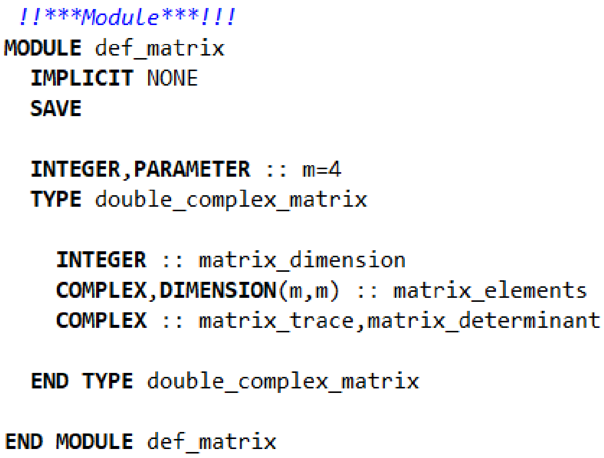
Derived Data Type

Abstract

The aim of this exercise is understand how we can write a test program in which define and initialize subroutines/functions and it is also a possibility to learn how define ,in a module unit program, a new derived type matrix data.
In order to do this I defined a new data type called *double complex matrix* and later i declared and initialized all functions that exercise required.
Finally,as requested by exercise,in a subroutine procedure,i wrote the new data type in a readable file.

1 Code development

The code starts with a ***module unit program*** in which a new data type called ”double complex matrix” is defined.



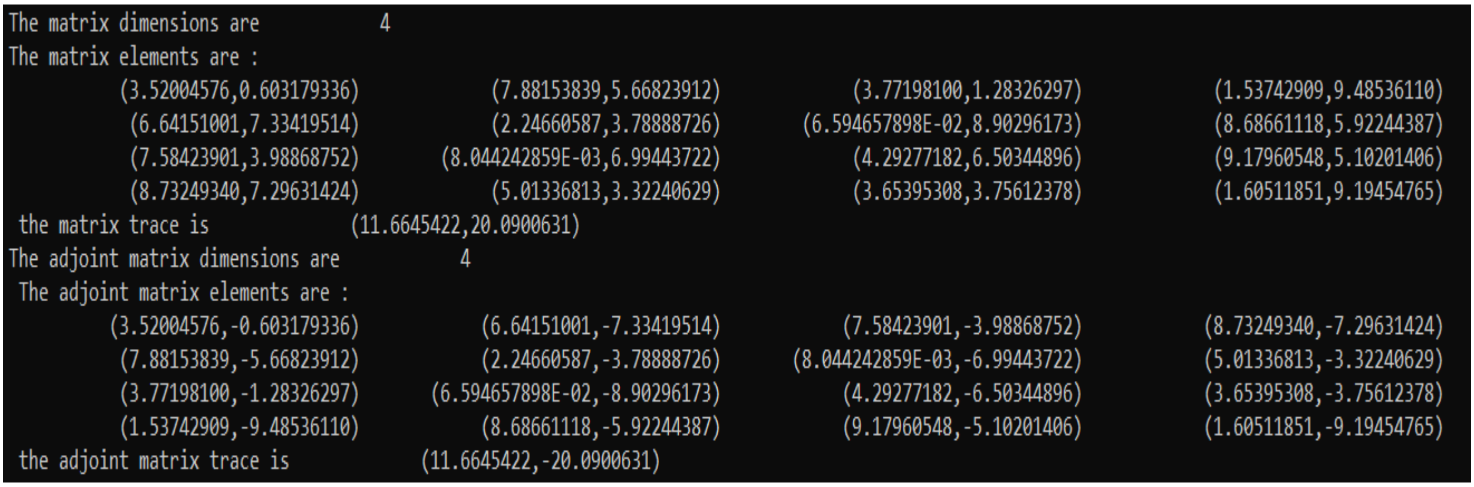
```
 !!***Module***!!!
MODULE def_matrix
  IMPLICIT NONE
  SAVE

  INTEGER,PARAMETER :: m=4
  TYPE double_complex_matrix

    INTEGER :: matrix_dimension
    COMPLEX,DIMENSION(m,m) :: matrix_elements
    COMPLEX :: matrix_trace,matrix_determinant

  END TYPE double_complex_matrix

END MODULE def_matrix
```

This object include an ***integer*** data as matrix dimension,a ***complex*** double array as matrix elements and another two ***complex*** variables as matrix trace and matrix determinant.

In the ***main program*** i defined function interfaces.

```
INTERFACE

     FUNCTION inizi_matrix(dcm)        !!! ****Inizi_matrix interface***

     USE def_matrix
     IMPLICIT NONE

     TYPE (double_complex_matrix) :: dcm

    END FUNCTION inizi_matrix

    FUNCTION  trace(dcm)             !!!***trace interface**!!!

    USE def_matrix
    IMPLICIT NONE

    TYPE (double_complex_matrix) :: dcm

    END FUNCTION trace


    FUNCTION  matrix_adjoint(dcm)   !!**matrix_adjoint interface**!!

    USE def_matrix
    IMPLICIT NONE

    TYPE (double_complex_matrix) :: dcm

    END FUNCTION matrix_adjoint


 END INTERFACE
```

Before the end of ***main program*** i tested functions defined outside printing their outputs.



```
The matrix dimensions are           4
The matrix elements are :
           (3.52004576,0.603179336)          (7.88153839,5.66823912)          (3.77198100,1.28326297)          (1.53742909,9.48536110)
            (6.64151001,7.33419514)          (2.24660587,3.78888726)      (6.594657898E-02,8.90296173)          (8.68661118,5.92244387)
            (7.58423901,3.98868752)      (8.044242859E-03,6.99443722)          (4.29277182,6.50344896)          (9.17960548,5.10201406)
            (8.73249340,7.29631424)          (5.01336813,3.32240629)          (3.65395308,3.75612378)          (1.60511851,9.19454765)
 the matrix trace is              (11.6645422,20.0900631)
The adjoint matrix dimensions are           4
 The adjoint matrix elements are :
          (3.52004576,-0.603179336)         (6.64151001,-7.33419514)         (7.58423901,-3.98868752)         (8.73249340,-7.29631424)
           (7.88153839,-5.66823912)         (2.24660587,-3.78888726)     (8.044242859E-03,-6.99443722)         (5.01336813,-3.32240629)
           (3.77198100,-1.28326297)     (6.594657898E-02,-8.90296173)         (4.29277182,-6.50344896)         (3.65395308,-3.75612378)
           (1.53742909,-9.48536110)         (8.68661118,-5.92244387)         (9.17960548,-5.10201406)         (1.60511851,-9.19454765)
 the adjoint matrix trace is              (11.6645422,-20.0900631)
```

From ***end program*** until the code last line i declared all ***functions*** such as matrix definition,trace matrix,and adjoint matrix and finally i declared also a ***subroutine*** used to write the new data type in a readable file.

```
 !********************Matrix_adjoint*****************!

TYPE(double_complex_matrix) FUNCTION  matrix_adjoint(dcm)

 USE def_matrix

 IMPLICIT NONE

 SAVE

 TYPE (double_complex_matrix) :: dcm,dcm1
 integer :: i,j

 dcm1=dcm

 DO i=1,m
    DO j=1,m

        dcm1%matrix_elements(i,j)=conjg( dcm%matrix_elements(i,j)   )

     END DO
  END DO


  dcm1%matrix_elements=transpose(dcm1%matrix_elements)

  matrix_adjoint=dcm1

  RETURN

 END FUNCTION matrix_adjoint
```

Exemple : adjoint matrix function definition

2 Self-evaluation

Achieved aims:

- how define *new data type* using MODULE unit program;
- how define function interfaces;
- how define functions and subroutine outside from main program and recall them during the process;
- how write in a txt file;

Next aim :

- If possible, define a new data matrix type, without dimensions previously defined in the same module and so,in other words, changeable directly in main program .I proved *ALLOCATABLE* method but it gives an error during compiling time.

Comment:

- Using different flags optimization run time compilation doesn't change.