Dal C++ al FORTRAN

Versione del 09/01/2002

Università degli Studi della Basilicata
Facoltà di Ingegneria
Corso di Informatica

Nota:

Questo manuale è basato sul contenuto della dispensa "Dal Pascal al FORTRAN" di Mirella Casini Schaerf e Daniele Nardi. Il manuale è finalizzato esclusivamente allo svolgimento delle attività didattiche del corso di Informatica della Facoltà di Ingegneria dell'Università della Basilicata. I contenuti del manuale rispecchiano gli argomenti trattati nel corso e non devono essere intesi come un riferimento esaustivo sulla sintassi e la semantica dei linguaggi trattati.
In questa dispensa ci siamo attenuti alle specifiche del FORTRAN77 con particolare riferimento al compilatore FORTRAN FTN77 Personal Edition della Salford (http://www.salfordsoftware.co.uk) che contiene anche alcuni costrutti tipici del FORTRAN90 (ad esempio l'istruzione DO WHILE). Si raccomanda comunque di consultare il manuale del compilatore utilizzato per verificare la sintassi dei costrutti disponibili.

Sommario

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 C++ 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, 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 C++ 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, assumendo che il lettore abbia una buona familiarità con i linguaggi di programmazione imperativi, ed in particolare con la programmazione procedurale in linguaggio C++. Per una trattazione completa del FORTRAN 77 si veda ad esempio [1,2,3].

2. CONSIDERAZIONI GENERALI SUL FORTRAN

Nella breve descrizione del FORTRAN e nel suo confronto con il C++, 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:

- la possibilità di usare numeri interi e reali con un elevato numero di cifre significative;
- la possibilità di utilizzare i numeri complessi e l'aritmetica complessa;
- l'esistenza di un gran numero di funzioni matematiche di libreria utilizzabili direttamente in qualsiasi punto di un programma o di un sottoprogramma;
- 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.
- 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.

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 e si segue il principio di “dichiarazione prima dell'uso”. 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 C++, 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 C++ ed istruzioni eseguibili che corrispondono alle istruzioni eseguibili di un programma C++. Le istruzioni dichiarative possono essere ordinate a piacere, purché precedano le istruzioni eseguibili. Le principali istruzioni dichiarative riguardano:

- dichiarazione di costanti
- dichiarazione di variabili.

Dopo le istruzioni dichiarative vengono poste le istruzioni eseguibili del programma, tra cui:

- istruzioni di assegnazione
- istruzioni di ingresso e uscita
- strutture di controllo, ovvero istruzioni condizionali e cicli
- istruzioni di chiamata di procedure.

Infine, il FORTRAN utilizza particolari righe di codice per definire le cosiddette “specifiche di formato di stampa”; queste particolari istruzioni 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.

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 nuovi tipi di dato.

4.1 Tipi di dato semplici

Interi (INTEGER)

Il tipo del FORTRAN per i numeri interi è il tipo INTEGER. I numeri interi si scrivono come nel C++. Il campo dei valori ammessi tuttavia può variare ed essere in generale più ristretto di quello offerto dal tipo “int” del C++.

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

Reali (REAL)

Il tipo per i numeri reali in precisione singola è il tipo REAL. 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 esponente
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
```

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. Per farlo, 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. 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 (COMPLEX)

Il FORTRAN, a differenza del C++, fornisce un tipo COMPLEX per la rappresentazione di numeri 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 + 0.000736i viene rappresentato con (3.14,.763E-3)
- il numero complesso 3 + 4i 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.

Come in C++, 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 C++ viene fatta una conversione di tipo nelle espressioni in cui compaiono termini di tipo numerico non omogenei. 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 C++ e sono riportati nella seguente tabella:

```
FORTRAN     C++
----------------
.EQ.        =
.NE.        !=
.GT.        >
.LT.        <
.LE.        <=
.GE.        >=
.EQV.       = tra valori di tipo LOGICAL
.NEQV.      != tra valori di tipo LOGICAL
```

Boolean (LOGICAL)

Il tipo booleano in FORTRAN si chiama LOGICAL. Le costanti hanno la seguente rappresentazione (si noti il punto prima e dopo i valori TRUE e FALSE):

```
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 (CHARACTER)

Il tipo carattere del FORTRAN, CHARACTER, è molto simile al tipo CHAR del C++. 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.

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, REAL, COMPLEX, LOGICAL, CHARACTER, eventualmente in precisione multipla.
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.

Nel più recente FORTRAN 90, per evitare i problemi legati alle dichiarazioni implicite, è stata introdotta l'istruzione

```
IMPLICIT NONE
```

che, utilizzata come prima istruzione dichiarativa del programma inibisce l'utilizzo delle dichiarazioni implicite. Alcuni tra i compilatori più moderni del FORTRAN 77 più moderni consentono di utilizzare questa istruzione.

Dichiarazione di costanti (PARAMETER)

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 definire una costante in termini di altre costanti.

4.3 Il tipo strutturato vettore

Il principale tipo strutturato offerto dal FORTRAN, il tipo vettore, è molto simile al tipo array del C++ 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.

4.4 Il tipo stringa

In FORTRAN è disponibile un tipo di dato stringa di caratteri. 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.

Operazioni sulle stringhe

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 cifre 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 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
```

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 paragrafi successivi. 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 C++ e, utilizzando i compilatori FORTRAN che consentono la programmazione strutturata (evitando l'uso delle istruzioni GOTO), come il compilatore della Salford, le differenze dal C++ 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.

5.1 Istruzioni particolari

L'istruzione `CONTINUE` è un'istruzione senza alcun effetto. 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.

5.2 Istruzioni di ingresso e uscita

Istruzioni READ e WRITE

Nelle istruzioni di ingresso e uscita 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). 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" "(" unità"," etichetta- FORMAT ")" variabili.
Istruzione-uscita = "WRITE" "(" unità "," etichetta-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 C++ le operazioni di ingresso e uscita normalmente leggono dati digitati dalla tastiera e scrivono dati sullo schermo. Per fare riferimento rispettivamente alla tastiera e allo schermo, 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.

5.3 Istruzioni di assegnazione

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

```
istruzione-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. 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, mentre non è possibile assegnare un valore di tipo vettore ad una variabile.

5.4 Istruzioni condizionali

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

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

La differenza principale con il C++ 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.

5.5 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 fa parte dello standard del FORTRAN 77, ma è fornito dai compilatori più moderni, tra cui quello della Salford.

Istruzione-DO

L’istruzione DO serve per eseguire cicli “chiusi”, ovvero cicli in cui il numero di iterazioni è finito e fissato univocamente all’inizio del ciclo. 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. Se L’istruzione DO si comporta come una istruzione FOR del C++ in cui:

- espr1 rappresenta il valore iniziale della variabile di ciclo;
- espr2 rappresenta il valore finale della variabile del ciclo (ovvero la condizione di uscita);

- il passo stabilisce l'incremento della variabile di ciclo; se il passo è 1, la variabile di ciclo viene incrementata di 1 ad ogni iterazione; se il passo è -1 la variabile viene decrementata, ecc.

In sintesi, la variabile del ciclo inizialmente assume il valore di espr1 e viene incrementata di passo ad ogni esecuzione del ciclo. Il ciclo termina quando la variabile di ciclo supera il valore di expr2. 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. Ad esempio consideriamo il seguente programma per il calcolo del fattoriale.

```
*********************************************
       PROGRAM FATTOR
       IMPLICIT NONE
       INTEGER FATT,N, I
       READ(*,*) N
       FATT=1
       DO 100 I=1,N
       FATT=FATT*I
100    CONTINUE
       WRITE(*,*) 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 Fortran della Salford:

```
blocco-WHILE = "DO WHILE" "(" condizione ")"
                  corpo-del-ciclo
               "ENDDO".
```

L'effetto dell'istruzione è del tutto analogo a quello dell'istruzione WHILE del C++. Si noti che, come già visto nel caso dell'istruzione IF-THEN-ELSE e DO, l'istruzione ENDDO serve a specificare dove termina il blocco di istruzioni che compone il corpo del ciclo (in FORTRAN, a differenza del C++, non è possibile specificare istruzioni composte attraverso l'uso delle parentesi graffe). Ad esempio consideriamo di nuovo un programma per il calcolo del fattoriale.

```
*********************************************
       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 FATT=FATT*I
         I=I+1
C        Dobbiamo incrementare la variabile I
       ENDDO
       WRITE(*,*) FATT
       STOP
       END
*********************************************
```

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 ax2+bx+c=0

```
*********************************************
       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
*********************************************
```

Esempio di uso delle istruzioni di ciclo

Un programma in FORTRAN per la ricerca di un elemento (Goal)in un vettore (Elem)

```
*********************************************
       PROGRAM CercaOrd
       PARAMETER (N=10)
       IMPLICIT NONE
       INTEGER Elem(N), Goal, I
       LOGICAL Trovato
C      Lettura degli elementi del vettore
       READ(*,*)(Elem(I) I=1,N)
C      Lettura dell'elemento da cercare
       READ(*,*) Goal
       Trovato=.False.
C      Ciclo di ricerca
       I=1
       DO WHILE ((I.LE.N).AND.(.NOT.Trovato))
         IF(Goal.Eq.Elem(I)) THEN
           Trovato=.True.
         ELSE
           I=i+1
         ENDIF
       ENDDO
C      Stampa del risultato sullo schermo
       IF (Trovato) THEN
          WRITE(*,*) I,Elem(I)
       ELSE
          WRITE(*,*)'elemento non presente'
       STOP
       END
*********************************************
```

6. LETTURA E STAMPA AVANZATE

6.1 L'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:

1. 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 precisione singola o 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 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)
```

6.2 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.

6.3 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; ovvero è' 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 chiudere lo 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.

6.4 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
************************************************
```

7. I SOTTOPROGRAMMI

7.1 Generalità

Nel FORTRAN sono previsti due tipi di sottoprogrammi: le FUNCTION e le SUBROUTINE che corrispondono rispettivamente alle FUNZIONI e alle PROCEDURE del C++. Come in C++, in FORTRAN i sottoprogrammi risultano separati dal testo del programma chiamante e possono quindi essere compilati separatamente. Lo scambio di dati tra programma principale e sottoprogrammi avviene tramite i parametri. Infatti non esistono variabili globali o non locali. Analogamente a quanto avviene per il C++, gli argomenti devono corrispondere in numero, ordine

e tipo ai parametri e normalmente vengono passati con la stessa modalità del passaggio per riferimento del C++. Inoltre in FORTRAN non esistono parametri standard o per valore.
I tipi dei parametri debbono essere dichiarati nei sottoprogrammi. In particolare i parametri di tipo vettore e di tipo stringa debbono essere dimensionati sia nel programma chiamante che nei sottoprogrammi chiamati, ma soltanto alle dichiarazioni del programma chiamante corrisponde una effettiva allocazione di memoria.

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 iperbolico	REAL	REAL
COSH(X)	coseno iperbolico.	"	"
TANH(X)	tangente iperbolica.	"	"
LOG(X)	logaritmo naturale	"	"
EXP(X)	esponenziale	REAL o INTEGER	REAL
SQRT(X)	radice quadrata	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.

7.3 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, I, J
       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)
       INTEGER K
       INTEGER I
       FATT=1
       DO 100, I=1,K
         FATT=FATT*I
100    CONTINUE
       RETURN
       END
************************************************
```

7.4 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 C++. 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)
      INTEGER K, N
      N=l
      DO 100, I=1,K
        N=N*I
100   CONTINUE
      RETURN
      END
```

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