Parla più FORTH

di Raffaello De Masi

Quarta parte

La matematica in virgola fissa

Nella puntata precedente avevamo preannunciato l'arrivo di un argomento spinoso, che è stato da sempre oggetto di aspra polemica non solo tra forthisti e non, ma anche tra programmatori di altri linguaggi. Chiunque ha un minimo di esperienza di programmazione sa come, appena possibile, sia sempre opportuno definire come Integer ... tutto quello che sia possibile. Chi conosce appena un po' il linguaggio assembly sa quanto sia complicato far capire ed eseguire al sistema operativo un' operazione con numeri decimali (immaginate cosa sia prendere due o più numeri decimali, magari con un numero di cifre diverse, ed elevarli tra di loro utilizzando le regole della matematica binaria). In pratica le operazioni con i numeri decimali, proprio per le enormi complicazioni che comportano, prolungano, talora in maniera estenuante, i tempi di esecuzione. Sono proprio questi, uniti ai tempi di interpretazione riga per riga, che rendono il Basic così lento e cigolante.

Che le operazioni con i numeri interi siano più veloci di quelle con i numeri decimali anche in Basic, è possibile evidenziarlo con un semplice loop. Ad esempio, un Apple IIe esegue 1000 volte il prodotto di due numeri interi di una cifra in 4.7 secondi; il prodotto di due numeri decimali ad una cifra viene eseguito in 11.8 secondi, mentre, se le stesse cifre decimali divengono 6, il tempo si allunga a quasi 50 secondi.

Le stesse operazioni, eseguite su un vecchio Hewlett-Packard 85, in possesso di un sistema operativo e di un Basic più efficienti, impiegano tempi meno che dimezzati, ma siamo sempre lontani dalla fulmineità di altri linguaggi.

Allorché Moore si accinse a progettare il primo compilatore Forth, evidentemente si trovò di fronte al problema della velocità. Gli scopi cui dovevano servire i suoi programmi rendevano questa caratteristica assolutamente prevalente e di importanza di gran lunga primaria. Da qui a scegliere la notazione numerica intera il passo fu breve, dando il via alle polemiche più diverse già accennate.

In verità, più che di notazione intera si preferisce parlare di notazione in virgola fissa. Vediamo come questa, se utilizzata con attenzione risulta, almeno in parte, altrettanto efficiente di quella in virgola mobile (che poi sarebbe quella che usiamo in pratica tutti i giorni).

Prendiamo una calcolatrice tascabile ed immaginiamo di dover eseguire l'operazione

$$3.2 \times 5.33$$

Avremo

tastiera	display
3.2	3.2
×	3.2
5.33	5.33
=	19.721

La virgola (il punto decimale) si sposta (fluttua) lungo il display secondo la bisogna. Questo tipo di rappresentazione è definito in virgola mobile (floating point notation).

La stessa calcolatrice, nell'eseguire l'operazione 3.2/1000000000 darà

3.2E-9

vale a dire che il risultato ottenuto viene conservato in memoria sotto forma di due numeri di cui il secondo rappresenta la potenza di 10 a cui va elevato il primo.

Lo scopo della notazione in virgola mobile è, come appare evidente, quello di rappresentare un enorme campo di valori con numeri piuttosto piccoli e semplici.

La rappresentazione in virgola fissa, invece, consente di manipolare numeri senza conservare in memoria, per ciascuno di essi, la posizione della virgola. Ad esempio, lavorando con valori di lunghezza in metri e sui sottomultipli (immaginiamo fino ai millimetri), tutti i dati vanno introdotti e vengono manipolati e conservati in memoria utilizzando come unità di misura il sottomultiplo più piccolo (nel caso particolare i mm). Si può, evidentemente, affidare poi al programma il compito di ripristinare, eventualmente, nei risultati, il punto decimale desiderato.

È evidente come in questo modo il sistema operativo non debba accollarsi l'ingrato e lento compito di manipolare dati di forma iniziale diversa, ma possa, per così dire, lavorare su dati portati "alla stessa scala". La domanda immediata che ci si pone è la seguente: è davvero vantaggioso l'uso di questa notazione rispetto a quella più convenzionale e perché no, più semplice, della virgola mobile? La risposta non è facile e, come in tutto nella vita, ci sono vantaggi dall'una e dall'altra parte. Tutto sta a vedere che cosa serve effettivamente al programmatore.

L'argomentazione più valida, ed evidentemente irreprensibile dei "mobilisti" è la seguente: per quale motivo devo accollarmi compiti, magari anche un po' barbosi, che posso lasciare volentieri al computer?

È una argomentazione validissima, specie nella elaborazione di programmi scientifici, dove il range dei valori può essere enorme; in tal caso l'uso della virgola mobile può rendere la vita del programmatore molto più facile. Il forthista, invece, percepisce il ruolo del computer in maniera diversa. Egli acquista, programma, ed usa un calcolatore con il primo scopo di massimizzare l'efficienza della propria macchina. Pertanto il suo fine non è quello di far fare, a qualsiasi costo, ogni cosa al computer, ma, si noti la sottigliezza, di far fare qualsiasi cosa al computer nel miglior modo. Perciò, in vista dell'efficienza, occorre seguire il programma nel modo più veloce e pulito possibile.

Facciamo un esempio pratico, peraltro credo interessante. Chi scrive esercita la professione di geologo ed utilizza un calcolatore Hewlett-Packard mod. 87. Uno dei compiti più gravosi ed ingrati nella normale routine professionale è quello di determinare la stabilità di un versante. Questa operazione, concettualmente, consiste nella individuazione di una arbitraria superficie circolare di scorrimento e nella suddivisione dei volumi compresi tra questa e la superficie topografica in prismi verticali, di cui occorre calcolare il volume, il peso, le superfici di scorrimento, l'azione di un sisma, i momenti stabilizzanti e ribaltanti anche in funzione degli attriti interni, della presenza di acqua, di eventuali sovraccarichi quale una costruzione od un terrapieno, di opere di sostegno, ecc. La risoluzione del sistema di vettori risultante dalla valutazione di tali parametri consente di ottenere un certo valore, definito coefficiente od indice di stabilità di un pendio (che, per la cronaca, diviene pericoloso se inferiore a 1.2).

Proprio perché la superficie di scorri-

mento (ed il suo raggio) è stata arbitrariamente scelta, occorre ricominciare daccapo, ipotizzandone un'altra e così via fino a trovare il minimo valore possibile. Trattandosi, come si vede, di un processo iterativo, basato su ipotesi arbitrarie, occorre eseguire un enorme numero di tentativi prima di essere ragionevolmente sicuri di aver individuato un valore prossimo al minimo possibile.

L'esecuzione manuale del metodo, che si basa molto sulla precisa ricostruzione grafica e valutazione numerica del pendio, dei cerchi di scorrimento e dei conci, è affidata ad un durissimo ed estenuante lavoro di annotazione (basti per tutte l'esatta valutazione del peso dei prismi, aventi al tetto una superficie irregolare ed alla base calotte cilindriche). Generalmente dopo 2 o 3 tentativi, che per pendii aventi anche solo una quindicina di punti quotati possono impegnare una intera giornata, la mente comincia a vacillare e si comincia a vedere la Madonna di Fatima. Appare quindi evidente come l'esatta applicazione del metodo sia pura utopia; credo, infatti, che neppure Fellenius, uno svedese ideatore del metodo (che appunto porta il suo nome), abbia davvero mai trovato, operando manualmente, il più basso indice possibile di sicurezza di un rilievo.

Il procedimento, per la sua esasperata iteratività, si presta in maniera eccellente ad essere sviluppato da un computer. La presenza di numerose routine di calcolo, dovute all'elevato numero di fattori concorrenti alla determinazione del coefficiente, porta ad un loop di ricerca del valore minimo molto lungo. Se questo va poi considerato moltiplicato per il numero dei conci, che possono essere anche diverse decine e più, appare evidente come l'insieme delle operazioni da eseguire possa per ogni tentativo, in condizioni anche non eccezionalmente complesse, raggiungere le diverse migliaia o anche le decine di migliaia.

Il programma iniziale fu sviluppato in Basic ed occupava, senza dimensionamento di variabili, oltre 45 K di RAM. Quando il numero dei conci superava la decina, il ciclo di calcolo completo diveniva estenuantemente lento per l'enorme massa di dati che il calcolatore era costretto ad elaborare. Pur se affascinato dal pensiero del terribile lavoro cui il computer era costretto, mi chiesi se era possibile rendere ancora più efficiente il tutto. Poiché il Forth, (in linguaggio macchina) di cui ero in possesso non ammetteva la coesistenza del Basic, dovetti elaborare una utility, in assembler, che consentiva, una volta acquisiti i dati del problema, di passarli al Forth per elaborarli numericamente, e li ritrasferiva al Basic per le fasi finali di output e disegno, sfruttando in tal modo la grafica dell'87, non implementata sul sistema Forth allora in mio possesso.

I dati venivano introdotti, in ambiente Basic, in notazione in virgola mobile. Una routine eseguiva la loro trasposizione in virgola fissa, scalandoli al sottomultiplo più piccolo. Quindi si passava in ambiente Forth utilizzando una fase intermedia di conservazione in file dati provvisori (sulle cui modalità non oso tediarvi) e si affidava ad esso l'elaborazione dati fino alla risoluzione. Di nuovo utilizzando la memoria di massa come area tampone si ritornava in Basic per l'output su stampante e plotter.

I risultati sono stati superiori a qualsiasi aspettativa. Uno per tutti basti un esempio: la verifica completa di un pendio composto di 15 conci, in presenza d'acqua, di sisma e di sovraccarico prodotto da una costruzione, eseguito su 125 tentativi che, nel pur rapido Basic HP, impiegava un'ora e venti minuti, con questo sistema ha richiesto solo 18 minuti, cioè meno di un quarto del tempo.

Il rovescio (immancabile) della medaglia, dato dal più preciso e meno rigido lavoro di programmazione (che per la cronaca ha impiegato 12 screen abbastanza pienotti) è stato ben sopportato. Credo che in questa fase si sia davvero centrato il problema e la differenza tra il Forth ed il Basic o Pascal, ad esempio. In questi, l'accuratezza e la cura della programmazione è lasciata un po' larga di manica, a vantaggio della facilità di redazione e di elasticità di disegno delle flow-chart.

Il Forth, invece, anche su sistemi dedicati, richiede programmi di fattura più fine, inevitabilmente più delicati, che richiedono un più accurato lavoro di progettazione iniziale del software ed un più lungo lavoro di messa a punto ed ottimizzazione, che, comunque, ripaga abbondantemente in termini di rapidità. Per la verità, disporre di una macchina, come già abbiamo detto, dedicata, porterebbe ad una vita più facile e credo che lo stesso Moore possegga oggi sistemi operativi e routine di base ben migliori di quelle oggi disponibili per noi Forthisti al di fuori dell'olimpo. Inoltre, generalmente, grossi computer destinati a lavorare in Forth utilizzano comunque un chip separato a frequenza elevata, destinato esclusivamente alle operazioni in virgola mobile. Ecco risolto il problema, in ossequio al proverbio delle mie parti "Tot pavatio, tot pittatio".

In conclusione l'uso della virgola mobile è preferibile quando:

— Il computer viene utilizzato molto come calcolatrice digitale.

— Occorre manipolare numeri molto grandi o molto piccoli e comunque superiori a 2 miliardi in valore assoluto (vedremo poi come questa limitazione possa essere in Forth, con un artificio, superata).

— La rapidità ed elasticità di redazione di un programma viene considerata molto più importante della rapidità di esecuzione.

Comunque anche una mezza cartuccia di linguaggio Forth, come se ne sono viste circolare ultimamente su alcuni home computer, dovrebbe disporre di una serie di comandi di alto livello chiamati "operatori scalari" che consentono di operare su numeri interi fornendo risultati paragonabili, in precisione ed accuratezza, al miglior sistema in virgola mobile.

Nuovi operatori numerici

Per introdurre in maniera adeguata gli operatori scalari è necessario definire ancora alcuni piccoli particolari ed alcuni nuovi operatori numerici, di facile comprensione. Una tabella ordinata di questi potrebbe essere (notare l'assenza di spazi).

```
1+
   1—
   2+
   2—
   2*
   2/
```

Essi eseguono esattamente le stesse operazioni come se numero ed operatore fossero separati. Gli ultimi due rappresentano i ben conosciuti smistamenti a destra ed a sinistra dell'assembler.

A prima vista sembrerebbe non esserci necessità evidente di inserire queste nuove word. In effetti, invece, trattandosi di operazioni estremamente diffuse specie quando si lavora in binario, ed essendo già presenti in dizionario in linguaggio macchina, ed essendo ancora interpretate direttamente dal microprocessore senza compilazione, la loro esecuzione è molto più veloce del conosciuto binario numero-operatore.

Esistono, inoltre, altri operatori numerici (peraltro esistenti in molti linguaggi) che in Forth assumono importanza rilevante per la tipica architettura dello stack.

Essi sono:

ABS n - - - (n) restituisce il valore assoluto.
MIN n1 n2 - - - nminimo lascia in stack il valore minimo
MAX n1 n2 - - - nmassimo idem con valore massimo.
MINUS (NEGATE) - - - -n cambia di segno a n
(ne esegue il complemento a 2).

Un esempio di applicazione del primo potrebbe essere la classica sottrazione senza darsi troppa cura dell'ordine di disposizione in stack. Scriviamo allora la seguente definizione:

```
: DIFFERENZA - ABS ;
```

ed immaginiamo di volere il risultato della sottrazione di 25 ed 8. Il risultato, ottenuto con la word DIFFERENZA, sarà sempre lo stesso, comunque si inseriscono i numeri.

Prima di passare agli operatori scalari è però ancora necessario definire una struttura nuova che rappresenta un vero biglietto da visita univoco del Forth, il Return Stack.

Il return stack

Finora avevamo sempre parlato di Stack tout-court; per la verità in Forth gli stack sono due. Quello finora conosciuto, più propriamente detto "parameter stack" o "data stack", che continueremo a chiamare col solo nome, ed il cosiddetto "Return Stack", uno stack anch'esso del tipo

LIFO, cui non si può accedere e che non accoglie dati dalla tastiera, ma serve a diverse altre cose, come area di parcheggio per dati intermedi, contatore per i limiti di loop, puntatore di sistema per le word che utilizzano altre word.

Inoltre può funzionare come buffer temporaneo di dati prelevati non da tastiera, con una sola ed inflessibile condizione: tutti i dati inseriti nel Return Stack nel corso della esecuzione di una word vanno rimossi prima del completamento dell'esecuzione della word stessa. Questo perché, all'inizio dell'esecuzione di una word, il sistema operativo lascia in TORS (Top Of Return Stack) un puntatore, che va a recuperare al termine del sistema.

L'elenco dei comandi riferibili al RS varia, come al solito, a seconda di sistemi. Il minimo vocabolario è però:

>R n --- preleva il valore in TOS e lo deposita in TORS (to R)

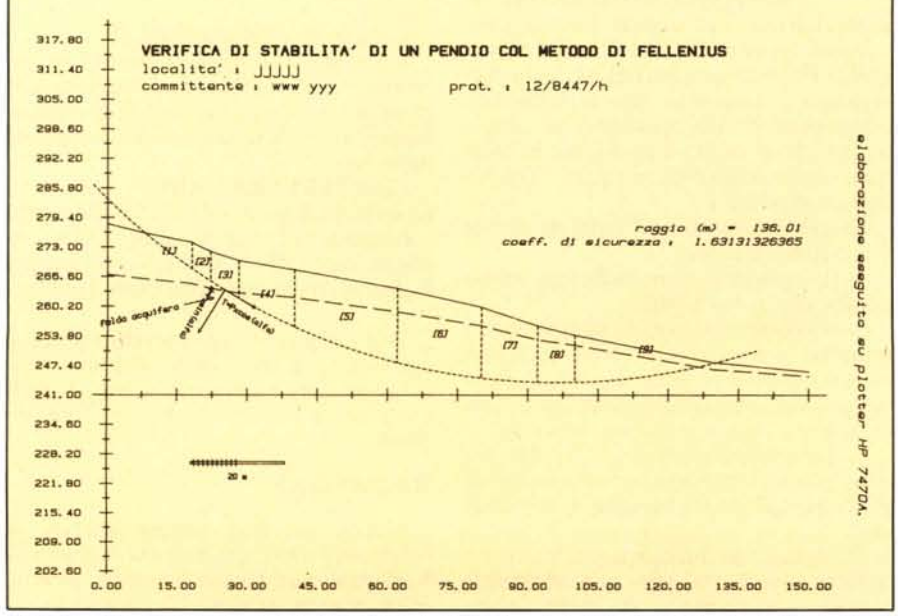
R> --- n preleva il valore in TORS e lo deposita in TOS (R from)

I --- n copia il TORS in TOS senza cancellarlo

I'--- n copia il secondo valore del Return Stack in TOS senza cancellarlo dal RS

J --- n copia il terzo valore del RS in TOS senza cancellarlo

```
VERIFICA DI STABILITA' DI UN PENDIO COL METODO DI FELLENIUS

ELABORAZIONE ESEGUITA SU COMPUTER HEWLETT-PACKARD MOD. 87

LOCALITA'              jjjjj
COMMITTENTE            www yyy
DATA                   10/10/84

Ipotesi per cerchio di scorrimento passante per i seguenti punti :
Punto superiore  -  coordinate : l. (m) =    8
                                 h. (m) =    276
Punto inferiore  -  coordinate : l. (m) =    130
                                 h. (m) =    248

E' stata presa in considerazione la presenza di falda acquifera.
E' stato considerato un carico verticale e centrato,in corrispondenza del concio
 3 ,con sovraccarico lineare di 21.4 t/ml .

          DATI RELATIVI AL CERCHIO DI MINORE COEFFICIENTE

Caratteristiche geotecniche :     peso specifico [ gamma ] (gr/cm3) : 1.7845
                                  coesione [ c' ] (kg/cm2):          .34
                                  angolo d'attrito [ phi' ] (deg):   21.5

Calcolo eseguito in fase sismica con incremento pari a  .1 , corrispondente alla cat. 1

DATI DIMENSIONALI DEL CERCHIO DI SCORRIMENTO :
raggio (in m)     :  136.01
corda  (in m)     :  125.17

coefficiente minimo ricavato su  271  tentativi

                    Tabella analitica dei valori

concio      volume(m3)     peso(t)     alfa(deg)     T (t)      N (t)

 1           28.476         50.815      52.339       57.575      35.07
 2           24.208         43.199      56.016       29.548      27.729
 3           43.426         98.894      58.514       55.116      60.085
 4           121.9          217.53      62.84        107.64      118.65
 5           310.39         553.89      70.6         206.47      236.22
 6           282.04         503.3       79.38        172.28      142.22
 7           165.65         295.6       85.776       105.66      51.252
 8           90.021         160.64      89.998       62.561      16.07
 9           153.59         274.08      98.08        261.51      -38.523

COEFFICIENTE DI SICUREZZA = ( 808.013  -  191.511  +  441.85 ) /  648.773  =  1.63131326365
```



Elaborati riassuntivi di una verifica di stabilità di un versante; i conci sono numerati in figura, e rappresentano la superficie di scorrimento ideale di una probabile frana rotazionale. La falda acquifera (in tratteggio elongato) influisce solo parzialmente sui conci, complicando ancora di più i già ardui calcoli. Si tiene anche conto dell'influenza del fattore sismico e di sovraccarichi centrati.

Gli operatori scalari

Il primo degli operatori scalari che vedremo è */ (notare l'assenza di spazio). Come il suo nome lascia intuire (multiply then divide) esso consente di calcolare la parte frazionaria di un numero. È molto più preciso della sequenza / * (notare che */ esegue le operazioni dal basso verso l'alto dello stack) utilizzando come risultato intermedio un numero in doppia precisione.

I semplici operatori numerici, come * e / utilizzano infatti, se non altrimenti specificato, sempre numeri in singola precisione. Questa è talvolta una limitazione, perché anche utilizzando numeri senza segno (e quindi sfruttando tutti gli otto bit di un byte), i risultati, anche intermedi, non possono essere superiori a 65535.

Pertanto il valore di 2200 * 50/40, non eseguibile immediatamente con operatori normali

```
2200
50
*    (110000 overflow!)
?
```

si fermerebbe al primo prodotto senza proseguire (se non specificando in precedenza che i risultati intermedi vanno calcolati in doppia precisione, con una tecnica che vedremo al più presto).

Invece:

```
2200
50
40
*/
```

fornirà il voluto 2750 OK

Inoltre, pur senza entrare in dimostrazioni lunghe e inutili, è importante sapere che i margini di approssimazione dell'operatore */ sono molto più ridotti.

Un altro operatore scalare è */MOD, perfettamente analogo al precedente tranne il fatto che conserva in TOS quoziente e resto.

Ancora una volta abbiamo finito. Se siete ancora saldi in sella, accampatevi e non temete. La prossima volta sarà ancora più dura: parleremo delle strutture decisionali e dei loop. È arrivato il momento di cominciare a scrivere qualche programma decente e di far vedere cosa il Forth è capace di fare, o come si dice a Merano, "che core tiene 'npietto". MC

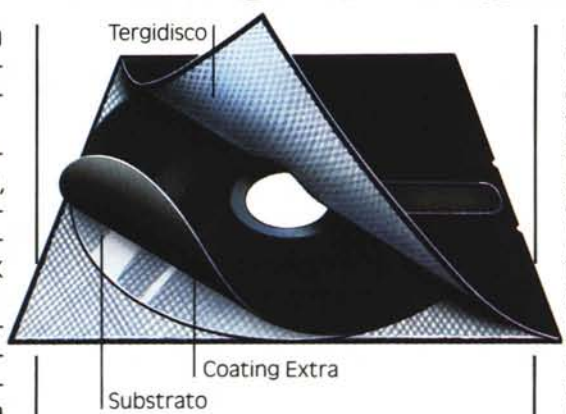
IN UN FLOPPY DISK DIASPRON LE QUALITA' SUPERFICIALI SONO LE PIU' PROFONDE.

E' dalla superficie che si giudica un floppy disk, ma solo un'analisi approfondita permette di apprezzarne le qualità.

Trattamento "Coating Extra": rivestimento di ossido magnetico, addittivato con lubrificante ad alta protezione anti-usura, che assicura lunga durata al floppy disk Diaspron.

Lappatura "Super Finish": finitura superficiale realizzata combinando in modo ottimale i parametri pressione, velocità e tempo: la superficie così ottenuta ha caratteristiche tali da assicurare il miglior contatto con le testine magnetiche e la massima protezione contro l'usura del disco e delle testine stesse.

Fabbricazione automatica in "camera bianca": processo produttivo esclusivo, automatizzato mediante robot ed effettuato in ambiente a livello di polverosità rigorosamente controllato; ciò garantisce il floppy disk Diaspron esente da particelle contaminanti di qualsiasi tipo, causa principale del "Soft Error".



Dispositivo tergidisco: la particolare fibrosità del liner assicura la costante pulizia del disco, garantendo contemporaneamente un attrito compatibile con le esigenze del drive.

Certificazione "Error Free 100%": la certificazione è REALMENTE l'ultima operazione del processo produttivo: per questo i floppy disks Diaspron sono davvero ERROR FREE 100% e superano gli standards qualitativi più severi delle principali Case costruttrici di hardware.

Ecco perché il floppy disk Diaspron vince in superficie!

Diaspron: microfloppy da 3,5", floppy disk da 5,25" e da 8", singola e doppia faccia, singola e doppia densità (da 80 Kbytes a 1 Mbyte), compatibili con tutti i principali sistemi EDP, Word Processors e Personal Computers esistenti sul mercato.

DIASPRON

AMICO DEI VOSTRI DATI E DEL VOSTRO DRIVE.

MODULAR ELECTRONICS

presenta

MTX 512

PROFESSIONAL COMPUTER ™

Tastiera in alluminio anodizzato nero • Chassis portatasti in acciaio • 24 K. ROM contenente MTX BASIC + comandi grafici LOGO-type + MTX NODDY • 16 K. RAM dedicata video • 64 K. RAM disponibile user (espansibile fino a 512 K.) • CPU Z80A (Zilog) (4MHz) • Set di caratteri maiuscoli e minuscoli (40 colonne/24 righe) • Uscita monitor colore/B.N. • Quattro canali suono controllati da software (uscita Hi.Fi.) • Porta per espansione ROM MTX PASCAL - MTX FORTH • 32 livelli di SPRITES • 8 virtual screens • In/out cassette fino a 2400 Baud • Interfaccia parallela (Centronics) • ASSEMBLER-DISASSEMBLER • Funzione front-panel con single step • 8 tasti funzione (16 funzioni con shift) • Orologio interno controllabile da software • Ingresso per due joystick • Tastierino numerico separato •

MTX 500 LIRE 599.000 IVA INCLUSA

MODULAR ELECTRONICS - VIA BRITANNIA 29 - 00183 ROMA - TEL. 06/7597701-6008340