EJERCICIO 16

Dados dos números naturales n ≥ m ≥ 0 se define el número combinatorio n sobre m como

$$C_{n,m} = \binom{n}{m} = \frac{n!}{m!\,(n-m)!}$$

Escribir una función en Fortran que calcule el factorial de un número. Utilizar esta función para calcular el número combinatorio de dos números introducidos por teclado.

Solución1: llama a la función "factorial" en el programa principal

```
Program Ejercicio16
Implicit none
! Variables                                  !declaramos la variable "factorial"
   Integer numero1, numero2, factorial       !para poder utilizar la función
! Cuerpo programa
   Print*, "Escribe dos naturales para calcular su número combinatorio"
   Print*, "El primero número tiene que ser mayor o igual que el segundo"
   Read*, numero1, numero2
   If (numero2>=0 .and. numero1>=numero2) Then  !comprobamos que se puede calcular
      Print*,"El número combinatorio de", numero1, "y", numero2, "es:"
      ! llamamos a la función factorial utilizando distintos argumentos
      Print*, factorial(numero1)/(factorial(numero2)*factorial(numero1-numero2))
   Else
      Print*, "El primer número ha de ser mayor o igual que el segundo y ambos naturales"
   End If
End Program Ejercicio16
! Funciones
! Función que calcula el factorial de un número natural
Integer Function factorial(numero)
implicit none
! Argumentos
   Integer numero
! Variables
   Integer i
! Cuerpo de la función factorial
   factorial=1
   Do i=1,numero
       factorial=factorial*i        !asignamos al nombre de la función el valor
   End Do                           !que devolverá
End Function factorial
```

Solución2: con otra función que calcula el número combinatorio y que es a la que se llama en el programa principal

```
Program Ejercicio16
Implicit none
! Variables
    Integer combinatorio, numero1, numero2
! Cuerpo programa
    Print*, "Escribe dos naturales para calcular su número combinatorio"
    Print*, "El primero número tiene que ser mayor o igual que el segundo"
    Read*, numero1,numero2
    If (numero2>=0 .and. numero1>=numero2) Then
        Print*,"El número combinatorio de", numero1, "y", numero2, "es:"
        Print*, combinatorio(numero1,numero2)
    Else
        Print*, "El primer número ha de ser mayor o igual que el segundo y ambos naturales"
    End If
End Program Ejercicio16
! Funciones
! Función  que calcula el factorial de un número natural
Integer Function factorial(numero)
Implicit none
! Argumentos
    Integer numero
! Variables
    Integer i
! Cuerpo de la función factorial
    factorial=1
    Do i=1,numero
    factorial=factorial*i
    End Do
End Function factorial
! Función que calcula el número combinatorio de dos números naturales tales
! que el primero es mayor o igual que el segundo
Integer Function combinatorio(n,m)
implicit none
! Argumentos
    Integer n, m
! Variables
    Integer i, factorial
! Cuerpo de la función combinatorio
    combinatorio = factorial(n)/(factorial(m)*factorial(n-m))
End Function combinatorio
! En este caso en el programa principal no se declara la función "factorial" como
! variable ya que no se utiliza. Sin embargo se declara la función "combinatorio"
! y en esta función si se utiliza la función factorial por lo que esta está en
! la declaración de variables de la propia función "combinatorio"
```

EJERCICIO 17

Escribir un programa en Fortran que calcule el Máximo Común Divisor y el Mínimo Común Múltiplo de dos números naturales introducidos por pantalla. Con este fin escribir:

- una función que calcule el Máximo Común Divisor de dos números naturales con el algoritmo de Euclides
- una función que calcule el Mínimo Común Múltiplo de dos números naturales utilizando la función anterior sabiendo que:

$$A * B = mcd(A, B) * mcm(A, B)$$

Solución

```
Program Ejercicio17
Implicit none
! Variables
    Integer numero1, numero2, mcd, mcm
! Cuerpo programa
    Print*, "Escribe dos naturales para calcular su Máximo Común Divisor"
    Print*, "y su Mínimo Común Múltiplo"
    Read*, numero1,numero2
    If (numero1>=0 .and. numero2>=0) Then
        Print*,"El Máximo Común Divisor de",numero1,numero2,"es:",mcd(numero1,numero2)
        Print*,"El Mínimo Común Múltiplo de",numero1,numero2,"es:",mcm(numero1,numero2)
    Else
        Print*,"Los números tienen que ser naturales"
    End If
End Program Ejercicio17
! Funciones
! Función que calcula el Máximo Común Divisor de dos números naturales
Integer Function mcd(num1,num2)
implicit none
! Argumentos
    Integer num1,num2
! Variables
    Integer i, aux1, aux2
! Cuerpo de la función
    aux1 = num1          !utilizamos variables auxiliares para no cambiar los
    aux2 = num2          !valores de las argumentos de la llamada a la función
    Do While (aux1/=0 .AND. aux2/=0)
        If (aux1 > aux2) Then
            aux1 = mod(aux1,aux2)
        Else
            aux2 = mod(aux2,aux1)
        End If
        mcd = aux1 + aux2
    End Do
End Function mcd
! Función que calcula el Mínimo Común Múltiplo de dos números naturales
Integer Function mcm(n1,n2)
implicit none
! Argumentos
    Integer n1,n2
! Variables
    Integer i,mcd,aux
! Cuerpo de la función
    aux = mcd(n1,n2)
    If (aux>0) Then      !protegemos la división por 0
        mcm = n1 * n2 / aux
    Else
        mcm = 0
    End If
End Function mcm
```

EJERCICIO 18

Realizar un programa en Fortran que representando las fracciones como vectores de 2 componentes enteras (numerador y denominador), implemente una serie de operaciones sobre estas. Debe ser capaz de sumar, restar, multiplicar y dividir dos fracciones y el resultado debe mostrarse como fracción irreducible.

Para dicho programa se utilizarán tanto funciones como subrutinas (al menos una por operación). El programa principal contendrá un menú en el que se permita elegir entre todas las operaciones posibles.

Solución

```
Program Ejercicio18
Implicit None
 ! Variables
       Integer A(2),B(2),C(2)
       Character(1) operacion
 ! Cuerpo del programa
    Call leerFraccion(A)
    Call leerFraccion(B)
    Call menu (operacion)
    If (operacion=='+') Then
        Call sumar(A,B,C)
    Else If (operacion == '-') Then
        Call restar(A,B,C)
    Else If (operacion == 'x') Then
        Call multiplicar(A,B,C)
    Else If (operacion == ':') Then
        Call dividir(A,B,C)
    Else
        Print*,"Operación no valida"
    End If
    Call escribirFraccion (C)
End Program Ejercicio18
!!! Subrutinas
Subroutine leerFraccion(F)
! Subrutina que lee una fracción introducida por teclado, previa petición
Implicit None
! Argumentos
    Integer F(2)
! Cuerpo de la Subrutina
    Print*,"Introduzca numerador y denominador de la fracción"
    Read*, F(1),F(2)
    Do While (F(2)==0)
        Print*,"El denominador ha de ser distinto de 0"
        Print*,"Introduzca de nuevo el denominador"
        Read*, F(2)
    End Do
End Subroutine leerFraccion
```

```
Subroutine menu (op)
! Subrutina que muestra el menu
Implicit None
! Argumentos
    Character(1) op
! Cuerpo de la subrutina
    Print*,"Elige la operación que quieres realizar"
    Print*,"+ ---> SUMAR"
    Print*,"- ---> RESTAR"
    Print*,"x ---> MULTIPLICAR"
    Print*,": ---> DIVIDIR"
    Read*,op
End Subroutine menu
Subroutine sumar(F1,F2,F3)
! Subrutina que calcula la suma de dos fracciones
Implicit None
! Argumentos
    Integer F1(2),F2(2),F3(2)
! Cuerpo de la subrutina
    F3(1) = F1(1)*F2(2) + F1(2)*F2(1)
    F3(2) = F1(2)*F2(2)
End Subroutine sumar
Subroutine restar(F1,F2,F3)
! Subrutina que calcula la resta de dos fracciones
Implicit None
! Argumentos
    Integer F1(2),F2(2),F3(2)
! Cuerpo de la subrutina
    F3(1) = F1(1)*F2(2) - F1(2)*F2(1)
    F3(2) = F1(2)*F2(2)
End Subroutine restar
Subroutine multiplicar(F1,F2,F3)
! Subrutina que calcula el producto de dos fracciones
Implicit None
! Argumentos
    Integer F1(2),F2(2),F3(2)
! Cuerpo de la subrutina
    F3(1) = F1(1)*F2(1)
    F3(2) = F1(2)*F2(2)
End Subroutine multiplicar
Subroutine dividir(F1,F2,F3)
! Subrutina que calcula la división de dos fracciones
Implicit None
! Argumentos
    Integer F1(2),F2(2),F3(2)
! Cuerpo de la subrutina
    F3(1) = F1(1)*F2(2)
    F3(2) = F1(2)*F2(1)
End Subroutine dividir
```

```
Subroutine escribirFraccion(F)
! Subrutina que simplifica una fracción y la imprime por pantalla
Implicit None
! Argumentos
    Integer F(2)
! Variables
    Integer mcd, aux
! Cuerpo de la subrutina
    aux=mcd(F(1),F(2))
    If (aux /= 0) Then
        F(1)=F(1)/aux
        F(2)=F(2)/aux
    End If
    Print*,"El resultado es:", F(1),"/",F(2)
End Subroutine escribirFraccion
!! Funciones
Integer Function mcd(num1,num2)
! Función que calcula el Máximo Común Divisor de dos números enteros
Implicit none
! Argumentos
    Integer num1,num2
! Variables
    Integer i, aux1, aux2
! Cuerpo de la función
    aux1 = num1
    aux2 = num2
    Do While (aux1/=0 .AND. aux2/=0)
        If (aux1 > aux2) Then
            aux1 = mod(aux1,aux2)
        Else
            aux2 = mod(aux2,aux1)
        End If
        mcd = abs(aux1 + aux2)       ! devolvemos el valor absoluto ya que
    End Do                           ! podemos tener valores negativos
End Function mcd
```

EJERCICIO 19

Realizar un programa en Fortran que implemente el juego del ahorcado. Para ello realizar las siguientes subrutinas:

- Subrutina que recibe como argumento el número de errores y dibuja el muñeco del ahorcado de la siguiente forma:

```
error=1     error=2     error=3     error=4     error=5     error=6
-           --          --          --          --          --
                          |           |           |           |
                                      O           O           O
                                                  /           / |

error=7     error=8     error=9     error=10
--          --          --          --
  |           |           |           |
 O           O           O           O
/ | \       / | \       / | \       / | \
              |           |           |
                          /           / \
```

- Subrutina que lee la palabra que escribe el primer jugador, la guarda en un vector de 20 caracteres, contabiliza el número de letras de la palabra y guarda en el resto del vector espacios en blanco. Además deberá ocultar la palabra al segundo jugador utilizando saltos de línea con la instrucción Print*, " ".

- Subrutina que comprueba si la letra elegida por el segundo jugador pertenece o no a la palabra. En caso afirmativo debe mostrar en que posición o posiciones aparece la letra utilizando otro vector de 20 caracteres que vaya mostrando los aciertos y los espacios de las letras que faltan por adivinar.

Ejemplo: si la palabra es "CASA" y la letra elegida es la "A" tendremos dos vectores

Palabra	C	A	S	A															

Aciertos	*	A	*	A															

Para realizar el resto del programa se pueden implementar mas subrutinas/funciones o incluir mas funcionalidades en las subrutinas anteriores.

Solución1: utilizando una subrutina que inicializa el vector de aciertos y los contadores de aciertos y errores.

```
Program Ahorcado
Implicit none
!Constantes
    parameter maxLetras=20, maxErrores=10, lineas=26
!Variables
    Character palabra1(maxLetras),palabra2(maxLetras),letra
    Integer numeroErrores,numeroLetras,numeroAciertos
!Cuerpo del programa
    Call leerPalabra(palabra1,numeroLetras)
    Call inicializarAciertos(palabra2,numeroAciertos,numeroErrores,numeroLetras)
    Print*, "               ", palabra2
    Print*, " "
    Do While (numeroAciertos<numeroLetras .and. numeroErrores<maxErrores)
        Print*, "Escribe una letra"
        Read*, letra
        Call  comprobarAciertos (palabra1,palabra2,letra,numeroLetras,numeroErrores,numeroAciertos)
        Print*, "               ", palabra2
        Print*, " "
        Call dibujarAhorcado(numeroErrores)
        Print*, " "
    End Do
    If (numeroErrores==maxErrores) Then
        Print*, "La palabra era:", palabra1
    Else If (numeroAciertos==numeroLetras) Then
        Print*, "Enhorabuena has acertado la palabra"
    End If
End Program
```

```
!Subroutinas
Subroutine dibujarAhorcado (contErrores)
!Subrutina que dibuja el muñeco del ahorcado teniendo
!en cuenta el número de errores
Implicit none
!Argumentos
    Integer contErrores
!Cuerpo de la subrutina
    If (contErrores==1) Then
        Print*, "-"
    Else If (contErrores==2) Then
        Print*, "--"
    Else If (contErrores==3) Then
        Print*, "--"
        Print*, "  |"
    Else If (contErrores==4) Then
        Print*, "--"
        Print*, "  |"
        Print*, "  O"
    Else If (contErrores==5) Then
        Print*, "--"
        Print*, "  |"
        Print*, "  O"
        Print*, " /"
    Else If (contErrores==6) Then
        Print*, "--"
        Print*, "  |"
        Print*, "  O"
        Print*, " /|"
    Else If (contErrores==7) Then
        Print*, "--"
        Print*, "  |"
        Print*, "  O"
        Print*, " /|\"
    Else If (contErrores==8) Then
        Print*, "--"
        Print*, "  |"
        Print*, "  O"
        Print*, " /|\"
        Print*, "  |"
    Else If (contErrores==9) Then
        Print*, "--"
        Print*, "  |"
        Print*, "  O"
        Print*, " /|\"
        Print*, "  |"
        Print*, " /"
    Else If (contErrores==10) Then
        Print*, "--"
        Print*, "  |"
        Print*, "  O"
        Print*, " /|\"
        Print*, "  |"
        Print*, " / \"
    End If
End Subroutine dibujarAhorcado
```

```
Subroutine leerPalabra (palabra, numLetras)
!Subrutina que lee la palabra, cuenta el número de letras que tiene e introduce
!espacios en blanco en el resto del vector. Además introduce lineas en blanco
!para ocultar la palabra al otro jugador.
Implicit none
!Constantes
    parameter maxLetras=20, lineas=26
!Argumentos
    Character palabra(maxLetras)
    Integer numLetras
!Variables
    Integer i,j
!Cuerpo de la subrutina
    numLetras=0
    i=1
    Print*,"Escribe la palabra letra a letra, escribe un punto para terminar"
    Print*, "La palabra no puede tener más de ", maxLetras, "letras"
    Read*, palabra(i)
    Do While (palabra(i)/= ".")
        numLetras=numLetras+1
        i=i+1
        Read*,palabra(i)
    End Do
    Do j=numLetras+1,maxLetras
        palabra(j)=" "
    End Do
    Do j=0,lineas
        Print*, " "
    End Do
End Subroutine leerPalabra
Subroutine inicializarAciertos (aciertos,nAciertos,nErrores,nLetras)
!Subrutina que inicializa el vector de aciertos a * hasta el número de letras
!y blancos en el resto. Además inicializa el número de aciertos y errores a 0
Implicit none
!Constantes
    parameter maxLetras=20
!Argumentos
    Character aciertos(maxLetras)
    Integer nAciertos,nErrores,nLetras
!Variables
    Integer i
!Cuerpo de la subrutina
    nAciertos=0
    nErrores=0
    Do i=1,nLetras
        aciertos(i)="*"
    End Do
    Do i=nLetras+1,maxLetras
        aciertos(i)=" "
    End Do
End Subroutine inicializarAciertos
```

```
Subroutine comprobarAciertos (buscada,coincidentes,caracter,numLet,numErrores,numAciertos)
!Subrutina que comprueba si la letra está en la palabra buscada y actualiza el
!vector de aciertos y los contadores de errores y aciertos.
Implicit none
!Constantes
    parameter maxLetras=20
!Argumentos
    Character coincidentes(maxLetras), buscada(maxLetras),caracter
    Integer numLet, numErrores,numAciertos
!Variables
    Integer i,ok        !ok variable auxiliar que guarda si la letra está o no en la palabra
!Cuerpo de la subrutina
    ok=1                 !persuponemos que no está la letra
    Do i=1,numLet
        If (caracter==buscada(i)) Then
            If  (caracter/=coincidentes(i))Then
            !Para proteger el n° de aciertos de la repeticion de letras que estan en la palabra
                coincidentes(i)=caracter
                numAciertos=numAciertos+1
            End If
            ok=0       !si la letra está ponemos 0
        End If
    End Do
    numErrores=numErrores+ok        !ok vale 1 si la letra no está, 0 si está
End Subroutine comprobarAciertos
```

Solución2: incluye la inicialización del vector de aciertos y los contadores de aciertos y errores en la subrutina "leerPalabra". Solo cambia el programa principal, ya que se elimina la llamada a la subrutina "inicializarAciertos" (que se elimina) y la propia subrutina "leerPalabra". Por esta razón solo se muestran a continuación el programa principal y la subrutina.

```
Program Ahorcado
Implicit none
!Constantes
    parameter maxLetras=20, maxErrores=10, lineas=26
!Variables
    Character palabra1(maxLetras),palabra2(maxLetras),letra
    Integer numeroErrores,numeroLetras,numeroAciertos,acierto,i
!Cuerpo del programa
    Call leerPalabra(palabra1,numeroLetras,palabra2,numeroAciertos,numeroErrores)
    Print*, "               ", palabra2
    Print*, " "
    Do While (numeroAciertos<numeroLetras .and. numeroErrores<maxErrores)
        Print*, "Escribe una letra"
        Read*, letra
        Call  comprobarAciertos (palabra1,palabra2,letra,numeroLetras,numeroErrores,numeroAciertos)
        Print*, "               ", palabra2
        Print*, " "
        Call dibujarAhorcado(numeroErrores)
        Print*, " "
    End Do
    If (numeroErrores==maxErrores) Then
        Print*, "La palabra era:", palabra1
    Else If (numeroAciertos==numeroLetras) Then
        Print*, "Enhorabuena has acertado la palabra"
    End If
End Program
```

```
Subroutine leerPalabra (palabra,numLetras,aciertos,nAciertos,nErrores)
!Subrutina que lee la palabra, cuenta el número de letras que tiene e introduce
!espacios en blanco en el resto del vector. Además introduce lineas en blanco
!para ocultar la palabra al otro jugador.
!Tambien inicializa el vector de aciertos a * hasta el número de letras
!y blancos en el resto, e inicializa el número de aciertos y errores a 0
Implicit none
!Constantes
    parameter maxLetras=20, lineas=26
!Argumentos
    Character palabra(maxLetras),aciertos(maxLetras)
    Integer numLetras,nAciertos,nErrores
!Variables
    Integer i,j
!Cuerpo de la subrutina
    numLetras=0
    i=1
    Print*,"Escribe la palabra letra a letra, escribe un punto para terminar"
    Print*, "La palabra no puede tener más de ", maxLetras, "letras"
    Read*, palabra(i)
    Do While (palabra(i)/= ".")
        numLetras=numLetras+1
        aciertos(i)="*"
        i=i+1
        Read*,palabra(i)
    End Do
    Do j=numLetras+1,maxLetras
        palabra(j)=" "
        aciertos(i)=" "
    End Do
    nAciertos=0
    nErrores=0
    Do j=0,lineas
        Print*, " "
    End Do
End Subroutine leerPalabra
```

EJERCICIO 20:

Realizar un programa en código FORTRAN que representando los números complejos como vectores de 2 componentes reales (parte real y parte imaginaria), implemente una serie de operaciones sobre estos.

Dichas operaciones serán:

1. Leer un número complejo.
2. Escribir un número complejo.
3. Multiplicar un número complejo por un número escalar.
4. Hallar el conjugado de un número complejo.
5. Sumar dos números complejos.
6. Restar dos números complejos.
7. Multiplicar dos números complejos.
8. Dividir dos números complejos.
9. Realizar el opuesto de un número complejo.
10. Realizar el inverso de un número complejo.

Para dicho programa se utilizarán subrutinas (Se realizará como mínimo una por operación).

Nota1:
La realización de dichas operaciones:

El número complejo se representará con la notación (a + bi) (donde a es la parte real y b, la parte imaginaria).

a) Multiplicar un número complejo por un número escalar.
   k * (a + bi) = k * a + k * bi

b) Hallar el conjugado de un número complejo.
   Conjugado(a + bi) = (a – bi)

c) Suma de dos números complejos.
   (a + bi) + (c + di) = (a + c) + (b + d)i

d) Diferencia de dos números complejos.
   (a + bi) - (c + di) = (a - c) + (b - d)i

e) Producto de dos números complejos.

   (a + bi) * (c + di) = (a * c – b * d) + (a * d + b * c)i

f) División de dos números complejos.
   (a + bi) / (c + di) = ((a + bi) * (c - di)) / (c2 + d2)
   Utilizar obligatoriamente para este apartado los subprogramas ya implementados en los apartados: e) la multiplicación de dos números complejos, b) el conjugado y a) la multiplicación de un escalar por un número complejo.

g) Opuesto de un número complejo.
   Opuesto (a + bi) = (-a - bi)
   Utilizar para este apartado el subprograma del apartado a)

h) Inverso de un número complejo.
   Inverso (a + bi) = 1 / (a + bi)
   Utilizar para este apartado el subprograma del apartado f)

Nota2:
Se deberá realizar un programa principal con un menú en el que se permita elegir entre todas las operaciones posibles.

Solución

```
PROGRAM NUMIMAG
! Calculadora elemental de complejos
        IMPLICIT NONE
! Declaración de variables
        REAL::F1(2), F2(2), SOLUCION(2)
        REAL::FE
        CHARACTER (1) OPCION
! Cuerpo del programa principal
        OPCION = '#'
        DO WHILE (OPCION <> 'S')
! Presentación menú
                PRINT *, "OPCIONES....."
                PRINT *, "E : MULTIPLICACION POR ESCALAR"
                PRINT *, "C : CONJUGADO"
                PRINT *, "+ : SUMAR"
                PRINT *, "- : RESTAR"
                PRINT *, "X : MULTIPLICAR"
                PRINT *, "D : DIVIDIR"
                PRINT *, "O : OPUESTO"
                PRINT *, "I : INVERSO"
                PRINT *, "S : SALIR"
                PRINT *, "Escriba su opción: "
                READ *, OPCION

                SELECT CASE (OPCION)
                CASE ('E')
                        PRINT *, "NUMERO IMAGINARIO"
                        CALL LEER(F1)
                        PRINT *, "NÚMERO ESCALAR"
                        CALL LEERESC(FE)
                        CALL ESCALAR(F1, FE)
                        PRINT *, "EL RESULTADO DEL PRODUCTO ESCALAR ES:"
                        CALL ESCRIBIR(F1)
                CASE ('+')
                        PRINT *, "PRIMER NUMERO IMAGINARIO"
                        CALL LEER(F1)
                        PRINT *, "SEGUNDO NUMERO IMAGINARIO"
                        CALL LEER(F2)
                        CALL SUMA(F1, F2, SOLUCION)
                        PRINT *, "EL RESULTADO DE LA SUMA ES:"
                        CALL ESCRIBIR(SOLUCION)
                CASE ('-')
                        PRINT *, "PRIMER NUMERO IMAGINARIO"
                        CALL LEER(F1)
                        PRINT *, "SEGUNDO NUMERO IMAGINARIO"
                        CALL LEER(F2)
                        CALL DIFERENCIA(F1, F2, SOLUCION)
                        PRINT *, "EL RESULTADO DE LA RESTA ES:"
                        CALL ESCRIBIR(SOLUCION)
                CASE ('X')
                        PRINT *, "PRIMER NUMERO IMAGINARIO"
                        CALL LEER(F1)
                        PRINT *, "SEGUNDO NUMERO IMAGINARIO"
                        CALL LEER(F2)
                        CALL PRODUCTO(F1, F2, SOLUCION)
                        PRINT *, "EL RESULTADO DE LA MULTIPLICACION ES:"
                        CALL ESCRIBIR(SOLUCION)
                CASE ('D')
```

```
                PRINT *, "PRIMER NUMERO IMAGINARIO"
                CALL LEER(F1)
                PRINT *, "SEGUNDO NUMERO IMAGINARIO"
                CALL LEER(F2)
                CALL DIVISION(F1, F2, SOLUCION)
                PRINT *, "EL RESULTADO DE LA DIVISION ES:"
                CALL ESCRIBIR(SOLUCION)
        CASE ('O')
                PRINT *, "NUMERO IMAGINARIO"
                CALL LEER(F1)
                CALL OPUESTO(F1)
                PRINT *, "EL NÚMERO IMAGINARIO OPUESTO ES:"
                CALL ESCRIBIR(F1)
        CASE ('C')
                PRINT *, "NUMERO IMAGINARIO"
                CALL LEER(F1)
                CALL CONJUGADO(F1)
                PRINT *, "EL NÚMERO IMAGINARIO CONJUGADO ES:"
                CALL ESCRIBIR(F1)
        CASE ('I')
                PRINT *, "NUMERO IMAGINARIO"
                CALL LEER(F1)
                CALL INVERSO(F1)
                PRINT *, "EL NÚMERO IMAGINARIO INVERSO ES:"
                CALL ESCRIBIR(F1)
        END SELECT
END DO
PRINT *, "FIN DEL PROGRAMA....."
END PROGRAM NUMIMAG


! Subprogramas

        SUBROUTINE LEER(A)
        ! Introduce en A un complejo  desde teclado
                IMPLICIT NONE
                REAL A(2)
                PRINT *, "Introduzca parte real y parte imaginaria"
                READ *, A(1), A(2)
        END SUBROUTINE


        SUBROUTINE LEERESC(A)
        ! Introduce en A un escalar desde teclado
                IMPLICIT NONE
                REAL A
                PRINT *, "Introduzca el número escalar"
                READ *, A
        END SUBROUTINE

        SUBROUTINE ESCRIBIR(A)
        ! Imprime el complejo A
                IMPLICIT NONE
                REAL A(2)
                IF (A(1) <> 0) THEN
                        PRINT *, A(1)
                END IF
                IF (A(2) <> 0) THEN
                        PRINT *, A(2), 'i'
```

```
        END IF
END SUBROUTINE

SUBROUTINE ESCALAR(A,K)
! Multiplica el complejo A por el escalar K
        IMPLICIT NONE
        REAL A(2)
        REAL K
        A(1) = A(1) * K
        A(2) = A(2) * K
END SUBROUTINE

SUBROUTINE CONJUGADO(A)
! Cambia de signo la parte imaginaria de A
        IMPLICIT NONE
        REAL A(2)
        A(2) = -A(2)
END SUBROUTINE

SUBROUTINE SUMA(A,B,C)
! Suma los complejos A y B y guarda el resultado en C
        IMPLICIT NONE
        REAL A(2), B(2), C(2)
        C(1) = A(1) + B(1)
        C(2) = A(2) + B(2)
END SUBROUTINE

SUBROUTINE DIFERENCIA(A,B,C)
! Resta los complejos A y B y guarda el resultado en C
        IMPLICIT NONE
        REAL A(2), B(2), C(2)
        CALL OPUESTO(B)
        CALL SUMA (A, B,C)
END SUBROUTINE

SUBROUTINE PRODUCTO(A,B,C)
! Multiplica los complejos A y B y guarda el resultado en C
        IMPLICIT NONE
        REAL A(2), B(2), C(2)
        C(1) = (A(1) * B(1)) - (A(2) * B(2))
        C(2) = (A(1) * B(2)) + (A(2) * B(1))
END SUBROUTINE

SUBROUTINE DIVISION(A,B,C)
! Divide los complejos A y B y guarda el resultado en C
        IMPLICIT NONE
        REAL A(2), B(2), C(2)
        CALL CONJUGADO(B)
        CALL PRODUCTO (A,B,C)
        CALL ESCALAR (C, 1/((B(1)*B(1)) + (B(2)*B(2))))
END SUBROUTINE

SUBROUTINE OPUESTO(A)
! Cambia el signo tanto de la parte real como de la imaginaria de A
        IMPLICIT NONE
        REAL A(2)
        A(1) = -A(1)
        A(2) = -A(2)
END SUBROUTINE
```

```
SUBROUTINE INVERSO(A)
! Inviete el complejo A
        IMPLICIT NONE
        REAL A(2),AUX(2)
        AUX(1) = 1
        AUX(2) = 0
        CALL DIVISION(AUX,A,A)
END SUBROUTINE
```