of `A`,
2. Print the array `A` out row by row,
3. Write a Fortran 90 program which use intrinsics to print out the following:
   (a) The sum of the product of the columns of `A` (use intrinsics)
   (b) The product of the sum of the row elements of `A` (use intrinsics)
   (c) The sum of squares of the elements of `X`
   (d) The mean of the elements of `X`
   (e) The sum of the positive elements of `X`
   (f) The infinity norm of `X` i.e. the largest of $(|x_i|, i = 1, n)$
   (g) The one norm of `A` i.e. the largest column sum of $|a_{ij}|$

Question 15: Vector Subscripts / `MAXLOC`

Generate an arbitrary 1D array, `vector`, filled with random numbers between 0 and 1. By using the `MAXLOC` intrinsic with a suitable mask set up an integer array, `VSubs`, which contains a permuted index set of `vector` such that `Vector(VSubs(i)) > Vector(VSubs(i+1))` for all valid `i`.

You may find it useful to use the `MAXLOC` intrinsic. For example,

```
MAXLOC(VSubs)
```

returns an array containing the index of the largest element of the array `VSubs`, and,

```
MAXLOC(VSubs,MASK=VSubs.LT.VSubs(i))
```

returns the position of the largest element that is less than the value of `VSubs(i)`. In both cases the result is a one element 1D array. (The array contains one element because `VSub` only has one dimension.)

Extra Questions

Question 16: Summation Example

Which five consecutive numbers have the greatest sum:

```
6.3 7.6 9.2 3.4 5.6 7.23 9.76 6.83 5.45 4.56
4.86 5.8 6.4 7.43 7.87 8.6 9.25 8.9 8.4 7.23
```

Question 17: Salaries Example

The salaries received by employees of a company are

10500, 16140, 22300, 15960, 14150, 12180, 13230, 15760, 31000

and the position in the hierarchy of each employee is indicated by a corresponding category thus

1, 2, 3, 2, 1, 1, 1, 2, 3

Write a program to find the total cost to the company of increasing the salary of people in categories 1, 2 and 3 by 5%, 4% and 2% respectively.

Question 18: Travelling Salesman Problem

A salesman travels between 5 towns `A, B, C, D, E` whose distances apart are given in the following table:-

	A	B	C	D	E
A	0	120	180	202	300
B		0	175	340	404
C			0	98	56
D				0	168
E					0

Which is the shortest route which takes in all the towns.

Practical Exercise 5: Procedures 1

Question 1: Simple example of Subroutine

Write a main program and internal subroutine that returns, as its first argument, the sum of two real numbers.

Question 2: Simple example of a Function

Write a main program and an internal function that returns the sum of two real numbers supplied as arguments.

Question 3: Random Number Generation

Write a function which simulates a throw of two dice and returns the total score. Use the intrinsic subroutine `RANDOM_NUMBER` to obtain pseudo-random numbers to simulate the throw of the dice.

Question 4: Local Variables

At each of the indicated points in the code, give the status (local, dummy argument, host associated or undefined) and, if appropriate, the values of the variables `v1, v2, v3, v4, r` and `i`.

```
 PROGRAM PerOg
  IMPLICIT NONE
  REAL    :: V1,V2
  INTEGER :: V3,V4
   V1 = 1.0
   V2 = 2.0
   V3 = 3
   V4 = 4
    ...
!------ Position 1
    ...
   CALL Inte(V1,V3)
    ...
!------ Position 2
    ...
   CALL Exte(V1,V3)
    ...
!------ Position 3
    ...
 CONTAINS
  SUBROUTINE Inte(r,i)
   REAL, INTENT(INOUT) :: r
   INTEGER, INTENT(INOUT) :: i
   INTEGER :: v2 = 25
```

```
    ...
!------ Position 4
    ...
    r = 24.7
    i = 66
    v4 = 77
    ...
  END SUBROUTINE Inte
END PROGRAM PerOg
```

Question 5: Standard Deviation

Write a program which contains an internal function that returns the standard deviation from the mean of an array of real values. Note that if the mean of a sequence of values $(x_i, i = 1, n)$ is denoted by m then the standard deviation, s, is defined as:

$$s = \sqrt{\frac{\sum_{i=1}^{n}(x_i - m)^2}{n}}$$

[Hint: In Fortran 90 `SUM(X)` is the sum of the elements of `X`.]

To demonstrate correctness print out the standard deviation of the following numbers (10 of 'em):

```
5.0 3.0 17.0 -7.56 78.1 99.99 0.8 11.7 33.8 29.6
```

and also for the following 14,

```
1.0 2.0 3.0 4.0 5.0 6.0 7.0 8.0 9.0 10.0 11.0 12.0 13.0 14.0
```

Question 6: Save Attribute

Write a skeleton procedure that records how many times it has been called.

Question 7: Erroneous Code

What is wrong with the following internal procedure?

```
SUBROUTINE Mistaken(A,B,C)
 IMPLICIT NONE
 REAL, INTENT(IN) :: A
 REAL, INTENT(OUT) :: C
 A = 2*C
END SUBROUTINE Mistaken
```

Question 8: Keyword Arguments

Write a main program and an internal subroutine that returns, as its first argument, the sum of two real numbers. Invoke this using keyword arguments.

What are the important things to remember when using keyword arguments?

Question 9: Erroneous Code

What is wrong with the following internal procedure?

```
SUBROUTINE Mistaken(A,B,C)
 REAL, INTENT(INOUT) :: A
 REAL, INTENT(OUT), OPTIONAL :: B
 REAL, INTENT(IN) :: C
 A = 2*C*B
 IF( PRESENT(B) ) B = 2*A
END
```

Extra Questions

Question 10: Draw a Circle

Write an internal subroutine which draws circles. The routine takes two arguments, the first defines the radius of the circle and the second is optional and defines the colour by which the circle is shaded. The default colour for the circle shading should be green. Assume you have two library subroutines available, one called `Circle` which takes one `REAL` argument, (the radius,) and draws the circumference and one called `Shade_Circle` which shades it in and takes one character argument that defines the colour as a character string, e.g. "`R`" for red or "`G`" for green.

Practical Exercise 6: Procedures 2

Question 1: Types of Arrays

In the following internal subroutine, which are the automatic and which are the assumed shape arrays?

```
SUBROUTINE Array_Types(A,B,C,D)
 INTEGER, INTENT(IN) :: D
  REAL, DIMENSION(:,:) :: A,C
  REAL, DIMENSION(:)   :: B
  REAL, DIMENSION(SIZE(A)) :: E
  INTEGER, DIMENSION(1:D,1:D) :: F
 ...
```

Question 2: Triangular Numbers — Array Valued Function

The numbers 1, 3, 6, 10, 15, 21, ... are called triangular numbers because the number of units in each can be displayed as a triangular pyramid of blobs. The i^{th}, p_i is computed from $p_i = p_{i-1} + i$. Write an array valued function which takes one argument (N) and returns a vector of the first N triangular numbers.

Make sure to specify the argument INTENT.

Write a test program to demonstrate the function and print out the sequence where $N = 23$.

Question 3: Encapsulation

Define a module called Simple_Stats which contains encapsulated functions for calculating the mean and standard deviation of an arbitrary length REAL vector. The functions should have the following interfaces:

```
REAL FUNCTION mean(vec)
 REAL, INTENT(IN), DIMENSION(:) :: vec
END FUNCTION mean


REAL FUNCTION Std_Dev(vec)
 REAL, INTENT(IN), DIMENSION(:) :: vec
END FUNCTION Std_Dev
```

[Hint: In Fortran 90, SIZE(X) gives the number of elements in the array X.]

You may like to utilise your earlier code as a basis for this exercise.

Add some more code in the module which records how many times each statistical function is called during the lifetime of a program. Record these numbers in the variables: mean_use and std_dev_use.

Demonstrate the use of this module in a test program; in one execution of the program give the mean and standard deviation of the following sequences of numbers:

```
5.0 3.0 17.0 -7.56 78.1 99.99 0.8 11.7 33.8 29.6
```

and

```
1.0 2.0 3.0 4.0 5.0 6.0 7.0 8.0 9.0 10.0 11.0 12.0 13.0 14.0
```

plus two other sequences.

Print out the values of `mean_use` and `std_dev_use` for this run of the program.

Question 4: Complex Arithmetic — Modules

A complex number with solely integer components can be represented as a two element `INTEGER` array. Write a `MODULE` called `Integer_Complex_Arithmetic` which contains 4 `FUNCTION`s each accepting two integer complex number 'operands' and delivering the result of addition, subtraction, multiplication and division. Where appropriate Fortran 90 integer division rules should be followed. The following rules for complex arithmetic may be useful,

$$(a + b\mathbf{i}) + (x + y\mathbf{i}) = (a + x) + (b + y)\mathbf{i}$$

and,

$$(a + b\mathbf{i}) - (x + y\mathbf{i}) = (a - x) + (b - y)\mathbf{i}$$

and,

$$(a + b\mathbf{i}) \times (x + y\mathbf{i}) = (a \times x - b \times y) + (a \times y + b \times x)\mathbf{i}$$

and,

$$\frac{(a + b\mathbf{i})}{(x + y\mathbf{i})} = \frac{(a + b\mathbf{i}) \times (x - y\mathbf{i})}{x^2 + y^2}$$

Also write a procedure to accept one integer complex number and one non-negative integral exponent which delivers the result of `**`.

Note, for $n \geq 0$,

$$(a + b\mathbf{i})^n = 1 \times \underbrace{(a + b\mathbf{i}) \times \cdots \times (a + b\mathbf{i})}_{n}$$

Extra Questions

Question 5: Maggot/Onion Recursive Procedure Conundrum

An onion with an unknown number of layers contains a maggot at an unknown layer. Write a recursive function which will determine the depth of the maggot using the probability of 0.1 of a maggot being at a particular layer and a random number generator.

Question 6: Vector Multiplication — Array Valued Function

Write a function `Outer` that forms the outer product of two vectors. If A and B are vectors then the outer product is a matrix C such that $C_{ij} = A_i \times B_j$.

Write a test program which accepts two integers giving the size of the `A` and `B` vectors, uses the `RANDOM_NUMBER` function to assign values and then prints the outer product.

Question 7: Binary Cut

Write a module containing a function which returns the position of a particular number in an array of sorted integers. The function should employ the so-called "binary cut" method. This method proceeds by determining in which half the number is and then concentrating on that half. It is easily implemented using two indices to point at the low and high positions of the current area of interest. It is assumed that if there is more than one occurrence of the number then the one with the higher index will be chosen. This method is very efficient for very large arrays.

Algorithm,

- Let i and j be the indices of the low and high marks.
- Initially set $i = 1$ and $j = n$ (the number in the list)
- Assume k is the number we are trying to find
- `DO`
  - `IF`$(i \geq j)$`EXIT`
  - determine the half way point ihalf $= \frac{i+(j-i)}{2}$
  - `IF` k is above ihalf put $i =$ ihalf $+ 1$
  - Otherwise put $j =$ ihalf
- `END DO`
- j will now point at the number k

Practical Exercise 7: Derived Types and Pointers

Question 1: Pointers and Assignment

Given the following declarations,

```
IMPLICIT NONE
REAL, POINTER :: a, b(:), c(:,:)
REAL, TARGET  :: e, f(100), g(100,100), h(100,100,1)
REAL          :: k(100,100)
INTEGER, POINTER :: o(:,:)
INTEGER, TARGET  :: r(100), s(100,100)
INTEGER          :: v(100)
```

7 of the lines below are incorrect. Which are they and why are they wrong?

```
a => h(1,1,:)
b => h(20,:,1:1)
o => s
c => g(100,100)
a => k(20,20)
a => e
b => v
c => g; b => c(32,40:42)
e => f(20)
a => f(5)
c => g(100:100,100:100)
b => REAL(r)
```

Question 2: Complex arithmetic

Modify your existing Integer_Complex_Arithmetic module and define a type INTCOMPLEX which is able to represent a complex number with integer components.

```
TYPE(INTCOMPLEX)
 INTEGER x,y
END TYPE
```

Modify the module procedures etc. to accept this datatype instead of the previously implemented 2 element array.

Add two functions UPLUS and UMINUS to the module which each take one INTCOMPLEX argument such that:

```
UPLUS(ZI)  = +ZI
UMINUS(ZI) = -ZI
```

Write a test program to demonstrate.

Question 3: Sorting Using Pointers

Using derived types and pointers to minimise storage, write a program to set up a user-specified number of 5 element arrays, and fill them with random numbers between 0. and 1000.0 and then using your `SimpleStats` `MODULE` sort them firstly on the basis of their mean and secondly on the basis of their standard deviation. Print out the results of the sorting so that the array with the greatest mean is printed out first alongside the actual value of its mean (to demonstrate correctness) and the array with the smallest mean is printed out last. Do the same with the sorted list of standard deviations, viz:

```
Position 1. Mean = 452.0
            Array is 960.66666 .... 34.87311
Position 2. Mean = 122.5
            Array is 460.92653 .... 50.80036

Position 10. Mean = 781.77
            Array is 160.23456 .... 4.80713
```

and so on.

[**Hint:** Use an array of the following user defined type:

```
TYPE ARRAY_INFO
 REAL, DIMENSION(:), POINTER :: Array
 REAL                        :: ArrayMean
 REAL                        :: ArrayDev
END TYPE ARRAY_INFO
```

to hold the array, its mean and its standard deviation. You should create two arrays where each element can point to objects of type `ARRAY_INFO` these arrays can be used to represent the sorted mean and standard deviation lists.

You may find it useful to employ other data structures to aid the solution. Do not worry about implementing an efficient sorting mechanism.]

Extra Questions

Question 4: Arrays of Pointers

Rewrite the '`MAXLOC`' sorting example using an array of pointers instead of a vector subscript to hold the order of the numbers.

Practical Exercise 8: Modules

Question 1: Series and Parallel Resistance, Defined Operators

Define a module called LECCY_OPS containing two operators .PARALLEL. and .SERIES. which, given two default REAL resistance values as operands will deliver the resistance obtained by connecting them in parallel or series.

For series resistance:

$$R = R_1 + R_2$$

and for parallel:

$$\frac{1}{R} = \frac{1}{R_1} + \frac{1}{R_2}$$

Use the following test program (which is available by anonymous ftp from ftp.liv.ac.uk in the directory /pub/f90courses/progs, filename DefinedOperatorResistanceQuestion.f90 to demonstrate correctness of the module.

```
PROGRAM Testo
 USE Resistance
 Print*, "10.0 .SERIES. 50.0", 10.0 .SERIES. 50.0
 Print*, "10.0 .PARALLEL. 50.0", 10.0 .PARALLEL. 50.0
END PROGRAM Testo
```

Question 2: Complex Arithmetic — Generic Interfaces

Enhance your Integer_Complex_Arithmetic module by adding the overload sets for the following intrinsics:

- ☐ REAL — returns INTEGER result; the real part of a complex number. Eg, REAL(x,y) is x
- ☐ INT — returns INTEGER result; as above.
- ☐ AIMAG — returns INTEGER result; the imaginary part of a complex number.Eg, AIMAG(x,y) is y
- ☐ CONJG — returns INTCOMPLEX result; the conjugate of a complex number. Eg, CONJG(x,y) is (x,-y).
- ☐ ABS — returns REAL result; absolute value of a complex number. Eg, ABS(x,y) is SQRT(x*x+y*y).

so that they accept arguments of type INTCOMPLEX. (REAL and INT should have the same functionality, ie, return the first (non-imaginary) component of the number.)

Demonstrate that the whole module works by USEing it with the following test program (which is available by anonymous ftp from ftp.liv.ac.uk in the directory /pub/f90courses/progs, filename IntegerComplex2zProgram.f90).

```
PROGRAM Testo
 USE Integer_Complex_Arithmetic
  IMPLICIT NONE

   PRINT*, "REAL(3,4)"
   PRINT*, REAL(INTCOMPLEX(3,4))
   PRINT*, "INT(3,4)"
   PRINT*, INT(INTCOMPLEX(3,4))
   PRINT*, "AIMAG(3,4)"
   PRINT*, AIMAG(INTCOMPLEX(3,4))
   PRINT*, "CONJG(3,4)"
   PRINT*, CONJG(INTCOMPLEX(3,4))
   PRINT*, "ABS(3,4)"
   PRINT*, ABS(INTCOMPLEX(3,4))

END PROGRAM Testo
```

Question 3: Complex Arithmetic — Overloading Operators

Modify your `Integer_Complex_Arithmetic` module so that the procedures which perform the five basic arithmetic operations overload the intrinsic operators. Do the same with the procedures that perform unary plus and unary minus.

Question 4: Complex Arithmetic — Overloading the Assignment Operator

Modify your `Integer_Complex_Arithmetic` module so that the assignment operator is overloaded to allow objects of type `INTEGER` and `REAL` to be assigned to `INTCOMPLEX` objects. (`REAL` values should be *truncated* before use.)

Question 5: Complex Arithmetic — Accessibility control

Modify your `Integer_Complex_Arithmetic` module to take advantage of the accessibility statements.

Use accessibility statements to restrict visibility of the module procedures, and type structure. Write a constructor function and I / O subroutines for the `INTCOMPLEX` type which have the following interfaces:

- construction

```
TYPE(INTCOMPLEX) FUNCTION Setup_INTCOMPLEX(i1,i2)
 INTEGER, INTENT(IN) :: i1, i2
END FUNCTION Setup_INTCOMPLEX
```

- output,

```
SUBROUTINE Put_INTCOMPLEX(ic)
 TYPE(INTCOMPLEX), INTENT(IN) :: ic
END SUBROUTINE Put_INTCOMPLEX
```

- input

```
SUBROUTINE Get_INTCOMPLEX(ic)
 TYPE(INTCOMPLEX), INTENT(OUT) :: ic
END SUBROUTINE Get_INTCOMPLEX
```

Demonstrate that the whole module works by USEing it with the following test program (which is available by anonymous ftp from `ftp.liv.ac.uk` in the directory `/pub/f90courses/progs`, filename `IntegerComplex3cProgram.f90`).

```
PROGRAM Testo
 USE Integer_Complex_Arithmetic
 IMPLICIT NONE
 TYPE(INTCOMPLEX) :: var1, var2, ans

 var1 = 3
 PRINT*, "var1 = 3"
 CALL Put_INTCOMPLEX(var1)
 PRINT*, ""

 var1 = 6.0
 PRINT*, "var1 = 6.0"
 CALL Put_INTCOMPLEX(var1)
 PRINT*, ""

 var1 = 6.2
 PRINT*, "var1 = 6.2"
 CALL Put_INTCOMPLEX(var1)
 PRINT*, ""

 var1 = 6.6
 PRINT*, "var1 = 6.6"
 CALL Put_INTCOMPLEX(var1)
 PRINT*, ""

 var1 = Setup_INTCOMPLEX(1,2)
 var2 = Setup_INTCOMPLEX(3,4)

 PRINT*, "(1,2)+(3,4)"
 ans = var1 + var2
 CALL Put_INTCOMPLEX(ans)
 PRINT*, ""

 PRINT*, "(1,2)-(3,4)"
 ans = var1 - var2
 CALL Put_INTCOMPLEX(ans)
 PRINT*, ""

 PRINT*, "(1,2)/(3,4)"
 ans = var1 / var2
 CALL Put_INTCOMPLEX(ans)
 PRINT*, ""

 PRINT*, "(3,4)/(3,4)"
 ans = var2 / var2
```

```
     CALL Put_INTCOMPLEX(ans)
     PRINT*, ""

     PRINT*, "(3,4)/(1,2)"
     ans = var2 / var1
     CALL Put_INTCOMPLEX(ans)
     PRINT*, ""

     PRINT*, "(1,2)*(3,4)"
     ans = var1 * var2
     CALL Put_INTCOMPLEX(ans)
     PRINT*, ""

     PRINT*, "(1,2)**3"
     ans = var1 ** 3
     CALL Put_INTCOMPLEX(ans)
     PRINT*, ""

     PRINT*, "+(1,2)"
     ans = +var1
     CALL Put_INTCOMPLEX(ans)
     PRINT*, ""

     PRINT*, "-(1,2)"
     ans = -var1
     CALL Put_INTCOMPLEX(ans)
     PRINT*, ""

     PRINT*, "Type in the two INTCOMPLEX components"
     CALL Get_INTCOMPLEX(var1)
     PRINT*, ""
     PRINT*, "This is what was typed in"
     CALL Put_INTCOMPLEX(var1)
     PRINT*, ""

     PRINT*, "Your number/(3,4)"
     ans = var1 / var2
     CALL Put_INTCOMPLEX(ans)
     PRINT*, ""

! Intrinsics

     PRINT*, "REAL(3,4)"
     PRINT*, REAL(var2)
     PRINT*, ""

     PRINT*, "INT(3,4)"
     PRINT*, INT(var2)
     PRINT*, ""

     PRINT*, "AIMAG(3,4)"
     PRINT*, AIMAG(var2)
     PRINT*, ""
```

```
    PRINT*, "CONJG(3,4)"
    CALL Put_INTCOMPLEX(CONJG(var2))
    PRINT*, ""

    PRINT*, "ABS(3,4)"
    PRINT*, ABS(var2)

  END PROGRAM Testo
```

Extra Questions

Question 6: Defined Types and Pointers

Write a program to implement a tree sort. The program should read in textual names from standard input and place each name in a binary tree. The algorithm is

1. read in a name
2. IF name = 'end' EXIT
3. add the name to the tree
4. print the tree

Steps 3 and 4 should be implemented with recursive procedures.

The algorithm for Step 3 is for each node:

1. if the node is null, add the name to the node and return
2. if the name is earlier in the alphabet than the node name
   - recursively go down the left branch

   otherwise
   - recursively go down the right branch

The algorithm for Step 4 is:

1. if the node is null return
2. recursively print the left branch
3. print the node name
4. recursively print the right branch

Practical Exercise 9: Advanced Fortran 90

Question 1: Complex Arithmetic — Mixing Types

Enhance your `Integer_Complex_Arithmetic` module by adding the overload sets for operations between `REAL` and `INTCOMPLEX` operands and `INTEGER` and `INTCOMPLEX` operands, these should be implemented for `+`, `-`, (dyadic) `*` and `/`. Clearly the result of an operation between a `REAL` and `INTCOMPLEX` expression and an `INTEGER` and `INTCOMPLEX` expression should be of type `INTCOMPLEX`. (`REAL` numbers should be *truncated* before use.)

Recall that if a is integer or real valued then,

$$a + (x + y\mathbf{i}) = (a + x) + y\mathbf{i}$$

and,

$$a - (x + y\mathbf{i}) = (a - x) - y\mathbf{i}$$

and,

$$a \times (x + y\mathbf{i}) = (a \times x) + (a \times y)\mathbf{i}$$

and,

$$\frac{a}{(x + y\mathbf{i})} = \left(\frac{a \times x}{x^2 + y^2}\right) - \left(\frac{a \times y}{x^2 + y^2}\right)\mathbf{i}$$

so

$$0.99 \times (3 + 4\mathbf{i}) = 0 + 0\mathbf{i}$$

and

$$1.0 \times (3 + 4\mathbf{i}) = 3 + 4\mathbf{i}$$

Demonstrate that the whole module works by `USE`ing it in the following test program (which is available by anonymous ftp from `ftp.liv.ac.uk` in the directory `/pub/f90courses/progs`, filename `IntegerComplex5Program.f90`).

```
PROGRAM Testo
 USE Integer_Complex_Arithmetic
 IMPLICIT NONE
 TYPE(INTCOMPLEX) :: var1, var2, ans
```

```
var1 = 3
PRINT*, "var1 = 3"
CALL Put_INTCOMPLEX(var1)

var1 = 5.99
PRINT*, "var1 = 5.99"
CALL Put_INTCOMPLEX(var1)

var1 = 6.01
PRINT*, "var1 = 6.01"
CALL Put_INTCOMPLEX(var1)

var1 = Setup_INTCOMPLEX(1,2)
var2 = Setup_INTCOMPLEX(3,4)

PRINT*, "(1,2)+(3,4)"
ans = var1 + var2
CALL Put_INTCOMPLEX(ans)

PRINT*, "(1,2)-(3,4)"
ans = var1 - var2
CALL Put_INTCOMPLEX(ans)

PRINT*, "(1,2)/(3,4)"
ans = var1 / var2
CALL Put_INTCOMPLEX(ans)

PRINT*, "(3,4)/(3,4)"
ans = var2 / var2
CALL Put_INTCOMPLEX(ans)

PRINT*, "(3,4)/(1,2)"
ans = var2 / var1
CALL Put_INTCOMPLEX(ans)

PRINT*, "(1,2)*(3,4)"
ans = var1 * var2
CALL Put_INTCOMPLEX(ans)

PRINT*, "(1,2)**3"
ans = var1 ** 3
CALL Put_INTCOMPLEX(ans)

PRINT*, "+(1,2)"
ans = +var1
CALL Put_INTCOMPLEX(ans)

PRINT*, "-(1,2)"
ans = -var1
CALL Put_INTCOMPLEX(ans)

PRINT*, "Type in the two INTCOMPLEX components"
CALL Get_INTCOMPLEX(var1)
```

```
    PRINT*, "This is what was typed in"
    CALL Put_INTCOMPLEX(var1)

    PRINT*, "Your number/(3,4)"
    ans = var1 / var2
    CALL Put_INTCOMPLEX(ans)

! Intrinsics

    PRINT*, "REAL(3,4)"
    PRINT*, REAL(var2)
    PRINT*, "INT(3,4)"
    PRINT*, INT(var2)
    PRINT*, "AIMAG(3,4)"
    PRINT*, AIMAG(var2)
    PRINT*, "CONJG(3,4)"
    CALL Put_INTCOMPLEX(CONJG(var2))
    PRINT*, "ABS(3,4)"
    PRINT*, ABS(var2)

! REAL | INTEGER .OP. INTCOMPLEX

    PRINT*, "2+(3,4)"
    ans =  2 + var2
    CALL Put_INTCOMPLEX(ans)

    PRINT*, "2-(3,4)"
    ans =  2 - var2
    CALL Put_INTCOMPLEX(ans)

    PRINT*, "2*(3,4)"
    ans =  2 * var2
    CALL Put_INTCOMPLEX(ans)

    PRINT*, "2/(3,4)"
    ans =  2 /var2
    CALL Put_INTCOMPLEX(ans)

    var1 = Setup_INTCOMPLEX(1,2)
    PRINT*, "4/(1,2)"
    ans =  4/var1
    CALL Put_INTCOMPLEX(ans)

    PRINT*, "2.5+(3,4)"
    ans =  2.5 + var2
    CALL Put_INTCOMPLEX(ans)

    PRINT*, "2.5-(3,4)"
    ans =  2.5 - var2
    CALL Put_INTCOMPLEX(ans)

    PRINT*, "2.5*(3,4)"
    ans =  2 * var2
    CALL Put_INTCOMPLEX(ans)
```

```
    PRINT*, "2.5/(3,4)"
    ans =  2.5 /var2
    CALL Put_INTCOMPLEX(ans)

    PRINT*, "4.7/(1,2)"
    ans =  4.7/var1
    CALL Put_INTCOMPLEX(ans)


    PRINT*, "-2.5+(3,4)"
    ans =  -2.5 + var2
    CALL Put_INTCOMPLEX(ans)

    PRINT*, "-2.5-(3,4)"
    ans =  -2.5 - var2
    CALL Put_INTCOMPLEX(ans)

    PRINT*, "-2.5*(3,4)"
    ans =  -2.5 * var2
    CALL Put_INTCOMPLEX(ans)

    PRINT*, "-2.5/(3,4)"
    ans =  -2.5 /var2
    CALL Put_INTCOMPLEX(ans)

    PRINT*, "-4.7/(1,2)"
    ans =  -4.7/var1
    CALL Put_INTCOMPLEX(ans)

! INTCOMPLEX .OP. INTEGER | REAL

    PRINT*, "(3,4)+2"
    ans =  var2 + 2
    CALL Put_INTCOMPLEX(ans)

    PRINT*, "3,4)-2"
    ans =  var2-2
    CALL Put_INTCOMPLEX(ans)

    PRINT*, "(3,4) * 2"
    ans =  var2 * 2
    CALL Put_INTCOMPLEX(ans)

    PRINT*, "(3,4)/2"
    ans =  var2/2
    CALL Put_INTCOMPLEX(ans)

    PRINT*, "(1,2)/4"
    ans =  var1/4
    CALL Put_INTCOMPLEX(ans)

    PRINT*, "(3,4)+2.5"
    ans =  var2+2.5
```

```
     CALL Put_INTCOMPLEX(ans)

     PRINT*, "(3,4)-2.5"
     ans =  var2-2.5
     CALL Put_INTCOMPLEX(ans)

     PRINT*, "(3,4)*2.5"
     ans =   var2*2.5
     CALL Put_INTCOMPLEX(ans)

     PRINT*, "(3,4)/2.5"
     ans =  var2/2.5
     CALL Put_INTCOMPLEX(ans)

     PRINT*, "(1,2)/4.7"
     ans =  var1/4.7
     CALL Put_INTCOMPLEX(ans)

     PRINT*, "-(3,4)-2.5"
     ans =   -var2-2.5
     CALL Put_INTCOMPLEX(ans)

     PRINT*, "(3,4)*(-2.5)"
     ans =  var2*(-2.5)
     CALL Put_INTCOMPLEX(ans)

     PRINT*, "0.99 * (3,4)"
     ans = 0.99 *var2
     CALL Put_INTCOMPLEX(ans)

     PRINT*, "1.0 * (3,4)"
     ans = 1.0 *var2
     CALL Put_INTCOMPLEX(ans)

    END PROGRAM Testo
```

Question 2: Kind Functions and Available Representations

Write a program which quizzes a processor and prints out the available kinds for the INTEGER data type. For each available kind, give the maximum exponent range that is supported. Also give the kind number corresponding to the default INTEGER. If an exponent is too large to be supported then SELECTED_INT_KIND will return -1. (You can assume that no number greater than 10^{32} will be supported.)

Question 3: Different precision INTCOMPLEX

Enhance your Integer_Complex_Arithmetic module by converting the code to a portable system which can represent INTCOMPLEX values, say in the range $(-10^9, 10^9)$.

Expressions containing INTCOMPLEXs may involve integers of two kinds of INTEGERs, (short_int and long_int which correspond to ranges of $(-10^4, 10^4)$ and $(-10^9, 10^9)$ respectively) so the overload set for numeric operators must be extended. There is still only one kind of REAL and COMPLEX data type.

Modify your program so that `Setup_INTCOMPLEX` is a generic function which constructs values for all combinations of `short_int` and `long_int`.

Demonstrate that the whole module works by `USE`ing it in the following test program (which is available by anonymous ftp from `ftp.liv.ac.uk` in the directory `/pub/f90courses/progs`, filename `IntegerComplex6Program.f90`.

```
PROGRAM Testo
 USE Integer_Complex_Arithmetic
 IMPLICIT NONE
 TYPE(INTCOMPLEX) :: var1, var2, ans

 var1 = 3
 PRINT*, "var1 = 3"
 CALL Put_INTCOMPLEX(var1)

 var1 = 5.99
 PRINT*, "var1 = 5.99"
 CALL Put_INTCOMPLEX(var1)

 var1 = 6.01
 PRINT*, "var1 = 6.01"
 CALL Put_INTCOMPLEX(var1)

 var1 = Setup_INTCOMPLEX(1,2)
 var2 = Setup_INTCOMPLEX(3,4)

 PRINT*, "(1,2)+(3,4)"
 ans = var1 + var2
 CALL Put_INTCOMPLEX(ans)

 PRINT*, "(1,2)-(3,4)"
 ans = var1 - var2
 CALL Put_INTCOMPLEX(ans)

 PRINT*, "(1,2)/(3,4)"
 ans = var1 / var2
 CALL Put_INTCOMPLEX(ans)

 PRINT*, "(3,4)/(3,4)"
 ans = var2 / var2
 CALL Put_INTCOMPLEX(ans)

 PRINT*, "(3,4)/(1,2)"
 ans = var2 / var1
 CALL Put_INTCOMPLEX(ans)

 PRINT*, "(1,2)*(3,4)"
 ans = var1 * var2
 CALL Put_INTCOMPLEX(ans)

 PRINT*, "(1,2)**3"
```

```
     ans = var1 ** 3
     CALL Put_INTCOMPLEX(ans)

     PRINT*, "+(1,2)"
     ans = +var1
     CALL Put_INTCOMPLEX(ans)

     PRINT*, "-(1,2)"
     ans = -var1
     CALL Put_INTCOMPLEX(ans)

     PRINT*, "Type in the two INTCOMPLEX components"
     CALL Get_INTCOMPLEX(var1)
     PRINT*, "This is what was typed in"
     CALL Put_INTCOMPLEX(var1)

     PRINT*, "Your number/(3,4)"
     ans = var1 / var2
     CALL Put_INTCOMPLEX(ans)

! Intrinsics

     PRINT*, "REAL(3,4)"
     PRINT*, REAL(var2)
     PRINT*, "INT(3,4)"
     PRINT*, INT(var2)
     PRINT*, "AIMAG(3,4)"
     PRINT*, AIMAG(var2)
     PRINT*, "CONJG(3,4)"
     CALL Put_INTCOMPLEX(CONJG(var2))
     PRINT*, "ABS(3,4)"
     PRINT*, ABS(var2)

! REAL | INTEGER .OP. INTCOMPLEX

     PRINT*, "2+(3,4)"
     ans =  2 + var2
     CALL Put_INTCOMPLEX(ans)

     PRINT*, "2-(3,4)"
     ans =  2 - var2
     CALL Put_INTCOMPLEX(ans)

     PRINT*, "2*(3,4)"
     ans =  2 * var2
     CALL Put_INTCOMPLEX(ans)

     PRINT*, "2/(3,4)"
     ans =  2 /var2
     CALL Put_INTCOMPLEX(ans)

     var1 = Setup_INTCOMPLEX(1,2)
     PRINT*, "4/(1,2)"
     ans =  4/var1
```

```
    CALL Put_INTCOMPLEX(ans)

    PRINT*, "2.5+(3,4)"
    ans =  2.5 + var2
    CALL Put_INTCOMPLEX(ans)

    PRINT*, "2.5-(3,4)"
    ans =  2.5 - var2
    CALL Put_INTCOMPLEX(ans)

    PRINT*, "2.5*(3,4)"
    ans =  2 * var2
    CALL Put_INTCOMPLEX(ans)

    PRINT*, "2.5/(3,4)"
    ans =  2.5 /var2
    CALL Put_INTCOMPLEX(ans)

    PRINT*, "4.7/(1,2)"
    ans =  4.7/var1
    CALL Put_INTCOMPLEX(ans)


    PRINT*, "-2.5+(3,4)"
    ans =  -2.5 + var2
    CALL Put_INTCOMPLEX(ans)

    PRINT*, "-2.5-(3,4)"
    ans =  -2.5 - var2
    CALL Put_INTCOMPLEX(ans)

    PRINT*, "-2.5*(3,4)"
    ans =  -2.5 * var2
    CALL Put_INTCOMPLEX(ans)

    PRINT*, "-2.5/(3,4)"
    ans =  -2.5 /var2
    CALL Put_INTCOMPLEX(ans)

    PRINT*, "-4.7/(1,2)"
    ans =  -4.7/var1
    CALL Put_INTCOMPLEX(ans)

! INTCOMPLEX .OP. INTEGER | REAL

    PRINT*, "(3,4)+2"
    ans =  var2 + 2
    CALL Put_INTCOMPLEX(ans)

    PRINT*, "3,4)-2"
    ans =  var2-2
    CALL Put_INTCOMPLEX(ans)

    PRINT*, "(3,4) * 2"
```

```
    ans =  var2 * 2
    CALL Put_INTCOMPLEX(ans)

    PRINT*, "(3,4)/2"
    ans =  var2/2
    CALL Put_INTCOMPLEX(ans)

    PRINT*, "(1,2)/4"
    ans =  var1/4
    CALL Put_INTCOMPLEX(ans)

    PRINT*, "(3,4)+2.5"
    ans =  var2+2.5
    CALL Put_INTCOMPLEX(ans)

    PRINT*, "(3,4)-2.5"
    ans =  var2-2.5
    CALL Put_INTCOMPLEX(ans)

    PRINT*, "(3,4)*2.5"
    ans =   var2*2.5
    CALL Put_INTCOMPLEX(ans)

    PRINT*, "(3,4)/2.5"
    ans =  var2/2.5
    CALL Put_INTCOMPLEX(ans)

    PRINT*, "(1,2)/4.7"
    ans =  var1/4.7
    CALL Put_INTCOMPLEX(ans)

    PRINT*, "-(3,4)-2.5"
    ans =   -var2-2.5
    CALL Put_INTCOMPLEX(ans)

    PRINT*, "(3,4)*(-2.5)"
    ans =  var2*(-2.5)
    CALL Put_INTCOMPLEX(ans)

    PRINT*, "0.99 * (3,4)"
    ans = 0.99 *var2
    CALL Put_INTCOMPLEX(ans)

    PRINT*, "1.0 * (3,4)"
    ans = 1.0 *var2
    CALL Put_INTCOMPLEX(ans)

! Extended precision

    var1 = Setup_INTCOMPLEX(3_short_int,1_long_int)
    PRINT*, "var1 = Setup_INTCOMPLEX(3_short_int,1_long_int)"
    CALL Put_INTCOMPLEX(var1)

    var1 = Setup_INTCOMPLEX(3_short_int,1_short_int)
```

```
 PRINT*, "var1 = Setup_INTCOMPLEX(3_short_int,1_short_int)"
 CALL Put_INTCOMPLEX(var1)

 var1 = Setup_INTCOMPLEX(3_long_int,1_short_int)
 PRINT*, "var1 = Setup_INTCOMPLEX(3_long_int,1_short_int)"
 CALL Put_INTCOMPLEX(var1)

 PRINT*, "var1 = Setup_INTCOMPLEX(3_long_int,1_long_int)"
 var1 = Setup_INTCOMPLEX(3_long_int,1_long_int)
 CALL Put_INTCOMPLEX(var1)

 PRINT*, "200_short_int * Setup_INTCOMPLEX(3_long_int,1_long_int)"
 ans = 200_short_int * var1
 CALL Put_INTCOMPLEX(ans)

 PRINT*, "Setup_INTCOMPLEX(3_long_int,1_long_int) - 20_short_int"
 ans = var1 - 20_short_int
 CALL Put_INTCOMPLEX(ans)

 PRINT*, "Setup_INTCOMPLEX(3_long_int,1_long_int) ** 200_short_int"
 ans = var1 ** 6_short_int
 CALL Put_INTCOMPLEX(ans)

 PRINT*, "var ** 6_long_int"
 ans = var1 ** 6_long_int
 CALL Put_INTCOMPLEX(ans)

 PRINT*, "var1 = Setup_INTCOMPLEX(30000_long_int,10000_long_int)"
 var1 = Setup_INTCOMPLEX(30000_long_int,10000_long_int)
 CALL Put_INTCOMPLEX(var1)

 PRINT*, "400_long_int / Setup_INTCOMPLEX(30000_long_int,10000_long_int)"
 ans = 400_long_int / var1
 CALL Put_INTCOMPLEX(ans)

 PRINT*, "Setup_INTCOMPLEX(30000_long_int,10000_long_int) / 4000000"
 ans = var1 / 4000000_long_int
 CALL Put_INTCOMPLEX(ans)

END PROGRAM Testo
```

Practical Exercise 10: I/O Plus

Question 1: Functions as dummy arguments

Write a subroutine that accepts a function with one real argument (user defined or intrinsic), a start value $i1$, end value $i2$ and stride $i3$ and prints out the value of the function at each point defined by the sequence, $i1, i1+i3, i1+2i3, \ldots$ Demonstrate its functionality by producing tables for two intrinsic and two user defined functions.

Question 2: Interfaces

What is the interface of the following procedure?

```
SUBROUTINE SGETRI_F90(A, IPIV, INFO )
   USE LA_PRECISION, ONLY:WP
   IMPLICIT NONE
   REAL(KIND=WP), INTENT(INOUT), DIMENSION(:,:) :: A
   REAL(KIND=WP), ALLOCATABLE, DIMENSION(:,:)   :: WORK
   INTEGER, INTENT(IN), DIMENSION(:)            :: IPIV
   INTEGER, INTENT(OUT)                         :: INFO
   INTEGER                                      :: N
   INTEGER                                      :: ILAENV
   EXTERNAL  ILAENV
   INTRINSIC MIN, MATMUL
   INFO = 0
   N = SIZE(A,1)
   IF (SIZE(A,2) /= N) THEN
      INFO = -1
   ELSE IF (SIZE(IPIV,1) /= N) THEN
      INFO = -2
   ENDIF
   IF( N.EQ.0 ) RETURN
     ....
END SUBROUTINE SGETRI_F90
```

Question 3: File IO

Write a program to open a new sequential file on unit 4 called `marks.dat`. The program should then read a student's name followed by his/her marks for four exams from the keyboard (the default unit) and write these to the file on a single line. Repeat the read/write process until a student with the name END is entered. Close the output file.

Question 4: Formatted IO

Given the statement:

```
READ(*,'(F10.3,A2,L10)') A,C,L
```

what would the variables A (a real), C (a character of length 2), L (a logical) contain when given the following input? (Note: b signified a blank space.)

1. `bbb5.34bbbNOb.TRUE.`
2. `5.34bbbbbbYbbFbbbbb`
3. `b6bbbbbb3211bbbbbbT`
4. `bbbbbbbbbbbbbbbbbbF`

Question 5: Formatted IO

Give the formats which would be suitable for use with the statements,

```
REAL      :: A
CHARACTER(LEN=12) :: B
INTEGER   :: C
READ(*,FORM) A,B,C
```

if the record to be read is, (b is blank),

```
  bbb354bbINTRODUCTORYb1993
```

and values to be assigned to A, B and C are given in the following table:

A	B	C
35.4	INTRODUCTORY	1993
3.54	DUCT	93
.354	TROD	19
354.0	TORY	99

Question 6: More Formatted IO

Given that A is an array declared with the statement,

```
INTEGER, DIMENSION(1:8) :: A
```

give a formatted input statement which will read in the values 1 to A(1), 2 to A(2), etc. from the following records:

```
12
34
56
78
```

Question 7: Formatted File IO

A file `personnel.dat` contains records of people's name (up to 15 characters), age (3 digit integer), height (in metres to the nearest centimetre), telephone number (4 digit integer). Write a program to read the file and print out the details in the following format:

```
                           Height
     Name          Age    (metres)    Tel. No.
     ----          ---    ------      --------
 Bloggs J. G.       45     1.80         3456
 Clinton P. J.      47     1.75         6783
 etc.
```

Extra Questions

Question 8: More File IO

Write a program to read the file produced by the exam marks program and to print a list of students and their average marks.