di Bruno Rosati

Speciale linguaggi: Fortran 77

terza parte

«Fortran has long been regarded as the programming language most suited to scientific and numeric applications...», dice l'introduzione al manuale del compilatore Acornsoft. E a noi, fra i tanti linguaggi di moda ed il correre dietro a tutte le versioni turbizzanti di questo o quell'altro high-level, fa davvero piacere, tornare a scrivere di Fortran. Non fosse altro che per rendere l'informazione a tutti gli «scientific» che posseggono Archie o che, pur non possedendolo ancora potrebbero decidersi al gran passo

Dall'ANSI Fortran all'Acorn F77

Se anche in campo informatico (in generale) e in quello della programmazione (in particolare) c'è posto per i sentimenti, la più grande storia d'amore si chiama allora Fortran. Una *neverending story* iniziata nei preistorici anni Cinquanta con l'unione di due parole — FORmula TRANsistion — contrattesi in una magica chiave di volta: FORTRAN.

Che fosse proprio questo il linguaggio giusto per tutti gli *scientific*, fu la stessa ratifica del 1966 a confermarlo, standardizzandone un high-level emblematicamente chiamato Fortran 66 (in realtà il vero nome sarebbe Fortran IV, ma viene usato il 66 per meglio distinguerne la codificazione cronologica). L'ultima ratifica ANSI in ordine di tempo è quella che ci ha infine reso l'attuale Fortran 77 a cui tutte le odierne implementazioni fanno riferimento.

Anche l'Acornsoft F77-compiler è pienamente conformato a tale standardizzazione — definita come Fortran X3.9/1978 — e al solito, fatte salve le particolarità dell'implementazione archimediana, concentrate in 42 pagine di manuale, per la sua completa assimilazione, c'è da prendere in riferimento le varie pubblicazioni che a quella ratifica si rifanno. L'*American National Standard Programming Language FORTRAN X3.9/1978* e, consigliatissimo, il diffuso *A Structured Approach to Fortran 77 Programming* della collana Addison Wesley. Le caratteristiche particolari dell'Acorn-F77 consistono, in linea generale, di una serie di estensioni allo standard come l'insieme dei cicli di WHILE (... ENDWHILE) con statement dedicati quali BLOCK DO, DO WHILE e END DO; l'interessante possibilità dello switching con il mitico Fortran 66; l'inserimento di costanti complesse COMPLEX*16, composte da una coppia di numeri a doppia precisione che rappresentano la parte reale e quella immaginaria di un numero complesso e tutta una serie di caratteristiche per un corretto debug. Cose queste che andremo ad illustrare esaustivamente.

Cominciamo intanto con una rapida passerella sulle caratteristiche principali del compilatore, il quale è strutturato nelle due sezioni principali: quella «Front End», in cui viene effettuato il controllo dei codici-sorgente in conformità con lo standard e quella del «Code Generator» con il quale si arriva alla creazione del programma (equivalente) in codice-macchina.

Un programma così creato si presenterà in versione AOF; ovvero: Acorn Object Format e dovrà essere linkato in una forma eseguibile.

Le due sezioni del compilatore possono essere eseguite automaticamente o separate una dall'altra, a seconda del tipo di comando che verrà impartito. Digitando **F77**, il compilatore provvederà ad eseguire, in sequenza, entrambe le sezioni; con **F77fe** attiveremo invece solo il **f**ront **e**nd e con **F77cg** il **c**ode **g**enerator. A ciò, infine, si aggiunge il comando **link**, con il quale, dopo il risultato ottenuto con le precedenti procedure, si arriva a produrre il linkaggio dell'AOF prodotto in un programma eseguibile.

Ed ora due parole sugli argomenti e le opzioni che ci permettono di modificare il comportamento del comando F77.

Lo facciamo prendendo a riferimento la sintassi della linea:

```
f77 [-from] <name> [-object <name>]
                   [-opt options]
```

nella quale è concentrato l'insieme delle caratteristiche a nostra disposizione per

Fortran 77

Produttore:
Acorn Computers Ltd, Fulbourn Road, Cherry Hinton, Cambridge, CB1 4JN, UK
Distributore:
G. Ricordi & C. S.p.A.
Via Salomone 77 - 20138 Milano
Prezzi *(IVA esclusa):*
Fortran 77 CSKL48 *L. 228.900*
Fortran 77 O.S. & Graphics Library *L. 149.330*

la compilazione, dove l'argomento è rappresentato dal **–from name**; ovvero il file-sorgente che contiene il codice da compilare e dove le **–opt options** (perlappunto... opzionali) rappresentano l'insieme delle forzature al sistema che l'implementazione ci fornisce. Le opzioni si dividono in due categorie: quelle a Codice Generato e quelle «Front End».

Della prima categoria fanno parte la **B**, la quale inserisce un limite al controllo del Codice Generato dal compilatore, costringendo questo a contenere il range di un Array o di eventuali substringhe; l'opzione **H** che inserirà l'uso delle costanti di Hollerith per inizializzare le variabili di tipo INTEGER (tale opzione è implicitamente usata allorché si inserisce il controllo dell'opzione **6**; ovvero: il funzionamento del sistema con tutte le caratteristiche del «vecchio» Fortran 66); e le opzioni **L***n*, e **M***n* che, a loro volta, predispongono rispettivamente all'indicazione del livello della linea numerata inclusa nel codice (nel caso di voler effettuare dei backtrace) e il limite alle unità di programmi (master e subprogram) da utilizzare nella compilazione.

Le operazioni di «Front End», in numero di quattro offrono tutta una serie di controlli per il debug; ad esempio l'opzione **W***n* — che setta i messaggi di livello (da 0 a 4) ed ha una corrispondenza diretta con la **7**, attraverso la quale i messaggi impostati con la **W***n*, possono essere settati o non utilizzati — e la **T***n* (prettamente da debug) che verrà usata dal programmatore per specificare che nel codice sono state inserite delle chiamate a routine particolari.

FUNZIONI DI BIT-MANIPULATION
(Argomenti di tipo INTEGER)

```
IAND  (I,J).............AND logico di I e J

IOR   (I,J).............OR logico di I e J

IEOR  (I,J).............OR esclusivo di I e J

NOT   (I,J).............Complemento logico di I

ISHFT (I,J).............Ritorna il valore dell'INTEGER
                        shiftato a sinistra  o a destra del
                        range (-32; +32) a seconda se J
                        risulta positivo o negativo.

IBSET (I,J).............Ritorna I il bit di J settato a 1
                        (il risultato è indefinito se J non
                        è compreso nel range 0-31)

IBCLR (I,J).............Ritorna I con il bit di J settato
                        a zero.

BTEST (I,J).............Testa il bit J di I e ritorna un
                        risultato logico.
```

**BTEST (I,J) è l'unica funzione a fornire un risultato logico; tutte le altre ritornano un INTEGER.*

Figura 1

Le estensioni dell'Acorn F77

Al solito, quello che è il Fortran, chi lo usa... lo sa. Quelle che sono invece le particolarità dell'implementazione Acorn andiamo ad elencarle ora cominciando col precisare subito che l'AF77 dispone di costanti esadecimali che possono essere utilizzate ogni volta che è permesso l'uso di una costante di tipo ordinario.

La forma delle costanti esadecimali è la seguente:

? <type><digits>

dove <type> è una lettera che specifica il tipo di costante assegnando alle varie INTEGER, REAL, DOUBLE PRECISION, COMPLEX, LOGICAL e CHARACTER, le rispettive iniziali I, R, D, C, L con H ad identificarsi per la costante Character.

<digits> infine è la cifra esadecimale (da 0 a 9; da A ad F).

Come già accennato nella parte introduttiva delle caratteristiche generali dell'AF77, con esso disporremo anche di una serie di costrutti per i cicli di loop. Al riguardo, (escluso il classico WHILE... ENDWHILE, compatibile con il Salford FTN77 ed il WATFIV) i costrutti DO WHILE e BLOCK DO (END DO) risulta-

Generiche	Specifiche
ABS	CDABS
CONJG	CDCONJG
SQRT	CDSQRT
EXP	CDEXP
LOG	CDLOG
SIN	CDSIN
COS	CDCOS

*Figura 2 - Questa è la lista dei nomi specifici dedicati alle funzioni intrinseche con argomenti del tipo COMPLEX*16 (che, ripetiamo, è una costante complessa a doppia precisione).*

```
        LIMITI DEL GENERATORE


GRANDEZZA CODICE..... 128 Kbytes
NUMERO ETICHETTE.....4096
VARIABILI LOCALI.....8192
COSTANTI.............8192
BLOCCHI COMUNI.......2048
SIMBOLI ESTERNI......2048
```

Figura 3

no essere delle forme pienamente compatibili agli equivalenti costrutti delle implementazioni VAX/VMS del compilatore.

Oltre ai loop, nell'AF77 potremo contare anche su due routine-generator di numeri random.

La REAL FUNCTION RNDO (1) che ci fornisce un numero «pseudo-casuale compreso fra 0.0 e 1.0 e la SUBROUTINE SETRND (I), la quale seleziona una nuova sequenza random condizionata al valore di I; se questo è pari a zero la sequenza non potrà più essere ripetuta.

Altra caratteristica estensiva dell'AF77 è la disponibilità del comando di tipo INCLUDE, nel quale sarà possibile tenere file di codice-sorgente che potranno esser letti dal compilatore al momento della richiesta del «filename». La sintassi: INCLUDE filename' è semplicissima.

Sia per le COMPLEX*16 — già citate nell'introduzione che per le funzioni per la bit — manipulation di argomenti di tipo INTEGER, rimandiamo infine alle figure 1 e 2, con le quali si completa questa prima escursione sulle estensioni «acorniane» all'F77.

Input/Output

Fra le varie precisazioni su che cosa s'intende, archimedianamente parlando, per unità numeriche, file sequenziali, letture e scritture di tipo formattato — possibili su ogni tipo di file — e quelle di tipo informattato — solo su file da disco — che vi lascio sorbire per vostro conto, sempre in riferimento all'importante capitolo dell'Input-Output, preferisco concentrare l'attenzione sull'argomento delle decodificazioni di formato che, l'implementazione della Acorn opera con minori limitazioni rispetto a quelle imposte dallo standard.

La cosa è molto interessante, giacché ci consente di verificare una ulteriore serie di caratteristiche dell'AF77 che, insieme alla lunga serie di estensioni già elencate, potranno renderci meglio l'effettivo valore dell'implementazione. Tanto per cominciare (o per continuare...) buona regola dell'AF77 è quella di ignorare qualsiasi tipo di conteggio ripetitivo (non desiderato) che può verificarsi prima di determinati descrittori (quali: **' T / : S B** e, nel caso particolare, il segno che precede il descrittore **P**) quindi di una virgola e di una parentesi chiusa.

Per quanto poi riguarda proprio il «comma» cioè la virgola, guarda proprio il «comma» cioè la virgola, l'AF77 ne omette l'uso, eccezion fatta nei casi in cui, tale omissione, possa causare ambiguità di lettura.

Le caratteristiche estensive legate al processo di decodifica non si fermano qui. È possibile difatti, disporre di un'estensione anche del gruppo degli edit-descriptor di tipo numerico. A quelli standard (**Iw**, **Iw.m.**, **Fw.d**, **Ew.d**, **Ew .dEe**, **Dw.d**, **Gw.d**, e **Gw.dEe**) vengono aggiunti: **Fw**, **Dw.dDe**, **Gw.dDe**, **Dw.dEe**, **Ew.dDe**, **Z** e **Zw**; con il risultato che il nostro lavoro di edit-aggio fortraniano acquisterà maggiori possibilità.

Il manuale dell'AF77, a riguardo del Format Decoding, parla di «... *more liberal manner than implied by the Fortran standard*» e alle «libertà» appena descritte, per confermare il concetto della massima manovrabilità, non rimane che aggiungere le ultime specifiche. Cominciando dall'e-descriptor **A** predisposto a maneggiare liste numeriche e da quello **I**, con il quale è possibile il trasferimento di valori reali e in doppia precisione. Chiudono la lista i descrittori **F**, **E**, **D** e **G**, abili a produrre un valore di tipo INTEGER.

Gestione degli errori

Nella gestione degli errori, il compilatore emette i relativi messaggi a secondo che, la trap, scatti a livello di Front-End o di Generatore. Nel primo caso, tali messaggi saranno molto concisi e facili da interpretare (in pratica si tratta di eventuali errori di sintassi). Nell'altro invece, gli errori rilevabili dal Generatore (costanti troppo larghe, argomenti fuori dal range della funzione in questione, il disequilibrio fra costanti e variabili, etc.) il discorso si fa meno immediato ed implica, oltre che la ricerca e l'individuazione dell'errore, anche la conoscenza dei limiti del generatore stesso (per una, anche sommaria informazione, occhio alla figura 3).

Ma c'è comunque una terza categoria di errori (e conseguente tipo di messaggio) da tenere in considerazione. Si tratta di errori che possono avvenire a livello della librerira di Run-Time. La sua forma è chiarissima:

++++ERROR N: text

dove **N** è il numero di un errore e **text** il messaggio relativo che descrive il tipo di errore commesso. Gli errori rilevabili in tale area di lavoro sono più che mai di tipo format ed interessano, in prevalenza, l'uso di tutti gli edit-descriptor.

Note conclusive

Un manuale stringatissimo, il rimando ai testi ufficiali e la pura trattazione delle particolarità relative all'implementazione, sono una costante delle presentazioni fatte dalla Acornsoft a cui neanche l'A-Fortran 77 fa eccezione. Chiudendo questa terza puntata del nostro speciale, ad un passo dall'ultimo articolo che ne completerà la panoramica, è solo questo l'appunto che mi sento il dovere di fare alla pur rimarchevole opera svolta dalla Acorn per garantire l'utilizzo più completo del nostro Archie. La golosità che tale macchina può indurre nei palati di tutti i programmatori esperti, è la chiave di volta in campo commerciale. Fare una semplice, a volte pedante elencazione, può sortire solo confusione ed alla lunga procurare degli effetti deleteri. Uno potrebbe anche dire: ma la Acorn sta facendo tutto da sola! Questo è vero e gliene diamo atto. Archie è grandioso qualsiasi «lingua» parli o gli si faccia parlare; la mia critica, velatissima e a lunga gittata vuol solo portare il valido contributo per vedere il nostro e chi lo vuol programmare, serviti nel miglior modo possibile. Gli *scientific* hanno un grande appetito di Fortran e questa di Archie, a sentirli parlare, è decisamente la più bella implementazione che abbiano mai visto e provato, ma guarda un po': trovano il manuale troppo stringato. Perché non servirli al meglio? MC