UNIVERSITA' DEGLI STUDI DI ROMA
"LA SAPIENZA"

FACOLTA' DI INGEGNERIA

Dal PASCAL al FORTRAN

Mirella Casini Schaerf

Daniele Nardi

Dispensa di
Fondamenti di Informatica

ANNO ACCADEMICO 1995/96

SOMMARIO

1 . 1 INTRODUZIONE

Il linguaggio FORTRAN (FORmula TRANslator) è nato negli anni '50, come un meccanismo per la gestione di librerie di programmi d'utilità, ed è stato uno dei primi linguaggi di programmazione ad alto livello. Dopo le prime versioni, apparve nel 1957 il manuale del FORTRAN II, ad opera di John Backus ed altri. Sono stati poi elaborati gli standard di nuove versioni (di cui la più fortunata è stata quella del FORTRAN IV), fino agli attuali FORTRAN strutturati FORTRAN77 e FORTRAN90.

Il FORTRAN si colloca fra i linguaggi imperativi ad alto livello, come uno dei più vicini alla struttura della macchina. Per questo, oltre che per il fatto di essere un linguaggio di largo uso da molti anni, possiede compilatori molto efficienti e robusti. Inoltre il corredo delle librerie di programmi di utilità, venutosi a comporre ed a consolidare nel tempo, è di gran lunga più vasto rispetto a quello disponibile per ogni altro linguaggio della stessa classe. Il FORTRAN è utilizzato soprattutto per elaborazioni di calcolo numerico, dove l'efficienza e la velocità di esecuzione che esso offre, e che sono dovute al suo orientamento alla macchina, sono ritenute più importanti rispetto ad altri aspetti del linguaggio.

Rispetto al PASCAL il FORTRAN risulta più essenziale. Innanzitutto esso ha una gestione completamente statica delle risorse, in particolare della memoria. I tipi di dato semplici sono di tipo numerico (con diverse possibilità in più rispetto al PASCAL), booleano e carattere. Il principale tipo di dato strutturato è il vettore di elementi omogenei; vi è inoltre il tipo stringa che è costituito da una sequenza di caratteri. Il controllo delle istruzioni, inizialmente basato sull'uso dell'istruzione di salto "GOTO", è stato nelle versioni più recenti aggiornato con le istruzioni strutturate più comuni dei moderni linguaggi imperativi. Le operazioni di ingresso/uscita sono piuttosto potenti ed in genere richiedono l'esplicita definizione del formato usato per i dati. I sottoprogrammi del FORTRAN utilizzano meccanismi meno sofisticati di quelli del PASCAL per il passaggio dei parametri e l'uso di variabili locali. Tuttavia i sottoprogrammi FORTRAN hanno il grande pregio di poter essere compilati separatamente. Ciò ha consentito lo sviluppo di librerie vastissime che costituiscono uno degli elementi fondamentali per la diffusione del linguaggio.

In questa dispensa verranno esaminati gli aspetti principali del linguaggio FORTRAN 77 e in particolare del Microsoft Professional Fortran, assumendo che il lettore abbia una buona familiarità con i linguaggi di programmazione imperativi, ed in particolare con il PASCAL. Per una trattazione completa del FORTRAN 77 si veda ad esempio [1,2,3].

1. 2. CONSIDERAZIONI GENERALI SUL FORTRAN

Nella breve descrizione del FORTRAN e nel suo confronto con il PASCAL, contenuti in questa dispensa, verranno evidenziati numerosi aspetti che differenziano i due linguaggi. I limiti principali del FORTRAN riguardano gli strumenti offerti per la tipizzazione e la strutturazione dei dati, la gestione completamente statica della memoria, alcuni aspetti della programmazione strutturata che ancora stentano a diffondersi nel linguaggio. Le caratteristiche del FORTRAN che lo hanno reso e che lo rendono ancora così popolare sono essenzialmente le seguenti:

1. la possibilità di usare numeri interi e reali con un elevato numero di cifre significative;

2. la possibilità di utilizzare i numeri complessi e l'aritmetica complessa;

3. l'esistenza di un gran numero di funzioni matematiche di libreria utilizzabili direttamente in qualsiasi punto di un programma o di un sottoprogramma;

4. la possibilità di costruire una propria libreria di sottoprogrammi già compilati e provati, riutilizzabili per la risoluzione di problemi diversi da quelli per cui sono stati sviluppati.

5. la possibilità di richiamare FUNCTION e SUBROUTINE scritte in linguaggi diversi dal FORTRAN.

Queste caratteristiche hanno favorito lo sviluppo di vaste librerie matematiche distribuite gratuitamente da Università e Centri di Ricerca oppure distribuite commercialmente da ditte produttrici di software. L'esistenza di tali librerie rende tuttora particolarmente attraente l'uso del FORTRAN per tutte le applicazioni numeriche.

1 .

3. LA STRUTTURA DI UN PROGRAMMA FORTRAN

La sintassi del FORTRAN è rimasta in parte vincolata all'uso delle schede, anche ora che il programma viene normalmente redatto da terminale tramite un editor. In FORTRAN una istruzione è di solito costituita da una linea del programma di lunghezza al massimo 80 caratteri. Una linea contiene informazioni valide fino alla colonna 72, le colonne 73-80 non vengono considerate dal compilatore. Le prime cinque colonne rappresentano il campo etichetta, la colonna 6 va riempita con un qualsiasi carattere se l'istruzione precedente continua sulla riga attuale, altrimenti va lasciata bianca, le colonne 7-72 sono utilizzabili per la scrittura delle istruzioni. Una "C" o un "*" in colonna 1 indicano una riga di commento.

Un programma FORTRAN è quindi costituito da una sequenza di istruzioni. L'ordine delle istruzioni in un programma FORTRAN è meno rigido che in PASCAL, tuttavia si segue il principio di dichiarazione prima dell'uso. Per descrivere la sintassi delle istruzioni verrà utilizzata una Backus Naur Form (BNF) estesa, del tipo di quella descritta in [4].

Il programma inizia in genere con una

istruzione-programma = "PROGRAM" *nome-programma*.

in cui *nome-programma* è un identificatore. L'istruzione programma può però essere omessa, anche se questo può facilmente dar luogo ad errori, specie se si hanno più programmi "anonimi" nello stesso file. Gli identificatori sono, come in PASCAL, definiti come sequenze di lettere o cifre che iniziano con una lettera; si richiede tuttavia che la loro lunghezza sia limitata a sei o otto caratteri e non esiste, in genere, distinzione tra lettere minuscole e maiuscole.

Seguono le istruzioni del programma che si possono suddividere in istruzioni dichiarative che corrispondono alle dichiarazioni di un programma PASCAL ed istruzioni eseguibili che corrispondono al corpo di un programma PASCAL. Le istruzioni dichiarative possono essere ordinate a piacere, purché precedano le istruzioni eseguibili. Le principali istruzioni dichiarative riguardano la dichiarazione di costanti e/o di variabili e la condivisione di memoria.

Dopo le istruzioni dichiarative vengono poste le istruzioni eseguibili del programma, tra cui: istruzioni di assegnazione, condizionali, cicli, istruzioni di salto, istruzioni di ingresso/uscita ed istruzioni di chiamata. Infine segnaliamo che le specifiche di formato di stampa FORMAT possono essere poste in qualsiasi punto del programma.

L'ultima istruzione deve essere una istruzione END, che segnala al compilatore la fine del testo del programma.

1 .

4. I TIPI DI DATO

Il FORTRAN consente il trattamento di dati interi, reali, complessi, logici e di tipo carattere. Due sono i tipi strutturati: i vettori e le stringhe di caratteri. In FORTRAN non è possibile definire tipi di dato.

a) 4.1 Tipi di dato semplici

Interi

I numeri interi si scrivono come nel PASCAL. Il campo dei valori ammessi tuttavia può variare ed essere in generale più ampio di quelli offerti dal PASCAL.

intero = [segno] intero-senza-segno
segno = "+" | "-"
intero-senza-segno = cifra {cifra}
cifra ="0" | "l" | ... | "9"

Reali

I numeri reali si scrivono sia con la notazione in virgola fissa che con la notazione in virgola mobile. La sintassi è la seguente:

reale = [segno] reale-senza-segno
reale-senza-segno = [intero-senza-segno] "." *intero-senza-segno [esponente]* |
intero-senza-segno "." *[esponente]* |
intero-senza-segno esponent.
esponente = "E" *intero* | "D" *intero*

I seguenti sono esempi di numeri reali ammessi dal FORTRAN:

.123 45.34 -0.32 -.32 13. 4.13E12 413E10 413E-10

Si noti che nella forma a virgola fissa a differenza del PASCAL sono ammessi i numeri senza cifre prima o dopo il punto decimale, e lo stesso vale per il numero che precede l'esponente; inoltre può comparire la lettera D al posto della E nell'esponente. Le cifre che descrivono un reale (non l'esponente) sono chiamate cifre significative. Il loro numero è vincolato alla rappresentazione fisica utilizzata nella memoria. E' possibile aumentare l'accuratezza (o la precisione) della rappresentazione usando i reali in precisione multipla (vedi sez. 2.2). Nella forma a virgola fissa, una costante reale in precisione multipla si distingue unicamente dal numero di cifre; in quella a virgola mobile occorre usare per l'esponente il carattere "D".

Complessi

I numeri complessi sono costituiti da coppie di reali o interi racchiuse fra parentesi:

complesso = "(" (intero | reale) "," (intero | reale) ")".

I seguenti sono esempi di numeri complessi ammessi dal FORTRAN:

il numero complesso 3.14 + 0.000736**i** viene rappresentato con (3.14,.763E-3)
il numero complesso 3 + 4**i** viene rappresentato con (3, 4)

Espressioni

In FORTRAN le espressioni numeriche possono essere formate usando i seguenti operatori:

OPERATORE	SIGNIFICATO
+	Addizione
-	Sottrazione
*	Moltiplicazione
/	Divisione
**	Elevamento a potenza

L'ordine di priorità di esecuzione delle operazioni è quello usuale con l'elevamento a potenza che precede moltiplicazione e divisione che, a loro volta, precedono somma e sottrazione; inoltre operazioni con lo stesso ordine di priorità vengono eseguite da sinistra verso destra, eccetto nel caso dell'elevamento a potenza. Per alterare l'ordine di esecuzione si usano le parentesi tonde.

Si noti che, a differenza del PASCAL non vi è distinzione tra divisione di interi e di reali. Occorre quindi fare attenzione all'uso dell'operatore "/" che, quando gli operandi sono interi, ritorna un valore intero.

Analogamente al PASCAL viene fatta una conversione di tipo nelle espressioni in cui compaiono termini di tipo numerico non omogenei. In PASCAL il caso che si presenta è quello di espressioni con operandi interi e reali che forniscono risultati reali. Ad esempio la somma di un intero e di un reale dà un risultato reale. In FORTRAN vi sono un maggior numero di casi. La gerarchia dei tipi è la seguente:

interi < reali < complessi

Ad esempio la somma di un intero e di un complesso dà un risultato complesso, così come la somma di un reale e di un complesso. Nella conversione da intero o reale a numero complesso si pone la parte immaginaria a zero. Con l'elevamento a potenza un complesso non può mai comparire come esponente. Si noti inoltre che le regole di conversione dei tipi vengono applicate seguendo l'ordine di valutazione dell'espressione e questo può dare con l'operatore di divisione risultati inattesi. Ad esempio sia data l'espressione 9/5*1.0+32. Il risultato è il numero reale 33 e non 33.8, poiché il calcolo di 9/5 produce un risultato intero.

Gli operatori relazionali del linguaggio sono analoghi a quelli del PASCAL e sono riportati nella seguente tabella:

FORTRAN	PASCAL
.EQ.	=
.NE.	<>
.GT.	>
.LT.	<
.LE.	<=
.GE.	>=

Boolean

Il tipo boolean che in FORTRAN si chiama LOGICAL, è del tutto analogo a quello del PASCAL. Le costanti hanno la seguente rappresentazione:

boolean = ".TRUE." | ".FALSE.".

Il linguaggio consente anche l'uso dei seguenti operatori logici (la priorità di esecuzione è dall'alto verso il basso):

OPERATORE	SIGNIFICATO
.NOT.	negazione
.AND.	congiunzione
.OR.	disgiunzione

L'esecuzione di espressioni con operatori eterogenei rispetta la seguente tabella di priorità (dall'alto verso il basso):

operatori aritmetici
operatori relazionali
operatori logici

Ad esempio il valore dell'espressione

Il+I2.GT.100.AND.B.OR.NOT.(A.OR.C.AND.D)

viene calcolato nel modo seguente. Dapprima viene calcolata la somma fra le due variabili intere I1 e I2 per confrontarla con la costante intera 100, il valore logico ottenuto viene usato nella congiunzione con la variabile logica B. Il valore finale è dato dal risultato della disgiunzione del valore sin qui calcolato e del valore ottenuto negando il risultato dell'espressione logica fra parentesi, cioè della disgiunzione della variabile A con il risultato della congiunzione di C e D.

Caratteri

Il tipo carattere del FORTRAN è molto simile a quello del PASCAL. Le costanti di tipo carattere si scrivono tra apici: ad esempio 'A' indica il carattere a maiuscola; il carattere apice va scritto due volte ''. Il set di caratteri cui si fa riferimento è normalmente il set dei caratteri ASCII, tuttavia per motivi di standardizzazione si fa anche riferimento al cosiddetto set dei caratteri FORTRAN, che comprende:

- i 26 caratteri alfabetici maiuscoli dell'alfabeto inglese;
- le dieci cifre;
- ' ', '=', '+', '-', '*', '/', '(', ')', ',', '.', '$', '"', ':', ';'

La relazione di ordinamento definita sui caratteri prevede che le lettere alfabetiche e le cifre siano nel loro ordine naturale con le cifre o tutte prima o tutte dopo le lettere, e che il carattere spazio preceda tutte le cifre e tutte le lettere. Questa restrizione è importante per la gestione delle stringhe (vedi sez. 2.4). Ad esempio la condizione 'A' .LT. 'B' è vera, mentre 'A' .LT.' ' è falsa.

a) 4.2 Dichiarazioni

In questa sezione vengono esaminate le principali dichiarazioni del FORTRAN, cioè dichiarazioni di variabili, di costanti e di condivisione di memoria. Non vi è un ordine rigido nella specifica delle dichiarazioni, tuttavia le dichiarazioni di costanti sono da intendere come il blocco (fissaggio) del valore di una variabile ed in quanto tali in genere seguono le

dichiarazioni di variabili.Tuttavia una costante può essere usata in una dichiarazione di variabile di tipo vettore o stringa ed in quel caso ovviamente la dichiarazione della costante deve precedere l'uso della costante stessa. Le dichiarazioni di condivisione di memoria in genere seguono le dichiarazioni di variabile in quanto possono fare ad esse riferimento.

Dichiarazione di variabili

Le variabili possono essere dichiarate in modo implicito o esplicito. Nelle dichiarazioni implicite il tipo viene definito dalla lettera iniziale del nome della variabile. Cioè se il nome inizia con:

- I,J,K,L,M,N viene assunto come identificatore d'una variabile intera;
- in ogni altro caso viene assunto come l'identificatore d'una variabile reale.

Le dichiarazioni esplicite di variabili per i tipi semplici sono realizzate da istruzioni che hanno la seguente forma:

dichiarazione-variabile = tipo nome-var {“,” nome-var}.

in cui il *tipo* è

- INTEGER per dichiarare variabili intere;
- REAL per variabili reali;
- COMPLEX per le variabili complesse;
- LOGICAL per le variabili booleane;
- CHARACTER per le variabili di tipo carattere.

Dopo il tipo viene specificata una lista di identificatori che rappresentano i nomi delle variabili dichiarate di quel tipo. Non è possibile usare lo stesso nome per due variabili di tipo diverso, anche se in qualche caso questo può essere tollerato dal compilatore per compatibilità con le precedenti versioni del linguaggio. Per esempio le istruzioni

```
INTEGER RESTO
REAL IDl, ID2
COMPLEX SPETTRO
```

definiscono le variabili: RESTO di tipo intero, IDl ed ID2 reali e SPETTRO complessa. L'uso di dichiarazioni implicite rende il programma meno leggibile e consente di introdurre errori non individuabili dal compilatore, poiché una variabile non dichiarata esplicitamente viene considerata di tipo reale o intero sulla base della lettera iniziale.
E’ possibile inibire l’uso delle dichiarazioni implicite utilizzando l’istruzione

```
IMPLICIT NONE
```

come prima istruzione dichiarativa del programma.

Precisione multipla

In FORTRAN è possibile definire variabili di tipo intero e reale aventi a disposizione una quantità di memoria supplementare (in genere doppia) per la memorizzazione dei valori, specificando il numero di byte da utilizzare per memorizzare una variabile. Ad esempio

```
REAL*8 DA
INTEGER*8 K
```

La prima dichiarazione definisce una variabile reale DA i cui valori vengono memorizzati utilizzando 8 byte, la seconda una variabile intera K i cui valori vengono memorizzati utilizzando 8 byte

Per le variabili in precisione multipla vale quanto detto per le variabili di tipo reale. Nelle espressioni miste con operatori di tipo intero e reale il risultato sarà in precisione multipla se uno degli operandi è in precisione multipla.

Per ragioni di compatibilità con le precedenti versioni del FORTRAN viene ancora accettata la dichiarazione DOUBLE PRECISION che permette di dichiarare variabili reali definite su 8 byte.
Ad esempio l'istruzione dichiarativa 1 può essere sostituita da

```
DOUBLE PRECISION DA
```

Dichiarazione di costanti

In FORTRAN è possibile dichiarare costanti attraverso l'istruzione PARAMETER, che ha la seguente sintassi:

istr-parametrica = "PARAMETER" "(" *def-costante {"," def-costante}* ")".
def-costante = nome-costante "=" *espressione-costante*.

Il tipo della costante è definito se il *nome-costante* è il nome di una variabile precedentemente dichiarata, altrimenti esso viene derivato dalle regole di dichiarazione implicita delle variabili. Ad esempio:

```
PARAMETER (PI=3.14)
```

definisce una costante di tipo reale con valore 3.14.
Una *espressione-costante* può contenere delle costanti definite in precedenza. In questo modo risulta possibile, a differenza del PASCAL, definire una costante in termini di altre costanti.

a) 4.3 Il tipo strutturato vettore

Il principale tipo strutturato offerto dal FORTRAN, il tipo vettore, è molto simile al tipo array del PASCAL e consente di definire collezioni di dati di tipo omogeneo, ai quali si può accedere individualmente attraverso il meccanismo dell'indice.
I vettori del FORTRAN si dichiarano tramite le istruzioni per la dichiarazione di variabili di tipo semplice in cui si aggiunge la specifica delle dimensioni, la cui sintassi è la seguente:

specifica-dimensioni = "(" *specifica-indice* {"," *specifica-indice*} ")"
specifica-indice = espressione-costante | espressione-costante ":" *espressione-costante*.

Nella forma più semplice le dimensioni vengono specificate con una lista di *espressione-costante* che devono essere di tipo intero, racchiuse tra parentesi tonde e separate da virgole, una per ogni dimensione. In questo caso l'indice del vettore assume i valori da 1 all'intero specificato. Ad esempio:

```
INTEGER C(3,4)
REAL*8 MATRIX(4,4,4)
```

dichiara una variabile C come un vettore bidimensionale di elementi di tipo intero in cui gli indici delle righe vanno da 1 a 3 e quelli delle colonne vanno da 1 a 4 ed una variabile MATRIX come un vettore tridimensionale di 4x4x4 elementi reali in precisione multipla.
E' inoltre possibile specificare l'estremo inferiore e superiore di ogni dimensione del vettore scrivendo due costanti intere separate dai due punti. Ad esempio:

```
LOGICAL B(-1:1)
```

dichiara un vettore di elementi di tipo boolean in cui l'indice va da -1 ad 1.

Per ragioni di compatibilità con le precedenti versioni del FORTRAN, esiste ancora la possibilità di dichiarare vettori, tramite l'istruzione DIMENSION, mentre la dichiarazione del tipo degli elementi può essere implicita o esplicita. Ad esempio le istruzioni dichiarative

```
INTEGER C
DIMENSION C(3,4), I(32)
```

dichiarano un vettore C bidimensionale 3x4 con elementi interi, ed un vettore I monodimensionale di elementi di tipo intero definito implicitamente. Si noti che la dichiarazione di C è equivalente a quella riportata nell'esempio precedente.

Ai singoli componenti di un vettore si fa riferimento attraverso il nome del vettore e la specifica dell'indice racchiusa fra parentesi tonde. Ad esempio gli elementi di un vettore bidimensionale 2x2 di nome A, sono selezionati con le seguenti notazioni:

```
A(1,1)    A(1,2)
A(2,1)    A(2,2)
```

Il valore dell'indice è dato dal risultato di una espressione intera. La seguente denotazione di un elemento di A è corretta purchè le variabili I,J,K siano variabili intere e forniscano al momento dell'esecuzione dei valori ammissibili per gli indici del vettore:

A(I/J,4-K)

Si noti che in FORTRAN non sono definite nè le istruzioni di assegnazione su variabili di tipo vettore (con l'eccezione delle istruzioni di inizializzazione di tipo DATA non trattate in questa dispensa), nè istruzioni di confronto. Per le modalità di ingresso/uscita dei vettori si rimanda alla sez. 4.3.

a) 4.4 Il tipo stringa

In FORTRAN è disponibile un tipo di dato stringa di caratteri, che è da considerare in relazione al tipo carattere piuttosto che come caso particolare del tipo vettore, come nel PASCAL. I valori del tipo stringa sono racchiusi tra apici e le variabili del tipo stringa si dichiarano attraverso l'istruzione CHARACTER in cui si specifica la lunghezza della stringa, nei modi seguenti:

CHARACTER* *lunghezza lista-nomi*
CHARACTER *nome1*lunghezza1, nome2*lunghezza2, ...*

in cui *lunghezza* è un valore intero che rappresenta la lunghezza della stringa. Ad esempio le due istruzioni

```
CHARACTER*8 A, B
CHARACTER C*5, D*8
```

dichiarano complessivamente quattro variabili di tipo stringa: A e B di 8 caratteri, C di 5 e D di 8.

<u>Operazioni sulle stringhe</u>

Il tipo stringa del FORTRAN consente di costruire stringhe attraverso l'operatore di concatenazione, di selezionare sottostringhe, di confrontare stringhe tramite gli operatori di confronto, di fare assegnazioni a variabili di tipo stringa. Sono inoltre definite modalità particolari per l'ingresso/uscita delle stringhe per le quali si rimanda al cap 4.

Concatenazione

La concatenazione, che si indica con il simbolo "//", permette di costruire una stringa mettendo in sequenza i caratteri delle due stringhe in ingresso. Ad esempio l'espressione 'AB' // 'CD' restituisce come valore la stringa 'ABCD'.

Selezione di sottostringhe

La selezione di una sottostringa si realizza specificando i due estremi della sottostringa tra parentesi tonde, separati dai due punti. Se uno dei due estremi non è specificato si intende l'inizio o la fine della stringa, rispettivamente. Quando i due estremi coincidono viene selezionato un valore di tipo carattere. Ad esempio sia A una variabile di tipo stringa dichiarata come sopra alla quale è stato assegnato il valore 'ABCDEFGH' L'espressione A(2:4) seleziona la sottostringa di tre caratteri 'BCD'. L'espressione A(:2) restituisce 'AB', A(5:5) restituisce 'E' e A(4:3) restituisce la stringa vuota.

Confronto

Il confronto tra le stringhe si basa sulla relazione di ordinamento definita sul tipo carattere. Il confronto è ammesso anche fra stringhe di lunghezza diversa.

E' necessario ricordare che:

- i caratteri delle stringhe vengono confrontati ad uno ad uno da sinistra verso destra;
- l'ordinamento dei caratteri e' quello alfabetico o quello delle cifre decimali;
- le cifre decimali precedono le lettere maiuscole che precedono le lettere minuscole
- lo spazio precede tutti i caratteri stampabili;
- il confronto tra stringhe non e' possibile se esse contengono caratteri diversi dalle lettere dell'alfabeto maiuscolo o minuscolo, dalle cifra decimali e dallo spazio bianco;
- se una delle stringhe e' piu' corta dell'altra viene completata con spazi bianchi a destra;

Esempio

'Adamo' .GT. 'Eva'	e' falso
'10' .GT. '9'	e' falso
'10' .LT. '11'	e' vero
'Eva' .LT. 'Evanio'	e' vero
'Alba' .LT. 'alba'	e' vero
'1alba' .GT. 'Alba'	e' falso

Assegnazione

L'assegnazione (vedi sez 3.1) di un valore di tipo stringa ad una variabile di tipo stringa prevede il troncamento nel caso in cui il valore di tipo stringa abbia un numero di caratteri superiore a quello della variabile, ed il riempimento con spazi nel caso in cui tale numero sia inferiore.Ad esempio l'istruzione

```
A ='abc'
```

assegna alla variabile A definita come sopra la stringa 'abc ', con 5 spazi nelle ultime cinque posizioni; l'istruzione

```
C = 'abcdefg'
```

assegna alla variabile C definita come sopra la stringa 'abcde', tralasciando gli ultimi due caratteri.

Funzioni intrinseche che operano sul tipo stringa

E' possibile operare su dati del tipo carattere e del tipo stringa con apposite funzioni intrinseche del FORTRAN:

Nome della funzione	Interpretazione
ICHAR(character)	Funzione intera che calcola il numero d'ordine del carattere nella sequenza ASCII
CHAR(I) (0<=I<128)	Funzione di tipo carattere che restituisce il carattere che si trova in posizione I nella sequenza ASCII
LEN(stringa)	Funzione intera che restituisce la lunghezza della stringa
INDEX(stringa1,stringa2)	Funzione intera che restituisce il valore della posizione iniziale di stringa2 in stringa1. *Se stringa2 non e' una sottostringa di stringa1 restituisce 0. Se stringa2 compare piu' di una volta fornisce l'indice della prima occorrenza da sinistra.*

Esempi

alfa = 'a'	
ind = ICHAR(alfa)	*ind assume il valore* 97
alfa = CHAR(72)	alfa assume il valore 'H'
k = LEN('abcdef')	k assume il valore 6
CHARACTER*10 beta	
j = LEN(beta)	j assume il valore 10, qualunque sia il contenuto di beta
n=INDEX('abcdef', 'cd')	n assume il valore 3
h=INDEX('abcdef', 'b')	h assume il valore 2 che rappresenta la posizione del carattere 'b' nella stringa
m=INDEX('abcde', 'g')	m assume il valore 0
L =INDEX('abcab', 'ab')	L assume il valore 1

a)

4.5 L'istruzione dichiarativa DATA

L'istruzione DATA effettua l'assegnazione di valori iniziali alle variabili contenute nella lista *varlist*
La forma e'

istruzione_ data = "DATA" *varlist1/vallist1/,varlist2/vallist2/*

```
DATA SUM/0.0/
DATA A, B, N/1.0, 2.0, 10/, NOME/'mio'/
```

Se piu' variabili debbono assumere lo stesso valore iniziale e' possibile indicare tale valore una sola volta specificandone la molteplicita'.

```
DATA I, J, K, L/ 4*0/
```

L'istruzione DATA deve seguire le istruzioni dichiarative e di solito precede le istruzioni esecutive.

1 .

5. LE ISTRUZIONI ESEGUIBILI DEL FORTRAN

In questo capitolo vengono esaminate le principali istruzioni eseguibili di un programma FORTRAN ad eccezione di quelle di ingresso/uscita, e di quelle di chiamata descritte nei capp. 4 e 5 rispettivamente. In particolare si considerano l'istruzione di assegnazione e le istruzioni di controllo, cioè quelle che determinano l'ordine di esecuzione delle istruzioni. L'ordine in cui vengono eseguite le istruzioni nel programma è sequenziale come in PASCAL e, nei FORTRAN strutturati, come quello in esame, le differenze dal PASCAL relativamente alle istruzioni di controllo sono di poca importanza.

Le istruzioni eseguibili (ed altre istruzioni come ad esempio le definizioni di formato) possono avere un' *etichetta* costituita da un numero intero specificato nelle prime cinque colonne dell'istruzione.

L'istruzione CONTINUE è un'istruzione senza alcun effetto, corrispondente all'istruzione nulla del PASCAL. Essa viene di solito usata per indicare la chiusura di un ciclo specificato da un'istruzione DO, di cui si parlerà nel seguito.

L'istruzione STOP determina l'arresto dell'esecuzione, cioè la terminazione del programma. Deve comparire almeno una istruzione di questo genere in ogni programma. A differenza del PASCAL questa istruzione permette di terminare l'esecuzione in un punto arbitrario del programma.

a) 5.1 Istruzioni di assegnazione

L'assegnazione è l'istruzione basilare che consente di modificare i valori delle variabili. La sintassi, in FORTRAN, è la seguente:

i*struzione-assegnazione = variabile* "=" *espressione*.

dove *variabile* è il nome di una variabile di tipo semplice oppure di un singolo elemento di vettore, oppure di tipo stringa. Normalmente il valore dell'espressione deve essere dello stesso tipo della variabile. Tuttavia vi sono numerose eccezioni che prevedono una conversione automatica del tipo dell'espressione per adattarlo a quello della variabile. Nel caso che il tipo della variabile sia intero e l'espressione sia reale, viene considerata la parte intera del valore dell'espressione (mentre in PASCAL il troncamento della variabile intera deve essere indicato esplicitamente dal programmatore). Se la variabile è reale mentre l'espressione è in precisione multipla, la rappresentazione di quest'ultima viene convertita per troncamento in precisione semplice. Se la variabile è complessa, mentre l'espressione è reale, viene modificata solo la parte reale e la parte immaginaria viene azzerata. Infine, se la variabile è reale o intera e l'espressione di tipo complesso, viene assegnata solo la parte reale (previa eventuale conversione per troncamento). Ad esempio:

```
N1(3,L)=ALFA(I)+BETA(I,K)*9.81
```

All'elemento L-esimo della terza riga di N1, viene assegnato il valore della somma dell'I-esimo elemento di ALFA con il prodotto fra la costante reale 9.81 e il K-esimo elemento della I-esima riga di BETA.

Si ricorda che nel caso delle stringhe è possibile assegnare un valore all'intera stringa (vedi sez. 2.4), mentre non è possibile assegnare un valore di tipo vettore ad una variabile.

a) 5.2 Istruzioni condizionali

La principale forma di condizionale del FORTRAN è l'IF-THEN-ELSE che è molto simile all'analoga del PASCAL. La sintassi è la seguente

blocco-if-then-else = "IF" *condizione* "THEN"
sequenza-istruzioni
"ELSE"
sequenza-istruzioni]
"ENDIF".

La differenza principale con il PASCAL è dovuta alla struttura in linee di un programma FORTRAN che rende lo IF-THEN-ELSE un blocco di istruzioni anziché una singola istruzione. Infatti la linea:

IF *condizione* THEN

costituisce una istruzione, così come la linea

ENDIF

Ad esempio

```
IF ((L-M).GE.0) THEN
        L=L-M
ELSE
        M=M-L
ENDIF
```

L'istruzione ENDIF è necessaria per via della mancanza dell'istruzione composta; l'istruzione ENDIF serve infatti a delimitare il blocco delle istruzioni del ramo-then nella forma senza ELSE e altrimenti del ramo-else. I blocchi IF-THEN-ELSE possono essere annidati in modo arbitrario ed in tal modo è possibile realizzare la selezione a più vie.

Quando la prima istruzione di un ramo-else è un blocco IF-THEN-ELSE questa si può scrivere su una singola linea:

ELSE IF *condizione* THEN
……
ENDIF

Ad esempio

```
……————————————
IF (N.LT.0) THEN
      WRITE(*,*) ‘ numero negativo’
ELSE IF (N.EQ.0) THEN
      WRITE(*,*) ‘ zero’
ELSE
      WRITE(*,*) ‘ numero positivo’
ENDIF
……————————————
```

Una ulteriore istruzione condizionale è l'IF-logico, presente in origine nel FORTRAN, e mantenuta per ragioni di compatibilità.

istruzione-if-logico = "IF" *condizione istruzione-ristretta.*

Si tratta di una istruzione IF-THEN con una sola istruzione sul ramo-then, per di più ristretta, nel senso che non tutte le istruzioni sono ammesse.

Ad esempio

```
IF(X.LT.0)     X=-X
IF(N.GT.Nmax) Nmax=N
```

a) 5.3 Esempio di uso delle istruzioni di assegnazioni e di scelta

Un programma FORTRAN per il calcolo delle radici reali di un'equazione di secondo grado a coefficienti reali.

Data l'equazione

$$ax^2 + bx + c = 0$$

sappiamo che le radici si ottengono applicando la formula seguente

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

Se a≠0 (equazione propria) l'espressione

$$d = \sqrt{b^2 - 4ac}$$

rappresenta il discriminante

```
************************************************
        PROGRAM Radici
        PARAMETER (Epsi=1E-20)
        IMPLICIT NONE
        REAL a, b, c, x1, x2, D,
        WRITE(*,*)'Immetti i tre coefficienti reali'
        WRITE(*,*)'Il coefficiente a deve essere ≠0'
        READ(*,*) a, b, c
        D=b**2-a*c
        IF (D.GT.Epsi) THEN
            x1=(-b+SQRT(D))/(a+a)
            x2=(-b-SQRT(D))/(a+a)
            WRITE(*,*)' rad1= ',x1,' rad2= ',x2
        ELSE
            IF (D.GT.-Epsi) THEN
                x1=-b/(a+a)
                WRITE(*,*)' Radice doppia = ',x1
            ELSE
                WRITE(*,*)' Nessuna radice reale'
            ENDIF
        ENDIF
        STOP
        END
************************************************
```

a) 5.4 Istruzioni di ciclo

Le forme di ciclo disponibili in FORTRAN sono il ciclo definito realizzato attraverso l'istruzione-DO e il ciclo WHILE. Quest'ultimo non è tuttavia fornito da tutte le implementazioni; ove assente, per esprimere i cicli indefiniti occorre usare le istruzioni di salto illustrate nella sez. 3.4.

Istruzione-DO

La sintassi dell'istruzione-DO è:

istruzione-DO = "DO" *etic* [","] *var* "=" *espr1* "," *espr2* [","*passo*].

Il corpo del DO, cioè la sequenza di istruzioni da eseguire iterativamente, è costituito dalle istruzioni comprese tra quella successiva al DO e quella etichettata con *etic*. Poiché vi sono delle restrizioni sul tipo di istruzione in cui specificare l'etichetta, di solito si preferisce introdurre per la chiusura del ciclo un'istruzione CONTINUE su cui porre l'etichetta.

La variabile *var* è normalmente una variabile intera che prende il nome di variabile del ciclo, *espr1* ed *espr2* sono due espressioni anch'esse di tipo intero, *passo* è una qualunque espressione intera e se non specificato è considerato unitario. Il FORTRAN permette di usare anche variabili del ciclo di tipo reale; per questi casi si rimanda ai testi citati in bibliografia.

Se il passo è unitario l'istruzione DO si comporta come l'istruzione FOR del PASCAL in cui *espr1* ed *espr2* rappresentano il valore iniziale e quello finale della variabile del ciclo; se il passo è -1 si ottiene il caso in cui il FOR del PASCAL usa la forma DOWNTO. Se il passo è specificato altrimenti occorre calcolare il valore dell'espressione

(espr2 - espr1+ passo)/passo

al momento dell'ingresso nel ciclo. Se tale valore è minore od uguale a zero il corpo del ciclo viene eseguito 0 volte, altrimenti il valore dell'espressione fornisce il numero di volte che viene eseguito il corpo del ciclo. La variabile del ciclo inizialmente assume il valore di *espr1* e viene incrementata di *passo* ad ogni esecuzione del ciclo. Il ciclo termina quando il suo corpo è stato eseguito un numero di volte pari al valore calcolato inizialmente.

In conformità con i principi della programmazione strutturata, all'interno del corpo del ciclo, non si deve modificare il valore della variabile indice, nè il valore dell'espressione finale e nè del passo; inoltre non si deve entrare nel ciclo con un'istruzione di salto, mentre è possibile che un'istruzione del corpo del ciclo provochi un salto all'esterno.
Ad esempio consideriamo il seguente programma per il calcolo del fattoriale.

```
      PROGRAM FATTOR
      IMPLICIT NONE
      INTEGER FATT,N, I

C  Si supponga di leggere un intero in
C  ingresso e di memorizzarlo in N

      FATT=1
      DO 100 I=1,N
            FATT=FATT*I
100   CONTINUE

C  Si supponga di stampare FATT

      STOP
```

Nel programma mancano le istruzioni di ingresso-uscita, al loro posto sono stati inseriti dei commenti.

Ciclo-WHILE

La sintassi del WHILE che, come detto in precedenza, non sempre è disponibile nei compilatori del FORTRAN 77, ma è implementato nel Microsoft Fortran :

1) blocco-WHILE = "DO WHILE" "(" *condizione* ")"
sequenza-istruzioni
"ENDDO".

L'effetto dell'istruzione è del tutto analogo a quello dell'istruzione WHILE del PASCAL. Si noti che, come il condizionale IF-THEN-ELSE, se un ciclo WHILE è costituito da un blocco di istruzioni e non da una singola istruzione, occorre il simbolo speciale ENDDO per terminare la sequenza di istruzioni del corpo del ciclo, in assenza dell'istruzione composta.

Ad esempio consideriamo di nuovo un programma per il calcolo del fattoriale.

```
      PROGRAM FATTOR
      INTEGER FATT, N, I

C  Si supponga di leggere un intero in
C  ingresso e di memorizzarlo in N
      READ(*,*) N
      FATT=1

      DO 100 I=1,N
         FATT=FATT*I

100   CONTINUE
C  Si supponga di stampare FATT
      WRITE(*,*) FATT
      STOP
```

```
      PROGRAM FATTOR1
      INTEGER FATT, N, I

      READ(*,*) N
      FATT=1
C  Dobbiamo inizializzare anche la variabile I
      I=1
      DO WHILE (I.LE.N)
         FATT=FATT*I
         I=I+1
C  Dobbiamo incrementare la variabile I
      ENDDO

      WRITE(*,*) FATT
      STOP
```

a) 5.5 Esempio di uso delle istruzioni di ciclo

Un programma in FORTRAN per la ricerca di un elemento (Goal)in un vettore ordinato (Elem) usando l'algoritmo di bisezione (ricerca binaria).

```
*************************************************
      PROGRAM CercaOrd
      PARAMETER (N=10)
      IMPLICIT None
      INTEGER Elem(N), Goal, I, Inf, Sup, Med
      LOGICAL Trovato
      READ(*,*)(Elem(I) I=1,N)
      READ(*,*) Goal
      Trovato=.False.
      Inf=1
      Sup=N
***se Inf>=Sup non è presente
      DO WHILE((Inf.Lt.Sup).And.(.Not. Trovato))
***calcolo l'indice mediano
            Med=(Inf+Sup)/2
***confronto con l'elemento di indice mediano
            IF(Goal.Eq.Elem(Med)) THEN
***se uguale ho finito
                  Trovato=.True.
***se minore cerco nella parte inferiore
            ELSEIF(Goal.Lt.Elem(Med))
                  THEN
                        Sup=Med-1
***se maggiore cerco nelle parte superiore
                  ELSE
                        Inf=Med+1
            ENDIF
      ENDDO
      IF (Trovato) THEN
            WRITE(*,*) Med,Elem(Med)
      ELSE
            WRITE(*,*)'elemento non presente'
      STOP
      END
*************************************************
```

a) 5.6 Istruzioni di salto

In FORTRAN l'ordine di esecuzione delle istruzioni è strettamente sequenziale e tutte le volte che il programmatore lo vuole alterare, può usare una istruzione di salto che specifica l'istruzione da eseguire successivamente. L'istruzione principale di salto prende il nome di salto incondizionato ed è analoga al GOTO del PASCAL. La sintassi della istruzione-GOTO è la seguente:

istruzione-goto = "GOTO" *etic.*

L'effetto è di trasferire il controllo all'istruzione di etichetta *etic*.

In FORTRAN le istruzioni di salto sono fondamentali per la programmazione dei cicli indefiniti nei casi in cui non è disponibile il ciclo WHILE. Infatti lo schema:

WHILE condizione DO sequenza-istruzioni

si può esprimere come

```
100     IF (condizione) THEN
                sequenza-istruzioni
                GOTO 100
        ENDIF
```

Analogamente lo schema

REPEAT sequenza istruzioni UNTIL condizione

si può esprimere come

```
100     sequenza-istruzioni
        IF (.NOT. condizione) GOTO 100
```

1 .

6. L'INGRESSO/USCITA

a) 6.1 Istruzioni READ,WRITE e PRINT

Nelle istruzioni di ingresso e uscita (READ,WRITE,PRINT) occorre specificare il codice della periferica addetta alle operazioni , il formato dei dati e i nomi delle variabili in cui i dati debbono essere memorizzati (READ) o da cui debbono essere prelevati (WRITE o PRINT). La descrizione del formato dei dati è contenuta in una particolare istruzione etichettata detta istruzione FORMAT e che verrà descritta nel paragrafo seguente. La sintassi delle istruzioni READ e WRITE è la seguente:

Istruzione-ingresso = "READ" "(" *numero-unità*"," *etichetta-istruzione*-FORMAT ")"
lista-variabili.

Istruzione-uscita = "WRITE" "(" *numero-unità* "," *etichetta-istruzione*-FORMAT ")"
lista-di-uscita.

Esempio:

```
        INTEGER giorno,mese,anno
        READ(5,100) giorno,mese
        WRITE(6,110) anno
100     FORMAT (2I2)
110     FORMAT (I4)
```

La prima istruzione legge dal canale 5 secondo un formato descritto nell'istruzione FORMAT con etichetta 100 e trasferisce il primo valore letto nella variabile giorno e il secondo nella variabile mese. E' prevista la lettura di due interi di due cifre. La seconda istruzione stampa il valore della variabile anno sulla stampante 6 secondo il formato descritto nell'istruzione FORMAT con etichetta 110. E' prevista la stampa di un intero di quattro cifre. Analogamente a quanto avviene per il PASCAL le operazioni di ingresso e uscita leggono e scrivono i dati su canali associati alle periferiche di ingresso e uscita. La specifica dell'unità e/o del formato può essere omessa sostituendo tale specifica con il simbolo *. Ad esempio:

```
READ(*,*) giorno,mese
WRITE(*,*) anno
```

In questo caso come unità di ingresso viene assunta la tastiera, come unità di uscita il video e il formato viene desunto dalle dichiarazioni delle variabili. Ogni istruzione di READ e WRITE con formato standard corrisponde alla lettura o scrittura di una linea.

Alcuni compilatori ammettono anche l'istruzione di uscita PRINT per compatibilità con precedenti versioni del linguaggio. La forma dell'istruzione PRINT è:

PRINT*, *lista -di-uscita*

a) 6.2 Istruzione FORMAT

La sintassi dell'istruzione è la seguente:

Istruzione-di-formato = FORMAT"("*lista-specifiche*")"

Si noti che l'istruzione FORMAT è una istruzione non eseguibile che contiene delle specifiche di un'altra istruzione nella quale è referenziata; pertanto essa deve essere sempre etichettata. La lista delle specifiche descrive le caratteristiche della rappresentazione dei dati.

Tali specifiche sono:

"I" per gli interi
"E" per i reali in forma esponenziale
"D" per i reali in precisione multipla in forma esponenziale
"F" per i reali
"X" per lasciare spazi (bianchi)
"H" per stampare stringhe di caratteri
"A" per leggere stringhe di caratteri
"L" per i booleani

Specifica I**:** "I"n

Dove n è un intero positivo che specifica con quante cifre decimali è rappresentato il dato. Per esempio I4 indica che l'intero é composto di 4 cifre(incluso il segno).

Specifica E**:** "E"n1","n2

Dove n1 è il numero complessivo di simboli utilizzati nella rappresentazione ed n2 quello delle cifre della parte decimale. Occorre tener presente che l'esponente occupa in genere 4 caratteri (nella forma E+cc dove la lettera c denota una cifra decimale) e che 2 caratteri vanno utilizzati per il punto e per il segno, deve quindi essere n1 ≥ n2+6. In generale valgono le seguenti regole:

l) se il segno è + può essere omesso;
2) se il segno dell'esponente è presente può essere omessa la lettera E;
3) l'esponente può avere anche una sola cifra;
4) non devono essere lasciati spazi bianchi alla destra dell'esponente;
5) in uscita l'esponente viene stampato nella forma E+cc oppure +cc (a seconda del compilatore il segno + può essere omesso).

Specifica D: "D"n1","n2

E' analoga alla specifica E ed è utilizzata per i reali in precisione multipla **.**

Specifica F: "F"n1","n2

E' usata per la rappresentazione dei reali, in singola o precisione multipla, in virgola fissa. Gli interi n1 ed n2 hanno lo stesso significato che avevano nelle due specifiche precedenti, in questo caso nl ≥ n2+2. Il formato F produce uscite più facili da leggere, ma richiede la conoscenza precisa delle dimensioni massime per la rappresentazione del dato. Infatti qualora n1 non sia sufficiente per la rappresentazione complessiva, il campo verrà riempito con asterischi. Se in lettura il dato contiene il punto decimale, verrà interpretato indipendentemente dalla specifica FORMAT. Se invece il punto decimale non è presente, viene automaticamente inserito nella posizione indicata dalla specifica.

Specifica X: n"X"

Serve per lasciare n spazi bianchi o per saltare n caratteri in lettura.

Specifica H**:** n"H"

Serve per stampare delle frasi di n caratteri. Dopo la H va scritta immediatamente la frase.E' anche possibile stampare delle frasi omettendo la specifica di formato e racchiudendo il testo tra apici. Ad esempio le due istruzioni seguenti sono equivalenti:

```
        WRITE(*,*) 'Arrivederci a presto'
        WRITE(*,100)
100     FORMAT(20HArrivederci a presto)
```

Specifica A: "A"n

Usata per stringhe di caratteri in ingresso. I caratteri letti vengono immagazzinati nelle variabili specificate dalla istruzione di lettura, senza alcuna conversione.

Specifica L: "L"n

Serve a caratterizzare la rappresentazione dei booleani. In ingresso è sufficiente che il campo indicato dalla specifica inizi con una T o con una F, per assegnare le rappresentazioni relative. In uscita vengono di solito stampate solo delle T o delle F.

Specifiche ulteriori

Nelle dichiarazioni di FORMAT viene di solito usata una barra "/" per indicare che occorre passare alla riga successiva. In tal modo possono essere descritti da un unico FORMAT dei dati contenuti in più righe.

Ad ognuna delle specifiche per rappresentazioni numeriche, può essere premesso un intero positivo avente la funzione di ripetitore. Ad esempio:

```
150     FORMAT (3I4,4F8.4)
```

descrive un formato costituito da 3 interi di 4 cifre e da 4 reali con 2 cifre intere e 4 cifre decimali.

Per leggere o stampare i numeri complessi si usa una coppia di descrittori per reali. Ad esempio:

```
        COMPLEX Z
        ................
        WRITE (8,200) Z
200     FORMAT ( 2F8.4)
```

Se l'uscita viene inviata ad una stampante è necessario utilizzare un FORMAT che non usi il primo carattere della riga. Infatti tale carattere non viene stampato, ma interpretato come carattere di controllo per la stampante stessa. Se viceversa si vuole controllare l'avanzamento delle carta nella stampante è possibile adoperare i seguenti caratteri:

carattere	*interlinea*
(spazio)	una linea
0	due linee
1	pagina successiva
+	stessa linea

Ad esempio i formati

```
100     FORMAT (1H ,I2)
101     FORMAT (1H1,I2)
102     FORMAT (1H+,I2)
```

producono la stampa di un intero di due cifre sulla riga successiva (100), all'inizio della pagina successiva (101) e sulla stessa riga (102).

a)

6.3 Le operazioni di ingresso e uscita per i vettori

La lettura o la stampa di un vettore possono essere effettuate mediante uno o più cicli DO e per alcuni compilatori la stampa può essere effettuata con una sola istruzione. Se il vettore ha più indici, nel caso di operazioni di uscita effettuate con una sola istruzione, varia più rapidamente l'indice di sinistra. Nel caso di vettore bidimensionale ciò corrisponde ad una stampa per colonne. Ad esempio, il seguente frammento di programma:

```
            INTEGER NVET(3,2)
      WRITE (*,50) NVET
50    FORMAT (1H  ,6(I4,3X))
```

stampa il vettore nel seguente ordine

NVET(1,1) NVET(2,1) NVET(3,1) NVET(1,2) NVET(2,2) NVET(3,2)

Un altro valido metodo per l'ingresso/uscita di un vettore consiste nell'uso di DO impliciti. Ad esempio:

```
      INTEGER W(3,2)
      WRITE (*,20)  ((W(I,J), J=1,2), I=1,3)
20    FORMAT (1H  ,2(I4,3X))
```

dove viene richiesta l'uscita del vettore W, stampando una coppia di interi su ogni riga e facendo variare più velocemente l'indice di destra.
Si noti che il seguente frammento non avrebbe lo stesso comportamento:

```
      INTEGER W(3,2)
      DO 10 I=1,3
            DO 10 J=1,2
            WRITE (6,20) W(I,J)
10    CONTINUE
20    FORMAT (1H  , 2(I4,3X))
```

infatti, ad ogni esecuzione del ciclo corrisponde l'uso di una nuova istruzione WRITE, su ogni riga viene stampato un solo intero e l'ordine è quello imposto dalle istruzioni di ciclo.

a) 6.4 Uso di file diversi da quelli standard

In FORTRAN è possibile leggere e scrivere dei dati su file diversi da quelli standard di ingresso/uscita associando ad essi un diverso canale.

I file del FORTRAN possono essere

FORMATTATI	(scritti usando FORMAT)
NON FORMATTATI	(scritti non usando FORMAT)
SEQUENZIALI	(accessibili in modo sequenziale)
AD ACCESSO DIRETTO	(accessibili in modo diretto, non sequenziale) *(E' possibile accedere direttamente all'ennesimo record)*

Le piu' importanti istruzioni per l'uso dei file sono:

READ/WRITE(unit,format,END,ERR)	istruzioni di lettura e scrittura
ENDFILE(lista_info)	istruzione di chiusura del file
OPEN(lista_info)	connessione e apertura di un file
CLOSE(lista_info)	disconnessione e chiusura di un file
BACKSPACE numero_unita'	posizionamento su file
REWIND numero_unita'	posizionamento su file

lista_info:

UNIT = u	*sempre presente*	numero unita'
FILE = fn	*opzionale*	nome file
STATUS = st	*opzionale*	OLD, NEW, ecc
ACCESS = acc	*opzionale*	SEQUENTIAL,DIRECT
END = s	*opzionale*	etichetta
ERR = s	*opzionale*	etichetta
IOSTAT = ios	*opzionale*	variabile

Esempi di uso

OPEN(9,FILE='MIOFILE') collega al programma il file MIOFILE, lo apre e gli assegna il numero di unita' 9

READ(9,111,END=999)(A(I),I=1,100)
legge dall'unita' 9 (file MIOFILE) 100 componenti del vettore, A con il format 111 e, se incontra un end-of-file prima di aver letto tutti gli elementi previsti, va all'istruzione con etichetta 999

WRITE(9,110)(A(I),I=1,100)
scrive sull'unita' 9 (file MIOFILE) 100 componenti del vettore, A con il format 110

BACKSPACE(9) si posiziona sul record che e' stato appena letto o scritto e puo' quindi rileggerlo o riscriverlo.

ENDFILE(9) inserisce un end-of-file nel file 9 dopo l'ultimo record inserito o letto

CLOSE(9) disconnette dal programma il file 9

REWIND(9) si posiziona all'inizio del file 9

Se si scrive un record in una qualsiasi posizione nel file tutti i record successivi vengono cancellati

Se si vogliono scrivere dei dati su un file il cui nome esterno è MIOFILE (se il file appartiene ad un direttorio diverso da quello corrente si deve indicare anche il nome del direttorio) è necessario scrivere le seguenti istruzioni:

```
        OPEN ( 9, FILE = 'MIOFILE' )
        WRITE ( 9, 110 ) lista-variabili
  110   FORMAT ( lista-specifiche )
```

Per chiuderelo stesso file si utilizza l'istruzione

```
        CLOSE (9)
```

I dati saranno memorizzati nel file MIOFILE e potranno essere riletti con un altro programma che utilizzi lo stesso file con lo stesso formato.
L'istruzione CLOSE è opzionale poiché il file viene comunque chiuso al termine dell'esecuzione del programma.

a) 6.5 Esempio di uso dei file

Questo programma legge un file contenente 10 interi, memorizzati 1 per riga, e li scrive, in ordine inverso, in un secondo file

```
            PROGRAM LSFILE
            INTEGER NUM(10), I
*apriamo il file DATI che deve gia' esistere e contenere i 10 interi scritti con un formato noto
            OPEN(3,FILE='DATI',ERR=901)
*apriamo il file RISULTATI che e' vuoto o cancellabile
            OPEN(4,FILE='RISULTATI',ERR=902)
*leggiamo l'intero vettore N dal file DATI con lo stesso formato con cui era stato scritto
            READ(3,101)NUM
*scriviamo sul file RISULTATI
            WRITE(4,101) (NUM(I),I=10,1,-1)
*scriviamo un end-of file sul file RISULTATI
            ENDFILE(4)
*disconnettiamo i due file
            CLOSE(3)
            CLOSE(4)
            STOP
901         WRITE(*,*) 'ERRORE SUL FILE DATI'
            STOP
902         WRITE(*,*) 'ERRORE SUL FILE RISULTATI'
            STOP
101         FORMAT(I10)
            END
```

1 .

7. I SOTTOPROGRAMMI

a) 7.1 Generalità

Nel FORTRAN sono previsti due tipi di sottoprogrammi: le FUNCTION e le SUBROUTINE che corrispondono rispettivamente alle FUNCTION e alle PROCEDURE del PASCAL. In FORTRAN i sottoprogrammi risultano separati dal testo del programma chiamante e vengono quindi compilati separatamente. L'ordine di definizione dei sottoprogrammi non è quindi significativo in FORTRAN.

Le ulteriori differenze nei sottoprogrammi del FORTRAN e del PASCAL nascono dal diverso modo in cui i linguaggi stabiliscono la comunicazione tra programma chiamante e programma chiamato, e dalla mancanza in FORTRAN della ricorsione.

Lo scambio di dati tra programma principale e sottoprogrammi avviene tramite i parametri o tramite l'istruzione COMMON (vedi sez. 5.6). Infatti non esistono variabili globali o non locali.

Analogamente a quanto avviene per il PASCAL, i parametri attuali devono corrispondere in numero, ordine e tipo ai parametri formali e vengono passati con la stessa modalità del passaggio per variabile del PASCAL. Inoltre in FORTRAN non esistono parametri per valore.

I tipi dei parametri debbono essere dichiarati nei sottoprogrammi. In particolare i parametri di tipo vettore o di tipo carattere debbono essere dimensionati sia nel programma chiamante che nei sottoprogrammi chiamati, ma soltanto alle dichiarazioni del programma chiamante corrisponde una effettiva allocazione di memoria.

In FORTRAN esiste anche una particolare istruzione dichiarativa, denominata istruzione dichiarativa funzionale, che consente la definizione di una funzione locale ad un programma (o ad un sottoprogramma). Le funzioni definite in questo modo si utilizzano per l'abbreviazione di espressioni ricorrenti dell'elaborazione.

a) 7.2 Funzioni matematiche predefinite

I compilatori FORTRAN sono in genere dotati di una ampia collezione di funzioni matematiche predefinite (dette anche intrinseche). Tali funzioni sono richiamabili in qualunque punto di un programma o di un sottoprogramma. Di seguito sono elencate alcune delle più importanti:

Nome	Funzione Matematica	Tipo argomenti	Tipo Funzione
SIN(X)	seno	REAL (radianti)	REAL
COS(X)	coseno	" "	"
TAN(X)	tangente	" "	"
SINH(X)	seno iper	REAL	REAL
COSH(X)	cos. iper.	“	“
TANH(X)	tan. iper.	"	"
LOG(X)	log.naturale	"	"
EXP(X)	esponenziale	REAL o INTEGER	REAL
SQRT(X)	radice quad	REAL o INTEGER	REAL
ABS(X)	valore assoluto	REAL o INTEGER	REAL o INTEGER

Per avere un elenco completo delle funzioni predefinite è consigliabile consultare il manuale del particolare compilatore FORTRAN che si intende usare perché molti compilatori hanno un insieme molto più esteso di quello qui riportato.

Tutte le funzioni elencate possono essere utilizzate anche con argomenti complessi o in precisione multipla. Più in generale si può dire che tali funzioni sono "generiche" nel senso che ad ognuna di esse corrispondono più sottoprogrammi che applicano la stessa o la analoga funzione ad argomenti di tipo diverso (ad esempio INTEGER, REAL, DOUBLE PRECISION, COMPLEX); il compilatore sceglie quale sottoprogramma attivare in base al tipo dei parametri.

a) 7.3 Istruzione dichiarativa funzionale

Oltre alle FUNCTION, che sono illustrate nella sez. 5.4, è possibile in FORTRAN definire funzioni, cui si associa un nome e dei parametri formali, per il calcolo del valore di singole espressioni (aritmetiche o logiche). Le funzioni di questo tipo sono locali ai programmi o sottoprogrammi in cui vengono definite e vanno dichiarate prima delle istruzioni eseguibili. A tal fine si fa uso della seguente sintassi:

dichiarazione-funzionale =nome-fun "("lista-argomenti")""="espressione.

Ad esempio:

FUN(X,Y)=SIN(X)**2 - COS(Y)**2

definisce la nuova funzione FUN(X,Y). Questa funzione può essere usata soltanto all'interno del programma in cui è stata dichiarata. Ad esempio se A,B sono due variabili reali dichiarate e inizializzate nel programma, l'istruzione:

Z = FUN (A,B)

assegna alla variabile Z il valore della espressione associata alla funzione FUN applicata a A e B.

Il tipo dei parametri e quello del risultato della funzione vengono definiti secondo le usuali regole di dichiarazione implicita ed esplicita. In caso di dichiarazione esplicita il nome della funzione deve comparire in una precedente dichiarazione di variabile. Nel caso in cui il tipo dell'espressione sia differente da quello della funzione, esso verrà convertito secondo le regole dell'assegnazione già viste precedentemente.

a) 7.4 Sottoprogrammi FUNCTION

La struttura di questi sottoprogrammi è:

[*tipo*] FUNCTION *nome"("lista-argomenti")*"
istruzioni-dichiarative-sottoprogramma
istruzioni-eseguibili-sottoprogramma
RETURN
[istruzioni-eseguibili-sottoprogramma]
END

Le FUNCTION sono sottoprogrammi compilabili separatamente ed in grado di restituire un valore di tipo INTEGER, REAL, DOUBLE PRECISION, COMPLEX, LOGICAL, CHARACTER. Esse vengono attivate dal programma chiamante all'interno di una espressione attraverso il loro nome con la specifica dei parametri attuali. Se il tipo della funzione è dichiarato esplicitamente, nel programma chiamante deve comparire una dichiarazione di variabile che specifichi il nome della funzione e il tipo del risultato.

Ogni FUNCTION, tra le istruzioni eseguibili che descrivono il comportamento del sottoprogramma, deve contenere almeno una istruzione RETURN, il cui compito è di trasferire il controllo al programma chiamante. Analogamente alla STOP del programma principale la RETURN può comparire in punti arbitrari del sottoprogramma.

Illustriamo un semplice esempio di FUNCTION relativo al programma per il calcolo del fattoriale:

```
**************************************************
*           Programma principale
                PROGRAM ESEMPIO
                INTEGER FATT
                READ (*,100) I
                IF (I.LT.0) GOTO 300
                J=FATT(I)
                WRITE (*,200) I,J
        100     FORMAT(I4)
        200     FORMAT (18H,' IL FATTORIALE DI  ',I4,4H,' E' ',I4)
        300     STOP
                END
**************************************************
*           Sottoprogramma
**************************************************
                INTEGER FUNCTION FATT (K)
                FATT=1
                DO 100, I=1,K
                        FATT=FATT*I
        100     CONTINUE
                RETURN
                END
**************************************************
```

5.5 I sottoprogrammi SUBROUTINE

La struttura della SUBROUTINE è:

SUBROUTINE *nome "("lista-argomenti")"*
istruzioni-dichiarative-sottoprogramma
istruzioni-eseguibili-sottoprogramma
RETURN
[istruzioni-eseguibili-sottoprogramma]
END

Le SUBROUTINE sono analoghe alle procedure del PASCAL. Come le FUNCTION possono essere composte da un numero qualsiasi di istruzioni, di cui almeno una deve essere una RETURN, e sono compilabili separatamente. Vengono attivate attraverso una esplicita istruzione di chiamata:

istruzione-chiamata= "CALL" *nome "("lista-parametri-attuali")".*

Vogliamo calcolare il fattoriale mediante un sottoprogramma di tipo SUBROUTINE.
Nel programma principale dell'esempio precedente, cambierà soltanto l'istruzione J=FATT(I), sostituito dalla seguente:

```
        CALL FATT(I,J)
```

Invece il sottoprogramma sarà:

```
        SUBROUTINE FATT (K,N)
        N=l
        DO 100, I=1,K
              N=N*I
100     CONTINUE
        RETURN
        END
```

a) 7.6 L'istruzione dichiarativa COMMON

I sottoprogrammi esaminati finora usano solo variabili locali e possono comunicare con il programma chiamante soltanto attraverso il passaggio dei parametri. Oltre al passaggio dei parametri, il meccanismo che in FORTRAN consente di realizzare una comunicazione tra programma chiamante e programma chiamato è costituito dall'istruzione COMMON. Questa istruzione assegna la stessa area di memoria, detta COMMON-area, a due o più variabili usate in diversi sottoprogrammi. La sua sintassi è la seguente:

istruzione-dichiarativa-common = "COMMON" ["/"*nome-blocco*"/"] *lista-var*"
{"/"*nome-blocco*"/"*lista-var* }.

dove *nomeblocco* è il nome scelto per designare la COMMON-area e *lista-var* contiene liste di variabili condivise. L'istruzione COMMON va posta fisicamente all'inizio del programma, insieme alle altre istruzioni dichiarative. La condivisione delle variabili in una COMMON-area è realizzata ponendo una istruzione COMMON con la specifica dello stesso nome-blocco nel programma chiamante e in uno o più sottoprogrammi chiamati. Nelle istruzioni COMMON che si riferiscono alla stessa COMMON-area, le variabili specificate devono in genere corrispondere in numero e tipo. Tuttavia sono possibili anche usi diversi dell'istruzione COMMON che in questa dispensa non vengono esaminati.

La COMMON-area viene strutturata a blocchi ed a ogni blocco corrisponde una propria lista di variabili. Il nome del primo blocco può non essere presente. In tal caso per la prima lista di variabili viene riservata la porzione iniziale della COMMON-area. Per esempio, se nel programma principale compare la seguente:

```
        COMMON / MIO/ A,B,C
```

in uno o più sottoprogrammi, ad esso correlati, potremmo avere:

```
        COMMON /MIO/ X,Y,Z
```

l'effetto è la condivisione della stessa area di memoria per le variabili A,B,C del programma principale rispettivamente con le variabili X,Y,Z del programma chiamante, considerando tutte le variabili dichiarate implicitamente di tipo reale.

Nel caso in cui si usi un solo blocco è possibile ometterne il nome e quindi nel caso dell'esempio precedente è possibile usare la seguente forma abbreviata:

```
        COMMON A,B,C

        COMMON X,Y,Z
```

Nel caso di dati strutturati, si può usare l'istruzione COMMON sia per dichiarare un vettore che per definire la sua appartenenza alla COMMON-area; per esempio:

```
COMMON A(30,5)
```

è equivalente a

```
REAL  A(30,5)
COMMON A
```

Se una variabile di tipo strutturato compare con il suo dimensionamento in una istruzione COMMON esse non deve essere ridichiarata.

a) 7.7 Le istruzioni dichiarative EXTERNAL e INTRINSIC.

Un sottoprogramma può essere argomento di un altro sottoprogramma. La possibilità di utilizzare delle funzioni come parametri è molto utile quando si vogliano creare dei sottoprogrammi di uso generale. Questa possibilità esiste anche nel PASCAL ed è ampiamente descritta in [4].

Affinchè un sottoprogramma possa essere passato come argomento ad un altro sottoprogramma, deve essere dichiarato mediante una istruzione EXTERNAL o INTRINSIC nel programma chiamante, rispettando la seguente sintassi:

dichiarazione-external = "EXTERNAL" *lista-nomi.*

dichiarazione-intrinsic= "INTRINSIC" *lista-nomi.*

Si usa l'istruzione INTRINSIC se il sottoprogramma passato come argomento è una funzione predefinita del FORTRAN, si usa EXTERNAL se è un sottoprogramma creato dall'utente.

Il programma seguente, costituito da un programma principale, da una subroutine con un parametro funzione e da una funzione, produce il grafico di una funzione su video o su stampante e mostra l’uso dei parametri funzione.

a)

7.8 Esempio di uso di subroutine e di parametri funzione

```
            PROGRAM StampaGraf
* Produce il grafico di una funzione su video o stampante. Non applica fattori di scala
* e va in errore se i valori della funzione sono fuori dell'intervallo 0÷Maxval
            EXTERNAL Quad
* QUAD e'una funzione utente
            INTRINSIC Cos
* COS e' una funzione di libreria
            INTEGER Xmax, Xmin
* Xmin eXmax sono gli estremi dell'intervallo sull'asse delle ascisse
            CALL Grafico(Quad, Xmax, Xmin))
* La SUBROUTINE Grafico ha come parametro la funzione di cui deve produrre il grafico
            CALL Grafico(Cos, Xmax, Xmin))
            STOP
            END

*****************************************************

            SUBROUTINE  Grafico(F, Tmax, Tmin)
            INTEGER Maxval, T, I, Funval, Tmax, Tmin
            PARAMETER (Maxval = 65)
            CHARACTER*1 Plot (0 : Maxval)
            WRITE (*, 5) (I, I = 0, Maxval, 5)
            WRITE (*,15) ('     |', I=0,Maxval,5)
            DO 10 I=0,Maxval
                  Plot=' '
 10         CONTINUE
            DO 20 T= Tmin, Tmax
                  Funval=F(T)
                  Plot (Funval)= '*'
                  WRITE(*,25) 'T'=T, (Plot (I), I=0,Maxval)
                  Plot(Funval)=' '
 20         CONTINUE
  5         FORMAT(1X, 14I5)
 15         FORMAT(1X, 14A)
 25         FORMAT(1X, 2A2, 12, 66A1)
            RETURN
            END
*****************************************************

            INTEGER FUNCTION Quad(X)
            INTEGER X
            Quad= X**2 - 4*X + 5
            RETURN
            END
```

1 . 8. RIFERIMENTI

[1] T.M.R.Ellis - *Programmazione strutturata in FORTRAN77* - Zanichelli,Bologna,1995

[2] D.M.Etter - *Structured FORTRAN77 for Engineers and Scientists* - 3d Ed,Benjamin-Cummings, Redwood City Cal,1990.

[3] E.B.Koffman, F.L.Friedman - *Problem Solving and Structured Programming in FORTRAN77* - 4th Ed, Addison-Wesley, New York, 1990

[4] J.Welsh, J.Elder - *Introduzione al Pascal* - 2d Ed,.Editoriale ESA, Milano 1989

Si raccomanda inoltre di consultare i manuali del compilatore che si intende usare. In questa dispensa ci siamo attenuti alle specifiche del FORTRAN77 come vengono implementate nel Fortran Professional della Microsoft) che contiene anche alcuni costrutti tipici del FORTRAN90 (ad esempio l'istruzione DO WHILE).

Nel caso in cui si scrivano programmi che scrivono e leggono da file è necessario consultare un manuale specifico perché le istruzioni relative sono molto dipendenti dall'ambiente.(sistema operativo) in cui si opera.

Le librerie di sottoprogrammi matematici sono, in genere, molto più ricche di quanto non richiedano gli standard del linguaggio e la documentazione relativa è contenuta nei manuali che accompagnano il compilatore.