Laboratori Nazionali di Frascati

LNF-66/71

A. Turrin : I CALCOLATORI ELETTRONICI NUMERICI AL SERVIZIO DEL CALCOLO SCIENTIFICO - II.

Estratto da: Giornal e di Fisica 7, n. 4 (1966)

Estratto dal *Giornale di Fisica* - Vol. VII - N. 4 - Ottobre-Dicembre 1966

I calcolatori elettronici numerici al servizio del calcolo scientifico - II.

A. TURRIN
Laboratori Nazionali del CNEN - Frascati

1. - Possibilità di commissionare al calcolatore programmi di calcolo espressi in linguaggio umano.

1·1. - *I linguaggi simbolici.* - La tecnica elettronica ci fornisce calcolatori potenti sotto ogni punto di vista (veloci, capaci di decisioni logiche, di vasta memoria) che comprendono, calcolano e si esprimono in linguaggio binario (linguaggio base, o assoluto). La tecnica di programmazione li rende capaci di tradurre istruzioni formulate in linguaggio umano in istruzioni formulate nel loro linguaggio base onde poi eseguire le elaborazioni.

In altre parole esistono programmi scritti in linguaggio base che hanno il compito di informare il calcolatore su come codificare nel proprio linguaggio assoluto un programma scritto in linguaggio umano. Questi programmi si chiamano traduttori. La procedura di traduzione è la seguente:

a) il traduttore entra (per esempio, mediante lettura da schede) nella memoria veloce;

b) la macchina legge la prima scheda del pacco di schede su cui è perforato il programma scritto in linguaggio umano (programma sorgente), codifica le informazioni contenute su questa scheda in una (o piú) istruzioni base, e perfora queste informazioni su schede;

c) legge la scheda seguente del programma sorgente e ripete il procedimento descritto in *b*), fino ad esaurimento delle schede del programma sorgente.

Il pacco di schede uscito dalla macchina si chiama programma oggetto, e su di esso è memorizzata la traduzione (in linguaggio assoluto) del programma sorgente.

L'esecuzione del calcolo vero e proprio e la produzione dei risultati si ottengono facendo leggere al calcolatore il programma oggetto e le schede contenenti i dati su cui operare. In altri termini la traduzione di un programma sorgente e l'esecuzione di un programma oggetto sono due elaborazioni del tutto distinte e separate l'una dall'altra.

Abbiamo parlato di « linguaggio umano », ma è implicito che questo linguaggio è soggetto a facili ma severe regole di grammatica, di punteggiatura, di modi di esprimersi, perché il traduttore deve essere sempre in condizioni di « comprendere » quello che vogliamo e non vogliamo. Questo linguaggio è piuttosto un compromesso tra le nostre esi-

genze di risolvere nel modo piú agevole possibile il problema e la struttura dell'insieme enorme di « si » e « no » mediante la quale dobbiamo operare. Diciamo allora meglio che scriviamo i programmi in *linguaggio simbolico*.

Comunque, esistono linguaggi simbolici grazie ai quali si può ignorare del tutto ogni nozione sul linguaggio base dei calcolatori (e lo sforzo di codificazione dei programmi viene sostenuto interamente da un traduttore molto elaborato e raffinato). Uno di essi è il FORTRAN.

Esistono anche linguaggi simbolici soggetti a regole derivate direttamente dalla natura delle istruzioni base di ogni particolare calcolatore (e in questo caso il traduttore è un programma relativamente piú semplice). Diciamo qualcosa di piú a proposito di questi ultimi.

Abbiamo a disposizione un calcolatore le cui istruzioni sono costituite da un codice operativo e da un indirizzo (vedi p. 180 della Parte prima[1]). Un programma scritto in linguaggio assoluto ha questo aspetto:

indirizzi	istruzioni:	
	cod. oper.	indirizzo
.....	.....	.....
001001	10101	010110
001010	00101	101001
001011	01010	101010
001100	01000	110100
.....	.....	.....

Nella colonna « indirizzi » a parte a sinistra sono rappresentati in ordine crescente gli indirizzi di memoria veloce nei quali sono contenute le istruzioni (colonna « istruzioni ») e i dati. Ogni codice operativo ha il suo significato (somma, sottrazione..., salto a, ...).

Un modo conveniente per redigere un programma in linguaggio base è il seguente. Si scrive in un primo tempo per propria comodità il programma, usando dei *simboli* al posto dei numeri 0 ed 1, ad esempio:

indirizzi	istruzioni	
	cod. oper.	indirizzo
.....	.....	.....
.....	.....	.....
MASS	SOMMA	DATO
.....	.....	.....
.....	VAI A	MASS
.....	.....	.....
.....	.....	.....

Cosí, in un certo indirizzo che per ora chiamiamo MASS mettremo una istruzione di somma; l'operazione andrà eseguita su un certo dato che vorremo contenere in un indirizzo chiamato per ora DATO. Ad un certo punto del programma ci serve ritornare (vedi per esempio istruzione *D* a p. 178 della Parte prima) ad eseguire il lavoro riprendendolo dalla istruzione di somma dello stesso dato e per far ciò scriveremo per ora VAI A MASS.

Una volta scritto il programma in *linguaggio simbolico* è relativamente facile codificarlo in linguaggio assoluto scrivendo in linguaggio binario i codici operativi e computandovi via via gli indirizzi partendo da un certo indirizzo iniziale.

Ma questo lavoro di codifica in linguaggio binario è del tutto banale e piatto: non lo dobbiamo fare noi, ma lo deve fare il calcolatore. Il traduttore, appunto, è un programma che codifica

---

(1) A. TURRIN: *Giornale di Fisica*, **6**, 171 (1965).

in linguaggio binario un programma scritto in linguaggio simbolico: assegna ad ogni codice simbolico del tipo SOMMA, VAI A, ecc. il codice in linguaggio binario corrispondente; assegna ad ogni indirizzo simbolico un indirizzo in linguaggio binario e computa via via tutti gli indirizzi in modo che il programma sia contenuto correttamente in memoria.

1·2. - *Macroistruzioni.* - Disponendo di un buon traduttore si ha la possibilità di scrivere in linguaggio simbolico i programmi adoperando codici operativi simbolici indicanti operazioni *non* elementari di macchina.

Ogni codice simbolico di questo tipo non ha ovviamente un codice corrispondente in linguaggio binario.

Una istruzione avente un simile codice operativo in linguaggio simbolico si chiama macroistruzione perché il traduttore la codifica in una serie di istruzioni base che nel loro insieme formano l'operazione *non* elementare individuata dalla macroistruzione stessa.

Per esempio la prescrizione *C* a pag. 178 della Parte prima non può essere espressa da una singola istruzione di macchina, ma costituisce di per sé un piccolo programma.

Supponiamo di disporre di un traduttore che sia in grado di codificare in una serie di istruzioni la prescrizione *C* a pag. 178 della Parte prima. La macroistruzione che darà luogo a questa codifica sarà del tipo

cod. operativo	indirizzo
BIVIO	SINISTRA, DESTRA, B,D

BIVIO è per il programmatore uno dei codici operativi di cui può far uso corrente. Nella parte indirizzo della macroistruzione *devono* comparire quattro indirizzi simbolici scritti in un ordine convenuto: il primo è l'indirizzo dove deve essere trasferito il dato maggiore tra i due dati che inizialmente sono contenuti negli indirizzi SINISTRA e DESTRA; il terzo è l'indirizzo dell'istruzione da cui il lavoro deve continuare nel caso che il dato in DESTRA sia maggiore del dato in SINISTRA, ed il quarto è l'indirizzo della istruzione da cui il lavoro deve continuare se il dato in SINISTRA è maggiore del dato in DESTRA.

Non rispettare una simile ben precisa convenzione significa compiere un errore nella stesura del programma.

I vantaggi offerti dal poter disporre di macroistruzioni sono evidenti.

È da richiamare inoltre l'attenzione sul fatto che perfezionando via via la tecnica delle macroistruzioni si possono creare linguaggi simbolici di natura indipendente da quella dei diversi linguaggi base dei calcolatori.

1·3 - *Il generatore dei programmi di organizzazione delle operazioni d'entrata ed uscita.* - Come detto nella Sez. 2·4 della prima Parte un calcolatore è usato in modo efficiente solo quando l'unità centrale è continuamente impegnata a compiere l'elaborazione, e non esistono intervalli di tempo morto durante i quali l'unità centrale deve aspettare che i dati entrino o escano.

Riprendiamo l'esempio di cui al quadro dei tempi di fig. 11 della prima Parte, supponendo che l'entrata e uscita vengano realizzate mediante due distinte unità a nastro magnetico. Supponiamo inoltre che o le freccie « legge » o le freccie « emette » (o anche sia le une che

le altre) abbiano lunghezza maggiore delle freccie « elabora ». In questo caso ci troviamo in una condizione di inefficiente uso del calcolatore.

Questa condizione è determinata dal fatto che per noi i blocchi di dati A, B, C, ... sono blocchi « logicamente uguali » tra loro, cosí come lo sono i corrispondenti blocchi A', B', C',.... Il programma di elaborazione è concepito in funzione di questi raggruppamenti, perché su ognuno di essi viene eseguita una elaborazione di tipo iterativo. Cosí, disponiamo di un nastro magnetico sul quale sono memorizzati i blocchi di dati A, B, C,.... .

Ognuno di questi blocchi di dati è separato fisicamente dal successivo da una porzione di nastro neutro, di lunghezza ~ 2 cm entro la quale riusciamo a frenare e riaccelerare il nastro nelle successive operazioni di lettura. Accade che il tempo richiesto per accelerare il nastro, leggere un blocco di dati, frenarlo, è maggiore del tempo impiegato dall'unità centrale ad eseguire l'elaborazione.

In piú, se ogni blocco è costituito da pochi dati, la maggior parte della lunghezza totale del nastro è neutra (per esempio, se ogni blocco contiene gli 80 caratteri di una scheda, circa tre quarti del nastro sono inutilizzati).

L'unico modo per evitare quest'uso dissipativo del calcolatore è quello di:

a) disporre di un nastro sul quale siano memorizzati i blocchi di dati « logicamente uguali » tra loro in gruppi di *piú* blocchi registrati ininterrottamente, la separazione fisica (realizzata dalla porzione di nastro neutro) esistendo solo tra un gruppo e il successivo, come in fig. 1. Cosí, leggendo i dati per gruppi di blocchi si ottiene un tempo medio di lettura di ogni singolo blocco logico minore del tempo impiegato per elaborarlo e si aumenta la densità media di registrazione utile.

ABCDEF GHIJKL MNOPQR .....

Fig. 1.

b) Caricare in memoria veloce oltre al programma di elaborazione un programma del tutto indipendente che cura l'entrata dei dati, che scinde ogni gruppo di blocchi entrato nei suoi costituenti e fornisce al programma di elaborazione uno alla volta successivamente i singoli « blocchi logici », che raggruppa i blocchi logici da emettere in gruppi di piú costituenti e che ne cura l'uscita.

Come il traduttore, cosí anche il generatore di programmi di organizzazione delle operazioni di entrata e uscita fa parte del normale corredo fornito al calcolatore dalla tecnica di programmazione.

Se durante una traduzione è presente in memoria veloce anche il generatore di programmi per l'entrata e uscita, poche macroistruzioni bastano ad informarlo onde rediga automaticamente il particolare programma di entrata/uscita che occorre per rendere ottime le entrate/uscite relative al lavoro che si dovrà successivamente compiere.

In questo modo il programmatore rimane svincolato dalla logica relativa ai « blocchi fisici » di dati memorizzati sui nastri magnetici e può dedicarsi soltanto alla tesura del programma di elaborazione dei « blocchi logici ».

Questo generatore (che è capace di redigere programmi di sovrapposizione nel tempo delle entrate/uscite e della elaborazione) serve anche alla stesura di

programmi di manutenzione e aggiornamento della biblioteca di nastri magnetici.

Ogni bobina di nastro magnetico di una biblioteca deve essere contraddistinta da una serie di informazioni registrate su di essa, che precedono i dati contenutivi. Queste informazioni sono: un nome assegnato a tutti i dati memorizzati su di essa, la data di registrazione, il periodo di tempo in cui è necessario mantenere inalterati i dati, la densità di registrazione, il numero dei blocchi fisici contenuti nella bobina, ecc.

Occorre sempre far compilare al generatore di programmi di organizzazione di entrata e uscita programmi che controllino durante le esecuzioni dei lavori che le bobine in lettura/scrittura siano inequivocabilmente quelle richieste, che siano montate sulle unità di lettura/scrittura scelte, che sia trascorso il periodo di tempo in cui è proibita la modifica dei dati contenuti nelle bobine destinate a ricevere nuovi dati dal calcolatore, ecc.

1·4. - *L'operatore automatico.* - La esecuzione dei lavori (traduzioni, prove di programmi, esecuzione di calcoli) compiuta mediante un calcolatore di modeste prestazioni comporta interventi manuali sulla macchina sia nei periodi di prova e messa a punto dei programmi, sia quando la macchina ha terminato un lavoro e dobbiamo farle iniziare l'esecuzione del successivo. Queste manualità determinano tempi morti che incidono sull'efficienza della macchina tanto meno quanto piú lento e piccolo è il calcolatore, ma che non sono permessi assolutamente quando si disponga di una macchina di elevate prestazioni.

Un calcolatore veloce e di vasta memoria deve poter avere l'unità centrale *sempre* impegnata.

L'operatore automatico è un programma che, una volta caricato (da nastro magnetico) in memoria, automatizza al massimo quelle operazioni che andrebbero eseguite manualmente su calcolatori piú modesti e mette la macchina in condizioni di formulare giudizi sui lavori presentati e di decidere quali provvedimenti adottare per ogni lavoro commissionato.

Quando la macchina funziona sotto il controllo dell'operatore automatico i vari lavori che via via assolve vengono prescritti ad essa mediante opportune schede controllo preparate a cura del personale addetto alla macchina.

Supponiamo di voler compiere successivamente i seguenti lavori distinti:

a) Traduzione di un programma sorgente scritto in FORTRAN.

a_1) Esecuzione di un calcolo mediante il programma oggetto ottenuto in *a*).

b) Esecuzione di un calcolo mediante un programma memorizzato nella biblioteca di nastri magnetici.

Essendo il lavoro indicato in *a*) il primo della serie, si fa precedere al pacco di schede del relativo programma sorgente le seguenti schede controllo:

una scheda che informa la macchina della data e dell'ora di inizio dei lavori;

alcune schede che la informano che il lavoro che segue è la traduzione di un programma scritto in FORTRAN.

Si fa seguire quindi ad esse il pacco di schede del programma sorgente *a*).

L'elaborazione a_1) verrà eseguita ovviamente su un insieme di dati numerici

memorizzati su un pacco di schede. Per far eseguire questa elaborazione si inserisce tra il pacco *a*) (sorgente) e a_1) (dati) una scheda che informa la macchina che si vuole compiere una elaborazione col programma testé tradotto.

Anche il lavoro *b*) andrà eseguito su certi dati memorizzati su schede, e pertanto si inseriscono tra il pacco a_1) e il pacco *b*) alcune schede che informano che il lavoro seguente è l'esecuzione di quel certo programma da far leggere da quel certo nastro della biblioteca dei programmi.

Se il lavoro *b*) è l'ultimo della serie si pospone al pacco di schede *b*) una scheda che informa la macchina che la serie di lavori è finita.

Cosí, non appena l'operatore automatico è entrato nella memoria veloce, vien letta la scheda della data e dell'ora d'inizio dei lavori. Preso atto che il prossimo lavoro è una traduzione di un programma sorgente scritto in FORTRAN l'operatore automatico fa entrare in memoria (da nastro magnetico) il traduttore FORTRAN. Il traduttore esegue il suo lavoro memorizzando via via su nastro magnetico il programma oggetto, finché il lavoro di traduzione è terminato, e il nastro riavvolto.

L'operatore automatico legge ora la scheda inserita tra *a*) e a_1), e fa entrare in memoria il programma oggetto testé ricavato (il traduttore non ha ora piú ragione di esistere in memoria). Il programma oggetto viene eseguito (e ciò comporta la lettura di tutte le schede dati). Alla fine di questa elaborazione l'operatore automatico legge le schede di controllo che precedono il lavoro *b*), chiama in memoria il programma voluto che legge le schede dati *b*). Alla fine di questa elaborazione l'operatore automatico è informato che la serie di lavori è finita, e la macchina si arresta.

Come si vede, l'operatore automatico ha il controllo totale della macchina tra un lavoro e l'altro. In questi periodi di tempo, dopo aver ricevuto le informazioni su ciò che sarà il prossimo lavoro, carica in memoria dalle unità a nastro magnetico i programmi necessari a compierlo, indi « sorveglia » il lavoro in corso. Poiché l'operatore automatico deve essere costantemente presente in memoria, questa parte della memoria della macchina è inviolabile. Se un errore di programmazione (contenuto nel lavoro da eseguire) porta a modificare il contenuto della memoria in cui è caricato l'operatore automatico, l'elaborazione in corso si arresta e viene comunicato un messaggio di errore sulla stampatrice veloce, e tutte le informazioni necessarie al programmatore per individuare l'errore.

Questo lavoro contenente l'errore viene quindi « levato »: la macchina « legge a vuoto » (cioè non le considera) le eventuali rimanenti schede relative al lavoro contenente l'errore, riavvolge i nastri magnetici, e passa a considerare le schede controllo relative al lavoro successivo.

La macchina possiede un orologio elettronico. Se noi informiamo mediante le schede controllo l'operatore automatico che una certa elaborazione non può durare piú di un certo intervallo di tempo, il lavoro viene « levato » dall'operatore automatico se la durata della elaborazione lo supera, ed il programmatore viene avvertito di ciò tramite la stampatrice veloce.

In ogni caso tra un lavoro e l'altro

viene stampata la data, e, volendo, la durata del lavoro.

Durante l'elaborazione esiste la possibilità di informare (manualmente) l'operatore automatico di abbandonare l'esecuzione del lavoro in corso (e di passare pertanto al successivo).

2. - Il linguaggio FORTRAN.

2·1. - *Descrizione informativa del linguaggio FORTRAN sulla base di qualche esempio. La scheda FORTRAN.* - Il nome FORTRAN deriva dall'abbreviazione di « FORmula TRANslation » (traduzione di formula), che è appunto il lavoro che compie il traduttore sul programma sorgente.

Un programma sorgente FORTRAN consiste in una serie di schede su ognuna delle quali è contenuta l'espressione di una *richiesta* d'esecuzione di una operazione aritmetica e/o logica, o di salto o di entrata/uscita. Esistono anche richieste che informano soltanto il traduttore su come preparare il programma oggetto (per esempio, lo informano di riservare un certo spazio in memoria dove si vorrà contenere durante la fase di elaborazione un vettore A_i ($i=1, \ldots, n$) o una matrice). La lingua usata è l'inglese.

In figura 2 è riportata una scheda contenente una richiesta FORTRAN. È facile descrivere il significato di questa richiesta: ricordando che in inglese « if » significa « se » e che « greater than » significa « maggiore di » (l'operatore di confronto .GT. definisce appunto questa situazione) la richiesta significa:

« Se il dato in DESTra è maggiore del dato in SINISTra *porta* in SINISTra il dato che c'è in DESTra », indi rivolgiti alla richiesta che segue; se ciò non è rivolgiti alla richiesta che segue (e ometti di eseguire il trasferimento).

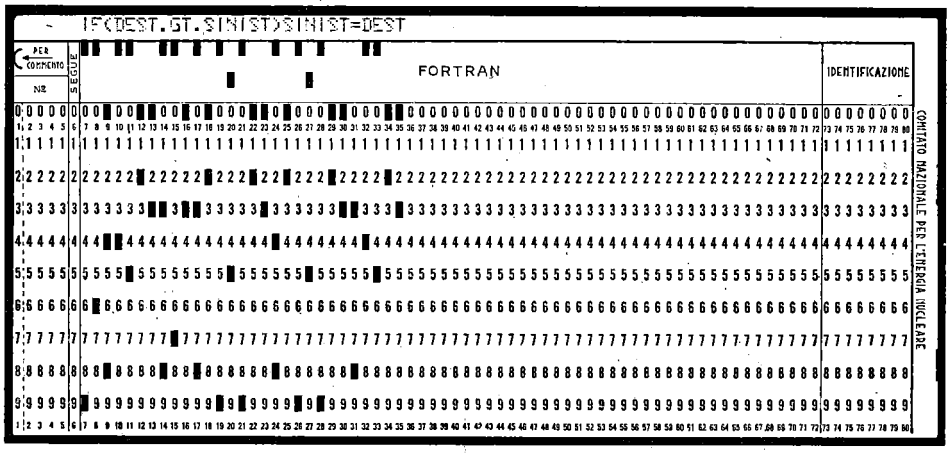

Fig. 2.

Questa richiesta è leggermente diversa dalla *C* di pag. 178 della Parte prima.

Come si vede dalla fig. 2 la richiesta va « scritta » a partire dalla colonna 7 della scheda; il traduttore ignora tutto ciò che è perforato nelle ultime 8 colonne. Una richiesta esprimibile con un nu-

mero di caratteri maggiore di 66 può prendere posto su due o piú schede successive a patto che tutte le schede occorrenti a partire dalla seconda portino in colonna 6 un carattere qualunque che non sia lo zero.

Una richiesta può essere contraddistinta da un numero (e costituisce una richiesta numerata). Questo numero può essere scritto nelle prime 5 colonne, con incolonnamento sulla quinta.

derazione l'intera scheda. Cosí si possono scrivere parole di commento nel corso del programma. In ogni caso come detto si può perforare ciò che si vuole nelle ultime 8 colonne di qualunque scheda.

Si ottiene il pacco di schede sorgente presentando al personale addetto il programma scritto su carta quadrettata. Ogni riga del foglio contiene una richiesta. (Vedi il frontespizio della scheda in fig. 2.) Per esempio

```
colonne   12345 6 7 8 9
          .....|.|.....................................
             21|A=A1+A2
          .....|.|.....................................
               |IF(DEST.GT.SINIST)SINIST = DEST
          .....|.|.....................................
               |GOTO 21
          .....|.|.....................................
```

Per esempio,

14
oppure
2
oppure
231
oppure
12377

Per « saltare » nel corso del programma ad una certa richiesta è necessario che questa sia numerata. Il salto si ottiene con la richiesta GO TO (vai alla).

Cosi la richiesta GO TO 21 fa continuare i calcoli a partire dalla richiesta numerata col numero 21. Non è necessario numerare ordinatamente le richieste: questi numeri di richiesta sono « nomi » piuttosto che numeri.

Se nella prima colonna è perforata una C il traduttore non prende in consi-

La richiesta 21 significa: metti in A la somma dei dati numerici contenuti in A1 e A2. La chiarezza e l'ordine del manoscritto tendono ad evitare la prima serie di errori che ci stanno attendendo.

Si può ottenere la lista stampata del programma facendo leggere (al di fuori del calcolatore) il pacco sorgente ad una tabulatrice di schede.

Si può ordinare a calcolatore di stampare la lista del programma durante la fase di traduzione.

2·2. - *Calcolo numerico di integrali.* - Come esempio che possieda un certo carattere di completezza scriveremo un programma per il calcolo di un integrale definito

$$\text{area} = \int_a^b f(x)\,dx$$

mediante il metodo numerico di Simpson.

Quando è impossibile o malagevole calcolare analiticamente un integrale è necessario ricorrere a uno (dei tanti) metodi di integrazione numerica, tra i quali la formula di Simpson costituisce il metodo piú diffuso. Il calcolo numerico di un integrale dà sempre un risultato approssimato, ma l'approssimazione può essere spinta fino al grado voluto a prezzo di calcoli numerici adeguatamente piú lunghi.

Il metodo di Simpson consiste nel suddividere l'intervallo (a, b) in un numero *pari* $(n-1)$ di intervalli uguali h; nel calcolare la funzione $f(x)$ negli n punti $a, a+h, a+2h, \ldots, a+(n-1)h(\equiv b)$; nell'approssimare la funzione $f(x)$ in tutto l'intervallo (a, b) mediante una spezzata costituita da $\frac{n-1}{2}$ tratti di parabola ad asse verticale dei quali:

il primo passa per i tre punti $f(a)$, $f(a+h)$, $f(a+2h)$;

il secondo passa per i tre punti $f(a+2h)$, $f(a+3h)$, $f(a+4h)$;

il terzo passa per i tre punti $f(a+4h)$, $f(a+5h)$, $f(a+6h)$;

. . . . . .

l'$\left(\frac{n-1}{2}\right)$-esimo passa per i tre punti $f[a+(n-3)h]$, $f[a+(n-2)h]$, $f(b)$;

nell'integrare sull'intervallo (a, b) la curva costituita da questa spezzata di tratti di parabola.

Si ottiene cosí la formula:

$$(6)\quad \int_a^b f(x)\,dx \cong \frac{h}{3}\{f(a)+4f(a+h)+2f(a+2h)+4f(a+3h)+\ldots+2f[a+(n-3)h]+4f[a+(n-2)h]+f(b)\},$$

che è appunto la formula di Simpson.

È evidente che quanto piú grande sarà il numero n di punti in cui è calcolata la funzione, tanto meglio i tratti di parabola approssimeranno la funzione $f(x)$ nell'intervallo (a, b).

La (6) può essere scritta nel'a seguente forma:

$$(7)\quad \int_a^b f(x)\,dx \cong \frac{h}{3}\sum_1^n c_i f_i\,,$$

dove le c_i sono, nell'ordine,

$$1, 4, 2, 4, 2, 4, 2, \ldots, 4, 2, 4, 2, 4, 1\,,$$

e le f_i sono, nell'ordine,

$$f_1 \equiv f(a),\ f_2 \equiv f(a+h), \ldots, f_n \equiv f(b)$$

(n *dispari*).

La formula ricorrente per i coefficienti c_i è la seguente:

per $x_i = a$ ed $x_i = b$, $c_i = 1$;

per $x_i \neq a$ ed $x_i \neq b$, le c_i sono date dalla successione *alternata*

$$4, 2, 4, 2, \ldots, 4, 2, 4, 2, 4,$$

che si può generare usando un deviatore (DEV), cioè una variabile logica che può assumere solo due valori possbili, per esempio:

$$\mathrm{DEV} = \uparrow \quad \text{oppure DEV} = \downarrow\,.$$

Costruiamoci (fig. 3) un diagramma a blocchi per il calcolo della (7), e seguiamo su di esso l'evoluzione del calcolo fin dall'inizio (nella locazione « area » verrà memorizzato il valore finale dell'integrale, ma questa locazione può servire come locazione di lavoro per memorizzarvi durante il calcolo la somma a secondo membro della formula (7)):

Nel blocco 1 viene (tra l'altro) azzerata la locazione (« area ») in cui si intende memorizzarvi la somma a secondo membro della formula (7); viene inizializzata x dandole il valore $a-h$, e il de-

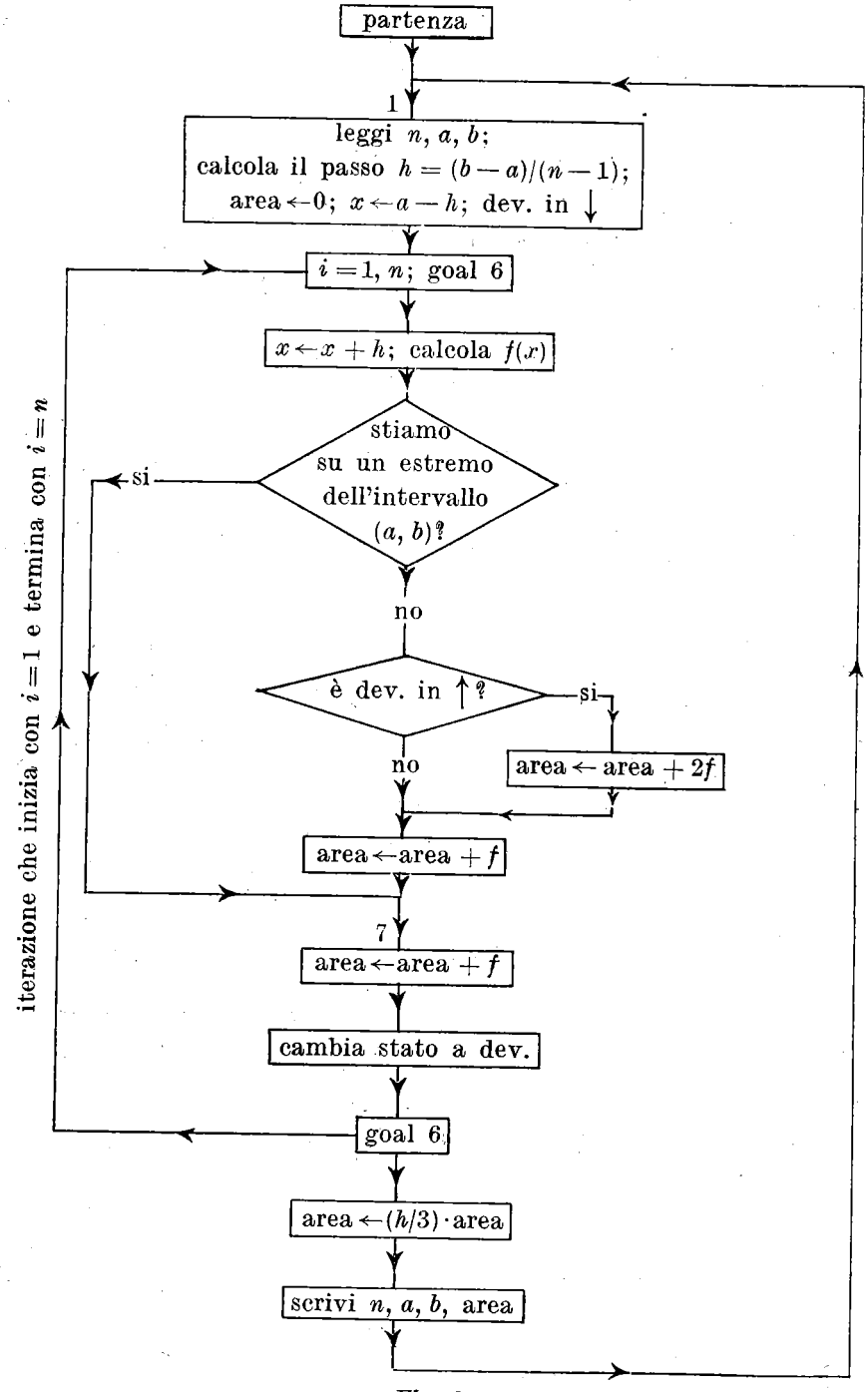
partenza
1
leggi n, a, b;
calcola il passo $h = (b-a)/(n-1)$;
area $\leftarrow 0$; $x \leftarrow a-h$; dev. in $\downarrow$
$i = 1, n$; goal 6
$x \leftarrow x+h$; calcola $f(x)$
stiamo su un estremo dell'intervallo (a, b)?
si
no
è dev. in $\uparrow$?
si
no
area $\leftarrow$ area $+ 2f$
area $\leftarrow$ area $+ f$
7
area $\leftarrow$ area $+ f$
cambia stato a dev.
goal 6
area $\leftarrow (h/3) \cdot$ area
scrivi $n, a, b,$ area
iterazione che inizia con $i = 1$ e termina con $i = n$

Fig. 3.

viatore viene posto in $\updownarrow$. Successivamente x assume il valore a e si calcola $f_1(\equiv f(a))$. Poiché si sta su un estremo dell'intervallo d'integrazione si va direttamente al blocco 7, *ed in « area » viene memorizzata* f_1.

Il deviatore cambia stato, cioè si mette in $\uparrow$ e si ritorna all'inizio del ciclo. Viene calcolata x_2 e la f_2 corrispondente. Poiché il deviatore sta ora in $\uparrow$, si intraprende un percorso per cui, giungendo al blocco 7, ad « area » è stata sommata la quantità $4f_2$; indi cambia lo stato del deviatore, cioè questo diventa $\updownarrow$, e riprende il calcolo con $i = 3$. Calcolate x_3 ed f_3 viene preso un percorso per cui, giungendo al blocco 7, ad « area » è stata sommata la quantità $2f_3$; cambia lo stato del deviatore ($\uparrow$ è il nuovo stato), per cui alla successiva iterazione ad « area » viene a sommarsi la quantità $4f_4$, e così via, fin quando ($i = n$) x assume il valore b per cui, trovandoci su un estremo dell'intervallo (a, b) si va direttamente al blocco 7, nel quale viene sommato ad « area » l'ultimo termine $f_n(\equiv f(b))$ della somma.

Finalmente ora si scavalca il goal 6; in « area » viene memorizzato il secondo membro della formula (7); si stampano i risultati e l'elaborazione ha termine.

Ecco come può essere scritto un programma in FORTRAN relativo a questo diagramma a blocchi:

```
                    12345 6 7 8 9 .......
                    C   CALCOLO NUMERICO DI UN INTEGRALE DEFINITO
                    C   MEDIANTE LA REGOLA DI SIMPSON
                      111   FORMAT (8I10)
                      222   FORMAT (13I10)
                      333   FORMAT (5E16.7)
                      444   FORMAT (8E16.7)
                            LOGICAL DEV
                    C   N   DEVE ESSERE UN NUMERO DISPARI
                        1   READ (5, 111)N
                            READ (5,333)A, B
                            H = (B - A)/FLOAT(N - 1)
                            X = A - H
                            AREA = 0.
                            DEV = .FALSE.
                            DO 6 I = 1,N
                    C   CALCOLO X ET F
                            X = X + H
                            .....
                            .....
                            .....
                            F = .....
                    C   FINE CALCOLO X ET F
                            IF(I.EQ.1.OR.I.EQ.N)GO TO 7
                            IF(DEV)AREA = AREA + F + F
                            AREA = AREA + F
                        7   AREA = AREA + F
                        6   DEV = .NOT.DEV
                            AREA = .333333333*H*AREA
                            WRITE (6, 222)N
                            WRITE (6, 444) A, B, AREA
                            GO TO 1
                            END
```

iterazione che inizia con $i = 1$ e termina con $i = n$

Seguono:

una scheda col valore numerico di N, secondo il formato specificato a programma;

una scheda coi valori numerici di A e B, secondo il formato specificato a programma.

Le richieste del tipo FORMAT denunciano come sarà il formato (incolonnamenti su schede e sui fogli) dei dati numerici che veranno letti/scritti nella fase di esecuzione del programma, e non è nostra intenzione intrattenerci a discuterle.

La richiesta LOGICAL DEV denuncia che il DEViatore è da considerare una variabile capace di assumere soltanto o l'uno o l'altro dei due valori .TRUE. e .FALSE..

Le richieste di lettura e di scrittura contengono rispettivamente il numero 5 e 6 tra parentesi. Questi numeri significano che sono chiamati a servire queste operazioni di ingresso/uscita il lettore di schede e la stampatrice veloce invece che, per esempio, le unità a nastro magnetico. Gli altri numeri in parentesi sono i numeri delle richieste relative al formato dei dati in ingresso e uscita.

Si è dovuto scrivere, per il calcolo del passo d'integrazione,

$$H = (B - A)/\mathrm{FLOAT}(N - 1)$$

invece che

$$H = (B - 1)/(N - 1)$$

perché ci sono alcune regole formali da osservare nel redigere un programma in FORTRAN.

La richiesta DO (in italiano si direbbe « fa », imperativo del verbo fare) provoca un calcolo ciclico.

DO 6 I=1, N significa: in tutte le richieste che seguono e fino alla richiesta numerata con 6, che è chiamata « goal » del DO, dà al contatore I il valore 1 ed eseguile. Quando hai eseguito l'ultima (il goal, cioè la 6), dà al contatore I il valore precedente incrementato di 1 e rieseguile, e prosegui cosí finché I ha acquistato il valore N; esegui per l'ultima volta la serie delle richieste, indi rivolgiti alla richiesta successiva al goal.

Non abbiamo specificato in questo programma la funzione $f(x)$. Se, per esempio, $f(x)$ è una curva di Gauss, cioè del tipo e^{-x^2}, si può scrivere la richiesta $F = EXP(-X**2)$ per calcolare questa funzione. I due asterischi denotano elevamento a potenza. Un solo asterisco denota segno di moltiplicazione. La domanda se si è o no su un estremo dell'intervallo (a, b) è espressa dalla IF(I.EQ. 1.OR.I.EQ.N), e l'espressione in parentesi significa *I uguale ad* 1 *oppure I eguale ad N*.

Se è vero che è cosí si va alla 7, se no si va alla successiva richiesta che è ancora un bivio: *Se* DEV sta in .TRUE. viene eseguita la somma AREA = = AREA + F + F, se DEV sta in .FALSE. si va alla successiva richiesta omettendo di eseguire la somma detta. Si còmmuta lo stato del deviatore scrivendo DEV =.NOT.DEV.

Dopo che sono stati stampati i risultati ci si rivolge (GO TO 1) alla richiesta di lettura di nuovi dati per il calcolo dello stesso integrale tra altri due limiti (a, b) e un nuovo valore di n. Se le schede dati sono terminate entra in memoria, invece di una scheda dati, una scheda controllo relativa ad un lavoro che ha aspettato il

completamento del precedente. La macchina avverte, mediante la stampatrice veloce, che considera terminato il nostro lavoro e inizia a considerare il successivo.

La richiesta END che compare alla fine del programma deve essere scritta alla fine di qualunque programma sorgente. Non è una richiesta di fermata, ma è invece una segnalazione al traduttore che è terminata la lista di richieste FORTRAN da codificare in linguaggio base.

In altre parole scrivere END alla fine del proprio programma è l'unico tributo che si paga per poter commissionare servendosi di un linguaggio umano il lavoro ad una macchina che intende soltanto un linguaggio fatto di successioni di soli « si » e « no ». Questo tributo non è molto esoso, in verità; ed allora, visto che ci si esprime in inglese, ci conviene considerare la richiesta END come un THANK YOU da apporre (non farlo, altrimenti il calcolatore protesta!) alla fine delle nostre richieste FORTRAN. Un « thank you » rivolto alla tecnica elettronica ed alla tecnica di programmazione che insieme riescono a fornire a questa nostra società di oggi possibilità strabilianti in questo campo, e che forniranno per quella del duemila possibilità che nessuno può prevedere.