Corso di Laurea in Matematica - Laboratorio di Programmazione gr.2 – a.a. 2014/15

SECONDA esercitazione in laboratorio

--- Creazione e modifica di files ASCII per mezzo del programma ***gedit***.
--- Uso del compilatore ***gfortran*** : debugging formale, compilazione, linking, esecuzione di programmi *Fortran* in ambiente *Linux.*
--- Redirezione dell' input e dell' output del programma da/verso files di testo
--- Esempi: accumulo di somme ; calcolo massimo e minimo ; risoluzione equazione II grado

PROMEMORIA: Come ***creare***, ***compilare*** ed ***eseguire*** programmi in Fortran 95

Accedere al sistema Fedora inserendo nella schermata di benvenuto le proprie *username* e *password*, **aprire il *terminale Linux*** per operare in modalità *linea di comando* [dal menu principale di Fedora (barra in alto, a sinistra) scegliere APPLICAZIONI > STRUMENTI SISTEMA > TERMINALE] nella finestra che verrà aperta (sfondo bianco) **operare in *modalità terminale*, usando il linguaggio di comando del sistema operativo *Linux*** .
Importante!! : una volta finito di operare, chiudere il terminale digitando **EXIT**, e chiudere la sessione Fedora scegliendo **ESCI** dal menu a tendina in alto a destra e poi **Termina Sessione**.

Operando a terminale Linux, in *modalità linea di comando* :

1) Per prima cosa creare una nuova directory (in cui si opererà) con il comando ***mkdir*** , ed entrare in questa directory usando il comando ***cd*** . Ad es. :

[prompt]$ **mkdir eser2**

[prompt]$ **cd eser2**

2) Per creare o modificare un file ASCII contenente un programma simbolico eseguire il programma **gedit**. Al prompt, digitare: **gedit *<nomefile>*** (dove <nomefile> è il nome del file da creare o modificare) e poi operare attraverso la interfaccia grafica di **gedit**.
Il file ASCII contenente un programma in *Fortran* deve avere un nome con estensione *".f90".*
Quindi, per esempio, il comando da digitare per mandare in esecuzione *gedit* sarà del tipo:

[prompt]$ **gedit filesorgente.f90**.

Una volta chiuso **gedit**, ed ottenuto nuovamente il prompt dalla shell di LinuX, si potrà verificare l'esistenza del file *filesorgente.f90* digitando il comando **ls** nel terminale LinuX.

3) Il nome del file contenente il compilatore Fortran è **gfortran.exe** , quindi il comando per mandare in esecuzione il compilatore sarà del tipo :

[prompt]$ **gfortran -<opzioni> <nome/i files>**

dove con <nome/i files> si è indicato il nome del file (o i nomi dei files) contenenti il sorgente.

----- Caso 3A : solo verifica della correttezza del codice Fortran: usare l'opzione **c**

[prompt]$ **gfortran -c filesorgente.f90**

----- Caso 3B : Verifica correttezza, compilazione, creazione dell'eseguibile (opzione **o**)

[prompt]$ **gfortran -o *eseguibile* *filesorgente.f90*** [*file-2.f90….. file-n.f90*]

per effetto di questo ultimo comando il codice sorgente contenuto in *filesorgente.f90* verrà compilato e poi linkato agli altri rilocabili ed alle librerie, e verrà creato l'eseguibile che verrà memorizzato su disco nel file con il nome da noi indicato nel comando come primo operando (nell'esempio: *esegubile[.exe]*). Se si sono indicati più files sorgente (nomi opzionali tra parentesi quadre), tutti verranno compilati, creando i corrispondenti moduli rilocabili, e nella fase di *linking* tutti i moduli rilocabili verranno linkati insieme per ottenere un unico programma eseguibile, che prenderà nome da noi indicato (*eseguibile.exe*). L'estensione *.exe* esiste ma non è necessario indicarla esplicitamente. Per effetto della compilazione, il contenuto dei files sorgente, *filesorgente.f90* ecc., non sarà modificato. Dopo l'esecuzione del comando, al ritorno del *prompt* della shell, si può verificare l'esistenza del file *esegubile.exe*, ed in generale il contenuto della cartella corrente, usando il comando **ls**.

4 -- Per mandare in esecuzione il programma eseguibile appena creato digitare, al prompt :

[prompt]$ ***./eseguibile*** (tutto attaccato, senza *blanks*…..dite voi: perché?)

N.B. E' indispensabile far precedere il nome del file contenente il programma da eseguire dai caratteri " ***./*** " (punto slash).

REDIREZIONE DEI FLUSSI DI DATI IN INGRESSO AL PROGRAMMA ED IN USCITA DAL PROGRAMMA

Quando il programma è in esecuzione sono definite automaticamente le unità di ingresso ed uscita per mezzo delle quali il programma "comunica" con l'utilizzatore ricevendo dati in input (esecuzione di una istruzione READ) o **fornendo dati in output** (esecuzione di una istruzione di WRITE), o anche **comunicando il verificarsi di eventuali condizioni di errore**. Le unità (o meglio, i flussi di dati scambiati tra il programma e tali unità, i cosiddetti "canali standard") prendono il nome convenzionale di **stdin** (standard input [channel]), **stdout** (standard output [channel]), e **stderr** (standard error [channel]) rispettivamente**. La condizione di default** è che lo stdin corrisponde alla tastiera, lo stdout e lo stderr corrispondono allo schermo del terminale; ma è possibile **modificare tale scelta effettuando quella che è detta una redirezione dell'I/O**

Per redirigere l'output del programma *prog.exe* verso il file chiamato *fileout* mandare in esecuzione il programma digitando al prompt:

[prompt]$ ***./prog > fileout***

In questo modo tutto l'output del programma (tutti i dati che vengono "scritti" con istruzioni WRITE) verrà scritto nel file ***fileout*** e non sullo schermo.

Analogamente, **per redirigere l'input del programma *prog.exe* acquisendo tutti i dati da un file chiamato *fileout* anziché dalla tastiera, mandare in esecuzione il programma digitando al prompt:**

[prompt]$ ***./prog < fileinp***

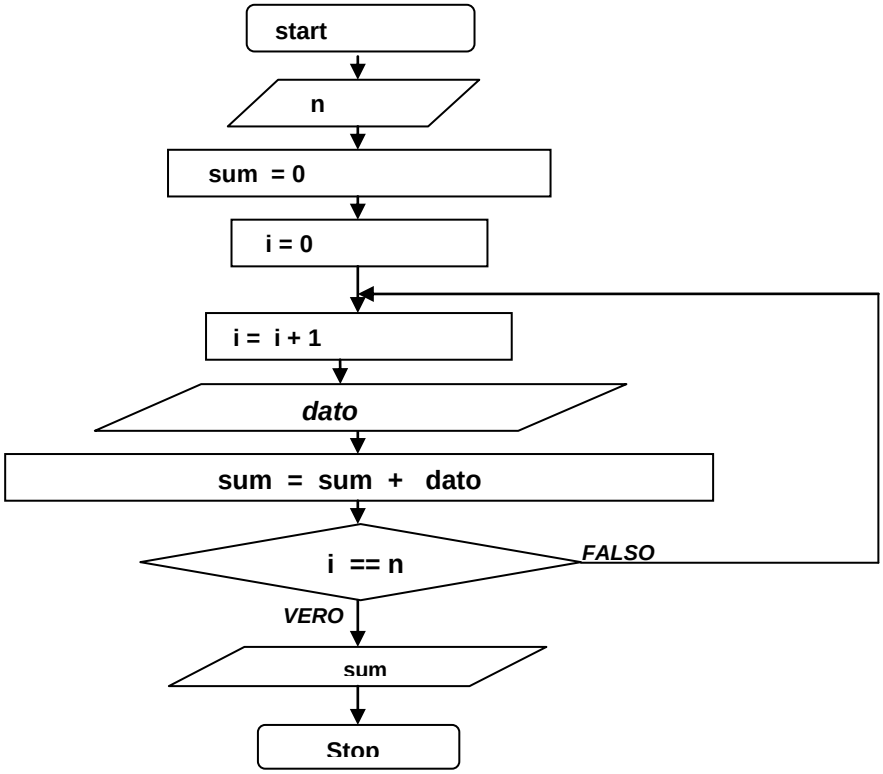
Per **redirigere entrambi i flussi** è possibile scrivere

[prompt]$ ***./prog < fileinp > fileout***

ESERCIZIO 1 --- Creazione ed esecuzione di un programma Fortran che calcola la somma di n numeri interi forniti in input tramite tastiera (dati in input : n, a1, a2…., an).

COME PROCEDERE IN LABORATORIO

Il flowchart ed il codice Fortran qui sotto illustrano un programma che accumula la somma di *n* addendi forniti in input (dove anche *n* è fornito in input): creare il sorgente poi compilare, linkare eseguire con vari insiemi di dati. [Per risparmiare tempo partire dal programma per il calcolo della somma dei primi *n* naturali creato nella prima esercitazione; creare la directory eser2 copiare il vecchio file in eser2, usando il comando "**cp**", e poi modificatelo opportunamente usando **gedit**]



CODICE 1 : QUESTO E' IL CODICE FOTRAN CHE IMPLEMENTA L'ALGORITMO PER IL CALCOLO DELLA SOMMA DI *n* NUMERI ILLUSTRATO NEL FLOWCHART QUI SOPRA

```
PROGRAM provasomma
IMPLICIT NONE
INTEGER :: n, i, dato, somma
WRITE (*,*)
WRITE (*,*) ' Questo programma calcola la somma'
WRITE (*,*) ' di  n numeri forniti in input'
WRITE (*,*) ' uno alla volta'
WRITE (*,*) ' Inserire il numero di dati da sommare'
READ  (*,*) n
somma = 0
DO i  = 1, n
  WRITE (*,*) ' Inserire il prossimo addendo'
  READ  (*,*) dato
  somma = somma + dato
END DO
WRITE (*,*) ' Il valore della somma degli', n
WRITE (*,*) ' numeri inseriti e'' :', somma
STOP
END PROGRAM prova somma
```

Eseguite il programma una o due volte fornendo i dati in input dalla tastiera, poi provate a creare (usando **gedit**) un file di dati opportuno (chiamatelo ade es, "*datisomma*" ed eseguite il programma acquisendo i dati in input da tale file **usando la redirezione dello standard input**.

[prompt]$ ***./somman < datisomma***

Provate anche a redirigere l'output verso un file chiamato "*fileout*". Dopo l'esecuzione del programma controllate il contenuto del file *fileout* usando il comando **cat.**

ESERCIZIO 2 --- Creazione ed esecuzione di un programma Fortran che calcola la somma di n numeri interi forniti in input tramite tastiera, nel caso in cui non è noto il numero di addendi, ma il flusso di dati in input è chiuso da un "tappo" (dati in input : a1, a2…., an, atappo).

Per ottenere un programma che implementi l'algoritmo richiesto il codice Fortran dell'esercizio precedente potrebbe, per esempio, essere modificato come segue:

```
PROGRAM provasomma
IMPLICIT NONE
INTEGER :: i, dato, tappo, somma
WRITE (*,*)
WRITE (*,*) ' Questo programma calcola la somma di alcuni '
WRITE (*,*) ' numeri interi forniti in input uno alla volta'
WRITE (*,*)
WRITE (*,*) ' Inserire il valore che chiudera'''
WRITE (*,*) ' il flusso di dati in input'
READ  (*,*) tappo
WRITE (*,*)
WRITE (*,*) ' Il tappo e'' :' , tappo
WRITE (*,*)
somma = 0
i = 0
  DO
    WRITE (*,*) ' Inserire il prossimo addendo'
    READ  (*,*) dato
    IF ( dato == tappo ) EXIT
    somma = somma + dato
    i = i +1
  END DO
WRITE (*,*)
WRITE (*,*) ' Il valore della somma dei/degli', i
WRITE (*,*) ' numeri inseriti e'' :', somma
STOP
END PROGRAM provasomma
```

Anche in questo caso eseguite il programma una o due volte fornendo i dati in input dalla tastiera, poi provate a creare eseguire il programma acquisendo i dati in input da tale file **usando la redirezione dello standard input**.

[prompt]$ ***./sommatappo < dati***

Potete usare lo stesso file dati che avete creato prima ? Se no, perché ?

ESERCIZIO 3 --- Modificate i programmi degli esercizi 1 e 2 in modo tale che anziché la somma dei dati forniti calcolino il minimo ed il massimo di tali insiemi di dati.

Eseguite sia con input ed output standard che con redirezione dello I/O

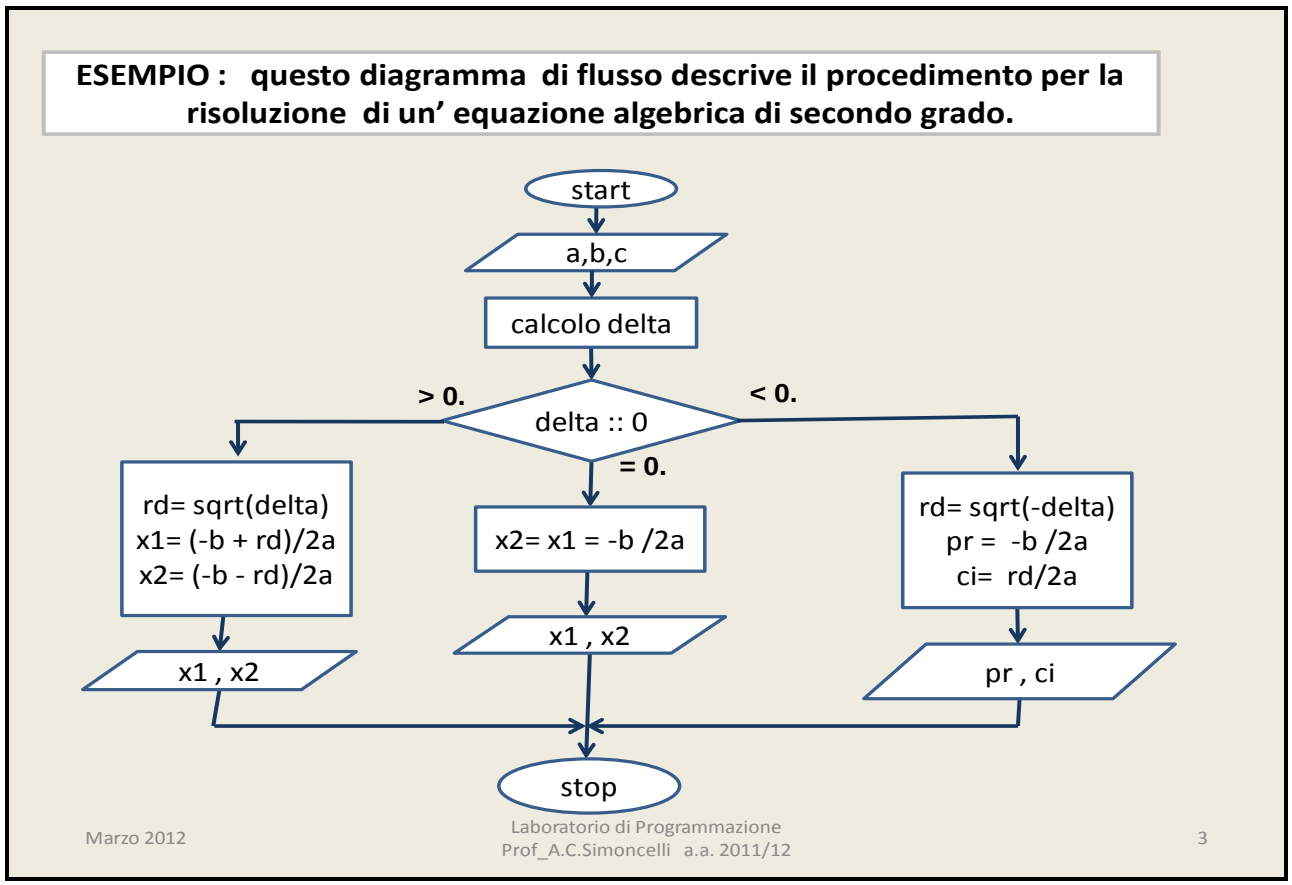
ESERCIZIO 4. Risoluzione di una equazione algebrica di secondo grado.

Studiare il flowchart riportato qui sotto, studiare il codice Fortran riportato nella prossima pagina, poi passare alla implementazione al computer.

SUGGERIMENT MOLTO UTILE: Nel digitare il programma, saltare tutti i commenti (risparmierete molto tempo) ma leggeteli per capire cosa fa il programma; questi sono stati inseriti proprio come chiarimento per chi usa il programma.

Eseguire il programma con diversi set di dati, così da verificarne il corretto funzionamento in tutti i casi possibili (delta positivo, delta negativo, delta nullo)

Ad esempio : (1) a = 1. b = 1. c = - 12. risultati_____
(2) a = 1. b = - 2. c = 1. risultati_____
(3) a = 1. b = - 4. c = 13. risultati_____



PROGRAM eq2grado
!----------
! Questo programma calcola le radici della equazione algebrica di secondo grado,

```
!     a coefficienti reali qualsiasi:     a*x**2 + b*x + c = 0
!     i valori dei  coefficienti a, b, c sono forniti  in input  da tastiera;   le radici, reali o
!    complesse, vengono calcolate e fornite  in output in forma adeguata ai vari casi possibili
!-------------------------------------------------------------------------------------------------------------------------
!--------------------------Parte dichiarazioni--
IMPLICIT NONE
! elenco delle variabili con dichiarazione del tipo
REAL :: a        ! Coefficiente del termine di grado 2
REAL :: b        ! Coefficiente del termine di grado 1
REAL :: c        ! Termine noto
REAL :: delta  ! Discriminante dell'equazione
REAL :: x1      ! Prima radice reale
REAL :: x2      ! Seconda radice reale
REAL :: pimag    ! Parte immaginaria delle radici complesse
REAL :: preal      ! Parte reale delle radici complesse
!-------------------------------------------------------------------------------Parte esecutiva-----
WRITE (*,*)               !  lascia un rigo bianco nell'output
WRITE (*,*) 'Programma per la risoluzione di una equazione di secondo grado'
WRITE (*,*) 'del tipo: A * x**2 + B * x + C = 0 '
WRITE (*,*)
WRITE (*,*)  'Introdurre i coefficienti A, B, and C: '             ! Sollecita l'input dei coefficienti
READ  (*,*) a, b, c                                                ! Input dei coefficienti
WRITE (*,*)
WRITE (*,*) 'I coefficienti  immessi sono: ', a, b, c             ! Eco dei coefficienti per la verifica
WRITE (*,*)
delta = b**2 - 4. * a * c     !  calcolo del discriminante
! In base al valore di delta,  il flusso delle istruzioni sarà diverso nei vari casi
IF ( delta > 0.) THEN         ! delta positivo: radici reali
   x1 = ( -b + sqrt(delta) ) / ( 2. * a )
   x2 = ( -b - sqrt(delta) ) / ( 2. * a )
! --- Output dei risultati ---
   WRITE (*,*) 'L''equazione ammette le due radici reali distinte:'
   WRITE (*,*) 'X1 = ', x1
   WRITE (*,*) 'X2 = ', x2
ELSE IF ( delta < 0.) THEN    ! delta negativo: radici complesse
! calcolo parte reale e coefficiente dell'immaginario
   preal = ( -b ) / ( 2. * a )
   pimag = sqrt ( abs ( delta)) / ( 2. * a )
! --- Output dei risultati (numeri complessi) in forma opportuna
   WRITE (*,*) 'L''equazione ammette due radici complesse coniugate:'
   WRITE (*,*) 'X1 = ', preal, ' + ', pimag, 'i'
   WRITE (*,*) 'X2 = ', preal, ' - ', pimag, 'i'
ELSE IF ( delta == 0. ) THEN     ! delta nullo: radice doppia
! calcolo
   x1 = ( -b ) / ( 2. * a )
!--- output della radice doppia---
   WRITE (*,*) 'Questa equazione ammette la sola radice, doppia:'
   WRITE (*,*) 'X1 = X2 = ', x1
END IF
END PROGRAM eq2grado

==========================================================================
```