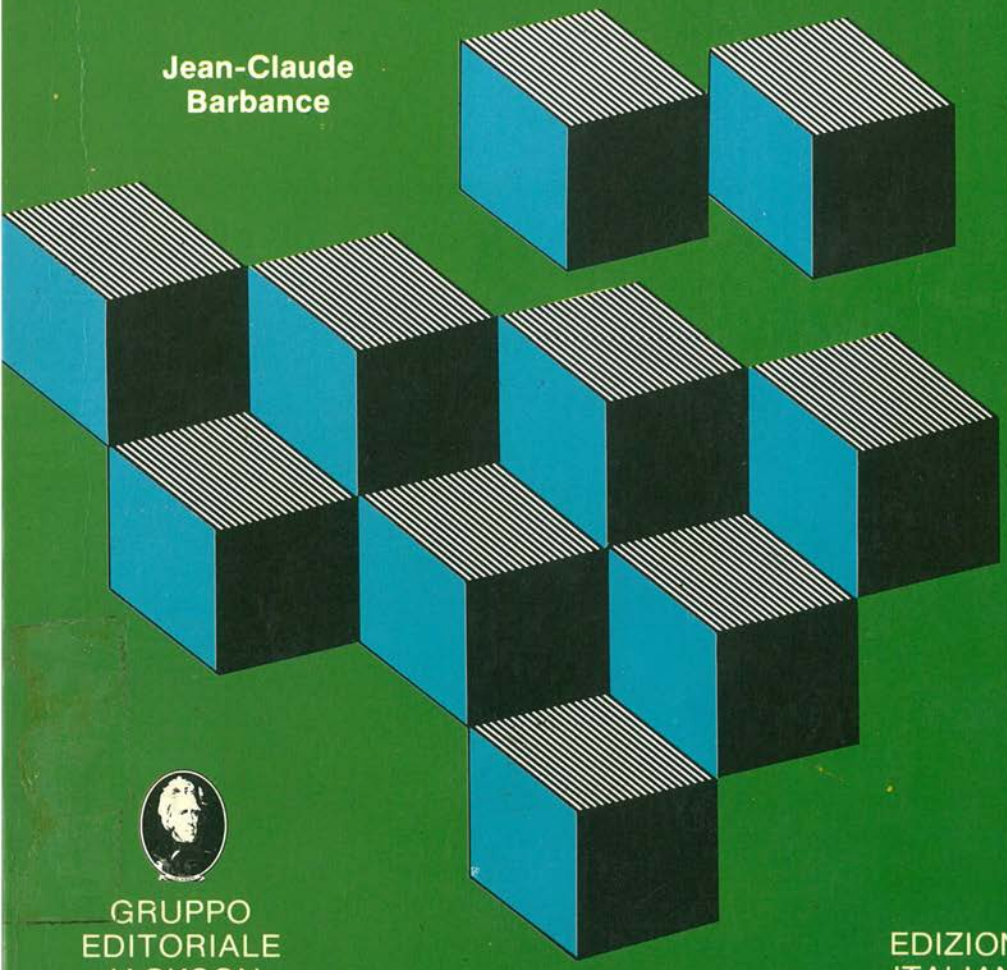
COME PROGRAMMARE

Jean-Claude
Barbance



GRUPPO
EDITORIALE
JACKSON

EDIZIONE
ITALIANA

COME PROGRAMMARE

di
Jean-Claude
Barbance

GRUPPO
EDITORIALE
JACKSON
Via Rosellini, 12
20124 Milano

Jean-Claude Barbance lavora nel campo degli elaboratori dal 1958 ed ha avuto occasione di utilizzare sia quelli di piccole che di grosse dimensioni.
Egli è persuaso che esistono in informatica dei concetti di base che permettono di realizzare programmi sia per i microcalcolatori che per quelli superpotenti.

L'autore ringrazia per il prezioso lavoro svolto nella stesura dell'edizione italiana la signora Francesca di Fiore e l'ing. Roberto Pancaldi.

Stampato in Italia da:
S.p.A. Alberto Matarelli - Milano - Stabilimento Grafico

SOMMARIO

PREFAZIONE 4

CAPITOLO 1 - REALIZZAZIONE DI PROGRAMMI: LE FASI 7

CAPITOLO 2 - LA DEFINIZIONE DEGLI OBIETTIVI 13

CAPITOLO 3 - L'ANALISI 23

CAPITOLO 4 - LA CODIFICA E LA MESSA A PUNTO DEL PROGRAMMA 33

CAPITOLO 5 - PRESENTAZIONE DEGLI ESEMPI 51

CAPITOLO 6 - RAPPRESENTAZIONE DI UN NUMERO DECIMALE MEDIANTE UNA STRINGA DI CARATTERI ALFABETICI 53

CAPITOLO 7 - IL GIOCO DEL 421 79

CAPITOLO 8 - LA CONTABILITA' PERSONALE 135

PREFAZIONE

A chi si rivolge questo libro?

Questo libro è destinato a tutti coloro che, senza un'esperienza specifica in campo informatico, cominciano ad utilizzare un piccolo calcolatore o un Personal Computer, sia che lo facciano a casa, sia presso un rivenditore o in qualche club.
Immagino che queste persone abbiano imparato un linguaggio di programmazione, molto spesso il Basic, o attraverso un manuale o frequentando qualche corso.

Era il caso di alcuni miei amici: essi avevano acquistato un Personal Computer, e dopo averne studiato con cura il manuale d'uso, avevano imparato a scrivere dei brevi programmi. Facendo inoltre parte di un club di giovani, ne dovevano organizzare l'attività e quindi parve loro utile poter utilizzare il Personal Computer per piccoli lavori; dopo essersi opportunamente documentati, si convinsero che diverse loro idee erano realizzabili utilizzando l'elaboratore che avevano a disposizione.

Ma a questo punto sorse un problema: come fare a realizzare un programma di qualche centinaio di istruzioni e non più di qualche istruzione soltanto.
Tale esperienza è alla base di questo libro: utilizzando i consigli dati qua e là e le annotazioni preparate, ho costruito un metodo, non certo originale ma accessibile anche a coloro che si avvicinano all'informatica senza una preparazione di base adeguata.

Si tratta di passare da un'idea vaga di un problema di programmazione (contabilità di un dettagliante, gioco del poker,...) alla realizzazione pratica del programma che svolga le funzioni richieste.

Gli argomenti trattati:

Per codificare i programmi da noi realizzati era necessario scegliere un linguaggio che fosse utilizzabile sulla maggior parte dei Personal in commercio. Poiché tra tutti i linguaggi il più diffuso è senza dubbio il Basic sono stato indotto a scegliere una versione di questo linguaggio (che chiamerei impropriamente un "dialetto Basic") che possedesse solo quelle funzioni disponibili sulla maggior parte dei Personal, eliminando invece il set delle istruzioni specifiche previste solo su questa o quella macchina particolare.

Una volta che il lettore avrà compreso il metodo, nulla gli impedirà di modificare gli esempi proposti utilizzando un linguaggio più completo, per sfruttare completamente le capacità dell'elaboratore a sua disposizione.

Per ciò che riguarda le dimensioni dei programmi, il metodo proposto è facilmente applicabile anche nel caso di programmi di grossa mole. Ciononostante non ho ritenuto opportuno, in questa sede, estendere la trattazione a problemi di dimensioni tali da richiedere, per la realizzazione del programma, l'impegno di una équipe di più persone per lunghi periodi di tempo. Ciò premesso diciamo che la nostra trattazione si limiterà a programmi che non richiedono più di qualche centinaio o qualche migliaio di istruzioni e che possono facilmente "girare" su un computer con una memoria compresa tra 4K e 64K, provvisto solo di qualche semplice unità periferica: una tastiera, un video, un registratore a cassette ed eventualmente una stampante e/o un'unità a disco.

Come conseguenza di tutto quanto finora detto, non porrò l'accento, in quest'opera, sulla spiegazione e l'esemplificazione delle singole istruzioni di un linguaggio in quanto per questo esiste già un'esauriente letteratura formata da testi e manuali.

Cercherò invece di sviluppare subito i seguenti punti: come precisare adeguatamente un'idea iniziale, come analizzarla, come trasformarla e come verificare la correttezza della stessa ottenendo alla fine qualcosa di documentato, leggibile e facilmente modificabile. (È certo che chi ha detto: "La donna è assai mutevole..." non aveva mai avuto a che fare con persone che chiedono di realizzare dei programmi di tipo informatico).

Presentazione

I concetti utilizzati in informatica sono di per sé molto astratti; per esprimerli in modo più chiaro ho scelto di esporre rapidamente tali concetti e di illustrarli poi con degli esempi trattati molto dettagliatamente: Partendo dall'enunciazione iniziale del problema fatta con una sola frase mi sono proposto di giungere alla codifica del programma in Basic, esplicitando tutte le fasi intermedie del lavoro: tutte le vie alternative che via via si presentano e tra cui bisogna scegliere, le eventuali estensioni, i ripensamenti, le prove e le verifiche che occorre fare per ottenere un programma che dia dei risultati conformi a quanto ci si era proposti.

Il metodo descritto è quello utilizzato generalmente, indipendentemente dal linguaggio prescelto, e che mi ha permesso di realizzare, personalmente o lavorando in un'équipe, dei programmi composti da qualche centinaia di migliaia di istruzioni.

Questo metodo, reso sempre più rigoroso con il crescere delle dimensioni del programma, permetterebbe la stesura di programmi di ben maggiore portata.

In alcuni paragrafi si precisa il modo in cui realizzare una documentazione relativa al programma che permetterà sia all'autore che ad altri di apportare delle aggiunte o modifiche ad un programma già collaudato anche dopo un certo tempo dalla sua stesura originale.

Infine vogliamo osservare che i programmi proposti sono stati tutti codificati utilizzando una versione Basic comune a molti Personal (TRS-80, APPLE, PET...) In particolare questi programmi sono stati effettivamente provati su un Personal Computer e da queste prove sono stati ricavati i listing proposti. Poiché nel corso dell'esposizione si è fatto a volte uso di particolari funzioni tipiche del sistema TRS-80, in tutti i casi in cui ciò si è verificato se ne è fatta specifica menzione in modo da rendere facile la modifica dei programmi per poterli utilizzare su altri elaboratori.

Non verranno utilizzati in generale, negli esempi proposti, né stampante nè unità a disco. La stampante infatti può essere facilmente sostituita dal video, essendo nel nostro caso la differenza tra l'una e l'altro trascurabile. Del resto la sostanziale differenza consiste nel fatto che mediante una stampante non è possibile scrivere su delle righe già precedentemente impresse. (Comunque se anche vi fosse questa possibilità, verrebbe utilizzata assai raramente).

Per quanto riguarda le unità a disco, queste cambierebbero in modo essenziale la gestione dei files poiché su disco possono essere organizzati anche dei files ad accesso diretto. Tuttavia questo fatto, di cui occorre tener conto in fase di analisi del problema, non modifica minimamente il metodo con cui tale analisi va fatta. L'uso delle memorie di massa a disco diviene assai utile in fase di messa a punto e di verifica in quanto rende più pratica e veloce la gestione dei files.

CAPITOLO 1

REALIZZAZIONE DI PROGRAMMI: LE FASI

Per realizzare un programma si parte a volte da un'idea vaga del lavoro, di cui si ha solo una visione di insieme, a volte da una formulazione del problema e da una moltitudine di piccoli dettagli da riordinare.

In entrambi i casi occorre definire, attraverso fasi successive, la struttura del programma e dei dati con precisione sempre maggiore fino a giungere alla stesura del programma definitivo.

In informatica, i criteri fondamentali sono stati codificati in modo sistematico partendo dalle idee più o meno "vaghe" presenti nella mente dei "buoni" programmatori che le sperimentavano nella stesura di programmi via via sempre più complessi, e costituiscono una solida base per tutti i metodi di programmazione.

Definizione degli obiettivi

Innanzitutto conviene definire con molta cura ciò che ci si prefigge di realizzare, precisando il punto di partenza, i risultati che si vogliono ottenere ed il modo con cui si vuole passare dall'uno agli altri. Chiariamo il concetto con degli esempi.

Nel caso di un *problema di tipo scientifico* (cioè di calcoli tecnico-matematici od altro) conviene definire le variabili di partenza, il loro valore iniziale, le variabili che conterranno i risultati finali, la sequenza delle operazioni necessarie per ottenere tali risultati.

Nel caso di un *problema di tipo gestionale* il numero dei dati iniziali e dei risultati sarà, in genere, molto alto mentre il metodo per passare dagli uni agli altri sarà molto semplice.

In questo caso converrà fare particolare attenzione nel definire gli elementi di partenza, il modo di raccoglierli e di controllarne la validità. In seguito ci si occuperà del metodo con cui ottenere i risultati in quanto, anche se le operazioni sono assai semplici, contengono, in genere, un numero enorme di casi particolari. Infine ci si dovrà anche preoccupare della presentazione dei risultati: infatti il loro numero, in genere notevole, richiede una presentazione adeguata per evitare di ottenere in stampa dei tabulati di difficile consultazione.

I calcolatori comunque sono delle macchine logiche capaci di risolvere, oltre a problemi di tipo scientifico o gestionale, altri tipi di problemi che richiedono, per la definizione dei dati, dei risultati e delle procedure risolutive, un vocabolario specifico. In tal caso è necessario esprimere, nel linguaggio tecnico proprio del problema trattato, le relazioni che uniscono gli elementi di partenza con quelli di arrivo, esplicitando, eventualmente con fasi intermedie, il modo con cui si passa dagli uni agli altri.

Nella *trattazione di testi* gli elementi di partenza e di arrivo sono delle righe di testo con delle informazioni sul modo con cui andare a capo, sul come individuare un capoverso, ad esempio saltando una riga,... Le operazioni che si effettuano in genere su delle righe di testo sono: aggiungere un paragrafo, una frase o alcune parole in un punto ben preciso del testo preesistente; sostituire una parte del testo con un'altra in modo globale o elemento per elemento (una frase, una parola, ...); effettuare una corretta impaginazione con righe di lunghezza fissa e pagine formate da un numero fisso di righe; compilare il testo con caratteri di tipo e grandezza diversi, ...

Nella realizzazione di *giochi su elaboratore* bisogna fissare in partenza quali sono gli elementi del gioco: carte, navi nella battaglia navale, astronavi e altro nello "starstreek", ... Occorre inoltre stabilire le regole del gioco, cioé le possibili mosse di ogni pezzo, le interazioni che possono esistere (prese, affondamenti, inserimento di nuovi pezzi, ...) e lo scopo che ci si prefigge con l'uso dell'elaboratore: il calcolatore infatti può svolgere le funzioni di arbitro tra due giocatori e controllare la validità delle mosse effettuate dai giocatori stessi, può controllare la validità delle mosse in un solitario o addirittura fare la parte di uno dei giocatori.

In quest'ultimo caso sarà necessario assegnare alla macchina anche una condotta di gioco. La definizione di questa condotta comporta notevoli difficoltà come è dimostrato dalla realizzazione di giochi del tipo scacchi, dama, bridge, poker, ... In genere, prima di lanciarsi in un'impresa del genere, conviene consultare ciò che è già stato realizzato e pubblicato in proposito.

Quando si utilizzano delle *interfacce speciali* o dei *relais che controllano delle apparecchiature esterne* conviene dare anche delle informazioni più specifiche su ciò che si vuole controllare: è diverso controllare un sistema di allarme o un trenino in miniatura o un impianto di riscaldamento. In questo caso è necessario definire ogni volta qual'é il mezzo che fornisce le informazioni e come si manifesterà la risposta. Inoltre bisognerà specificare anche quale sarà la risposta ad ogni stimolo elementare: osserviamo a questo proposito che tale risposta può anche dipendere da eventuali stimoli precedenti di cui si è mantenuta traccia. È evidente che in questo caso il problema è notevolmente complesso in quanto bisogna tener conto anche dei fenomeni che avvengono nell'ambiente esterno.

Ora, tutto ciò che io chiamo *definizione degli obiettivi* è spesso chiamato, specie nei problemi di tipo gestionale, *analisi funzionale*. Analisi in quanto si tratta di definire e precisare con la massima cura dei problemi (o dei sottoproblemi) in modo via via sempre più dettagliato, funzionale in quanto questa analisi viene applicata, in problemi di tipo gestionale, a delle funzioni, già esistenti o da creare, che devono essere svolte in un particolare ambito (amministrativo contabile, di produzione,...). Questo non ha comunque carattere di generalità in quanto in diversi casi (vedi ad esempio il problema degli scacchi) la scelta della funzione da valutare deve essere fatta con tecniche dei tutto particolari.

La consultazione della letteratura già esistente

Nel precedente paragrafo ho accennato brevemente ad un punto che tuttavia riveste notevole importanza nello sviluppo di un programma: documentarsi su ciò che è già stato fatto in precedenza in relazione al problema da risolvere.

Gli ottimisti stimano che il 75% dei programmi che vengono elaborati siano già stati sviluppati in precedenza da altri, i pessimisti portano la loro stima al 90%. Infatti è assai raro non poter trovare il programma di cui si ha bisogno già svolto da altri, per intero o in buona parte, a cui apportare eventualmente alcune modifiche.

Purtroppo, al di fuori della sfera scientifica o tecnica, la maggior parte delle persone non si rende conto che spesso è utile pensare che un universo non può essere generato da una sola persona!

L'analisi

Una volta definiti in modo rigoroso ed organico gli obiettivi, vediamo come il nostro problema può essere risolto con l'uso di un elaboratore. Per far ciò occorre definire una corrispondenza tra le singole operazioni e gli elementi del problema, limitatamente all'ambito informatico, in cui tutto si riduce alla memorizzazione di dati e a operazioni aritmetiche o logiche su di essi.

È indispensabile analizzare tutte le operazioni necessarie tenendo conto delle informazioni raccolte nella prima fase e precisare via via la corrispondenza tra gli elementi reali esterni e la loro rappresentazione all'interno dell'elaboratore.

Se ad esempio si vuol realizzare un gioco che necessita dell'uso di una scacchiera, non bisogna porsi come problema prioritario quello di riuscire a disegnare sul video una scacchiera con i singoli pezzi, in quanto è molto più importante definire tutte le operazioni che si vogliono eseguire sulle rappresentazioni astratte della scacchiera e dei singoli pezzi disposte nella memoria dell'elaboratore; solo in seguito ci si occuperà della visualizzazione dei risultati sul video.

Procedendo in modo opposto si rischia di accrescere le difficoltà in misura tale da rendere quasi impossibile la risoluzione del problema per chi ha poca esperienza di programmazione (Chi ha già una certa esperienza avrà l'intelligenza di tralasciare il problema relativo alla rappresentazione su video e di riprenderlo solo dopo avere affrontato correttamente il problema più generale).

Il modo con cui effettuare correttamente l'analisi sarà sviluppato dettagliatamente in uno dei prossimi capitoli e verrà illustrato con una serie di esempi.

In ambito gestionale questa tappa è definita anche *analisi organica*. In essa si trovano meno differenze, rispetto al problema specifico trattato, che nella definizione degli obiettivi: infatti in ambito scientifico si insiste maggiormente sugli *algoritmi* da usare, cioé sulla sequenza di operazioni necessarie per passare dai dati iniziali ai risultati, mentre in ambito gestionale si insiste maggiormente sulla *rappresentazione dei dati*, ma a ben guardare questa differenza è assai marginale. In effetti non vi sono differenze sostanziali tra "informatica scientifica", "informatica gestionale" ed altri tipi di informatica; prova ne sia che tutta la gamma di questi problemi può essere risolta utilizzando uno stesso elaboratore. Osserviamo tra l'altro che gli algoritmi di calcolo utilizzati in ambito scientifico possono venire molto convenientemente utilizzati in ambito gestionale così come le rappresentazioni dei dati usate in ambito gestionale possono essere utilizzate anche in campo scientifico.

In tutto il resto dell'opera questa tappa verrà chiamata analisi in quanto il suo scopo principale è quello di analizzare in tutti i dettagli il lavoro che si deve eseguire sul calcolatore in modo da poter poi effettuare facilmente la codifica.

In alcuni testi l'analisi e la codifica sono unificate in un'unica tappa detta "programmazione" (del resto anche per noi l'espressione "programmazione strutturata" comprende sia l'analisi che la codifica), in quanto il passaggio da una tappa all'altra è del tutto naturale. In effetti l'analisi termina normalmente quando si arriva a poter scrivere un algoritmo in modo che ad ogni singolo passo corrisponda una sola istruzione o per lo meno un numero assai limitato di istruzioni.

La codifica

Si intende per codifica la stesura della lista delle istruzioni da dare alla macchina che indicano alla stessa sia come organizzare i dati in memoria sia quali operazioni eseguire su di essi. Se l'analisi è stata ben effettuata, la codifica diventa un'operazione semplice e rapida: sul piano professionale si stima che la codifica occupi tra il 5% e il 15% del tempo necessario per realizzare un progetto informatico.

La messa a punto

La tappa successiva è la messa a punto del programma: si tratta di controllare che il risultato del processo sia conforme a quanto ci si era prefissato di ottenere, vale a dire che il programma approntato realizzi in modo corretto ciò che, all'inizio, si era definito come obiettivo.

Questa tappa è estremamente importante e corrisponde a ciò che si fa generalmente nell'industria: prima di porre in commercio un'automobile o prima di aprire al traffico un ponte vengono fatti numerosi e rigorosi collaudi.

La messa a punto deve permettere di individuare tutti gli errori eventualmente passati inosservati in fase di progettazione, da quelli banali di ortografia in cui si incorre in fase di battitura del programma a quelli molto più gravi di tipo logico in cui si può incorrere in fase di analisi.

Vi possono poi essere degli errori già presenti in fase di definizione degli obiettivi: questo tipo di errore si verifica quando non si sono capite in modo corretto le richieste dell'utente oppure quando lo studio del problema è stato fatto troppo velocemente ed in modo superficiale.

La durata di questa tappa, purché portata avanti in modo corretto, può fornire un valido criterio per giudicare la qualità del lavoro di preparazione.

La documentazione

Una volta realizzato e messo a punto un programma, dopo averlo utilizzato per un certo periodo di tempo, può sorgere l'esigenza di modificarlo per migliorarlo oppure per adattarlo a dei casi particolari.

È proprio in casi come questo che si è contenti di ritrovare una traccia del lavoro effettuato nelle varie tappe di preparazione e cioé nelle fasi di definizione degli obiettivi, di analisi, di codifica e di messa a punto. Per questo è bene creare una raccolta di documenti, detta appunto documentazione, in cui siano riuniti tutti gli elementi utilizzati per la realizzazione pratica del programma.

CAPITOLO 2

LA DEFINIZIONE DEGLI OBIETTIVI

L'esperienza dimostra che la definizione degli obiettivi è un'operazione molto complessa: spesso capita infatti che l'utente, a programma ultimato, abbia l'impressione di non aver ottenuto ciò che desiderava.
Quando il richiedente e il programmatore sono due persone distinte (e la cosa diviene anche peggiore quando anzicché di due persone si tratta di due équipes) è evidente che possono nascere problemi di comunicazione.

Mi è capitato di assistere personalmente ad una "battaglia verbale" tra due équipes che utilizzavano nella stessa discussione i medesimi vocaboli attribuendo però loro significati diversi e pretendendo di ottenere con lo stesso procedimento i medesimi risultati.

È evidente che, poiché i due interlocutori attribuivano significati diversi ad uno stesso vocabolo, per esprimere lo stesso concetto ciascuno utilizzava un'espressione diversa che, ovviamente, veniva interpretata in modo ancora diverso dall'interlocutore.

Ho pensato di risolvere la questione facendo la parte dello sprovveduto e facendomi nuovamente spiegare da entrambi gli interlocutori i termini usati ed il procedimento che intendevano seguire nella risoluzione del problema, ottenendo così la conferma di ciò che avevo già intuito: le due équipes stavano parlando della medesima cosa.

Questo problema può sorgere, entro certi limiti, anche quando chi sviluppa il programma è la stessa persona che poi dovrà utilizzarlo, in particolare se costui cerca di realizzare un problema informatico su un argomento che conosce bene, avendo però una conoscenza imperfetta dello strumento che ha a disposizione.

Non vi è pertanto che una soluzione: usare solo termini definiti in modo rigoroso. In questo risultano particolarmente facilitati gli scienziati ed i giuristi, in quanto la loro formazione li ha portati a sviluppare spesso questa forma di lavoro mentale.

Inoltre bisogna pensare che, inizialmente, gli elaboratori vennero progettati da scienziati per risolvere problemi di tipo scientifico e che anche nei calcolatori moderni è rimasta traccia di questo orientamento.

Da tutto ciò è facile dedurre perché è indispensabile definire ogni cosa con un grado molto elevato di precisione.
Chi non è mentalizzato a questo genere di lavoro spesso è irritato dalle continue richieste di dettagli provenienti da coloro che devono preparare il programma. Ma quando i programmatori, credendo di aver capito le richieste, non vanno a fondo nelle questioni i risultati diventano addirittura catastrofici.

Cosa dunque bisogna definire e come si può essere aiutati per ottenere delle definizioni corrette?

Nel corso di questa trattazione sono già stati elencati rapidamente gli elementi che si devono definire: i dati iniziali, i risultati, le operazioni che si devono fare per passare dagli uni agli altri. Naturalmente le singole operazioni possono condurre a dei risultati intermedi; in tal caso anche questi dovranno essere definiti con la medesima cura dei risultati finali.
Esaminiamo ora questi elementi uno per uno.

I dati

La prima cosa da precisare è il loro elenco completo. Tale elenco deve comprendere non solo i dati variabili ma anche le eventuali costanti che vengono utilizzate. Queste possono essere, ad esempio, il numero "pi greco" (se si lavora su cerchi e circonferenze) o la lunghezza del pollice espressa in millimetri (se si vuole effettuare la trasformazione di misure inglesi nel sistema MKS).

Per ogni dato bisogna poi precisare, se necessario, l'unità di misura: una lunghezza può essere espressa in cm, m, km.

Inoltre bisogna evitare, per quanto possibile, una trasformazione dei dati da parte dell'utente prima di introdurli nella macchina. Sarebbe auspicabile che l'utente potesse comunicare alla macchina i dati nella forma in cui li utilizza abitualmente in quanto si eviterebbero così gli eventuali errori dovuti alla manipolazione dei dati.

Bisogna inoltre tener conto anche della natura dei dati, dei possibili vincoli su di essi e delle eventuali mutue esclusioni: un dado può assumere solo valori interi compresi tra 1 e 6; in un mazzo di carte da gioco le carte sono caratterizzate dal seme (cuori, quadri, fiori, picche) e da un valore che varia dal 2 all'asso. Vi è indipendenza tra il valore di una carta ed il seme, ma esiste nel mazzo un'unica carta corrispondente ad una particolare coppia valore-seme. Inoltre, scelta comunque una delle 52 possibili coppie valore-seme, esiste una ed una sola carta ad essa associata.

È evidente che tutti questi elementi dovranno essere tenuti in considerazione anche al momento dell'analisi, quando sarà necessario effettuare un gran numero di tests.

Tutto ciò porta a parlare dei controlli che si possono fare sui dati di partenza. Tali controlli, come di seguito esplicitato, possono essere di diverso tipo.

I controlli di attendibilità. Nelle carte dei tarocchi vi è un cavaliere mentre questa figura non è presente nelle carte del bridge. L'altezza di un uomo non può essere m 5,12. Il numero di figli di una donna deve essere un numero intero ed è senz'altro inferiore a 53. Il nome di una persona non contiene cifre mentre ciò è possibile per la ragione sociale di una ditta. Tutto ciò va precisato con cura all'inizio in quanto in genere non verranno in seguito effettuati altri controlli.

I controlli di coerenza: A volte si è portati a considerare i dati per un certo problema sia singolarmente sia con un unico dato globale che ne rappresenti la sintesi: ad esempio, delle somme parziali e la somma totale o, nel gioco del bridge, il gioco che ha in mano ciascuno dei quattro giocatori e la situazione globale della smazzata. In questo caso occorre effettuare dei controlli sistematici di coerenza. Vi sono però anche dei controlli di coerenza creati per delle necessità informatiche e che consistono in tests da effettuare sui dati tramite delle chiavi di lettura (numeri di conto corrente bancario, codice fiscale, ...) Queste chiavi vengono in genere calcolate partendo dal lato da controllare e permettono di verificare se la coppia chiave-dato che si ottiene è corretta (questo si ricollega ai metodi per la costruzione dei codici di verifica degli errori che tuttavia non verranno trattati in quest'opera).

Nonostante queste precauzioni, non si potrà però ugualmente avere, nella maggior parte dei casi, la certezza assoluta della correttezza di un dato.

Se, ad esempio, sul vostro libretto di assegni avete registrato come valore di un assegno L. 354300 mentre l'assegno stesso era stato emesso per un importo di L. 534300, vi accorgerete dell'errore solo quando riceverete dalla banca l'estratto conto: tra il vostro totale ed il loro vi sarà una differenza di cui bisognerà trovare l'origine.

Questo è uno dei problemi più seri dell'informatica gestionale, in quanto degli operatori devono nel corso della giornata "registrare", cioé ricopiare, dei dati numerici. La risoluzione di questo problema ha impegnato gli esperti fin dagli albori dell'informatica.

Un rimedio consiste nel far registrare due volte le medesime informazioni o da due persone diverse, l'una che esegue la registrazione e l'altra che fornisce lo stesso dato ad una macchina che ha il compito di controllare le due scritture (principio della perfo-verifica), o da una sola persona in due tempi diversi (per esempio utilizzando una macchina di tipo conversazionale che faccia questo tipo di richiesta).

Il fatto di scrivere l'importo sull'assegno sia in cifre che in lettere parte dallo stesso principio anche se poi il controllo viene effettuato raramente. Infatti se gli importi in cifre ed in lettere su un vostro assegno sono diversi è assai probabile che nessuno se ne accorga e che vi venga addebitato l'importo scritto in cifre ...
Non basta avere la possibilità di effettuare i controlli ma occorre che i controlli siano fatti realmente.

L'uso di elaboratori di tipo conversazionale permette di ridurre le probabilità di errore. Si trasmettono all'elaboratore solo pochi dati iniziali ed è lui che poi, in base a queste informazioni, sceglie la via da percorrere e richiede i nuovi dati. Ad esempio i dati utilizzati da un assicuratore variano con il bene da assicurare. Definito il dato iniziale, cioé il bene, (una casa, un'automobile, ...) e trasmessolo al calcolatore sarà quest'ultimo a visualizzare la richiesta per il dato successivo (numero di locali, numero di cavalli fiscali, ...) e così via.

Questo metodo è facilmente utilizzabile sui Personal Computers o più in generale su tutti gli elaboratori che utilizzano un linguaggio "conversazionale", e sostituisce vantaggiosamente l'elenco dei dati che prima doveva essere redatto completamente all'inizio.
Per poter utilizzare correttamente questo metodo occorre distinguere, in fase di definizione, i dati che vengono comunque utilizzati da quelli che vengono richiesti solo in casi particolari.

Infine è indispensabile prevedere, nel caso di registrazioni pilotate dal sistema, la possibilità di modificare alcuni dei dati già registrati o addirittura di annullare tutte le registrazioni effettuate fino a quel momento e di ripartire da zero (Questo comunque è un aspetto puramente informatico del problema).

Ho insistito particolarmente sul controllo dei dati in ambito gestionale in quanto questi sono meno frequenti in ambito scientifico anche se la probabilità di errore non è molto diversa. In quest'ultimo caso si effettuano solo i controlli di attendibilità ed alcuni controlli di coerenza.

Per concludere, esiste anche un altro modo per trasmettere dati all'elaboratore: partire da alcuni dati già utilizzati per un altro problema e modificarne eventualmente solo una parte. Questo metodo verrà esaminato più dettagliatamente in seguito nel quadro generale dell'uso di files, generati anche da altri programmi.

I risultati

Molto evidentemente bisogna definire anche per i risultati la loro natura ed il loro elenco. Quest'ultimo deve comprendere oltre ai risultati ordinari anche quelli che si ottengono in casi particolari.

Se i risultati vengono poi visualizzati su video, poiché in genere non è possibile farli comparire su questo tutti insieme per le dimensioni dello schermo, occorre prevedere la possibilità di visualizzarne solo una parte e di far comparire in seguito i rimanenti solo su richiesta dell'utente.

Questo tipo di problema sorge anche per la stampa di dati in uscita quando ad esempio si vogliono utilizzare dei moduli prestampati, cosa tipica in ambito gestionale. Si tratta pertanto di predisporre la trasmissione dei risultati in modo che sia poi facilmente utilizzabile.
Per far ciò è efficace disegnarsi la forma del video o del modulo prestampato, definendo di volta in volta la posizione in cui i vari dati dovranno in seguito essere visualizzati o stampati, e scrivervi infine i vari risultati man mano che il progetto prosegue.

L'elaborazione

In questa fase bisogna descrivere con la massima esattezza tutte le operazioni che si intendono svolgere.

In ambito scientifico la descrizione sarà spesso facile in quanto la successione delle operazioni è in genere esprimibile tramite delle equazioni. Ciononostante non bisogna dimenticarsi di descrivere alcuni particolari come le unità di misura utilizzate ed il metodo utilizzato per renderle tra loro omogenee mediante opportuni coefficienti.

Vi sono inoltre certi tipi di problemi che non possono essere risolti con l'uso di un calcolatore come se si risolvessero manualmente: per esempio su un elaboratore non è pratico utilizzare tavole numeriche (tralascerei comunque la trattazione di quei problemi scientifici che per la loro complessità non possono essre risolti con l'uso dei Personal Computers).

Conviene pertanto ricercare il metodo migliore per risolvere il proprio problema nella letteratura già esistente, in quanto la maggior parte dei problemi risolvibili con dei Personal Computers è già stata trattata, a volte anche in diversi modi differenti.

In ambito gestionale le operazioni da svolgere sono in genere molto semplici. Conviene comunque essere anche in questo caso meticolosi e non sottovalutare il problema: quelli di arrotondamento, ad esempio, vanno trattati con discernimento (la somma di più numeri arrotondati non è uguale all'arrotondamento della somma).

Nella realizzazione di giochi è necessario specificare oltre alle regole fondamentali anche la linea di condotta dell'elaboratore ed abbiamo visto che ciò è tutt'altro che semplice.

Nel trattamento di testi è indispensabile precisare con cura tutto ciò di cui si ha bisogno: larghezza dei margini, capoverso, numero di caratteri per riga, numero di righe per pagine, caratteri speciali o di dimensione diversa, tipo di caratteri utilizzabili ...

Infine nella applicazioni *in tempo reale* occorre definire, oltre alle operazioni, il tempo massimo in cui il programma deve essere eseguito: problema, questo, molto complesso.

In tutti i casi bisogna comunque descrivere tutte le operazioni come se dovesse eseguirle una persona poco intelligente, anche se fedele ed instancabile.

Esempi completamente svolti

Vedremo in seguito che per verificare la correttezza di un programma occorre preparare manualmente un certo numero di esempi svolti interamente, per poterne poi paragonare i risultati con quelli ottenuti dal calcolatore.

La preparazione di questi esempi può essere un buon esercizio per controllare se ci si è dimenticati di definire qualche dato o di prendere in esame qualche caso particolare.

Appurato questo, può quindi succedere in fase di analisi di dover precisare qualche dettaglio. Attenzione che questo non sia però un elemento essenziale, altrimenti si rischia di rimettere in discussione tutto quanto è stato già fatto fino a quel punto. Ecco il motivo per cui è necessario descrivere molto bene tutte le fasi del lavoro.

L'uso di files comuni a più programmi

Tra i dati in entrata oppure tra i risultati possono esservi dei files, generati da altri programmi o che devono venire usati in seguito da questi, od ancora files che servono per la conservazione temporanea di risultati intermedi.

In un programma per la gestione della contabilità di cassa è utile creare un file che contenga il valore di tutti gli assegni emessi ma non addebitati ancora in conto corrente. Questo permetterà al momento opportuno di aggiornare la contabilità senza dover registrare nuovamente tali operazioni.

In questo caso, per quanto riguarda la situazione di cassa, il totale indicato nell'estratto della banca non è attendibile in quanto a questo andrebbero tolti tutti gli importi registrati sul file temporaneo attinenti ai movimenti effettuati in quel periodo e non ancora registrati sull'estratto conto.

Nel caso di *uso ripetuto di files* durante un programma, è conveniente realizzarli utilizzando delle memorie a disco, perché l'uso delle cassette magnetiche non consente il movimento automatico del nastro in entrambi i sensi.

Quando un file deve essere utilizzato da diversi programmi è indispensabile analizzare attentamente tutte le necessità dei singoli programmi, in quanto il file dovrà contenere non solo i dati comuni a tutti questi ma anche quelli usati eventualmente anche da uno solo dei programmi.

Questo tipo di files, generati in un certo momento da un programma ed utilizzati in seguito anche in altri programmi, ha trovato applicazioni anche in campi diversi da quello gestionale: per esempio si può utilizzare un file per memorizzare particolari situazioni che possono verificarsi in un gioco (scacchi e bridge).

L'uso dei files permette anche, in caso di inconvenienti tecnici, di sospendere l'elaborazione memorizzando i risultati parziali ottenuti, per riprenderla poi in seguito: ad esempio, in ambito scientifico, se durante un calcolo non si ottiene, neppure dopo qualche ora di elaborazione, una convergenza accettabile, si può interrompere l'esecuzione, memorizzare nel file i risultati parziali ottenuti e cercare un nuovo algoritmo più conveniente con cui riprendere l'elaborazione partendo però dai risultati parziali già memorizzati.

Si può inoltre effettuare sistematicamente la memorizzazione di tutti i dati presenti nella memoria centrale su delle memorie ausiliarie (nastri o dischi), ad esempio ogni ora, se si teme che possa verificarsi qualche fenomeno esterno che possa turbare il buon funzionamento della macchina (caduta di tensione od altro). Nel caso in cui questo inconveniente si verificasse, si potrà riprendere il lavoro interrotto dall'ultima registrazione effettuata sulla memoria ausiliaria e perdere al più un'ora di lavoro (il tempo perso coincide con l'intervallo di tempo intercorrente tra l'ultima registrazione e l'interruzione).

Limitazioni dovute alla particolare configurazione del sistema utilizzato

Più volte queste limitazioni non vengono esplicitate chiaramente all'inizio del lavoro ma vengono individuate durante la fase di progettazione, a causa delle conseguenze che comportano.

Nel caso dei Personal Computers, in cui le periferiche sono poche (cassetta o disco), le limitazioni possono essere pesanti. Se è disponibile solo un registratore, ad esempio, per la memoria a cassette magnetiche, non si può usare che una sola cassetta per volta, immagazzinare tutto quanto necessita nella memoria centrale, eseguire l'elaborazione, che può comportare nei dati cancellature o aggiunte, ed infine trasferire il nuovo insieme di dati ancora sulla cassetta.

Se invece si hanno a disposizione due registratori è possibile leggere dalla prima cassetta un solo blocco di dati, memorizzandolo nella memoria centrale, elaborarlo, effettuando eventuali aggiunte e/o cancellazioni o creando addirittura nuovi blocchi di dati e scrivere i risultati ottenuti sulla seconda cassetta prima di leggere il successivo blocco di dati dalla prima.

In questo caso è evidente che la lunghezza dei files non dipende dalle capacità della memoria centrale ma solo dalla lunghezza dei nastri.

Lo stesso risultato si può ottenere più comodamente utilizzando un solo disco al posto di due registratori a cassetta.

Altre limitazioni sono talmente evidenti che non meritano alcun commento: senza una stampante non si può avere una traccia scritta dei risultati ottenuti se non ricopiando manualmente ciò che compare sul video, con un'alta possibilità quindi di errore. Per evitare questi errori è possibile fotografare di volta in volta ciò che appare sul video con un apparecchio a sviluppo istantaneo.

Infine osserviamo che se si vuole interfacciare l'elaboratore con una strumentazione scientifica esterna, che controlla o registra fenomeni del mondo fisico, si devono tener presenti le eventuali complicazioni dovute alle caratteristiche degli strumenti utilizzati. Questa scelta deve pertanto essere effettuata in modo corretto.

In genere sarà bene effettuare la scelta degli strumenti solo dopo aver analizzato attentamente il problema ed aver precisato, in fase di analisi, quali caratteristiche tali strumenti devono possedere; scegliendo invece la strumentazione a priori si possono creare delle complicazioni in sede informatica tali da vanificare tutto il resto del lavoro.

Il risultato di questo lavoro di definizione degli obiettivi dà luogo a quelle che in genere vengono chiamate "specifiche esterne", dove per esterno si intende tutto ciò che può essere percepito dall'utente in contrapposizione ad interno in quanto con quest'ultimo termine si vuole indicare tutto ciò che avviene all'interno del sistema di elaborazione (cioé all'interno del calcolatore e degli eventuali strumenti ad esso collegati).

Nella gestione di un progetto informatico di dimensioni considerevoli la definizione delle specifiche esterne viene affidata ad una équipe insieme al compito di approntare un certo numero di prove, che verifichino, a progetto ultimato, se questo è conforme alle richieste iniziali.

È evidente pertanto che la definizione delle specifiche esterne deve precedere ogni altra fase di progettazione.

Questa regola di carattere generale è completamente trasferibile sui Personal Computers. È infatti indispensabile che sia chiaro ciò che si vuole ottenere prima di affrontare l'analisi e la codifica. Nei prossimi capitoli verrà mostrato con degli esempi il cammino che si deve percorrere.

La documentazione

Ogni qualvolta si scrive un programma bisogna tener conto che la documentazione ad esso relativa deve essere proporzionale alle dimensioni del programma stesso. Converrà comunque sempre, anche per semplici programmi, produrre una buona documentazione.

La documentazione di sviluppa parallelamente al lavoro: comincia con la definizione degli obiettivi, momento in cui si raccolgono tutte le notizie relative a ciò che si vuole ottenere con il programma e tutte le precisazioni necessarie riguardo alle specifiche esterne.

In questa prima fase conviene articolare la documentazione in modo tale che sia possibile verificare facilmente se si è tralasciato qualche punto, ed in particolare:

- che siano stati descritti in modo esauriente gli algoritmi e le procedure che si intendono usare per ottenere i risultati desiderati;

- che siano stati definiti correttamente, in un modo o nell'altro, i dati, le variabili e le costanti, utilizzati negli algoritmi, e che per le prime sia stato definito anche il campo di variabilità;

- che siano state approntate delle tabelle descriventi i risultati da ottenere che ricoprano tutto l'insieme dei risultati possibili.

La documentazione verrà poi ampliata nelle tappe successive fino alla conclusione del lavoro.

CAPITOLO 3

L'ANALISI

Dopo aver definito accuratamente gli obiettivi si passa, con l'analisi, ad uno stadio in cui la successiva codifica diverrà un fatto semplice e naturale.

Occcore precisare subito che l'analisi non dipende dal particolare linguaggio adottato per la codifica anche se utilizzando un linguaggio di tipo evoluto (ad esempio il Basic) è evidente che non occorrerà dettagliare ogni singola operazione, mentre ciò si renderà necessario programmando in un linguaggio di tipo assemblativo. Ribadiamo comunque che a livello di metodologia questa differenza risulta piuttosto marginale.

Prima di descrivere come procedere nell'analisi ricordiamo che il nostro scopo ultimo non è la codifica ma il raggiungimento di quei risultati che sono stati descritti precedentemente come obiettivi, anche se questi alla fine si ottengono tramite un programma eseguito automaticamente.

Spingendo il ragionamento un pò oltre si può dire che la definizione degli obiettivi rappresenta una prima formulazione simbolica della procedura da eseguire (alcuni non disperano di poter vedere, un giorno, una macchina eseguire un programma formulato solo in questo modo).

Così, se l'obiettivo è la rappresentazione del valore di un dado, non si considera il fatto che questo abbia sei facce numerate ma che il valore del dado coincide con quello indicato sulla sua faccia superiore e che può assumere con la stessa probabilità uno qualsiasi dei valori interi compresi tra 1 e 6. Quindi, se si memorizza in una variabile chiamata DADO il valore della faccia superiore, si dice che tale variabile rappresenta il dado.

Un programma che modifica casualmente il valore di questa variabile sarà quindi la rappresentazione astratta del lancio di un dado. Facendo questa astrazione sono stati trascurati molti fenomeni secondari (il fatto che il dado venga fatto rotolare su una pista, che ne urti le sponde, che il lancio sia irregolare, ...) e si è posto l'accento soltanto su un fatto essenziale: i valori che può assumere il dado devono essere equiprobabili, interi e compresi tra 1 e 6.

L'analisi deve, man mano che viene sviluppata, precisare sempre più le operazioni che si devono svolgere.

Riprendendo l'esempio del dado, all'inizio si può avere semplicemente la seguente formulazione:

- il lancio di un dado

Approfondendo però di più l'analisi si può giungere alla seguente formulazione del problema:

- Lancio di un dado
  - ● Estrazione casuale di un valore intero compreso tra 1 a 6
  - ● Il dado assume questo valore

Questa rappresentazione può essere facilmente codificata in Basic

- Lancio di un dado
  - ● Estrazione casuale di un valore intero compreso tra 1 e 6 → RND(6)
  - ● Il dado assume questo valore → DADO=RND(6)

(Si presuppone che un programmatore con un minimo di esperienza riesca a scrivere direttamente questo programma in Basic saltando le tappe intermedie)

Questo semplice esempio illustra il meccanismo con cui viene sviluppata l'analisi per giungere alla fine alla codifica in Basic, che non è altro che una nuova rappresentazione del problema.

I diagrammi a blocchi

Prima di proseguire è necessario introdurre uno strumento molto utile al programmatore in fase di analisi: il diagramma a blocchi o diagramma di flusso.

Un diagramma a blocchi è un insieme di simboli grafici, racchiudenti delle funzioni, di forma diversa (rettangoli, rombi, ...) collegati tra loro con delle linee orientate (frecce).

Il diagramma a blocchi è utilizzabile per rappresentare graficamente le funzioni logiche del programma.

In ogni diagramma a blocchi il simbolo iniziale è un ellisse contenente la parola INIZIO e quello finale un ellisse contenente la parola FINE, come illustrato nella figura 1.

Nella stessa figura sono rappresentati due cerchi contenenti un carattere (nel nostro caso A). Questi simboli permettono di realizzare il collegamento logico tra due blocchi che non possono essere collegati direttamente (ad esempio l'ultimo blocco di una pagina con il primo della pagina seguente)

Figura 1

Il simbolo grafico che in un digramma a blocchi racchiude una funzione di elaborazione è il rettangolo. Nell'esempio del lancio di un dado questo simbolo avrebbe potuto contenere una delle seguenti funzioni: lancio di un dado; estrazione di un numero intero comrpeso tra 1 e 6; il dado assume questo valore. L'uso di questo simbolo viene illustrato nella figura 2.

Figura 2

La figura 3 mostra, mediante l'uso dei diagrammi a blocchi, come l'operazione "lancio del dado" viene scomposta in due operazioni più semplici. Ciò dimostra come sia facile e naturale passare da enunciati generali a programmi

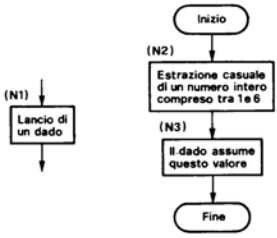


Figura 3

particolareggiati. I caratteri tra parentesi in alto a sinistra dei vari blocchi (N, N1, N2, N3) permettono di poter riprendere poi questi blocchi in momenti successivi dell'analisi per scomporli ulteriormente. Nel nostro caso N1 è stato scomposto in N2 ed N3.

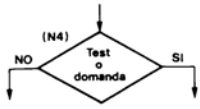


Figura 4

Un altro simbolo usato frequentemente è quello di decisione. Esso ha due uscite, caratterizzate dalle risposte SI e NO, che indicano la strada da seguire a seconda della risposta data alla domanda contenuta nel simbolo (rombo) come illustrato nella figura 4.

Figura 5

La figura 5 illustra invece il simbolo utilizzato per rappresentare operazioni di input (lettura) o di output (scrittura o visualizzazione), che per la sua forma particolare è facilmente individuabile all'interno di un diagramma a blocchi.

Figura 6

L'ultimo simbolo che bisogna esaminare è rappresentato nella figura 6. Esso ha una sola entrata e più uscite. La via da percorrere viene scelta in funzione del valore assunto dalla variabile contenuta nel blocco (nel nostro caso i). Osserviamo che il numero di uscite di questo blocco non è fisso ma viene definito ogni volta dal programmatore.

La figura 7 illustra il diagramma a blocchi relativo allo sviluppo di un esempio molto semplice: il calcolo del valore della radice nell'equazione $Ax + B = 0$ ($x = - B/A$). Nel blocco 1 viene considerato l'inserimento dei coefficienti dell'equazione e la loro visualizzazione sul video. Nel blocco 2 viene fatto un controllo sul valore di A, onde evitare la divisione per zero, cosa che l'elaboratore non gradisce. Quindi se A è uguale a zero si prende la via a destra, che prevede solo la stampa di un messaggio per l'utente (blocco 5) mentre se A è diverso da zero si prende la via di sinistra e si calcola il valore $X = - B/A$ della radice (blocco 3) e lo si visualizza sul video in modo da renderlo utilizzabile dall'utente (blocco 4). Le due vie poi si ricongiungono confluendo nel blocco 6 che permette di eseguire un controllo sulle intenzioni dell'utente (se vuole risolvere altre equazioni) ed in base a ciò si decide la via da seguire.

La figura 7 mostra in modo chiaro come vengono effettuati i collegamenti logici tra i vari blocchi mediante delle linee orientate e come vengono gestite le intersezioni tra le varie linee, dando un'idea completa dello sviluppo di un diagramma a blocchi.

Man mano che vengono sviluppati, i diagrammi a blocchi si devono includere nella documentazione di cui formeranno la parte essenziale in quanto permetteranno in ogni momento di avere una veduta di insieme del programma (dei programmi che compongono un certo lavoro).

Ritorniamo ora all'analisi, fase in cui, specificando progressivamente i dettagli, si passa dalla definizione degli obiettivi ad una sequenza logica di operazioni elementari facilmente traducibile in un linguaggio simbolico.

Lo sviluppo dell'analisi si realizza prendendo un blocco del diagramma iniziale e scomponendolo gradatamente in più blocchi (in partenza il diagramma a blocchi può essere anche formato da un solo elemento).

Questa scomposizione può essere fatta a piacere o occorre seguire delle regole ben precise?

Gli esperti in informatica hanno impiegato molto tempo (circa un decennio) per trovare una risposta adeguata a questa domanda, in quanto con il complicarsi dei problemi da risolvere venivano posti in evidenza i difetti dei vari metodi proposti: vi era chi, ad esempio, si scatenava ed eseguiva moltissime

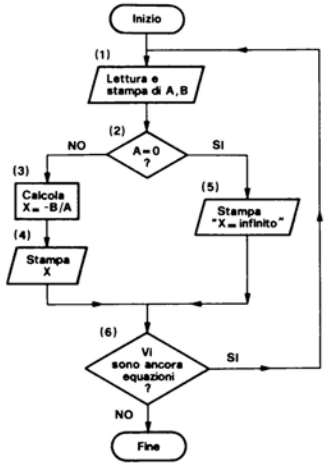


Figura 7

prove sulla macchina cercando ogni volta di mettere a punto il proprio programma e chi per contro sembrava lavorare con troppa calma, effettuando pochissime prove dilazionate nel tempo. Quest'ultimo però riusciva a completare il lavoro in un tempo inferiore. Inoltre il suo programma era documentato con maggiore chiarezza, era più facile da leggere e da capire e permetteva di apportare facilmente delle eventuali modifiche.
Attraverso uno studio sia teorico che pratico gli universitari delle due sponde dell'Atlantico hanno finito per approntare un metodo di programmazione che

permette di sviluppare in modo organico un qualsiasi programma.
Questo metodo non vuole certo essere rivoluzionario in quanto era già usato, anche se non con sistematicità, da parecchi programmatori di vecchia data.

Comunque la conclusione a cui si giunge è che il metodo succitato permette di risolvere nello stesso tempo tre grossi problemi:

1. *La codifica di programmi di grosse dimensioni* che necessitano del lavoro di svariate équipes per la loro realizzazione

2. *La difficoltà di verifica della correttezza del programma*, cioé di verificare che esso fornisca correttamente i risultati richiesti

3. *La leggibilità del programma*, intendendo con ciò il fatto che anche chi non ha partecipato ai lavori preparatori deve essere in grado, leggendo il programma e la documentazione, di capire facilmente le operazioni che il programma esegue.

Si potrebbe aggiungere che questo metodo aiuta anche a formulare una prima prova del programma, cioé la dimostrazione che il programma stesso esegue, in ogni caso, solo ciò che si è previsto e niente altro.
Il problema di verificare completamente la correttezza di un programma non è stato ancora risolto. In pratica ci si limita a "supporre" che ciò sia vero, "sperando" che nella sua "vita", cioé in tutti i casi in cui verrà utilizzato, e non in assoluto, il programma funzioni come si deve.
È evidente che qui giocano un ruolo fondamentale l'esperienza, l'intuito e una salda formazione professionale.

Le tecniche di scomposizione

Quando si vuole scomporre un elemento del diagramma a blocchi per illustrare in modo più dettagliato l'operazione che esso descrive, è evidente che ciò comporta anche una modifica della struttura del diagramma a blocchi stesso.

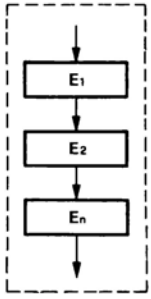


Figura 8

Il caso più semplice di scomposizione è quello in cui si scompone un blocco in una sequenza di blocchi rappresentanti delle operazioni logicamente consecutive. Nell'esempio del lancio del dado, già esposto in precedenza, il blocco descrivente il lancio del dado veniva scomposto in due blocchi successivi.

La figura 8 evidenzia questa scomposizione. Gli elementi E_1 ..., E_n rappresentano la scomposizione dell'elemento iniziale, simboleggiato dal rettangolo tratteggiato.

Un'altra scomposizione è quella che dà luogo a delle vie alternative: si esegue un'operazione o un'altra a seconda della risposta data ad una certa domanda.

Nell'esempio riportato in figura 9 si esegue il blocco E se la condizione espressa nel blocco precedente è verificata, altrimenti il programma prosegue senza che venga eseguito il blocco E.

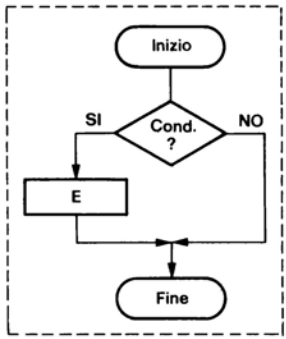


Figura 9

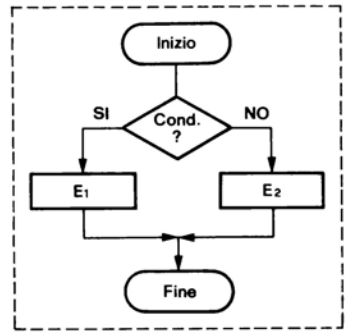


Figura 10

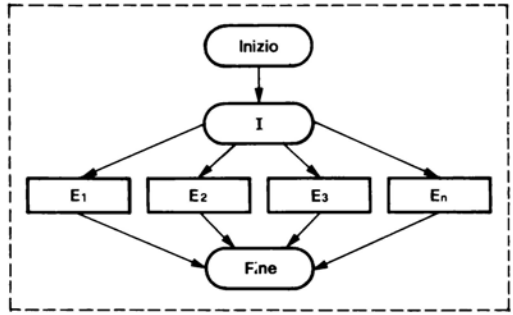


Figura 11

Nell'esempio riportato nella figura 10 si ha invece un'alternativa vera e propria: o si esegue E_1 o si esegue E_2. Un esempio di questo tipo di alternativa è già stato illustrato nel caso della risoluzione dell'equazione $Ax + B = 0$ con il test sul valore del divisore A (cfr figura 7).

In figura 11 è generalizzata l'alternativa nel caso in cui si abbiano più di due possibilità di scelta. Nell'esempio si hanno n possibilità di scelta che si escludono a vicenda (si può sciegliere una e una sola via).
Il valore assunto dalla variabile i determina la via che deve essere intrapresa all'uscita del simbolo "case" e quindi quale tra i blocchi E_i del diagramma dovrà essere preso in considerazione.
Bisogna poi considerare le strutture ripetitive che vengono utilizzate anche quando non si conosce a priori il numero di iterazioni che dovranno essere eseguite.

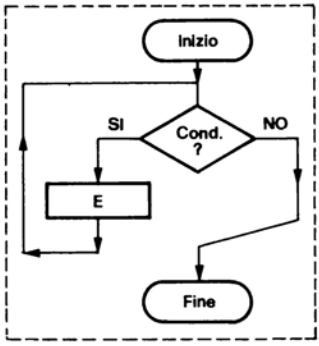


Figura 12

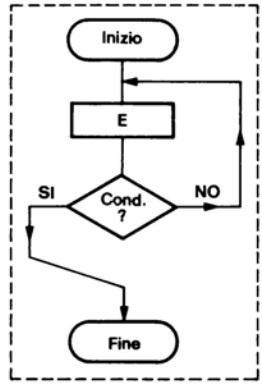


Figura 13

Le figure 12 e 13 illustrano le due strutture iterative:

- *mentre* Cond *fai* E: fino a quando è verificata la condizione Cond esegui il blocco E

- *ripeti* E *finché* Cond: ripeti E finché non si verifica Cond

Osserviamo che nel secondo caso il blocco E viene comunque eseguito almeno una volta. Le istruzioni FOR per il Basic e DO per il Fortran sono casi particolari di queste strutture:
Abbiamo visto che la scomposizione di un blocco può dare origine a tre diversi tipi di strutture che chiameremo:
- sequenza, nel caso in cui il blocco iniziale sia scomposto in una serie di elementi concatenati in modo univoco,
- selezione, con la sua naturale generalizzazione nel "case",
- iterazione.

Si vede chiaramente che la scomposizione di un rettangono che ha una sola entrata ed una sola uscita e che descrive un'azione dà luogo ad un insieme di blocchi in cui i rettangoli non hanno che un'entrata e un'uscita.

Con questi modelli è possibile rappresentare qualsiasi programma ed anche descrivere, attraverso dei ragionamenti classici, le varie regole di scomposizione, nonché verificare, almeno parzialmente, l'utilità di questo metodo.

Raccomandiamo perciò al lettore l'uso di questo metodo, che viene chiamato "programmazione strutturata". È stata fatta una buona trattazione teorica di questa metodologia e ne è stata ampliamente dimostrata la validità. Il diagramma a blocchi realizzato in questo modo può essere direttamente codificato in un linguaggio creato appositamente, ed insegnato con sempre maggiore frequenza nelle Università, che è ora disponibile su moltissimi sistemi: il Pascal.

I dati

Oltre che delle operazioni che devono essere effettuate, specificate progressivamente nei dettagli mediante le tecniche di scomposizione, occorre tener conto anche degli elementi che saranno l'oggetto dell'elaborazione.

Uno dei fini dell'analisi è quello di trasformare la descrizione di un fenomeno reale in un insieme di simboli che possono essere elaborati da un calcolatore, cioé valori numerici, stringhe di caratteri, variabili logiche.

Come per le operazioni da eseguire si passa, tramite la scomposizione e per tappe successive, da una prima rappresentazione schematica del problema ad una rappresentazione talmente dettagliata da permettere la codifica in un linguaggio simbolico, così, anche per i dati si giunge ad una loro corretta rappresentazione mediante tappe intermedie.
L'analisi ci porterà spesso a creare dei dati intermedi, chiamati variabili di lavoro, che permettono di semplificare lo sviluppo del diagramma a blocchi.

Anche se in casi particolari bisogna tener conto della reale rappresentazione dei dati all'interno della macchina, in generale, per facilitare la scrittura del diagramma a blocchi, basterà riflettere sulla loro struttura astratta e rimandare ad una fase successiva la scelta della loro reale rappresentazione.

Anche nel definire le condizioni a cui devono sottostare i dati, come si era fatto nella descrizione delle procedure, occorre lavorare con una meticolosità molto elevata. Bisogna inoltre osservare che queste condizioni devono essere espresse inzialmente in un linguaggio tecnico attinente al problema in esame, e solo in seguito tradotte in un linguaggio di tipo informatico.

Ad esempio, in un gioco di carte, non bisogna iniziare l'analisi dicendo: "il seme Cuori sarà rappresentato dall'indice 3 ed il Re dall'indice 13 per cui le carte da gioco verranno rappresentate con delle variabili a due indici", ma occorre assegnare prima un nome ad ogni carta, (il Re di cuori).

Spesso un gioco di carte è rappresentabile più convenientemente in altri modi: essendo l'asso la carta con valore maggiore converrà rappresentarla mediante il numero più grande anzicché con il numero 1. Nei giochi come la briscola o il tresette, in cui certe carte giocano un ruolo particolare, occorre analizzare tutto ciò attentamente prima di scegliere la rappresentazione per le relative carte.

Per non creare intralci al lavoro, bisogna evitare di entrare nei dettagli sulle caratteristiche dei dati finché ciò non diventi indipensabile per la prosecuzione del lavoro stesso: infatti solo di fronte a difficoltà reali le decisioni vengono prese con cognizione di causa e nel modo più opportuno.

Del resto è abbastanza frequente giungere alle decisioni sulla rappresentazione dei dati non in fase di analisi ma in fase di codifica, in quanto solo a quel punto una scelta definitiva si impone!

CAPITOLO 4

LA CODIFICA E LA MESSA A PUNTO DEL PROGRAMMA

Terminata l'analisi, si devono avere a disposizione tutti gli elementi necessari per poter facilmente codificare il programma in un linguaggio simbolico.

Poiché, come è stato più volte ribadito, l'analisi procede indipendentemente dal particolare linguaggio adottato per la codifica, bisogna effettuare la scelta di tale linguaggio in fase di codifica, tenendo conto delle caratteristiche del problema trattato.

Per i nostri esempi abbiamo scelto il linguaggio Basic in quanto è senza dubbio il linguaggio più diffuso sui Personal Computers e al tempo stesso si presta più facilmente di altri a rappresentare una vasta gamma di problemi.

Descrizione del Basic utilizzato in questo libro

Per gli esempi proposti nei capitoli seguenti viene utilizzato un Basic molto simile a quello utilizzato sulla maggior parte dei Personal Computers in commercio. Qui di seguito vengono descritte le caratteristiche fondamentali di questo linguaggio e vengono inoltre segnalate quelle particolari istruzioni che eventualmente devono essere trasformate per poter essere utilizzate su particolari calcolatori.

Variabili: il nome di una variabile è formato da 1 o 2 caratteri alfanumerici di cui il primo deve essere alfabetico. Il nome di una variabile seguito dal carattere % sta ad indicare che la variabile stessa può assumere solo valori interi. Se invece esso è seguito dal carattere $, ciò significa che la variabile rappresenta una stringa di caratteri (queste notazioni sono comuni a tutti i Basic). Non possono essere usate come nomi di variabili le seguenti parole chiave del linguaggio Basic: IF, ON, OR, TO.

Operazioni aritmetiche: i simboli sono +, -, *, /.

Operazioni logiche: gli operatori di confronto possono essere usati sia tra variabili o costanti numeriche che tra variabili o costanti alfanumeriche (stringhe di caratteri). Tali operatori sono: <, <=, >, >=. Il risultato di un'opera-

zione logica è uno dei due valori che può assumere una variabile logica: FALSO (0) e VERO (1).
Tra due variabili logiche possono essere applicati anche i seguenti operatori logici: AND, OR, NOT (Anche se nei nostri programmi non vengono utilizzati che nelle frasi di commento, occorre notare che sul PET e sul TRS 80 questi tre operatori possono essere utilizzati anche con variabili intere e con variabili binarie formate da 16 bit).

Istruzioni di Input/Output: Le istruzioni classiche che permettono la gestione di un terminale o di una stampante sono:

```
PRINT A,B,C          con tabulazione automatica
PRINT A;B;C
PRINT TAB(N) A
INPUT A,B,C,
INPUT "stringa di caratteri" A,B,C
```

Se invece si vogliono eseguire operazioni di lettura/scrittura su un nastro magnetico si useranno istruzioni del tipo:

```
INPUT#-1,A,B,C
PRINT#-1,A,B,C
```

Istruzioni di assegnazione queste saranno del tipo:

```
A = B        (non verrà usata la forma LET A = B)
```

Istruzioni di salto:

```
GOTO N
ON J GOTO N1,N2,N3        } salti ad altra istruzione
```

```
GOSUB N
ON J GOSUB N1,N2,N3       } salti ad un sottoprogramma e
RETURN                      ritorno al programma principale
```

Istruzioni di ciclo:

```
FOR I = 1 TO 50 STEP 2
NEXT I                    (o solo NEXT)
```

Istruzioni condizionali:

```
IF A > B THEN C = C + 3
```

In questo esempio si è utilizzata la sola forma IF ... THEN ... Verrà utilizzata

però, quando sarà possibile, anche la forma IF ... THEN ... ELSE ... Questa seconda forma (disponibile ad esempio sul TRS 80) permette di esprimere la struttura illustrata in figura 10. Poiché però non è un'istruzione standard presente in tutti i Basic, il suo uso verrà segnalato ogni volta.

Istruzioni di inizializzazione

```
READ A,B,C,D
DATA 317, -29, 43, 537
RESTORE
```

Permettono agevolmente di assegnare dei valori iniziali a delle variabili.

Istruzione di arresto:

```
STOP
```

Si può riprendere l'esecuzione del programma con un CONTINUE. La procedura per riprendere l'esecuzione può essere diversa su altre versioni del Basic.

Istruzione di fine programma:

```
END
```

Istruzione di dimensionamento per vettori e matrici:

```
DIM A(2,3), A$(4)
```

Il valore minimo dell'indice è zero.

Frasi di commento:

```
REM IL GIOCO DELLE CARTE
```

Dopo il REM la fine della riga è trascurata.

Separatore di istruzione:

Due istruzioni, oltre che su due linee differenti, possono essere scritte sulla stessa riga purché separate dal carattere: (due punti). Questo risulta particolarmente utile nella forma IF ... THEN in quanto permette di scrivere su una stessa riga tutto ciò che si esegue nel caso in cui la condizione sia verificata. In caso contrario si esegue l'istruzione della riga seguente (o il ramo ELSE se esiste).

Le funzioni utilizzate:

Negli esempi trattati verranno utilizzte solo poche funzioni che vengono elencate qui di seguito.

Funzioni algebriche:

Le funzioni algebriche che verranno utilizzate sono le seguenti:

INT (expr)	che dà la parte intera dell'espressione tra parentesi. I valori limite per questa funzione, come del resto accade di frequente, sono +32767 e -32768
SGN (expr)	Questa funzione restituisce uno dei tre valori +1, 0, -1, ed in particolare +1 se il valore dell'espressione tra parentesi è positivo, 0 se tale valore è nullo e -1 se è negativo.

Funzioni operanti su stringhe di caratteri:

LEN (A$)	dà la lunghezza della stringa contenuta nella variabile A$
VAL (B$)	dà il valore numerico relativo ad una stringa di caratteri rappresentanti le cifre di un numero decimale

Funzioni che generano stringhe di caratteri:

C$ = A$ + B$	L'operazione + tra due stringhe fornisce la concatenazione delle stesse
C$ = LEFT$ (A$, N)	Considera solo i primi N caratteri a partire da sinistra
C$ = RIGHT$ (A$, N)	Considera solo i primi N caratteri partendo da destra
C$ = MID$ (A$, I, J)	Considera solo J caratteri a partire dall'I-imo
CHR$ (X)	Restituisce il carattere il cui codice ASCII è contenuto in X
STR$ (X)	Restituisce una stringa di caratteri (cifre) rappresentanti il numero contenuto in X

Funzioni che generano numeri casuali:

In questo campo esistono le più evidenti differenze tra le varie versioni del Basic. Negli esempi da noi proposti verranno utilizzate queste istruzioni:

RANDOM	Questa funzione inizializza il generatore di numeri pseudocasuali
RND (0)	Genera un numero casuale tra 0 e 1
RND (X)	Genera un numero casuale intero compreso nell'intervallo 1, INT(X) con X< = 32767

Cancellazione del video

Per cancellare ciò che è visualizzato sul video e riposizionare il cursore in alto a sinistra si usano comandi particolari a seconda dell'elaboratore che si ha a disposizione. Per ottenere questo nel seguito dell'opera verrà utilizzato il comando CLS.
Come abbiamo già detto però i vari sistemi di elaborazione utilizzano funzioni assai diversificate e pertanto il lettore dovrà preoccuparsi, per quanto riguarda la cancellazione del video, di adattare i programmi da noi proposti alle caratteristiche dell'eleboratore che ha a disposizione.

Esaminiamo ora le caratteristiche ed i limiti del Basic da noi adottato.

Esistono due tipi di variabili utilizzabili: le variabili numeriche e quelle alfanumeriche, contenenti cioé stringhe di caratteri. A loro volta le variabili numeriche si suddividono in intere e reali.
All'interno del calcolatore i numeri interi vengono rappresentati nella loro forma binaria utilizzando parole di 16 bit di cui il primo è rappresentativo del segno. Potranno venire rappresentati tutti i numeri interi compresi tra +32767 e -32768. I numeri reali vengono rappresentati in modo diverso a seconda della precisione, cioé del numero di cifre significative; in generale tale numero è 7 o 9 o 16 (quest'ultimo caso è ammesso ad esempio sul TRS 80 utilizzando la rappresentazione in doppia precisione). Se l'elaboratore permette una scelta tra diversi gradi di precisione occorre effettuare la scelta con molta oculatezza, vagliando attentamente i problemi di arrotondamento, che affronteremo più a fondo nei prossimi paragrafi.
Le variabili alfanumeriche possono contenere un numero di caratteri compreso tra 0 (stringa vuota) e 80 o 256 (la lunghezza massima varia da macchina a macchina).
Ricordiamo infine che esistono particolari funzioni che permettono di trasformare stringhe di caratteri in dati numerici e viceversa.

Tipi di precisione per le variabili numeriche

Quando si utilizzano variabili numeriche di tipo reale (sia in semplice che in doppia precisione) è opportuno sapere con quali metodi vengono rappresentati all'interno della macchina i valori da noi introdotti.

Partendo da sinistra verso destra il valore di ogni cifra del numero inserito in macchina viene sommato a quello ricavato dalla elaborazione delle cifre precedenti, e passando da una cifra alla successiva il risultato ottenuto viene moltiplicato per 10. Si prosegue così fino al punto decimale, alla destra del quale le cifre assumono valori inferiori all'unità.

Occorre osservare che il risultato di questa elaborazione è un numero, espresso in codice macchina, che in genere può occupare solo un numero prefissato di posizioni binarie, oltre cui viene troncato, dando luogo ad una perdita di informazione sulle cifre meno signficative.

Ad esempio, se rappresentando i numeri in virgola mobile, si utilizzano solo tre cifre di mantissa, i numeri 12300, 12312, 12287 verrebbero rappresentati tutti come 0.123E5. (In realtà le cose sono ancora più complesse in quanto l'elaboratore usa una numerazione binaria anziché decimale).

Bisogna pertanto sapere assolutamente quale è il numero di cifre significative necessario per rappresentare adeguatamente i dati di un problema: ad esempio, in un problema i cui dati possono assumere valori oscillanti tra un milione ed un centesimo (caso della contabilità in valuta straniera) occorre assicurarsi che il numero 1000000 ed il numero 999999.99 vengano rappresentati in modo diverso. In questo caso saranno necessarie pertanto 8 cifre significative. (Occorre chiarire che per ottenere 3 cifre signficative occorrono ben 10 cifre binarie poiché $2^{18} = 1024$ è di poco superiore a $10^3 = 1000$).

La precisione ottenuta con questa rappresentazione (cioé il numero di cifre significative dopo il punto decimale) dipende dal valore rappresentato. Ad esempio con un numero totale di 8 cifre significative il valore 1000 sarà rappresentato con una precisione alla quarta cifra decimale. Questo fatto comunque non genera complicazioni in quanto in corso di elaborazione è sempre possibile arrotondare i risultati, ad esempio al centesimo.

In ambito scientifico subentra poi anche un altro tipo di problema: capita spesso di dover calcolare la differenza dei valori assoluti di due numeri (o per una sottrazione o per una addizione tra due numeri di segno opposto). Se, per disgrazia, i numeri sono dello stesso ordine di grandezza e differiscono solo per dei millesimi del loro valore, il risultato sarà circa 1000 volte più piccolo rispetto ai numeri iniziali. Poiché in una somma algebrica l'errore assoluto si conserva, l'errore nel risultato permane comunque dello stesso ordine di grandezza di quello presente nei dati iniziali.

Se l'errore relativo iniziale era di 10^{-8}, e cioé l'errore si ripercuoteva sull'ottava cifra decimale degli addendi, è evidente che, utilizzando sempre 8 cifre significative, sul risultato, che si è presupposto 1000 volte più piccolo, l'errore sarà dell'ordine di $10^3/10^8$. Esso verrà perciò rappresentato solo con 5 cifre significative.

Ciò porta ad una duplice conseguenza:

- durante l'analisi occorrerà ricercare le possibili cause di perdita di precisione e possibilmente eliminarle;

- non bisognerà però neppure sottovalutare il problema relativo alla scelta del numero di cifre significative da adottare nei calcoli: questo perché sugli elaboratori di grosse dimensioni i numeri reali possono essere rappresentati anche utilizzando 60 o 120 bit (che corrispondono a circa 18 o 36 cifre decimali) mentre in generale i dati relativi ad un certo problema non richiedono per la loro rappresentazione più di 4 o 6 cifre.

Per i Personal Computers invece il discorso relativo alla scelta corretta del numero di cifre significative da utilizzare si giustifica con il fatto che nello sviluppo di un programma di tipo scientifico occorre prestare molta attenzione alla precisione: infatti l'inversione di una matrice di rango superiore a 5 può creare seri problemi se non si usano dati con almeno 7 o 8 cifre significative.

I nomi delle variabili

Determinata la natura di un dato, questo verrà identificato mediante il nome della variabile che lo contiene. Nel Basic utilizzato sui Personal Computers si possono dare alle variabili dei nomi di lunghezza a piacere, purché non contengano nessuna parola chiave del linguaggio. L'interprete considererà comunque sempre solo i primi due caratteri del nome stesso. Occorre perciò prendere alcune precauzioni in quanto, ad esempio, le parole VALORE e VACANZA rappresenteranno la stessa variabile VA. (Invece le due scritture VA% e VA$ individueranno variabili diverse rappresentando la prima il nome di una variabile di tipo intero e la seconda il nome di una variabile di tipo alfanumerico). Le parole TOTALE e CONTANTE non potranno essere usate come nomi di variabili perché contengono le parole chiave TO e ON.

Per tutti questi motivi si è preferito usare negli esempi solo nomi di variabili composti da due caratteri, di cui il primo alfabetico' Con questo criterio si ottengono 936 combinazioni possibili, di cui solo 932 utilizzabili, in quanto occorre scartare le 4 parole chiave di due lettere IF, ON, OR, TO.

Questo numero di combinazioni è in genere più che sufficiente e può addirittura venire ridotto per comodità. In particolare si raccomanda, per evitare la classica confusione tra lettera O e lo zero, di non usare per quanto possibile né un simbolo né l'altro per indicare i nomi di variabili (esistono varie convenzioni che per distinguere tra loro questi due caratteri ne barrano uno: Ø. Sfortunatamente però, se si usa la convenzione francese AFNOR si deve barrare la lettera O, mentre se si usa la convenzione americana ANSI si deve barrare il numero zero. Pertanto per i Francesi che lavorano spesso con strumentazione americana questo rimedio risulta peggiore del male!). Osserviamo inoltre che non utilizzando la lettera O, vengono automaticamente eliminate anche tre delle quattro parole chiave di due caratteri e precisamente ON, OR, TO. (Alcuni interpreti considerano poi i nomi formati da una sola lettera, per

esempio F, come nomi di due caratteri aventi per secondo carattere lo zero, nel nostro caso F0. Per evitare dunque ulteriori ambiguità è consigliabile non usare la cifra 0 per definire i nomi di variabili).

Un'altra buona norma è quella di redigere un elenco alfabetico dei nomi di variabili utilizzati durante il lavoro, comprendente anche i nomi delle variabili di trasferimento e degli indici che controllano i cicli di FOR. Nell'ordinare i nomi si adotterà la convenzione che le cifre numeriche precedano nell'ordinamento le lettere alfabetiche. Bisognerà infine provvedere a scrivere, in quest'elenco, di fianco al nome utilizzato una breve descrizione di ciò che la variabile rappresenta.

La compilazione accurata di questo elenco ed il suo costante aggiornamento, ottenuto inserendo nell'elenco stesso *nelle fasi precedenti la codifica* i nomi delle variabili principali ed *in fase di codifica* quelli delle variabili di lavoro che eventualmente si rendono necessarie, evita il pericolo di utilizzare lo stesso nome per due variabili diverse oppure di assegnare nomi diversi alla stessa variabile. Questi errori infatti sono difficilissimi da individuare, in seguito, specie se il programma è lungo.

La codifica in Basic

Esaminiamo ora le istruzioni o gli insiemi di istruzioni che permettono di esprimere in Basic le varie strutture esposte nel capitolo precedente, dedicato all'analisi.

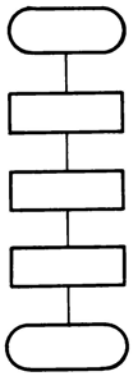


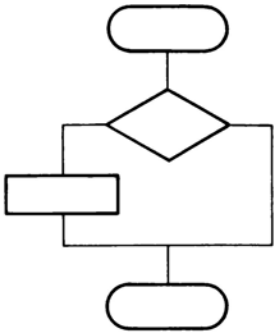
Figura 14a

La prima struttura, richiamata nella figura 14.a, è la sequenza.

La codifica di questa struttura non presenta particolari difficoltà in nessun tipo di linguaggio in quanto essa consiste in un insieme di istruzioni consecutive.

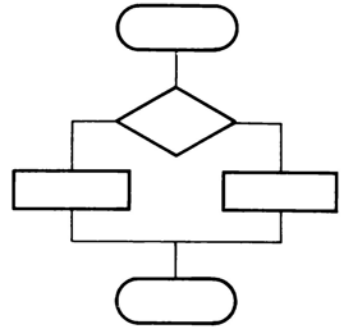
La seconda struttura presa in considerazione è quella che permette delle vie alternative e che si può presentare nelle tre forme distinte illustrate nelle figure 14.b, 15.c, 14.d.

La prima forma del tipo Se ... fai, si traduce in Basic con l'espressione IF ... THEN ... . Poiché il linguaggio Basic prevede una lunghezza massima per un'istruzione, potrebbero sorgere dei problemi nel momento in cui la codifica delle operazioni contenute nel blocco che logicamente dovrebbe seguire il THEN richiedesse più spazio di quello disponibile sulla riga: in questo caso si fa seguire il THEN da un GOSUB che rimandi ad un sottoprogramma in cui è stata codificata la routine descritta nel blocco in questione, oppure si utilizza



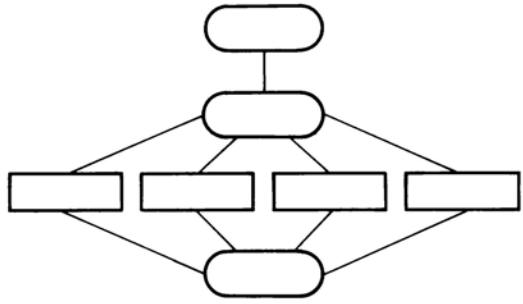
L'alternativa:
SE... FAI...

Figura 14b



L'alternativa:
SE ... ALLORA ... ALTRIMENTI ...

Figura 14c



L'alternativa:
NEL CASO... FAI (...)

Figura 14d

dopo l'IF la condizione inversa e si fa seguire il THEN da un GOTO che permetta di saltare quel blocco di istruzioni, successive alla linea IF ... THEN ..., che rappresentano la codifica delle routine in questione.

La seconda forma, Se... allora... altrimenti..., trova la sua naturale rappresentazione della classica espressione Basic: IF... THEN... ELSE... . È ovvio che anche per questo caso valgono le osservazioni effettuate precedentemente per la lunghezza delle istruzioni. Comunque, poiché il Basic utilizzato in quest'opera non ammette questa forma (non comune a tutte le versioni del Basic) si dovrà esprimere questa struttura in modo diverso. Di seguito vengono presentate le due soluzioni che si utilizzano più spesso:

Si possono usare due forme consecutive del tipo IF... THEN..., la prima con la condizione richiesta e la seconda con quella inversa (se nella prima si ha la condizione A > B nella seconda si deve avere B >= A). Se nello svolgimento del blocco relativo al primo THEN vengono modificati o il valore di A o

il valore di B o entrambi, occorre concludere tale blocco con un'istruzione GOTO che eviti l'esecuzione del secondo test non più attendibile. In questo caso il secondo IF risulta addirittura inutile in quanto a questo test si giungerebbe solo nel caso in cui il primo non fosse stato verificato.

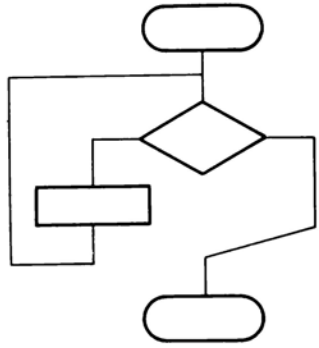
si può utilizzare una forma del tipo IF... THEN GOTO... per andare a piazzarsi, in caso di risposta affermativa, in un'altra regione del programma.

La terza forma, illustrata in figura 14.d, non è direttamente traducibile né in Basic né nella maggior parte dei linguaggi in uso. Questa forma è infatti prevista solo dai linguaggi PASCAL, ALGOL 68 e loro derivati.
Poiché comunque questa struttura nella pratica è molto utile, illustreremo come possa essere sostituita nel linguaggio Basic. Per rappresentarla, a seconda dei casi, si può:

far ricorso ad un'istruzione del tipo ON... GOTO (o GOSUB)

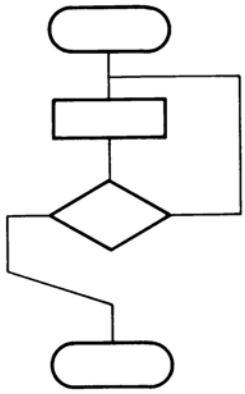
utilizzare una serie di forme del tipo IF... THEN... in cascata.

Quest'ultima soluzione è senza dubbio la più usata anche se rende il programma di difficile lettura.
Restano ancora da esaminare le due strutture iterative illustrate nelle figure 14.e e 14.f.



L'iterazione:
MENTRE... FAI...

Figura 14e



L'iterazione:
RIPETI... FINCHE'...

Figura 14 f

La prima, Mentre... fai..., può essere rappresentata in Basic mediante un IF posto all'inizio del ciclo, che permetta di saltare interamente il blocco delle istruzioni del ciclo quando la condizione non è verificata, oppure, nel caso in cui la condizione sia verificata, di eseguire tutte le istruzioni del blocco. L'ultima istruzione del blocco sarà ovviamente un GOTO di rinvio al test iniziale.

La seconda struttura, Ripeti... finché..., può essere in alcuni casi rappresentata con la coppia di istruzioni FOR... NEXT. In questo caso il test finale è fatto su una variabile numerica. In generale comunque questa struttura viene adeguatamente rappresentata mediante un IF finale che rimanderà, quando necessario, all'inizio del ciclo.

Per realizzare un programma in un linguaggio è utile continuare a fare uso della documentazione di cui si è spesso parlato.

Usando correttamente la documentazione prodotta, la codifica diventa molto facile e rapida e richiede solo una certa attenzione per evitare errori materiali di battitura.

Per coloro che hanno poca dimestichezza con le macchine da scrivere è sconsigliabile tentare di codificare il programma scrivendolo direttamente sulla tastiera del calcolatore. È preferibile invece giungere alla battitura del programma solo dopo aver effettuato la codifica: in pratica converrà compilare il programma utilizzando carta, gomma e matita e solo a lavoro ultimato passare alla battitura.

Sarebbe auspicabile inoltre lasciare trascorrere un pò di tempo, circa una settimana, tra la stesura del programma ed il suo inserimento in macchina: l'esperienza dimostra infatti che rileggendo il programma prima di batterlo sulla tastiera è possibile eliminare tanti piccoli errori di cui altrimenti ci si accorgerebbe solo in fase di esecusione.

Non insisterei sulle precauzioni da prendere al momento di generare il programma, sul numero di copie dello stesso da conservare, sulla frequenza con cui conviene salvaguardare il lavoro già svolto e sulla cura con cui bisogna conservare la documentazione relativa al programma.

Bisogna comunque tener conto di tutto ciò anche in fase di messa a punto e conservare tutte le versioni del programma, sperando ogni volta che sia l'ultima ... finché non si scopre un nuovo errore.

La messa a punto del programma

Malgrado tutte le precauzioni prese è molto improbabile che un programma "giri" alla prima prova. E se per caso ciò succede bisogna convincersi che nel programma vi sono probabilmente errori che si annullano a vicenda!

Gli errori possibili sono numerosi: cominciando da quelli banali di battitura fino a quelli molto più gravi dovuti ad una codifica errata o, ancor peggio, ad una errata interpretazione in fase di analisi.

Come fare allora per sapere se un programma è corretto? In teoria la risposta è semplice: salvo che in casi elementari non sarà mai possibile provare che il programma funzioni correttamente in tutti i casi in cui si può applicare.

Procedendo però con spirito pratico diremo che si può provare l'affidabilità di un programma e la sua quasi sicura correttezza. Come si può arrivare a questo?

Per ottenere un buon grado di affidabilità si possono percorrere due vie distinte e complementari:

- la prima consiste nel preparare una serie di esami già svolti completamente e verificare se il programma funziona in questi casi. Ovviamente gli esempi devono essere numerosi e coprire una vasta gamma di casi.
  la seconda consiste invece nell'eseguire delle prove, non sul programma completo, ma di volta in volta su alcune sue parti (moduli).

Questo secondo metodo si può utilizzare già in fase di stesura del programma. Infatti con questo sistema è possibile provare una parte del programma ancor prima di aver ultimato il programma stesso.

In pratica occorre sviluppare un "programma di prova" che permetta di provare e verificare i risultati di un modulo senza usare gli altri moduli del programma di partenza.

Ha importanza l'ordine con cui si controllano i vari moduli: conviene infatti iniziare dalla prova di quei moduli che, con struttura di sottoprogramma, vengono più spesso richiamati dal programma principale, fino a giungere alla prova di quelli che invece vengono utlizzati solo in casi particolari.

Per controllare un sottoprogramma occorrono due "programmi prova": uno che fornisca i dati al sottoprogramma, *rendendoli noti anche all'utente*, e l'altro che raccolga i risultati e li comunichi poi sul video. Ovviamente anche la correttezza di questi programmi di prova andrà verificata accuratamente.

Nel caso si debbano controllare degli algoritmi molto complessi e si conoscono altri metodi per ottenere gli stessi risultati, converrà codificare anche questi, pur se non utilizzati nel programma, e confrontare poi i vari risultati ottenuti.

Perché la verifica di un modulo abbia una certa validità occorrerà effettuare u-n gran numero di prove sul modulo in esame modificando le condizioni iniziali e controllando, tramite il diagramma a blocchi, di aver utilizzato almeno una volta le diramazioni che esso prevede.

Dopo tutte queste verifiche il modulo esaminato potrà essere considerato corretto e potrà anche venir utilizzato per effettuare la messa a punto di altri moduli del programma.

Procedendo così, modulo dopo modulo, si giungerà a controllare tutto il programma.

Un'istruzione (tipica di alcuni Personal Computers) che aiuta il programmatore in questa fase è la TRACE. Essa viene utilizzata durante la messa a punto di programmi molto complessi che presentano molte diramazioni. Occorre però

premunirsi per conservare quei dati che vengono di volta in volta visualizzati sul video perché questi vengono poi cancellati. Conviene perciò ricopiarli man mano che compaiono e controllarne solo in seguito la validità.
Anche se pochi programmatori hanno il coraggio di farlo, l'ideale sarebbe costruirsi l'insieme di tutti i risultati, anche quelli parziali, manualmente per poi poter effettuare un controllo completo dei dati ottenuti automaticamente. Ciò tra l'altro eviterebbe di avere grosse delusioni in seguito!
L'istruzione TRACE, a causa dei troppi risultati prodotti, risulta poco pratica quando si devono testare dei moduli contenenti delle iterazioni molto lunghe.

Un metodo alternativo consiste nell'inserire in posti del programma scelti accuratamente delle stampe di controllo contraddistinte da numeri di linea facilmente rintracciabili. Se ad esempio, come spesso accade, le linee del programma sono state numerate di 10 in 10, le stampe di controllo potranno essere caratterizzate da numeri di linea aventi come ultima cifra il 5. In questo caso tali istruzioni potranno essere facilmente riconosciute e cancellate una volta ultimata la messa a punto.

Le istruzioni di stampa inserite in questo modo permettono di controllare:

i valori delle variabili, che in genere non vengono visualizzati: ad esempio in un gioco di carte, quelle carte che "ha in mano" il calcolatore e che non vengono in genere visualizzate. Per la messa a punto del programma però è molto utile, se non addirittura indispensabile, conoscere questi valori;

i risultati intermedi relativi ad un calcolo che ha dato risultati errati ad un primo controllo;

gli elementi che vengono memorizzati in un file esterno: verificandone la correttezza si evitano spiacevoli sorprese nel momento in cui questi verranno utilizzati nel programma;

l'indice di un ciclo ed i valori particolari assunti da certe variabili che permettono di interrompere un'iterazione prima della sua naturale conclusione.
Per concludere segnaliamo che a volte, in fase di messa a punto, può accadere di voler scomporre una linea di programma, contenente più istruzioni, in diverse linee distinte, e ciò per poter inserire tra un'istruzione e l'altra delle stampe di controllo.

Si segnalano ora alcune cause di errore, di per sé banali, che possono però dare del filo da torcere durante la fase di messa a punto:

- su alcuni elaboratori come il TRS 80, è possibile battere contemporaneamente una lettera ed il tasto "Shift". Poiché in genere la scrittura corrente è quella maiuscola, con questa alternativa si dovrebbe ottenere un carattere

minuscolo. Ma poiché in generale non è prevista la visualizzazione su video di questo tipo di carattere, sul video stesso comparirà la normale scritta maiuscola e l'utente avrà l'impressione che non sia accaduto nulla di strano. Invece, dato che l'interprete Basic distingue il carattere minuscolo da quello maiuscolo, in fase di esecuzione verrà segnalato l'errore. In questi casi, in cui non si riesce ad individuare l'errore, l'unica soluzione consiste nel ribattere completamente tutta la riga. (Usando una stampante che preveda sia caratteri minuscoli che maiuscoli, l'errore verrebbe individuato immediatamente). Questo genere di errore è assai frequente quando si utilizzano molti caratteri speciali che prevedono l'uso del testo "Shift".

quando non si ha molta esperienza, l'uso dei cicli può generare grossi problemi: occorre pertanto verificare sempre con gran cura la correttezza delle condizioni iniziali e finali dell'iterazione; è infatti molto facile far iniziare o far terminare un'iterazione troppo presto o troppo tardi. Se poi si utilizzano cicli annidati tra loro, in cui per giunta i cicli interni dipendono da parametri il cui valore viene alterato dai cicli più esterni, le difficoltà diventano maggiori. Pertanto non conviene utilizzare questi metodi se non dopo aver fatto uno studio approfondito alla ricerca di eventuali soluzioni alternative.

notevoli difficoltà possono trarre origine dall'uso di un diagramma a blocchi troppo contorto, che comprenda cioé molte diramazioni. I salti incondizionati e troppo frequenti possono essere evitati usando la metodologia della programmazione strutturata, come è stata esposta nel capitolo dedicato all'analisi.

Non bisogna includere tra i salti incondizionati quelli che rimandano ad un sottoprogramma, in quanto quest'ultimo è un insieme di istruzioni distinto ed autonomo, anche se nel Basic viene considerato come parte integrante del programma principale, a causa della numerazione delle linee... (in altri linguaggi comunque l'uso dei sottoprogrammi genera problemi di altro tipo).

È in questo caso particolare che l'istruzione TRACE risulta di estrema utilità. Infatti essa permette di controllare quale chiamata ha dato origine all'errore, nel caso l'errore sia stato individuato in un sottoprogramma richiamato più volte.

Rintracciato un errore, occorre prendere molte precauzioni prima di correggerlo, a meno che non si tratti manifestamente di un errore di battitura. È necessario innanzitutto ricercare accuratamente la causa dell'errore, dato che questa non ha sempre origine nel posto in cui l'errore viene segnalato. Infatti esso può anche dipendere dal valore assunto da una certa variabile durante l'esecuzione di un'altra parte del programma.

Un controllo poco accurato sui dati comunicati dall'esterno genera spesso pseudoerrori. Esempio capitato effettivamente: in un elenco di nomi, ordinato alfabeticamente, un solo elemento non risultava correttamente ordinato. Era un nome che iniziava con la lettera G e che precedeva tutti gli altri, compresi quelli che cominciavano per A. Come era potuto succedere? Si trattava di un errore di programmazione? ... No! Era semplicemente accaduto che in fase di battitura l'operatore aveva inavvertitamente lasciato uno spazio bianco prima della lettera G. Poiché nel programma non era stato previsto alcun test per controllare la presenza di spazi bianchi all'inizio del nome e poiché per permettere il corretto ordinamento di due parole del tipo ART e ARTISTA il codice dello spazio "bianco" precede il codice di ogni altro carattere, si è avuto quell'ordinamento, solo apparentemente errato.

Rintracciato l'errore con le sue cause, occorre apportare le dovute correzioni ripartendo dal diagramma a blocchi, su cui verranno fatte le modifiche del caso; indi bisogna accertarsi della validità di queste correzioni.

Se questa verifica viene effettuata in modo frettoloso, possono esserci in seguito delle grosse perdite di tempo in quanto, correggendo il programma si rischia di introdurre errori complementari; le ricerche successive, partendo dal presupposto che la parte corretta sia efficiente, saranno lunghe e penose.

Pertanto, ultimata questa verifica formale, si passerà alla correzione vera e propria del programma. Infine si effettueranno tutte le prove necessarie per verificare la correttezza del programma verificato.

Anche dopo aver controllato tutte le varie parti del programma, occorrerà stare attenti: infatti il programma potrebbe contenere degli errori non ancora individuati. Pertanto se un risultato sembrerà errato od anche soltanto curioso, sarà necessario controllarne la correttezza.

Si sente spesso dire: "Il calcolatore ha sbagliato!" ma nella maggior parte dei casi l'errore è da imputare al programmatore od all'analista (È come quando si dice che un'auto è finita contro un albero, mentre in realtà ve l'ha portata il guidatore).

Richiami sulla documentazione

Prima di passare alla trattazione degli esempi vogliamo fare alcune considerazioni sul modo di raccogliere la documentazione relativa al programma. Abbiamo già visto che la documentazione deve essere prodotta man mano che si procede nel lavoro ed essere proporzionale alle difficoltà incontrate. A programma ultimato la documentazione deve contenere:

la definizione degli obiettivi con tutte le note esplicative
l'analisi con i relativi diagrammi a blocchi
- l'elenco delle variabili con le relative specifiche
il programma codificato
gli esempi utilizzati per la messa a punto
le routines di lavoro eventualmente utilizzate per effettuare le prove dei sottoprogrammi
se possibile, un paio di esempi concreti in cui è stato utilizzato il programma

È evidente che durante la fase di messa a punto occorrerà aggiornare continuamente la documentazione introducendo opportunamente tutta la documentazione relativa alle modifiche effettuate.

La documentazione relativa ad un programma può andare da 2 o 3 pagine, per programmi di piccole dimensioni, a qualche decina di pagine per programmi di una certa complessità (occorrerebbe comunque mezzo armadio per contenere tutta la documentazione relativa ad un programma con un numero di istruzioni aggirantesi intorno al milione).

Nel caso in cui la documentazione raggiunga dimensioni considerevoli, converrà dividerla in "capitoli". Ogni capitolo conterrà tutta la documentazione relativa ad un singolo modulo di programma e comprenderà:

l'elenco delle variabili utilizzate sia in questo modulo che in moduli precedenti,
l'elenco delle cosiddette variabili locali, utilizzate cioé solo all'interno del modulo in esame,
l'elenco delle variabili generate in questo modulo, che verranno usate poi anche in altri,
le caratteristiche dei files utilizzati,
i diagramma a blocchi relativi al modulo.

Per avere una veduta d'insieme di tutto il programma converrà poi collegare tra loro i diagrammi a blocchi di tutti i moduli utilizzando i simboli illustrati nella figura 1 del capitolo 3, al fine di ottenere un'unica configurazione rappresentante tutto il programma.

La documentazione relativa ad un modulo dovrà poi contenere anche la parte relativa alle routines che si sono utilizzate durante la messa a punto del modulo stesso.

Vogliamo ora segnalare un caso particolare: l'utilizzo di parti di un programma già sperimentato ed utilizzato in altre occasioni. In questo caso la documentazione relativa a questo modulo sarà molto succinta, in quanto, per la

maggior parte, farà riferimento al programma originario dal quale è stato estratto il modulo.

Programmando in Basic, comunque, poiché questo linguaggio tratta i sottoprogrammi come parte integrante del programma principale, occorrerà riscrivere tutto il modulo con nuovi numeri di linea.

Proprio per evitare questo si è soliti riunire tutte le routines relative alle funzioni maggiormente usate in quelle che si è soliti chiamare libreria di sistema e libreria di programmi/utente. Esse contengono i programmi relativi alla risoluzione di problemi classici di tipo matematico: funzioni algebriche e trigonometriche, risoluzioni di equazioni ... Inoltre generalmente contengono anche programmi per la generazione di grafici funzionali ed è in questo campo che in genere l'utente si crea una propria libreria per generare alcuni tipi di grafici che ritiene possano essergli utili.

CAPITOLO 5

PRESENTAZIONE DEGLI ESEMPI

Per illustrare i principi esposti nei capitoli precedenti e mostrare come si possa attuare concretamente un lavoro di programmazione abbiamo esposto esempi pratici.

I tre esempi, trattati nei capitoli rimanenti, sono di diversa natura. Essi comunque non sono né troppo semplici - in tal caso non sarebbero di alcuna utilità - né troppo complessi, perché una lunga trattazione risulterebbe troppo noiosa per persone non direttamente interessate al problema.

In tutti e tre gli esempi, comunque, si inizierà con spiegazioni molto dettagliate per giungere ad una trattazione più sciolta nel momento stesso in cui il lettore avrà preso confidenza con l'argomento trattato.

Il primo esempio affronta il *problema di creare una stringa di caratteri*: si tratta di *visualizzare un numero anziché espresso in cifre, espresso in lettere*, come si fa comunemente su assegni, contratti o altri documenti legali. Poiché questo tipo di problema viene in genere visto come una routine di utilità da usare come sottoprogramma, nell'esempio non sono previste né istruzioni di lettura (input) né di stampa (output). Per il modo con cui è stato concepito questo programma potrebbe essere benissimo inserito in una libreria-utente di routines di utilità.

Il secondo è un **esempio** *di gioco su elaboratore: il "421"*. In questo gioco il calcolatore farà la parte di un giocatore. Ricordiamo che in genere la programmazione delle regole di un gioco viene facilitata dal fatto che la definizione stessa del gioco prevede già un buon grado di astrazione. Inoltre si può utilizzare comodamente come periferica il video terminale. La parte più difficile del lavoro non consisterà quindi nell'analisi delle regole del gioco ma nella ricerca di una "condotta di gioco" che poi il calcolatore dovrà seguire. È evidente che, poiché non esiste mai una sola condotta di gioco, quella che verrà alla fine scelta rispecchierà la tecnica che il programmatore ritiene essere migliore.

Il terzo esempio è dedicato ad un *problema di tipo gestionale*: l'elaborazione di un *budget economico familiare* (tenuta di conti, previsione di spesa, ...). In questo caso le operazioni di input/output sono molto numerose: vengono

usati dei files sui quali si effettuano operazioni di cancellazione, di aggiornamento, di rettifica; inoltre vengono a volte creati dei files temporanei. Al contrario gli algoritmi su cui si basano le varie operazioni sono molto semplici.

In ogni caso si dovrà prestare molta attenzione al modo con cui i risultati dovranno essere presentati sul video o sulla stampante.

Questi tre esempi, anche se molto diversi tra loro, non ricoprono certo tutta la gamma dei problemi informatici: partendo da essi quindi il lettore dovrà poi approfondire la sua preparazione, consultando programmi ed articoli pubblicati su riviste specializzate e programmi di utilità utilizzati presso aziende o rintracciabili dai rivenditori di elaboratori.

Si consiglia al lettore, per trarre il maggior vantaggio possibile da quest'opera, di avvicinarsi alla lettura dei prossimi capitoli seguendo i criteri menzionati qui di seguito:

leggere la parte del capitolo che descrive una fase del lavoro (definizione degli obiettivi, analisi, ...) e **riflettere attentamente** su di essa prima di passare alla lettura della fase successiva;

confrontare il risultato delle proprie riflessioni con quanto esposto nel testo. A questo proposito va ricordato che, quando ci si accinge a formulare delle risoluzioni personali, il testo non deve essere considerato come un "errata-corrige" del programma prodotto, in quanto anche utilizzando le stesse tecniche di programmazione e lo stesso linguaggio si possono ottenere soluzioni diverse per uno stesso problema. Un confronto con quanto proposto dall'autore risulterà invece molto utile. È ovvio che nulla impedisce al lettore di approntare una diversa soluzione ai problemi proposti, che rispecchi il suo modo di pensare.

crearsi una propria documentazione relativa al problema trattato come più volte richiamato nel corso dei capitoli precedenti: ciò permetterà al lettore di seguire più agevolmente i discorsi fatti nel testo.
Tra i vari elementi che si consiglia di prendere in considerazione rivestono particolare importanza:

- l'elenco delle variabili utilizzate con le relative specifiche come ripetuto più volte;
- I principali diagrammi a blocchi con i loro sviluppi
- i tracciati record dei files utilizzati.

CAPITOLO 6

RAPPRESENTAZIONE DI UN NUMERO DECIMALE MEDIANTE UNA STRINGA DI CARATTERI ALFABETICI

In questo capitolo affrontiamo il problema di creare un sottoprogramma che permetta di trasformare un numero espresso nella sua forma decimale nella parola della lingua italiana corrispondente (cioé trasformi "23" in "ventitré".

Questo sottoprogramma prevede l'uso di numeri compresi tra 999999999.99 e 0.01. Gli eventuali valori decimali verranno comunque arrotondati sempre al centesimo! Pertanto il numero 1234.564 verrà trasformato in "milleduecentotrentaquattro e cinquantasei centesimi).

La limitazione superiore a numeri inferiori al miliardo è stata posta esclusivamente per motivi di ordine pratico e non lede in modo essenziale la generalità del problema.

In questo modo il nostro problema è completamente definito se si tralascia un piccolo particolare: il calcolatore non conosce le regole della lingua italiana e pertanto non sa scrivere in parole un numero! Occorrerà pertanto insegnargli come deve fare.

Conviene allora, prima di continuare, ricordare brevemente quali sono le regole fondamentali della grammatica che regolano l'uso degli aggettivi e sostantivi numerali:

1) *I numeri vengono rappresentati dagli aggettivi numerali cardinali* che, *tranne "uno" e "mille" sono indeclinabili, e dai sostantivi* milione e miliardo (quest'ultimo da noi non utilizzato);

2) Nella rappresentazione dei numeri mediante gli aggettivi numerali cardinali composti, *quando tale aggettivo risulta composto da "uno" o "otto" e da una parte rappresentante il valore delle decine, quest'ultima perde la vocale finale* (per esempio: venti-uno = ventuno; cinquanta-otto = cinquantotto)

3) *Nella rappresentazione delle migliaia*, per i multipli di mille, *viene usata la forma plurale "mila"* (duemila, trecentomila)

4) *l'aggettivo "uno" se seguito da un sostantivo* (milione, centesimo) *assume la forma sincopata "un".*

5) *Il numero tre non si accenta mentre si accentano i suoi composti*

A proposito della regola 5, precisiamo che nel corso della nostra trattazione, per non appesantire notevolmente il discorso relativo alla trattazione di costanti alfanumeriche, *non abbiamo tenuto conto di questa distinzione.*
Queste semplici regole ci permettono di scrivere in lingua italiana un numero ma non ci dicono la cosa essenziale: come passare da un numero espresso in cifre alla sua rappresentazione letterale! Non sono riuscito a trovare questo genere di regole in nessuna delle grammatiche consultate e suppongo che non ci riesca neppure il lettore.
Inoltre, se si guarda la cosa da vicino, si vede che queste regole non permettono di trattare in modo rigoroso le varie parole. Il numero "uno" può assumere due forme distinte: "uno" e "un" come del resto succede anche per "venti", "trenta", ... L'aggettivo tre può essere o non essere accentato ...
Questo genere di ambiguità è assai frequente quando si prendono in esame problemi pratici da trattare automaticamente: le regole utilizzate nella vita di tutti i giorni non hanno il rigore richiesto per il trattamento automatico delle informazioni e sarà pertanto necessario riorganizzare il tutto!
Fatta questa precisazione si può dire di aver puntualizzato in modo sufficiente il problema e definito con precisione il lavoro che si vuole fare. (Si sono ovviamente tralasciate tutte le rimanenti regole di morfologia occorrenti, che verranno supposte note). Non ci resta quindi, definito l'obiettivo, che passare all'analisi ... .
Come utilizzare questo ammasso disordinato di notizie per generare un programma che realizzi il nostro obiettivo, tenendo conto di tutto quanto è stato detto in precedenza?

Cominciamo a fare una semplice osservazione: 123 123 123,23 si scrive centoventitremilionicentoventitremilacentoventitré e ventitre centesimi". È ragionevole quindi suddividere il nostro numero in "blocchi di tre cifre" trattando separatamente i milioni, le migliaia, le unità, i centesimi e usare un sottoprogramma per la trasformazione in lettere di numeri di tre cifre. Tale sottoprogramma viene ovviamente usato anche per la parte decimale, in quanto un numero di due cifre è un caso particolare di un numero di tre cifre (un numero inferiore a 100 è anche inferiore a 1000!). Ad esempio il numero 23 023 023,23 si scrive "ventitremilioniventitremilaventitré e ventitré centesimi".

È evidente da quanto appena detto che la parte più difficoltosa del nostro lavoro sarà quella di creare questo sottoprogramma.

Prima di affrontare il problema relativo alla trasformazione di un numero espresso in cifre nella relativa forma letterale, bisogna affrontare quello di separare tra loro i milioni, le migliaia, le unità, i centesimi.

Senza entrare in merito allo sviluppo delle singole funzioni enunciate, possiamo cominciare a costruire un rudimentale diagramma a blocchi (figura 15) che espliciti le fasi fondamentali del lavoro da svolgere.

È evidente che questo diagramma a blocchi rappresenta solo un primo approccio al problema e che la funzione contenuta in ciascun blocco andrà ulteriormente dettagliata. Osserviamo in particolare che in questo diagramma non sono presenti i simboli di INIZIO e FINE. Questo permetterà in seguito di aggiungere, se necessario, dei blocchi prima della trattazione dei milioni e dopo la trattazione dei centesimi, cosa del resto prevedibile. A questo livello di analisi anche le funzioni definite nei vari blocchi son piuttosto vaghe (isolare, tradurre) e dovranno essere in seguito precisate.

Prima rappresentazione del problema

Figura 15

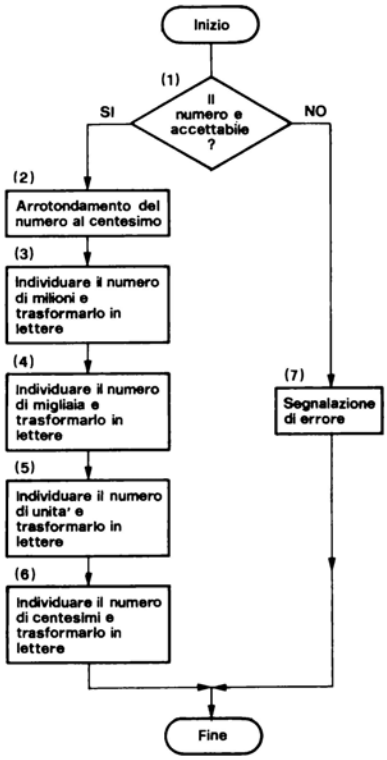
Per il momento sappiamo solo che bisogna lavorare su un numero di tre cifre decimali, compreso tra 0 e 999, e che la funzione di trasformazione di un numero in una stringa di caratteri che lo rappresenti, per numeri di tre cifre, sarà espletata da un apposito sottoprogramma. Dobbiamo invece ancora definire i legami che esistono tra questo sottoprogramma ed il programma principale, con particolare riguardo alle condizioni che devono essere verificate affinché il sottoprogramma venga chiamato.

Queste condizioni riguardano le limitazioni che abbiamo posto sui numeri da trattare: non devono raggiungere il miliardo e devono essere eventualmente arrotondati al centesimo.

La prima condizione è molto importante in quanto, una volta stabilito il valore massimo da trattare, occorrerà prevedere un controllo che, segnalando errore, inibisca tutte le operazioni nel caso in cui il numero inserito sia maggiore di quello massimo permesso. Si avrà allora una segnalazione di errore e l'interruzione dell'esecuzione del programma.

La seconda condizione è meno critica e ci permetterà in seguito di non preoccuparci del problema dell'arrotondamento. Infatti se si lavora sul numero 9999,997 si troverà nella scomposizione in migliaia, unità, centesimi: 9 per le migliaia, 999 per le unità e 99+1 (per arrotondamento) per i centesimi. Ma con 1000 centesimi bisogna aumentare di 1 la cifra delle unità ed azzerare il valore dei centesimi; inoltre poiché il valore delle unità in questo caso sarebbe 100 occorrerà aumentare di 1 le migliaia ed azzerare le unità ... Questo procedimento classico risulta, come si vede, piuttosto laborioso. Esiste però un metodo molto semplice che permette di giungere allo stesso risultato. Se in partenza si somma al valore da elaborare 0.005 (mezzo centesimo) e si tronca poi il risultato alla seconda cifra decimale si ottiene infatti il valore arrotondato al centesimo. (Esempi: 999.997 + 0.005 = 1000.002 che troncato come abbiamo già detto dà 1000.00; 9999.992 + 0.005 = 9999.997 che troncato dà 9999.99).

Tenendo conto di queste considerazioni si giunge alla stesura del diagramma a blocchi presentato in figura 16. Esso è formato da 7 blocchi, anche se si può sottintendere già l'esistenza di un altro blocco, utilizzato nei precedenti, rappresentante il sottoprogramma per il trattamento dei numeri di tre cifre.
Gli 8 blocchi così individuati sono quindi:
(1) : Controllo sull'accettabilità del valore da esaminare
(2) : Arrotondamento al centesimo
(3) : Isolamento e trasformazione del numero di milioni
(4) : Isolamento e trasformazione del numero di migliaia
(5) : Isolamento e trasformazione del numero di unità
(6) : Isolamento e trasformazione del numero di centesimi
(7) : Segnalazione di errore
(8) : Sottoprogrmma per la trasformazione di un numero di tre cifre.



Trasformazione di un numero in una stringa di caratteri

Figura 16

Abbiamo così individuato una prima soluzione del nostro problema.
Una regola classica della programmazione è: un problema non ha mai un'unica soluzione!

Dobbiamo allora ricercare anche le altre?

Non necessariamente, anche se sarà conveniente sfruttare questa possibilità nel caso in cui il metodo adottato risulti troppo contorto e pieno di difficoltà.
Per inciso notiamo che il diagramma a blocchi presentato nella figura 16 rispetta le regole strutturali proposte nel capitolo dedicato all'analisi. Possiamo infatti pensare il diagramma a blocchi in oggetto come lo sviluppo di una forma alternativa del tipo presentato nella figura 10, in cui la parte di destra è ridotta al solo blocco (7) mentre quella di sinistra è stata scomposta nei blocchi dal (2) al (6), secondo lo schema della figura 8.

D'ora in avanti eviteremo di soffermarci su argomentazioni di questo tipo tralasciando gli aspetti formali del procedimento.

Come proseguire ora nel lavoro?

Dettagliando sempre più il nostro diagramma a blocchi. Per far ciò ripartiamo dalla definizione degli obiettivi e classifichiamo le regole della grammatica italiana, già ricordate in precedenza, in modo più pertinente al nostro particolare problema:

a) i numeri si rappresentano tramite gli aggettivi numerali indicati nel presente elenco: uno, due, tre, quattro, ..., cento, ..., mille;

b) a quest'elenco vanno aggiunti i sostantivi: milione e centesimo;

c) solo l'aggettivo mille ammette il plurale "mila" come lo ammettono i sostantivi "milione" e "centesimo";

d) gli aggettivi: venti, trenta, quaranta, ..., novanta perdono la vocale finale se seguiti da "uno" o "otto";

e) per dividere la parte intera da quella decimale, qualora nessuna delle due sia nulla, si utilizza la congiunzione "**e**";

f) l'aggettivo "uno" davanti ai sostantivi milione e centesimo assume la forma sincopata "un".

Le regole (a), (b), (c), (e) hanno carattere generale mentre le altre vengono utilizzate solo in casi particolari. Inoltre la regola (d) viene utilizzata solo all'interno del sottoprogramma per la trasformazione dei numeri di tre cifre. Le regole (c) ed (e) vengono utilizzate all'uscita del sottoprogramma di cui sopra e la regola (f) permette di generare direttamente le stringhe: "**un milione**" e "**un centesimo**".

Tenendo conto di quanto espresso nella regola (d) dovremo prevedere all'interno del sottoprogramma per il trattamento dei numeri di tre cifre un test che ci permetta di decidere quale delle due forme (normale o sincopata) utilizzare per le decine.

La regola (e) viene usata solo quando si verificano contemporaneamente due condizioni particolari: la parte intera e la parte decimale del numero da trasformare sono entrambe diverse da zero.

Ricapitoliamo ora quali sono i passi che si dovranno compiere per trasformare in lettere i centesimi:

- calcolare il numero di centesimi
  se il numero di centesimi è nullo non fare niente
  se il numero di centesimi è uno generare la scritta "UN CENTESIMO"
  altrimenti ricorrere al sottoprogramma per la trasformazione di un numero di tre cifre spesso citato (blocco 8)
  completare a destra la strin-ga di caratteri generata dal sottoprogramma citato, nel caso in cui il numero di centesimi sia maggiore di 1, con i caratteri "CENTESIMI"

se la parte intera e la parte decimale del valore da trattare sono entrambe non nulle completare a sinistra la stringa di caratteri ottenuta con i metodi indicati nei punti precedenti con i tre caratteri "E". Osserviamo a questo proposito che il carattere bianco viene considerato come un particolare carattere da stampare.

Da quanto sopra si può ricavare una condotta risolutiva che permetta di sviluppare adeguatamente il blocco (6). Abbiamo inoltre ricavato alcune informazioni utili per lo sviluppo completo del programma. In particolare osserviamo che occorrerà comunque memorizzare, almeno fino all'esecuzione del blocco (6), il valore iniziale inserito per poter decidere se occorre o no applicare la regola (e). A questo proposito osserviamo che, poiché questa necessità compare solo nel caso in cui il numero di centesimi è diverso da zero (nel caso in cui è zero si è detto di non fare nulla), basterà controllare se il valore iniziale è maggiore di 1 per assicurarsi che la parte intera non sia zero.

Prima di descrivere questi nuovi risultati tramite un diagramma a blocchi che rappresenti la scomposizione del blocco (6), riassumiamo la filosofia del sottoprogramma che permette di trasformare un numero di tre cifre in lettere: viene creata progressivamente una stringa di caratteri, da destra a sinistra, come se venisse scritta manualmente.

Ogni blocco (3), (4), (5), (6) produce quindi una stringa relativa al trattamento della parte del numero inserito che gli compete; tale stringa potrà anche essere vuota (caso in cui la parte di numero considerata sia nulla).

La figura 17 illustra il diagramma a blocchi relativo alla scomposizione del blocco (6), in cui sono stati evidenziati tutti i casi particolari enunciati:

a) il caso in cui il numero di centesimi è zero
b) il caso in cui il numero di centesimi è 1 (in questo caso non si ricorre al sottoprogrmma (8) per evitare delle inutili elaborazioni)
c) il caso in cui la parte intera del valore di partenza è zero (non bisogna applicare la regola (e)).

Il diagramma a blocchi che si ottiene è composto quindi dai seguenti blocchi logici:

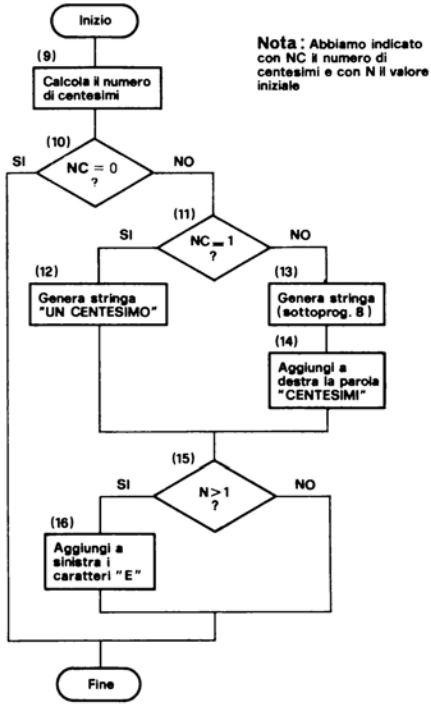
(9) : Calcolo del numero di centesimi
(10) : Controllo: il numero di centesimi è zero?
(11) : Controllo: il numero di centesimi è 1?
(12) : Generazione della stringa nel caso di un centesimo
(13) : Generazione della stringa nel caso generale con l'uso del blocco (8)

(14) : Completamento a destra della stringa ottenuta

(15) : Controllo sulla parte intera del numero

(16) : Completamento della stringa a sinistra

Come si può facilmente intuire dal contenuto dei vari diagrammi a blocchi, spesso capita che la traduzione in Basic della parte funzionale di un'istruzione sia di gran lunga inferiore alla parte di testo che essa deve produrre.



Generazione della stringa corrispondente ai centesimi (6)

Figura 17

In modo del tutto analogo a quanto fatto per il blocco (6) possiamo sviluppare i blocchi (3), (4), (5) relativi ai milioni, alle migliaia, alle unità.

Per il blocco (3) osserviamo che se il numero di milioni è zero non occcorre fare nulla, se è uno occorre produrre la stringa "UNMILIONE" e se è maggiore di uno occorre produrre, tramite il sottoprogramma più volte citato, la stringa

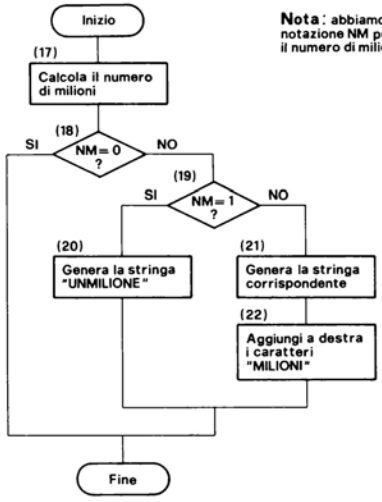
corrispondente al numero di milioni, facendola seguire dai caratteri "MILIONI". Quanto sopra esposto viene chiaramente illustrato in figura 18 con un diagramma a blocchi di cui specifichiamo i contenuti:

(17) : Calcolo del numero di milioni

(18) : Controllo: il numero di milioni è 0?

(19) : Controllo: il numero di milioni è 1?

(20) : Generazione della stringa nel caso in cui il numero di milioni sia 1

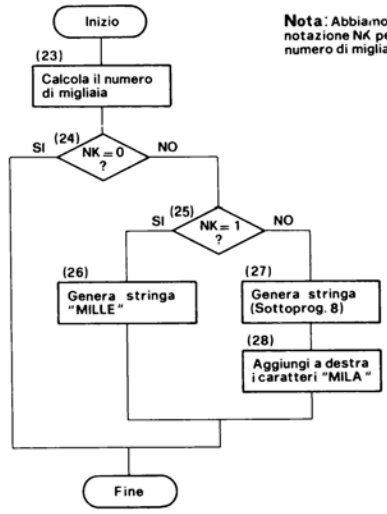


Generazione della stringa corrispondente ai milioni (3)

Figura 18

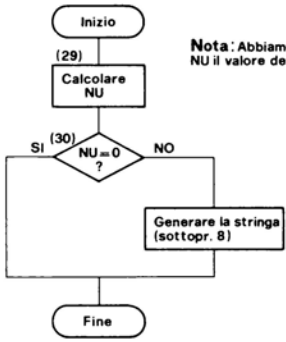
(21) : Generazione della stringa nel caso generale con l'uso del sottoprogramma 8
(22) : Completamento a destra della stringa ottenuta

Anche nel caso del trattamento delle migliaia si dovrà distinguere il caso in cui il numero di migliaia è 1 (bisognerà pertanto produrre i caratteri "MILLE") ed il caso in cui tale numero è maggior-e di 1 (si produrrà prima la stringa corrispondente al numero di migliaia e poi la si farà seguire dai caratteri "MILA").



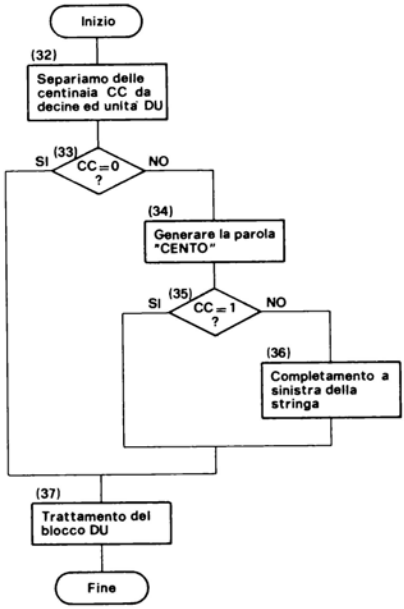
Generazione della stringa corrispondente alle migliaia (4)

Figura 19



Generazione della stringa corrispondente alle unità (5)

Figura 20



Primo sviluppo del sottoprogramma per il trattamento di numeri di tre cifre (8)

Figura 21

Il diagramma a blocchi relativo al trattamento delle migliaia, del tutto simile al precedente, è rappresentato nella figura 19 e contiene i blocchi dal (23) al (28).

La scomposizione del blocco (5) è molto semplice in quanto non prevede casi particolari. L'unico controllo da eseguire è quello relativo all'esistenza o meno di un valore non nullo per le unità. Per completezza, comunque, riportiamo anche il diagramma a blocchi relativo a questo caso.

Non ci resta ora che esaminare attentamente il sottoprogramma per la trasformazione di un numero di tre cifre, cioé per numeri compresi tra 1 e 999. Questo sottoprogramma, ricevuto un numero, restituisce una stringa di caratteri che rappresenta il numero scritto in lingua italiana.

Per poter costruire un diagramma a blocchi piuttosto dettagliato che rappresenti questo sottoprogramma, occorre analizzare le varie fasi che il sottoprogramma deve effettuare per giungere alla generazione della stringa.

Osserviamo innanzitutto che in italiano esistono delle forme particolari per indicare i numeri compresi tra 1 e 19. Pertanto converrà separare la cifra delle centinaia, che d'ora in avanti verrà indicata con il simbolo CC, dal blocco decine-unità (DU). Se quest'ultimo è un numero inferiore a 20 permetterà di generare immediatamente la stringa corrispondente, altrimenti occorrerà procedere ad una nuova scissione in cifra delle decine (CD) e cifra delle unità (CU).

Le operazioni che il sottoprogramma deve eseguire per giungere alla generazione della stringa sono pertanto:

separazione della cifra delle centinaia dal blocco decine-unità

trattamento della cifra delle centinaia (CC):
- Se CC=0 non bisogna generare nessuna stringa
- se CC≠0 occorre generare la parola "CENTO" e farla precedere, nel caso in cui il numero delle centinaia sia maggiore di 1, dalla parola corrispondente al valore di CC (Nel caso in cui CC=1 basterà invece la sola parola "CENTO")

trattamento del blocco DU:
- se DU = 0 non occorre generare alcuna stringa
- se DU < 20 si genera la parola corrispondente al valore di DU
- se DU > 19 occorre suddividere il blocco DU in cifra delle decine CD e cifra delle unità CU e trattarle separatamente come segue:

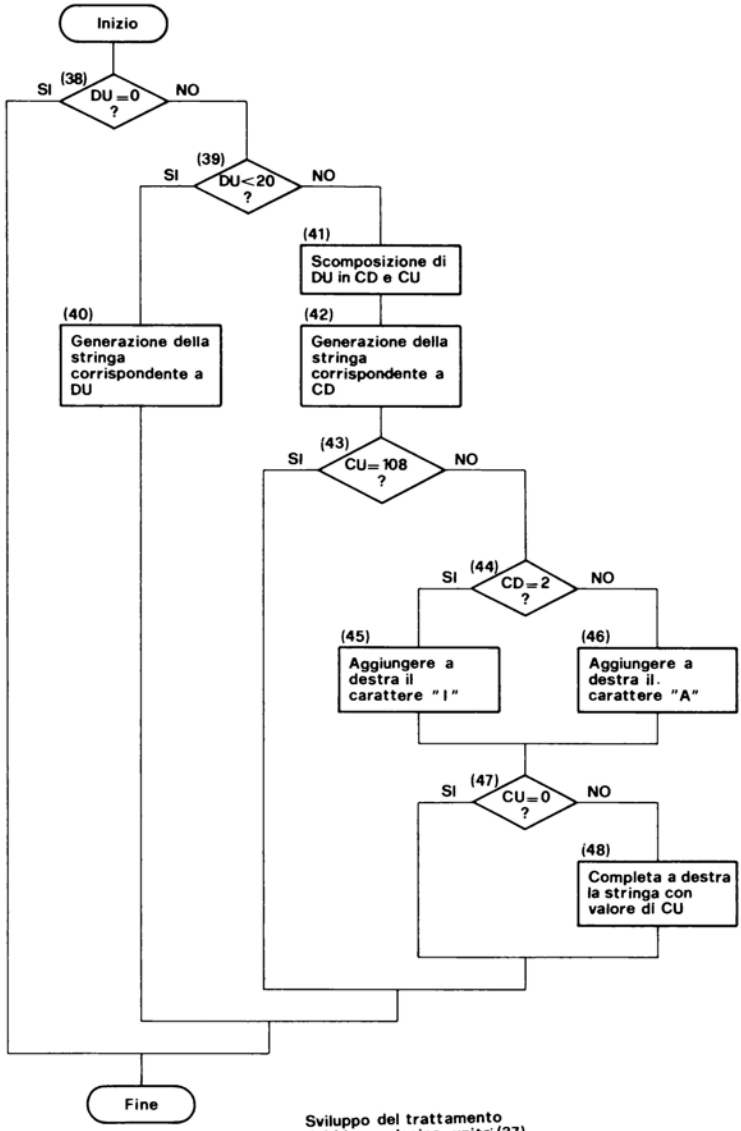
  * generare la stringa corrispondente al valore di CD (se ad esempio CD=2 occorre generare la parola "VENTI")
  * troncare la stringa così ottenuta, elidendo l'ultima vocale, nel caso in CU=1 o CU=8
  * completare a destra la stringa, aggiungendo la parola corrispondente al valore di CU, se tale valore non è zero (Se CU=0 la stringa risulta già completa).

Un primo diagramma a blocchi relativo al trattamento dei numeri di tre cifre è rappresentato nella figura 21. In tale diagramma a blocchi, per non appesantire troppo il discorso, è stata considerata come funzione il trattamento del blocco DU. La scomposizione di tale funzione verrà illustrata a parte nella figura 22.

Prima di passare allo sviluppo del blocco (37) facciamo un'osservazione di carattere pratico: poiché utilizzando il Basic dei Personal Computers è molto semplice operare l'unione di due stringhe o l'aggiunta di un certo numero di caratteri a destra di una stringa già generata mentre spesso non è possibile effettuare il troncamento eliminando dei caratteri, converrà predisporre per le decine la generazione delle stringhe tronche (VENT, TRENT, ...) che verranno eventualmente completate con l'aggiunta dell'ultima vocale nel caso in cui il numero delle unità non sia 1 o 8. Notiamo infine che la stringa VENT verrà completata con l'aggiunta della lettera I mentre le altre con l'aggiunta della lettera A.
La figura 22 illustra il diagramma a blocchi relativo alla scomposizione del

blocco (37). Esso è formato dai blocchi da (38) a (48) che svolgono le seguenti funzioni:

(38) : Controllo sul fatto che il valore di DU sia zero
(39) : Controllo sul fatto che il valore di DU sia minore di 20
(40) : Generazione della stringa nel caso in cui DU < 20



Sviluppo del trattamento del blocco decine-unita (37)

Figura 22

(41) : Scomposizione di DU in decine (CD) ed unità (CU)
(42) : Generazione della parola troncata corrispondente al valore di CD
(43) : Controllo sul valore di CU (1 o 8)
(44) : Controllo sul valore di CD (è 2?)
(45) : Completamento stringa tronca nel caso in cui CD = 2
(46) : Completamento stringa tronca negli altri casi

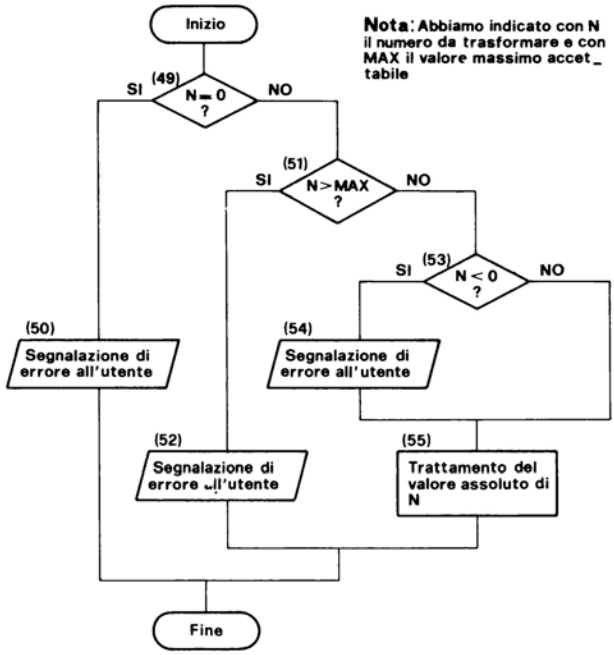


Figura 23

(47) : Controllo sul fatto che CU non sia zero
(48) : Completamento della stringa con l'aggiunta della parola corrispondente al valore di CU

È evidente a questo punto che per ottenere un trattamento corretto e veloce di un numero è necessario mantenere in memoria le parole standard che vengono utilizzate in più parti del programma e mi riferisco in particolare alle parole corrispondenti ai numeri da 1 a 19 ed alle parole (troncate) corrispondenti alle decine (VENT, TRENT, ...).

Un metodo comodo per memorizzare ed utilizzare queste è quello di usare dei vettori alfanumerici nei quali il valore dell'indice rappresenta anche il valore del numero corrispondente alla stringa contenuta in quella posizione. Per esempio, denominando NUM il vettore contenente le stringhe corrispondenti ai numeri da 1 a 19 si avrà: NUM(1) = "UNO", NUM(2) = "DUE", NUM(11) = "UNDICI", e così via. Ovviamente non verrà presa in considerazione la posizione NUM(0).

Una tabella simile a quella presentata può essere utilizzata per la memorizzazione di dati relativi alle decine.

Osserviamo a questo proposito che il metodo da noi proposto per il trattamento del blocco DU non è l'unico possibile. Strutturando il programma a moduli, se si desidera modificarne una parte, ad esempio la parte relativa al trattamento del blocco DU, è possibile farlo senza dover per questo modificare la parte rimanente.

Questa tecnica di programmazione è stata per un certo tempo chiamata programmazione modulare e le parti "indipendenti" in cui viene diviso il programma venivano chiamate moduli. Il modulo descriveva una "funzione" che poteva essere realizzata in più modi. (Nel nostro caso la funzione è la generazione della stringa corrispondente al blocco DU). Uno dei vantaggi della programmazione strutturata è proprio quello di evidenziare tali moduli, anche se le funzioni non sono state evidenziate tutte all'inizio.
Dopo questa breve digressione ritorniamo all'esame del diagramma a blocchi proposto nella figura 22. Il blocco (39) serve per distinguere dal caso generale il caso in cui il valore di DU è inferiore a 20: esistono infatti in italiano delle parole particolari per rappresentare tali valori. La generazione della stringa, nel caso generale, viene sviluppata nei blocchi dal (41) al (48). Il blocco (43) seguito dal blocco (47) (caso in cui CU = 1 o 8) sembra generare una ridondanza in quanto è ovvio che se CU assume i valori 1 o 8 non può assumere anche il valore 0. Ma, se per questo motivo si eliminasse il secondo quesito, collegando direttamente il blocco (43) con il blocco (48), non si otterrebbe più una struttura di programmazione corretta, cioé conforme a quelle presentate nel capitolo dedicato all'analisi.

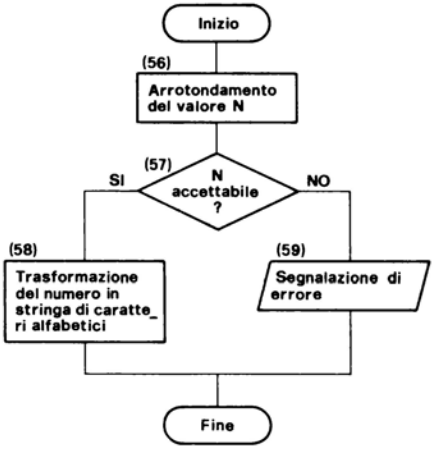
Ritornando ora ad esaminare la figura 16 osserviamo che abbiamo ormai sviluppato tutti i blocchi da (3) a (8) (quest'ultimo rappresenta il sottoprogramma per il trattamento di numeri di tre cifre).

La funzione richiesta nel blocco (2) può essere eseguita applicando il metodo di arrotondamento a cui si è già accennato in precedenti occasioni e cioé quello che consiste nell'aggiungere al numero iniziale 0.005 e nel considerare poi il risultato troncato alla seconda cifra decimale.

Per poter realizzare le funzioni rappresentate nei blocchi (1) e (7) occorre definire un valore massimo da trattare, in modo che, se questo viene superato, il programma non venga eseguito e vi sia una segnalazione di errore.

Riflettendo bene ci si accorge che si sono trascurati dei casi particolari:

il caso in cui il numero iniziale sia zero. Applicando il programma da noi realizzato si otterrebbe in uscita una stringa vuota (non vi sono infatti né milioni, né migliaia, né unità, né centesimi). Bisognerà pertanto prevedere anche in questo caso una particolare segnalazione all'utente.



Trasformazione di un numero nella sua forma letterale

Figura 24

Programma-prova

Figura 25

il caso in cui il numero iniziale sia negativo. Converrà predisporre il programma affinché segnali all'utente questa anomalia e trasformi in lettere il valore assoluto del numero stesso.

Tenendo conto di queste osservazioni modifichiamo i blocchi (1) e (7) nel modo seguente: il blocco (1) diventa "il numero da esaminare è superiore al valore massimo prefissato oppure è nullo oppure è negativo?" ed il blocco (7) si espanderà per permettere la trattazione dei singoli casi.

Poiché lo scopo per cui è stato realizato il programma è quello di generare una stringa di caratteri, cominciamo con l'esaminare attentamente come questa viene creata. Osserviamo innanzitutto che la stringa non viene generata tutta insieme ma suddivisa in 4 segmenti (sottostringhe): uno per ciascuno dei blocchi di tre cifre in cui è stato suddiviso il valore iniziale.

Per il numero 123 456 789.01 i segmenti sono rispettivamente: "CENTOVENTITREMILIONI"; "QUATTROCENTOCINQUANTASEIMILA"; "SETTECENTOOTTANTANOVE"; "E UN CENTESIMO".
Per il numero 0.23 vi saranno invece tre segmenti vuoti ed uno formato da: "VENTITRE CENTESIMI".

Per poter memorizzare separatamente questi quattro segmenti utilizzeremo un vettore alfanumerico di 4 elementi che chiameremo TX$.

Il programma prevede poi la presenza di segnalazioni di casi anormali: per comodità questo tipo di segnalazione verrà realizzato anziché con parole con valori numerici. Si utilizzerà per questo scopo una variabile numerica che assumerà solo i seguenti valori particolari: 10 nel caso in cui il numero introdotto sia troppo grande (in valore assoluto); 11 nel caso in cui tale numero (già arrotondato) sia 0; 1 nel caso in cui il valore introdotto sia negativo e 0 in tutti gli altri casi. (Nei primi due casi il valore della variabile rappresenta l'unico risultato del programma). Questa variabile può essere considerata di tipo intero in quanto essa può assumere solo i valori sopra citati: tale variabile verrà indicata con ER%.

Al sottoprogramma verrà poi trasmesso un altro parametro: il valore da noi inserito. Per rappresentare questo valore verrà usata la variabile reale N.

Esaminiamo ora le variabili locali.

Innanzitutto vi sono quelle che dovranno contenere le parole standard che si utilizzeranno nel sottoprogramma: UNO, DUE, ... DIECI, UNDICI, ...

Vi sono diversi modi per manipolare queste parole:

sia direttamente mediante delle particolari istruzioni: se si deve aggiungere alla stringa TX$ (1) i caratteri MILA si può usare l'istruzione Basic

TX$ (1) = TX$ (1) + "MILA"

- sia memorizzando precedentemente queste parole in tabelle e poi utilizzando ciò che in queste è contenuto.

Nel caso del sottoprogramma da noi realizzato vengono utilizzati entrambi i metodi.

In particolare sono utilizzate due tabelle, realizzate mediante vettori alfanumerici, per memorizzare alcune delle parole più usate e precisamente:

un vettore UN$ di 19 elementi che conterrà le parole corrispondenti ai valori da 1 a 19

un vettore DE$ di 8 elementi che conterrà le parole sincopate corrispondenti alle decine (VENT, TRENT, ... NOVANT)

Nota: poiché i vettori hanno la prima posizione corrispondente al valore zero dell'indice, non vi sarà corrispondenza perfetta tra il valore da considerare ed il valore dell'indice che individua la corrispondente parola nel vettore. L'indice sarà comunque facilmente ricavabile dal valore del numero da trattare. Infatti nel primo vettore la parola corrispondente al valore K sarà contenuta nella posizione K-1. (Per esempio per il valore 3 la parola TRE sarà contenuta in UN$ (2)) mentre per il secondo vettore, posto K il valore delle decine si troverà la stringa corrispondente nella posizione K-2.

Per inserire le parole standard nei due vettori si possono seguire due metodi: utilizzare le istruzioni READ e DATA, che usano un file interno, oppure la sola istruzione INPUT per la lettura dall'esterno. (Se si avesse a disposizione una memoria di massa a disco si potrebbe anche generare un file su disco contenente tutti i nostri dati e leggerli poi da programma direttamente su questo file).

Il più pratico, quando i dati non siano eccessivamente numerosi, è senza dubbio il primo, anche se richiede l'uso di un maggior numero di locazioni di memoria; pertanto nel nostro programma verrà utilizzato questo metodo.

Inoltre, poiché i dati vengono letti una sola volta all'inizio e non è stata prevista nel sottoprogramma da noi approntato una fase di lettura, abbiamo utilizzato, nel programma-prova, un apposito sottoprogramma di inizializzazione. Tale sottoprogramma, che viene richiamato all'inizio del programma principale, contiene tra l'altro le dichiarazioni relative al dimensionamento dei tre vettori utilizzati DE$, UN$, TX$.

Non ci resta ora che esaminare le operazioni da eseguire sul numero N. Tale valore va innanzitutto scomposto in: segno (NS) e valore assoluto (NA). La prima variabile (NS) verrà utilizzata per segnalare eventualmente che N è negativo. Le operazioni successive vengono invece effettute su NA, che a sua

volta viene suddiviso in quattro parti intere di due cifre: NM% (numero di milioni), NK% (numero di migliaia), NU% (numero di unità), NC% (numero di centesimi). Ciascuna di queste variabili potrà assumere valori compresi tra 0 e 999. (Nella figura 26 viene illustrato, come esempio, il metodo usato per isolare il numero di milioni).

Poiché di volta in volta ognuna di queste variabili dovrà essere trasmessa al sottoprogramma per il trattamento di un numero di tre cifre (blocco 8) occorre predisporre una variabile dello stesso tipo, che chiameremo NN%, detta variabile di trasmissione.

A sua volta questà verrà scomposta in' CC% (cifra delle centinaia) e DU% (blocco decine e unità); quest'ultimo dovrà poi essere eventualmente ancora suddiviso in CD% (cifra decine) e CU% (cifra unità).

I nomi dele variabili sono stati scelti in modo da mantenere una corrispondenza con la simbologia utilizzata nei diagrammi a blocchi.

Facendo riferimento a questi diagrammi possiamo facilmente eseguire la codifica in linguaggio Basic. In tale codifica sono state utilizzate, oltre alle variabili sopra elencate, altre tre variabili: I% di controllo per i cicli: XP come indice del vettore TX$; NR come variabile di lavoro.

Pur non utilizzando nella codifica le forme particolari della programmazione strutturata, non presenti in tutte le versioni del linguaggio Basic, si è cercato di mantenere una certa modularità in modo da rendere il programma di facile lettura ed interpretazione.

Il lettore potrà così facilmente notare la corrispondenza esistente tra le istruzioni e le funzioni rappresentate nei blocchi dei vari diagrammi.

Tutti gli argomenti trattati in questo capitolo possono rappresentare la documentazione relativa al programma realizzato. La lista delle variabili è molto utile, come precedentemente già detto, in fase di codifica.

Vogliamo notare infine come l'inserimento nel programma di numerose frasi di commento faciliti notevolmente l'interpretazione del programma stesso anche per chi lo volesse esaminare senza consultare tutta la documentazione.

```
4 REM ------------------------------------------------------------
5 REM   PROGRAMMA DI PROVA PER LA ROUTINE DI TRASFORMAZIONE DI
6 REM   UN NUMERO IN UNA STRINGA DI CARATTERI
7 REM ------------------------------------------------------------
8 REM
10      GOSUB 30000: REM SOTTOPROGRAMMA PER INIZIALIZZAZIONE
20      INPUT N:     REM RICHIESTA NUMERO DA ESAMINARE
```

```
30      GOSUB 30200: REM ROUTINE DI TRASFORMAZIONE
40      PRINT TX$(0):REM    STAMPE
50      PRINT TX$(1):REM    SEPARATE
60      PRINT TX$(2):REM    PER
70      PRINT TX$(3):REM    CONTROLLO
80      PRINT "ERR= ";ER%
90      PRINT TX$(0);TX$(1);TX$(2);TX$(3)
100     GOTO 20:      REM  NUOVA PROVA
999     END
1000 REM
1001 REM
1002 REM ------------------------------------------------------------
29900 REM  SOTTOPROGRAMMA DI INIZIALIZZAZIONE
29901 REM
30000       DIM UN$(18),DE$(7),TX$(3)
30010       FOR I%= 0 TO 18:READ UN$(I%):NEXT
30020       FOR I%= 0 TO 7:READ DE$(I%):NEXT
30030       RETURN
30040 REM
30050       DATA UNO,DUE,TRE,QUATTRO,CINQUE,SEI,SETTE,OTTO,NOVE
30060       DATA DIECI,UNDICI,DODICI,TREDICI,QUATTORDICI,QUINDICI
30070       DATA SEDICI,DICIASSETTE,DICIOTTO,DICIANNOVE
30080       DATA VENT,TRENT,QUARANT,CINQUANT,SESSANT,SETTANT
30090       DATA OTTANT,NOVANT
30100 REM
30101 REM ------------------------------------------------------------
30110 REM
30120 REM
30190 REM  ROUTINE PER LA TRASFORMAZIONE DI UN NUMERO IN LETTERE
30191 REM
30200       FOR I%=0 TO 3:TX$(I%)="":NEXT
30210       ER%=0: XP=0
30220       NS=SGN(N):NA=SGN(N)*N
30225 REM                                 ARROTONDAMENTO
30230 NA=NA+.005
30235 REM
30240 IF NA<.01 THEN ER%=11:RETURN
30250       IF NA>=1E+09 THEN ER%=10: RETURN
30260       IF NS=-1 THEN ER%=1
30265 REM                   TRATTAMENTO MILIONI
30270       NM%=INT(NA/1E+06):NR=NA-NM%*1E+06
30280       IF NM%=0 THEN 30330
30290       IF NM%=1 THEN TX$(0)="UNMILIONE":XP=XP+1:GOTO 30330
30300       NN%=NM%: GOSUB 30500
30310       TX$(0)=TX$(0)+"MILIONI"
30320       XP=XP+1
30325 REM                  TRATTAMENTO MIGLIAIA
30330       NK%=INT(NR/1000):NR=NR-NK%*1000
30340       IF NK%=0 THEN 30390
30350       IF NK%=1 THEN TX$(XP)="MILLE":XP=XP+1:GOTO 30390
30360       NN%=NK%: GOSUB 30500
30370       TX$(XP)=TX$(XP)+"MILA"
30380       XP=XP+1
30385 REM                   TRATTAMENTO UNITA'
30390       NU%=INT(NR):NR=NR-NU%
30400       IF NU%=0 THEN 30440
30410       NN%=NU%
30420       GOSUB 30500
30430       XP=XP+1
```

```
30435 REM                  TRATTAMENTO CENTESIMI
30440       NC%=INT(NR*100)
30450       IF NC%=0 THEN RETURN
30460       IF NC%=1 THEN TX$(XP)="UN CENTESIMO": GOTO 30495
30470       NN%=NC%
30480       GOSUB 30500
30490       TX$(XP)=TX$(XP)+" CENTESIMI"
30495       IF NA>1 THEN TX$(XP)=" E "+TX$(XP)
30496       RETURN
30497 REM ------------------------------------------------------
30498 REM SOTTOPROGRAMMA PER IL TRATTAMENTO DI NUMERI DI 3 CIFRE
30499 REM
30500       CC%=INT(NN%/100):DU%=NN%-CC%*100:REM SCANSIONE NUMERO
30510       IF CC%=0 THEN 30540
30515 REM                          TRATTAMENTO CENTINAIA
30520       TX$(XP)="CENTO"
30530       IF CC%>1 THEN TX$(XP)=UN$(CC%-1)+TX$(XP)
30540       IF DU%=0 THEN RETURN
30545 REM                          TRATTAMENTO DECINE E UNITA'
30550       IF DU%<20 THEN TX$(XP)=TX$(XP)+UN$(DU%-1):RETURN
30560       CD%=INT(DU%/10):CU%=DU%-CD%*10
30570       TX$(XP)=TX$(XP)+DE$(CD%-2)
30580       IF CU%=1 THEN 30620
30590       IF CU%=8 THEN 30620
30600       IF CD%=2 THEN TX$(XP)=TX$(XP)+"I":GOTO 30620
30610       TX$(XP)=TX$(XP)+"A"
30620       IF CU%>0 THEN TX$(XP)=TX$(XP)+UN$(CU%-1)
30630       RETURN
30640 REM  ------------------------------------------
```

Elenco delle variabili con riferimenti al programma

Nomi delle variabili	Dimensione	Descrizione	Riferimenti
CC%		cifra delle centinaia	30500-30510-30530
CD%		cifra decine	30560-30570-30600
CU%		cifra unità	30560-30580-30590-30620
DE$	8	tabella decine	30570-30000-30020
DU%		blocco decine-unità	30500-30540-30550-30560
ER%		simbolo di errore	80-30210-30240-30250-30260
I%		indice di ciclo	30010-30020-30200-
N		valore iniziale	20-30220

Nomi delle variabili	Dimensione	Descrizione	Riferimenti
NA		valore assoluto di N	30220-30230-30240-30250-30270-30495
NC%		numero centesimi	30440-30450-30460-30470
NK%		numero migliaia	30330-30340-30350-30360
NM%		numero milioni	30270-30280-30290-30300
NN%		variabile di trasmissione	30300-30360-30410-30470-30500
NR%		variabile di lavoro	30270-30330-30390-30440
NS		segno di N	30220-30260
NU%		numero unità	30390-30400-30410
TX$	4	stringa finale	40-50-60-70-90-30000-30200-30290-30310-30350-30370-30460-30490-30495-30520-30530-30550-30570-30600-30610-30620
UN$	19	tabella valori inf. 20	30000-30010-30530-30550-30620
XP		indice di vettore	30210-30290-30350-30370-30380-30430-30460-30490-30495-30520-30530-30550-30570-30600-30610-30620

Corrispondenza
tra blocchi funzionali ed istruzioni

(1) 30240/30260	Controlli su N
(2) 30230	Arrotondamento di N
(3) 30270/30320	Trattamento dei milioni
(4) 30330/30380	Trattamento migliaia
(5) 30390/30430	Trattamento unità
(6) 30440/30496	Trattamento centesimi
(7) 80	Segnalazione di errore
(8) 30500/30630	Sottoprogramma per il trattamento di numeri di tre cifre
(9) 30440	Calcolo numero di centesimi
(10) 30450	Controllo sui centesimi
(11) 30460 inizio	Controllo sui centesimi
(12) 30460 fine	Generazione scritta: "UN CENTESIMO"
(13) 30470/30480	Chiamata al sottoprogramma (8)
(14) 30490	Completamento della stringa
(15) 30495 inizio	Controllo sul valore di N
(16) 30495 fine	Completamento a sinistra della stringa
(17) 30270	Calcolo numero di milioni
(18) 30280	Controllo sui milioni
(19) 30290 inizio	Controllo sui milioni
(20) 30290 fine	Generazione della scritta: "UNMILIONE"
(21) 30300	Chiamata al sottoprogramma (8)
(22) 30310	Completamento stringa
(23) 30330	Calcolo numero di migliaia
(24) 30340	Controllo sulle migliaia
(25) 30350 inizio	Controllo sulle migliaia
(26) 30350 fine	Generazione della scritta: "MILLE"
(27) 30360	Chiamata al sottoprogramma (8)
(28) 30370	Completamento stringa
(29) 30390	Calcolo numero unità
(30) 30400	Controllo sulle unità
(31) 30410/30420	Chiamata al sottoprogramma (8)
(32) 30500	Scansione numero di tre cifre
(33) 30510	Controllo sulle centinaia
(34) 30520	Generazione della stringa "CENTO"
(35) 30530 inizio	Controllo sulle centinaia
(36) 30530 fine	Completamento a sinistra della stringa
(37) 30540/30630	Trattamento blocco decine-unità
(38) 30540	Controllo sul blocco decine-unità
(39) 30550 inizio	Controllo sul blocco decine-unità
(40) 30550 fine	Generazione stringa

(41) 30560	Scansione del blocco decine-unità
(42) 30570	Generazione della strin-ga corrispondente alle decine
(43) 30580/30590	Controllo sul valore delle unità
(44) 30600 inizio	Controllo sul valore delle unità
(45) 30600 fine	Completamento a destra della stringa
(46) 30610	Completamento a destra della stringa
(47) 30620 inizio	Controllo sulle unità
(48) 30620 fine	Completamento della stringa
(49) 30240 inizio	Controllo sulla validità del valore introdotto
(50) 30240 fine	Segnalazione di errore
(51) 30250 inizio	Controllo sulla validità del valore introdotto
(52) 30250 fine	Segnalazione di errore
(53) 30260 inizio	Controllo su N
(54) 30260 fine	Segnalazione caso anomalo
(55) 30270/30496	Trasformazione del numero in lettere
(56) 30230	Arrotondamento di N
(57) 30240/30270	Controllo su N
(58) 30270/30496	Trasformazione del numero in lettere
(59) 30240/30260	Segnalazione di errore
(60) 29990/30100	Inizializzazione
(61) 30200/30496	Trasformazione di un numero in lettere
(62) 50/90	Stampa dei risultati
(63) 30270 inizio	Divisione del numero per un milione
(64) 30270 parte centrale	Calcolo della parte intera del risultato precedente (numero milioni)
(65) 30270 fine	Memorizzazione della parte rimasta del valore iniziale

Problemi relativi all'analisi

Confrontando i diagrammi a blocchi con le istruzioni Basic, si può notare che alcuni test sono stati invertiti o suddivisi in più test consecutivi come ad esempio il blocco (43).

Un altro problema è quello relativo alla realizzazione della doppia alternativa (se ... allora ... altrimenti ...): in questo caso, poiché il THEN rappresenta solo il ramo positivo del test, il ramo negativo dovrà essere sviluppato disgiuntamente. In genere si sviluppa il ramo negativo nelle linee di programma che seguono immediatamente la riga in cui è formulato il test IF ... THEN ...; per evitare però che dopo la parte relativa al THEN, in caso di risposta affermativa al test, vengano eseguite anche le istruzioni del ramo negativo, si dovrà concludere la serie di operazioni relative al THEN con un GOTO che permetta di "saltare" le istruzioni relative al ramo negativo.

Questo tipo di realizzazione non complica in modo apprezzabile la codifica ma rende il programma di difficile lettura ed interpretazione.
Un linguaggio che permette di esprimere questa struttura e le altre da noi proposte in modo lineare è, come abbiamo già avuto occasione di notare, il Pascal.
Un'ultima osservazione: nel corso dell'analisi abbiamo trascurato il problema relativo all'inserimento nella stringa di eventuali spazi bianchi tra due parole successive, visto che ciò risulta implicito e naturale quando si scrive manualmente.

Individuazione del numero di milioni
Figura 26

Proprio per questo un programmatore alle prime armi si dimenticherà facilmente di tenere in considerazione anche cose di questo genere nel corso dell'analisi e si accorgerà di eventuali errori o dimenticanze solo al momento della messa a punto.

Nel nostro caso occorre prevedere uno spazio bianco tra il valore dei centesimi e la parola "CENTESIMI", tra la parte intera del numero e la congiunzione "E" ed infine tra quest'ultima e la parte decimale.

Problemi relativi alla codifica

La precisione richiesta sul numero N è molto alta: un centesimo su qualche decina o centinaia di milioni. Occorrono quindi ben 10 o 11 cifre significative. Solo raramente sui Personal Computers, lavorando in semplice precisione, è possibile ottenere una precisione così alta.

Non vi sono che due soluzioni: o ridurre il valore massimo accettabile o lavorare in doppia precisione.

Se si decide di lavorare in doppia precisione è necessario far seguire il nome delle variabili N, NA, NR dal carattere indicativo di questo tipo di variabile (in genere viene utilizzato il carattere #). In questo modo si potrebbero trattare anche valori superiori al miliardo!

Nella codifica da noi proposta sono state raggruppate in modo sistematico più istruzioni su una stessa linea: questa è solo un'abitudine e non una regola. Lo stesso programma infatti si può realizzare utilizzando una linea di programma per ogni istruzione. Neppure in questo caso però sarà possibile ottenere una corrispondenza biunivoca tra blocco logico ed istruzione. Alcune di queste infatti rappresenteranno sempre più di un blocco: si tratta delle istruzioni del tipo IF ... THEN ... che contengono sia il test che il ramo relativo alla risposta affermativa. Per ottenere una reale corrispondenza uno a uno bisognerebbe appesantire notevolmente il programma, facendo seguire ad ogni THEN un GOTO che rimandi ad un gruppo di istruzioni, od anche ad una sola istruzione, rappresentante il ramo positivo del test.

Il tipo di corrispondenza raggiunto nel programma da noi proposto è comunque abbastanza valido e dimostra a posteriori che il livello di dettaglio raggiunto nell'analisi é coerente con l'utilizzo del linguaggio Basic.

Nella codifica non è stata ricercata sistematicamente l'ottimizzazione nell'uso della memoria. Segnaliamo ad esempio che alcune variabili avrebbero potuto essere unificate (gli indici I% e XP) risparmiando così alcune locazioni di memoria. Con lo stesso criterio si poteva utilizzare per l'indice XP la forma intera XP%. Questo comunque non avrebbe permesso, come invece sembra a prima vista, di risparmiare spazio in memoria: infatti anche se effettivamente una variabile di tipo intero fa risparmiare da 2 a 4 bytes rispetto ad una variabile reale, è pur vero che per identificare una variabile intera occorre aggiungere al suo nome il carattere % ogni qualvolta la si usa all'interno del programma. Nel nostro caso caso la variabile XP viene utilizzata ben 46 volte e quindi utilizzando al suo posto la variabile intera XP% si avrebbe una perdita di circa 40 bytes. Per contro però l'uso di variabili intere permette un'elaborazione molto più veloce.

Il programma poi avrebbe potuto essere ottimizzato utilizzando delle istruzioni particolari della macchina a disposizione ma è evidente che in questo caso il programma che si sarebbe ottenuto non avrebbe avuto i caratteri di facile trasferibilità ricercati.

Osserviamo infine che, eliminando tutte le linee REM di separazione tra i vari moduli del programma e tutti gli spazi bianchi superflui nelle varie linee, si sarebbe risparmiato diverso spazio, a discapito però della leggibilità.

Conclusioni

Come il lettore avrà potuto notare, quando si realizza un programma si devono obbligatoriamente fare delle scelte. Alcune di queste dipendono dalla macchina che si ha a disposizione ma per la maggior parte dipendono solo dal programmatore. Queste scelte vengono fatte spesso arbitrariamente e non sempre in modo assennato, in quanto si tende a tener conto solo delle proprie abitudini.

Proprio per questo motivo è essenziale che un programmatore alle prime armi acquisisca delle corrette abitudini di lavoro!

CAPITOLO 7

IL GIOCO DEL 421

Il secondo esempio è dedicato al gioco del 421. La scelta di un gioco da un punto di vista didattico è utile per varie ragioni.

Poiché molte persone conoscono già il gioco in questione, questo facilita la comprensione del lavoro e la trattazione può quindi essere fatta in modo più scorrevole.

Inoltre, visto che esistono precise "regole di gioco", la definizione del problema è molto semplificata.

La parte più complessa è quella relativa alla scelta di una strategia vincente, cioé di una tattica di gioco che permetta di vincere sempre.

Comunque la scelta della strategia migliore viene messa a punto in fase di analisi. In genere questa strategia, che deve essere adottabile automaticamente, è spesso meno "brillante" di quella che verrebbe seguita da un giocatore umano ma in compenso, una volta messa a punto, è senza dubbio esente da quegli errori che in genere si fanno nel corso del gioco o per scarso impegno o per stanchezza.

Le regole del gioco

Prendiamo ora il foglio illustrativo del gioco 421 e studiamone le regole.

Il 421 si gioca con 3 dadi normali (cioé con le facce numerate da 1 a 6) e con due o più giocatori e si basa sulle possibili combinazioni che si possono ottenere con i tre dadi.

Per ottenere il risultato corretto da ogni "lancio" dei dadi si segue la seguente convenzione: i tre valori che si ottengono lanciando i tre dadi vengono ordinati in modo decrescente (dal più grande al più piccolo) o, quanto meno, in modo non crescente, nel caso in cui si abbiano più valori uguali: si genera in questo modo un numero di tre cifre che rappresenta il risultato ottenuto dal giocatore. Così, ad esempio, un lancio che abbia generato i tre valori 5, 6, 3, darà origine al numero 653.

Elenchiamo ora i valori delle possibili combinazioni suddivise in cinque livelli, con importanza decrescente, ordinando pure in modo decrescente le varie combinazioni all'interno di ogni livello:

1 - il *421*, dato da un 4, un 2 ed un 1 (l'uno d'ora in avanti verrà denominato "asso"), che dà il nome al gioco;

2 - *le cinque coppie d'assi*, con due assi ed un altro valore qualsiasi (611, 511, 411, 311, 211);

3 - *i sei "tris"*, che comportano "l'uscita" di tre valori uguali (da 666 a 111);

4 - *le quattro sequenze*, chiamate impropriamente "scale", e cioé 654, 543, 432, 321;

5 - *le altre quaranta combinazioni* possibili dal 665 al 221.

All'inizio del gioco ogni giocatore ha a disposizione lo stesso numero di gettoni; inoltre al centro del tavolo viene messa una pila di undici gettoni denominata "piatto".

Il gioco si svolge in due parti: il carico, in cui i giocatori prelevano gettoni dal piatto, e lo scarico, in cui il vincitore del giro cede dei gettoni agli altri.

Vince la partita chi per primo riesce a sbarazzarsi di tutti i gettoni.

Il carico

Durante questa fase il gioco si svolge in questo modo: iniziando dal giocatore che ha perso il giro precedente, ogni giocatore lancia i tre dadi. Chi ottiene la combinazione peggiore perde mentre chi ottiene la migliore vince.

Il perdente deve prelevare dal piatto un numero di gettoni che si determina in base al risultato ottenuto dal giocatore vincente ed in particolare:

1 - tutti i gettoni del piatto tranne uno se la combinazione vincente è 421;

2 - un numero di gettoni pari al valore del terzo dado nel caso in cui la combinazione vincente sia una coppia d'assi;

3 - tre gettoni nel caso in cui la combinazione vincente sia un "tris";

4 - due gettoni nel caso in cui la combinazione vincente sia una "scala";

5 - un gettone negli altri casi.

Se il piatto non contiene un numero sufficiente di gettoni, il giocatore perdente preleva solo quelli disponibili.

Lo scarico

Una volta vuotato il piatto si passa a questa seconda fase del gioco, in cui ogni giro si gioca nel modo seguente:

Inizia il giocatore che ha perso il giro precedente. Egli lancia una prima volta i tre dadi. Se non è soddisfatto del risultato ottenuto può rilanciare uno o due dadi. Si possono fare al massimo tre tentativi.
Gli altri giocatori non possono effettuare un numero di tentativi superiore a quello dei tentativi effettuati dal primo giocatore.

Quando due giocatori ottengono lo stesso punteggio, cioé si viene a creare una situazione di parità, la classifica tra i due viene determinata mediante uno spareggio, che consiste nel lancio dei tre dadi senza possibilità di ripetizione. Se necessario lo spareggio viene ripetuto finché il risultato ottenuto dai due giocatori permette di definire il vincitore.

Alla fine del giro, il vincitore, cioé il giocatore che ha ottenuto la combinazione migliore, cede al perdente, cioé al giocatore che ha ottenuto il risultato peggiore, un numero di gettoni pari a 7 se la combinazione vincente è 421, oppure calcolato secondo le regole esposte nel paragrafo precedente.
Il gioco termina quando uno dei giocatori resta senza gettoni: tale giocatore viene dichiarato vincitore. Ciascuno degli altri giocatori dovrà pagare al vincitore una somma proporzionale al numero di gettoni che gli sono rimasti.
Spiegate le regole del gioco, vediamo ora di ricapitolare la situazione. Sia che il gioco si svolga tra uomini sia che si giochi con un calcolatore, occorre:

avere a disposizione tre dadi, o qualcosa che li sostituisca: osserviamo a questo proposito che si può simulare il lancio di un dado mediante l'estrazione a sorte di un valore intero compreso tra 1 e 6; per avere il lancio di tre dadi occorre eseguire questa operazione tre volte ricordando in qualche modo i valori via via ottenuti;

far lanciare i dadi a tutti i giocatori assegnando loro dei turni di gioco: nel caso in cui sia un calcolatore a dover lanciare i dadi, il lancio potrà venir effettuato materialmente da un uomo che comunicherà poi il risultato alla macchina in modo che essa, nel caso dello scarico, possa scegliere la strategia da seguire (se rilanciare o no i dadi). Occorrerà inoltre mantenere anche la contabilità dei gettoni in possesso del calcolatore.

- Tutte le operazioni da effettuare, tranne che nella fase di scarico, sono governate da regole ben precise. Nello scarico invece il giocatore è messo di fronte ad una possibilità di scelta, seppur limitata dalle regole del gioco (i giocatori sono limitati nella scelta da ciò che viene fatto dal primo).

Se si vuol far giocare un calcolatore bisogna quindi insegnargli, oltre alle regole del gioco, anche una strategia.

Questa situazione, in informatica, è abbastanza tipica: l'enunciato del problema non precisa una condotta risolutiva. Ciò non accade solo per i giochi ma anche per problemi di tipo scientifico: ad esempio, per calcolare il valore di un integrale si possono utilizzare diversi metodi.
In questo caso non basterà essere dei buoni informatici per effettuare la scelta migliore, ma occorrerà avere anche una conoscenza approfondita del problema da trattare.

Torniamo comunque al nostro esempio. Per stabilire una strategia di gioco esaminiamo qualche caso concreto attinente alla fase di scarico. (Nella fase di carico tutto si svolge in modo automatico).

All'inizio, il giocatore effettua, secondo la regola, il lancio dei tre dadi. Esaminiamo ora le varie possibilità (dopo aver preso nota della combinazione ottenuta).

Supponiamo che la combinazione uscita sia 421. È evidente che in questo caso il giocatore non ha alcun interesse a rilanciare nuovamente i dadi in quanto ha già ottenuto il risultato migliore e quindi con un nuovo lancio potrebbe solo peggiorare la sua posizione.

- Supponiamo invece che la combinazione uscita sia 221. Essendo questa la peggior combinazione è ovvio che occorre rigiocare.

Il problema però si complica nel momento in cui si deve decidere cosa fare: dato che non è possibile rilanciare tutti e tre i dadi, converrà lanciarne di nuovo uno o due? E in un caso o nell'altro, quali?

Esaminiamo, nel caso specifico, ciò che può succedere.

Se si decide di lanciare nuovamente un solo dado si hanno due possibilità: mantenere i due dadi con valore 2 e lanciare quello con valore 1, oppure mantenere la combinazione parziale 21 e lanciare uno dei dadi con valore 2.

Nel caso in cui di rilancia il dado 1, si possono ottenere queste nuove combinazioni: 622, 522, 422, 322, 222, 221. Osserviamo che solo la combinazione 222 ha, in gettoni, un valore più alto di quello della combinazione precedente. Infatti questa combinazione ha un valore teorico di 3 gettoni mentre le altre, anche se per la maggior parte dei casi migliori della combinazione di partenza, hanno tutte il valore di 1 gettone. La combinazione 221 risulterebbe comunque perdente, mentre 422 e 322 avrebbero poche possibilità di risultare vincenti.

Nel caso in cui invece si rilancia un dado 2, si possono ottenere le seguenti combinazioni: 421, 211, 321, 621, 521, 221. Osserviamo che in questo caso due combinazioni hanno un valore superiore a tre gettoni e quindi si ha una probabilità su tre di ottenere un risultato migliore di quello che si potrebbe ottenere lanciando il dado 1. Inoltre 3 combinazioni hanno il valore di due gettoni, con una probabilità su due di ottenerne una. Resta invece una sola possibilità su sei di ottenere nuovamente la combinazione 221.

È evidente che questa seconda tattica risulta più conveniente, in quanto dà maggiori possibilità di ottenere combinazioni con un valore alto.

Vediamo ora cosa succede nel caso in cui invece si lanciano due dadi: si può conservare o il valore 1 o il valore 2.

Nel primo caso le combinazioni che si possono ottenere, elencate in ordine di valore decrescente, sono: 421, 611, 511, 411, 311, 211, 111, 321, 661, 651, 641, 631, 621, 551, 541, 531, 521, 441, 431, 331, 221.

Tenendo conto delle probabilità di uscita della varie combinazioni, possiamo concludere che il 421 ha 1 probabilità su 18 di uscire; inoltre si ha 1 probabilità su 3 di ottenere una combinazione che valga almeno come una coppia d'assi, 13 probabilità su 36 per una combinazione che valga almeno come un "tris", 5 probabilità su 12 per una combinazione che valga almeno coma una "scala" e 1 probabilità su 2 di ottenere un risultato non peggiore di 651. Si ha infine 1 sola probabilità su 36 di ottenere nuovamente il 221.

Esaminando nello stesso modo il caso in cui si conserva il valore 2, si può osservare che vi sono 2 probabilità su 9 (meno del 25%) di ottenere una combinazione che valga almeno 2 gettoni, mentre se si vuole raggiungere una probabilità del 50% di ottenere un risultato minimo, bisogna scendere fino al 621.

In conclusione si vede che se il primo tentativo ha dato come risultato 221 è meglio, se si tiene un solo dado conservare un asso, mentre se si vuol cambiare un solo dado rilanciare un 2.

La scelta tra le due soluzioni dipende molto dal temperamento del giocatore: infatti mantenendo 21 si ha una probabilità maggiore di ottenere valori molto alti, ma anche una maggior possibilità di ottenere risultati scadenti; conservando solo l'asso vincente si ha una probabilità più distribuita su tutta la gamma dei valori possibili.

Esaminando in questo modo tutte le 56 combinazioni che si possono ottenere al primo tentativo si può giungere a stabilire una corretta strategia.

Osserviamo inoltre che la scelta della strategia può dipendere anche dalla posizione che un giocatore occupa nel giro.
Chi gioca per primo, ad esempio, deve cercare di ottenere un risultato difficilmente superabile con lo stesso numero di tentativi da lui effettuati: la combinazione 653 può essere considerata una combinazione soddisfacente se ottenuta al primo tentativo, dato che esiste solo una probabilità su tre di ottenere con un solo tentativo un risultato migliore; mentre non si può dire la stessa cosa se quella combinazione viene raggiunta al terzo tentativo. Chi gioca per ultimo si deve proporre innanzitutto di non rimanere con la combinazione peggiore e se possibile di ottenere la combinazione vincente.

Continuando di questo passo nella ricerca della strategia migliore rischieremmo di far diventare questo libro un'opera sul calcolo delle probabilità e perderemmo di vista lo scopo principale che è invece quello di esporre un metodo di programmazione. Pertanto diamo solo le regole che compongono la strategia che verrà adottata nel seguito:

Strategia per il primo giocatore

se al primo lancio si ottiene come risultato almeno 651 non si richiede di rilanciare dei dadi; altrimenti

se nella combinazione ottenuta vi è un 6 si rilanciano gli altri due dadi; altrimenti

se nella combinazione vi è almeno un asso si rilanciano gli altri due dadi; altrimenti

se nella combinazione vi è almeno un 4 si rilanciano gli altri due dadi; altrimenti

se nella combinazione vi è almeno un 5 si rilanciano gli altri due dadi; altrimenti
se si è ottenuto almeno un 3 si rilanciano gli altri due dadi.

Rispetto al nuovo risultato ottenuto:

- se si è ottenuta una "scala" non si rilanciano più i dadi; se no
  se si sono ottenuti due 6 o un 6 e un 5 si rilancia il terzo dado solo se è 1 o 2; altrimenti
- se si è ottenuto almeno un 6 si rilanciano gli altri due dadi; altrimenti
- se si è ottenuto almeno un "asso" si rilanciano gli altri due dadi; se no
- se si è ottenuto almeno un 4 si rilanciano gli altri due dadi; altrimenti
- se si è ottenuto almeno un 5 si rilanciano gli altri due dadi; altrimenti
- se si è ottenuto almeno un 3 si rilanciano gli altri due dadi.

se dal lancio dei dadi si è ottenuto un risultato superiore ad almeno uno di quelli ottenuti dagli avversari:

se si è al primo tentativo si rilanciano i dadi seguendo i criteri sotto esposti, se il risultato ottenuto è inferiore a 621 mentre se si è già al secondo tentativo ci si può accontentare se si è ottenuto almeno 661.

Se si deve effettuare un nuovo lancio:

- se si ha un "asso" si rilanciano gli altri due dadi;
- se si ha un 4 si rilanciano gli altri due dadi
- se si ha un 6 si rilanciano gli altri due dadi

Se dopo il secondo lancio ci si ritrova all'ultimo posto si segue la strategia seguente.

Se dal lancio dei dadi si è ottenuta una combinazione peggiore di quella ottenuta dagli avversari che hanno già giocato:

se si è ottenuta una "scala" o qualcosa di meglio non si rilanciano i dadi; altrimenti
se si hanno due 6 o un 6 e un 5 si rilancia il terzo dado; altrimenti
se si ha un asso si rilanciano gli altri due dadi; altrimenti
se si ha un 4 si rilanciano gli altri due dadi; altrimenti
se si ha un 6 si rilanciano gli altri due dadi; altrimenti
se si ha un 5 si rilanciano gli altri due dadi; altrimenti
se si ha un 3 si rilanciano gli altri due dadi.

Se il nuovo risultato ottenuto non è più il peggiore ci si accontenta. Non si effettua il terzo tentativo anche nel caso in cui il risultato ottenuto sia superiore a 653.

Strategia per l'ultimo giocatore

La strategia per questo giocatore, che ha un sicuro vantaggio in quanto conosce già il risultato ottenuto da tutti gli avversari, è la seguente:

- se ha una coppia d'assi, risulta vincitore e nessuno degli avversari ha un risultato superiore o uguale a un "tris", rilancia il terzo dado se questo è inferiore a 3, altrimenti non gioca;

- se non risulta né perdente né vincente, (o se è vincente ex-aequo):

  se ha una coppia d'assi e il risultato vincente è pure una coppia di assi (e non il 421) e se inoltre l'ultimo in classifica non ha un risultato superiore o uguale ad un "tris" rilancia il terzo dado;

se ha un 6 e il risultato vincente è inferiore a 611 e se l'ultimo in classifica non ha 6, rilancia gli altri due dadi;

se ha due 6, ed il valore vincente è un "tris" e se l'ultimo in classifica non ha più di un 6, rilancia il terzo dado;

se ha un 6 od un 5 ed il risultato vincente non è superiore a 654 e se l'ultimo in classifica non ha entrambi i valori 6 e 5, rilancia il terzo dado;

se è ultimo ex-aequo:

se ha un risultato inferiore al "tris" e il dado con valore minore è 1 o 2, rilancia tale dado

se no segue le regole esposte per il caso successivo

se è l'ultimo in classifica:

se il giocatore penultimo in classifica (che è anche il vincitore nel caso in cui si gioca in due) ha:

- 421: rigioca conservando quei dadi che hanno uno dei valori 4, 2, 1; se nessuno dei dadi ha uno di questi valori e si hanno a disposizione due lanci si conserva solo il dado con valore più alto, altrimenti si rinuncia;
- una coppia d'assi o un "tris": rigioca conservando, a seconda di quello che ha, due assi, un asso, il dado con valore più alto;
- una "scala":
  se ha un asso lancia gli altri due dadi
  se ha due dadi con valori relativi ad una "scala" superiore a quella del penultimo in classifica e se vi sono almeno due possibilità di ottenere una "scala" superiore a quella ottenuta dal penultimo in classifica, rilancia il terzo dado
  altrimenti gioca come nel caso seguente;
- al più un 6:
  se ha un 6 e il penultimo in classifica ha un risultato non sueriore a 641, conserva il 6 e rilancia gli altri due dadi

  altrimenti:
  - se ha un asso, rilancia gli altri due dadi;
  - se ha un 6 rilancia gli altri due dadi;
  - se ha un 4 rilancia gli altri due dadi;
  - se ha un 5 rilancia gli altri due dadi;
  - se ha un 3 rilancia gli altri due dadi

Questa non è senza dubbio la migliore strategia ma permette di prevedere numerosi casi particolari e di avere così la possibilità di automatizzarla. Inoltre questa strategia permette anche di fornire un criterio per paragonare la validità di due strategie diverse. In fase di messa a punto suggerirò il metodo da seguire per effettuare questo confronto.

L'ANALISI

Abbiamo definito oramai tutto ciò che il programma deve eseguire. Tenendo conto delle regole di gioco si può stendere un primo diagramma a blocchi simile a quello proposto nella **figura 27**.

Il gioco del 421
Figura 27

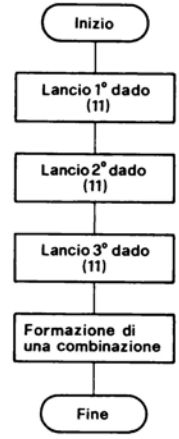
Fase di carico (2)
Figura 28

In questo caso non è contemplata la possibilità di effettuare direttamente una seconda partita; per ottenere questo risultato basterebbe aggiungere tra il blocco (3) e la FINE un test in modo che, a richiesta, ci sia un salto al punto (2). In questo diagramma il blocco (1) rappresenta la fase di Inizializzazione. In ogni programma occorre introdurre all'inizio i valori da assegnare ad alcune variabili, definire delle costanti, ... per cui è buona norma prevedere sistematicamente una fase di inizializzazione.

I blocchi (2) e (3) rappresentano le funzioni di carico e scarico nel nostro gioco.

La figura 28 illustra il diagramma a blocchi relativo alla fase del carico. Questa è molto semplice: ogni blocco rappresenta una regola del gioco; il blocco (4) rappresenta il fatto che ogni giocatore a turno può lanciare i dadi una sola volta; il blocco (5) serve per effettuare un bilancio dei risultati ottenuti dai vari giocatori e per assegnare al giocatore perdente il numero di gettoni che gli spetta; il blocco (6) permette, controllando il contenuto del piatto, di vedere se occorre effettuare un altro giro.

Lo scarico (3)
Figura 29



(12) Generazione di una combinazione
Figura 30

La figura 29 illustra il diagramma a blocchi relativo alla fase di scarico ed è molto simile, nella forma, al precedente.

Il blocco (7) serve per l'inizializzazione. Il blocco (8) descrive un giro: i giocatori, a turno, eseguono uno o più lanci (sia i giocatori umani che il calcolatore).

Il blocco (9) rappresenta l'operazione di calcolo e assegnazione dei gettoni al perdente da parte del vincente. Per questo all'uscita del blocco (8) devono essere conosciuti i punteggi relativi ai vari giocatori, al fine di poter stilare una classifica e sapere così chi è il giocatore vincente e chi il perdente. Nel blocco (9)

si sfrutteranno questi risultati, e il punteggio del vincitore verrà trasformato in numero di gettoni.

Il blocco (10) serve per capire quando è finito il gioco. In questo blocco si dovranno controllare i gettoni ancora in possesso dei vari giocatori e ripetere un altro giro nel caso in cui nessuno sia rimasto senza gettoni.

A questo punto il nostro problema può essere correttamente rappresentato dall'insieme dei blocchi (1) e dal (4) al (10). Non siamo però giunti ad un sufficiente livello di scomposizione. Conviene quindi suddividere questi blocchi in altri ancora più elementari.

Prendiamo in esame due funzioni che ricorrono sovente nel nostro programma: il lancio dei dadi (blocchi (4) e (8)) e la stima del numero di gettoni da assegnare al perdente (blocchi (5) e (9)). Conoscendo le regole del 421, queste due funzioni risultano elementari per un essere umano, ma non per il calcolatore. Utilizzando quest'ultimo converrà, prima di analizzare le combinazioni, disporre in ordine non crescente i valori ottenuti con i 3 dadi ed assegnare un valore in gettoni ad ogni combinazione.

Dato che queste funzioni sono utilizzate più volte nel programma, conviene creare dei sottoprogrammi che le svolgano e che possano essere richiamati da punti diversi del programma principale.

Il primo sottoprogramma è quello di simulazione del lancio di un solo dado: si tratta in pratica di generare un numero casuale intero compreso tra 1 e 6 (limiti inclusi). Tale sottoprogramma verrà individuato mediante il blocco (11).

Il secondo sottoprogramma è quello di generazione di un'intera combinazione: si tratta in pratica di generare i tre valori relativi al lancio di 3 dadi e di ordinarli secondo le regole del gioco (se escono successivamente i valori 3, 6, 5 la combinazione risultante sarà 653). Il diagramma a blocchi relativo a questo sottoprogramma è illustrato nella **figura 30**. Non abbiamo contrassegnato questi blocchi in quanto supponiamo che non sia necessario, data la loro semplicità, dettagliarli ulteriormente.

Resta ora da eseguire, mediante un programma, il confronto tra le varie combinazioni ottenute. Si possono classificare, assegnando loro un numero d'ordine, oppure confrontarle a due a due per giungere così, passo dopo passo, alla classifica finale.

In ogni caso ci si trova di fronte ad un primo problema: come confrontare tra loro due combinazioni?

Esponiamo ora varie soluzioni:

a) *Il confronto mediante un programma:*

a.1 : verificare che le due combinazioni siano diverse;
a.2 : se una di esse è 421 è senz'altro vincente;
a.3 : se una di esse contiene due 1 (ma non tre!) e la più forte. Se entrambe contengono due 1, la vincente è determinata dal terzo valore;

e così via per i "tris", le "scale", le altre combinazioni possibili confrontando tra loro due numeri o due stringhe di caratteri (621 > 532 oppure "621" > "532"). (Ricordiamo che non abbiamo ancora deciso il modo in cui rappresentare le nostre combinazioni e per questo ne abbiamo accennati almeno due).
Queste osservazioni ci permettono di realizzare un programma che rappresenta una possibile soluzione al nostro problema.

b) *Utilizzo di un elenco di combinazioni.*

Nel gioco del 421 le combinazioni possibili sono solo 56 e pertanto risulta abbastanza semplice costruire una lista di tali combinazioni disposte in ordine decrescente dal 421 al 221. Quando si devono confrontare due combinazioni, dopo aver verificato che sono distinte, basta confrontarle entrambe con tutti gli elementi dell'elenco, iniziando dal valore più elevato. La prima combinazione che si incontra è chiaramente la vincente. Con questo metodo si può facilmente stabilire anche il corrispondente valore in gettoni, ricavandolo dal posto occupato dalla combinazione nell'elenco: il primo elemento, 421, vale 7 gettoni in fase di scarico oppure il numero di gettoni presenti nel piatto meno uno in fase di carico. Il secondo vale 6 gettoni, e così via. A partire dal diciassettesimo posto tutte le combinazioni valgono 1 gettone.
Anche questa soluzione può essere realizzata mediante un programma.

c) *Uso di un elenco ordinato in modo decrescente e di un secondo elenco collegato a questo.*

Tenendo conto delle regole del 421, è possibile ordinare le 56 combinazioni in modo decrescente piuttosto che per valore, partendo cioé dalla 666 fino alla 111 (666, 665, 664, ..., 221, 211, 111). Utilizzando il metodo di ricerca dicotomica con al più 6 confronti si può trovare la combinazione cercata.

Nella ricerca dicotomica, l'elemento da identificare viene subito confrontato con quello che occupa il posto a metà elenco. Se da questo confronto risulta, ad esempio, che il valore ricercato è superiore a quello contenuto nella posizione intermedia della tabella, allora tale valore verrà confrontato con quello che occupa la posizione a metà della prima parte dell'elenco. Questo nuovo

paragone indicherà se il valore richiesto si trova nel primo o nel secondo quarto dell'elenco. Si nota subito che ad ogni confronto la dimensione del campo di ricerca viene dimezzata. La ricerca termina nel momento in cui da un confronto si ottiene un'uguaglianza. Questo metodo può essere usato oltre che per la ricerca, anche per inserire un nuovo elemento nell'elenco stesso, secondo un ordinamento prefissato.
Individuata così la posizione occupata nell'elenco dalla nostra combinazione, dobbiamo ottenere la sua posizione nella scala di valori del gioco (come è stato fatto ad esempio nel paragrafo b) con 1 per il 421 e 56 per il 221 e trasformare poi quest'ultimo nel corrispondente numero di gettoni. Occorre a questo proposito osservare che la scala di valori in gettoni non segue esattamente la scala di valori delle combinazioni: ad esempio a 211 corrispondono 2 gettoni mentre a 111, che nella scala di valori di gioco è inferiore, ne corrispondono 3. Per questo si potrebbero costruire due elenchi distinti, associati a quello iniziale: il primo contenente per ogni combinazione la sua posizione "gerarchica" ed il secondo contenente invece i corrispondenti valori in gettoni. Prendiamo ad esempio la combinazione 666 che occupa il primo posto nell'elenco iniziale: la sua posizione "gerarchica", essendo superata da 421 e dalle 5 coppie d'assi, è 7 ed il suo valore in gettoni, essendo un "tris", è 3. Pertanto, nel primo posto dell'elenco contenente le posizioni "gerarchiche" verrà posto 7 ed al primo posto dell'altro elenco, contenente i valori in gettoni, 3. Al secondo posto, corrispondente al valore 665, si avranno, rispettivamente, 17 e 1.
Questi nuovi elenchi si potrebbero unificare ottenendo dai due rispettivi valori un nuovo numero che li esprima entrambi: ad esempio moltiplicando uno dei due per 100 o 10 e sommando al risultato ottenuto l'altro valore. Dato che in genere si consulta più frequentemente il valore "gerarchico", che può assumere tra l'altro valori superiori a 10, converrà moltiplicare per 10 quest'ultimo valore, sommando poi al risultato ottenuto il valore dei gettoni, che è sempre compreso tra 1 e 7.

Definiti questi due elenchi, si può tradurre anche questo metodo in un programma.

d) *Sostituzione dell'elenco in modo decrescente con un algoritmo.*

Facendo riferimento al metodo esposto al punto c), si può notare che l'insieme delle combinazioni: 666, 665, ..., 661, 655, 654, ..., 651, 644, 643, ..., 641, 633, ..., 611, 555, ..., 111 è piuttosto regolare e permette di ricavare la posizione di una combinazione tramite il valore delle sue cifre. Infatti data una combinazione IJK dove I, J, K, rappresentano le singole cifre (con la solita condizione I >= J >= K) si può verificare che l'indice di posizione all'interno dell'elenco è dato dalla formula:

$$F(I, J, K) = 56 - K - J\,(J - 1) / 2 - (I - 1)\,I\,(I + 1) / 6$$

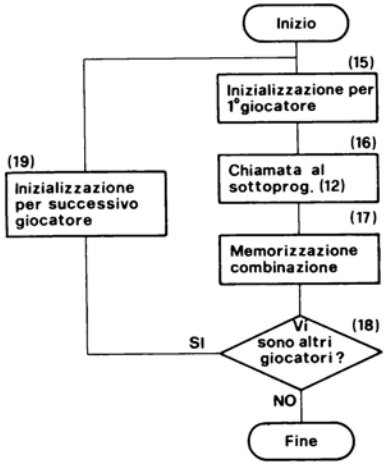
con la convenzione di assegnare indice 0 al primo elemento (I = 6, J = 6, K = 6) e 55 all'ultimo (I = 1, J = 1, K = 1).
Il lettore interessato può divertirsi ad estrarre questa formula utilizzando strutture a triangolo o a piramide, oppure procedendo per tentativi; esso dovrà tener conto del fatto che la funzione F deve essere di primo grado in K, di secondo in J e di terzo in I.

Avendo così eliminato l'elenco ordinato in modo decrescente, ci rimane un unico elenco di 56 valori interi, che può essere realizzato tramite un vettore, con indice variabile tra 0 e 55, i cui elementi, come abbiamo già visto, sono rispettivamente: 73, 171, 181, ... Per inizializzare questo vettore si potranno utilizzare le istruzioni di lettura da un file interno READ e DATA (bisogna però fare molta attenzione a non commettere errori!).
Abbiamo così trovato ben quattro soluzioni differenti per il nostro problema, realizzabili con difficoltà minima. Per la mia trattazione ho preferito, arbitrariamente, l'ultima soluzione. Comunque non bisogna effettuare queste scelte tenendo conto solo della velocità di esecuzione oppure dello spazio occupato in memoria, in quanto i programmi scritti tenendo conto stupidamente solo di queste esigenze perdono poi molto della loro validità nell'applicazione pratica.
Per concludere, conoscendo due combinazioni IJK e I' J' K', la classifica può essere ottenuta direttamente dal confronto tra i valori contenuti nel vettore sopra citato alle posizioni date dai valori F (I, J, K) e F (I', J', K') dell'indice. Inoltre, se la combinazione corrispondente al numero minore risulta vincente, il suo corrispondente in gettoni si può facilmente ottenere come resto della divisione del numero stesso (contenuto nel vettore) per 10.

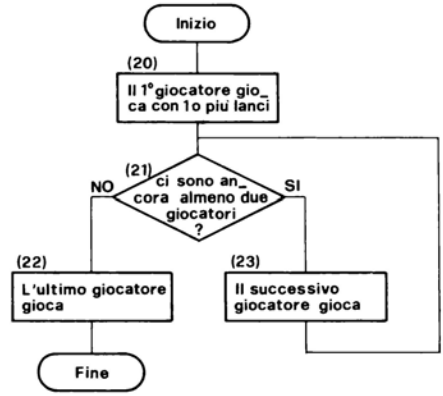
Nota: Se anziché moltiplicare il valore "gerarchico" per 10 lo avessimo moltiplicato per 16 prima di aggiungere il numero dei gettoni, ottenendo ovviamente valori finali che non alterano l'ordine, avremmo potuto avere (sul TRS 80 e sul PET) direttamente il valore in gettoni mediante l'operazione logica AND. Infatti detta NDE la variabile in cui è stato inserito il numero così ottenuto, relativo ad una combinazione, l'istruzione NJ = NDE AND 15 dà come risultato in NJ la forma binaria del valore iniziale in cui i 4 bit più significativi hanno assunto valore 0, mentre i 4 rimanenti sono rimasti invariati. Si è ottenuto così un valore compreso tra 0 e 15, che rappresenta il resto della divisione per 16. Si consiglia, se possibile, di utilizzare questa soluzione in quanto l'istruzione AND viene eseguita molto velocemente ed è più facile da codificare che non quella relativa al calcolo del resto di una divisione.

Abbiamo così esplicitato i sottoprogrammi per l'ordinamento delle combinazioni (13) e per il calcolo relativo al numero di gettoni (14). (Il numero di gettoni relativo alla combinazione 421 viene considerato 7. Al momento del "carico" per ottenere il numero di gettoni corretto, cioé il numero di gettoni del piatto meno 1, occorrerà applicare un metodo particolare).

Abbiamo ormai a disposizione tutti gli elementi necessari ad analizzare a fondo i blocchi (4) e (8).
Mediante le funzioni del blocco (4), ogni giocatore ottiene una combinazione, che si ricava tramite il sottoprogramma rappresentato in figura 30; infatti esso serve per generare tre numeri casuali e trasformarli in una combinazione del gioco. Per utilizzare questo sottoprogramma, occorrerà prevedere dei blocchi di inizializzazione (15) e (19) e di recupero dei risultati (17).
La figura 31 illustra il diagramma a blocchi relativo alla fase di carico. Il ciclo termina quanto tutti i giocatori hanno lanciato i dadi.
Dettaglio pratico da osservare: conviene inserire nel blocco (16) un ciclo di ritardo per aspettare che il giocatore umano faccia qualcosa ed avere così l'impressione che i lanci vengano effettuati realmente (in realtà il giocatore umano non deve fare nulla).



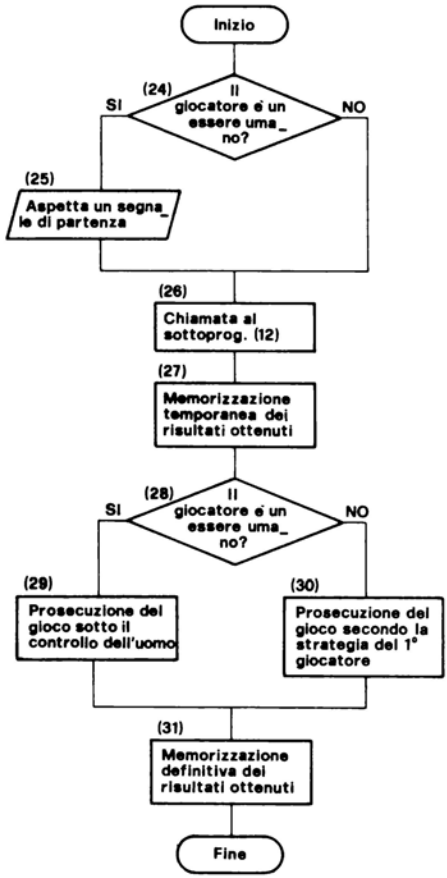
Ogni giocatore lancia una sola volta i dadi (4)
Figura 31



Ogni giocatore gioca al suo turno con uno o piu' tentativi (8)
Figura 32

La figura 32 illustra il diagramma a blocchi relativo alla scomposizione del blocco (8) che è composto da diversi blocchi rappresentanti le varie strategie per i singoli giocatori: come abbiamo visto infatti il primo giocatore tende a fare il minor numero di lanci possibile, specie se ha ottenuto un risultato discreto. L'ultimo giocatore invece deve cercare di ottenere, anche con un maggior numero di lanci, un risultato minimo. Infine i rimanenti giocatori devono seguire una strategia ancora differente.

Inoltre, per eseguire le funzioni rappresentate nei blocchi (20), (22) e (23) il calcolatore deve "cedere la mano" all'uomo, in quanto spetta a quest'ultimo la decisione sulla strategia da seguire.



Il 1° giocatore gioca (20)
Figura 33

Vogliamo a questo punto fare un' osservazione sulla posizione del blocco (21) nel diagramma e sul suo contenuto. Questo test è stato necessariamente inserito dopo la parte riguardante il primo giocatore perché, quando ci sono solo due giocatori, occorre avere la possibilità di passare direttamente dal blocco (20) al blocco (22). (Quindi in casi come questo, anche conoscendo a

priori il numero dei giocatori, non conviene utilizzare per la funzione descritta nel blocco (8) un ciclo FOR, per il Basic, o DO, per il Fortran, in quanto le operazioni del ciclo vengono eseguite almeno una volta).

Esaminiamo ora le funzioni (20), (22), (23) che sono molto simili tra loro.

La funzione rappresentata nel blocco (20) è stata scomposta in una serie di operazioni più semplici e ha dato origine al diagramma a blocchi della **figura 33**.

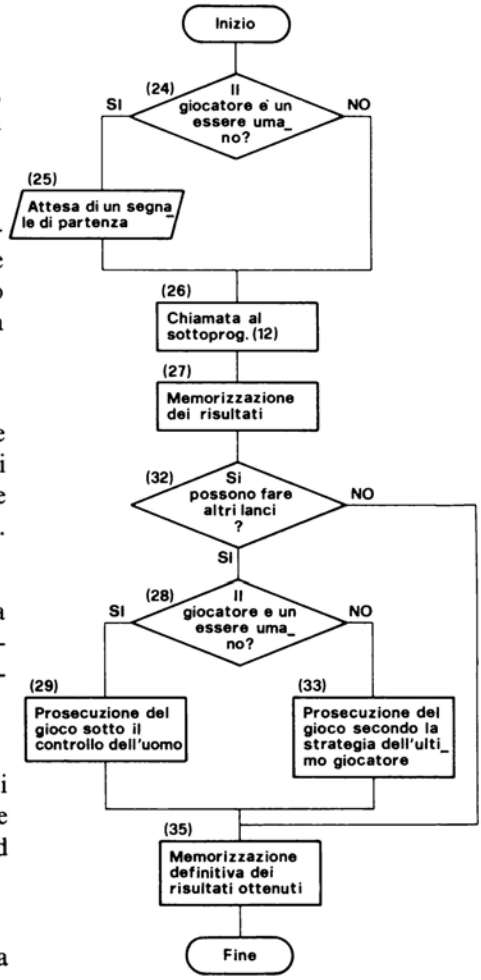
In tale diagramma, i blocchi (24) e (25) servono per generare un ciclo di ritardo, definito mediante un segnale dato dal giocatore in carne ed ossa.

Il blocco (26) serve per ottenere la combinazione relativa al primo tentativo ed il blocco (27) per classificare tale combinazione.

I blocchi (28) e (29) permettono di chiedere al giocatore umano, tramite video e tastiera, cosa vuole fare, ed esaminare la risposta.

Il blocco (30) rappresenta la strategia adottata automaticamente dal calcolatore in questo caso.

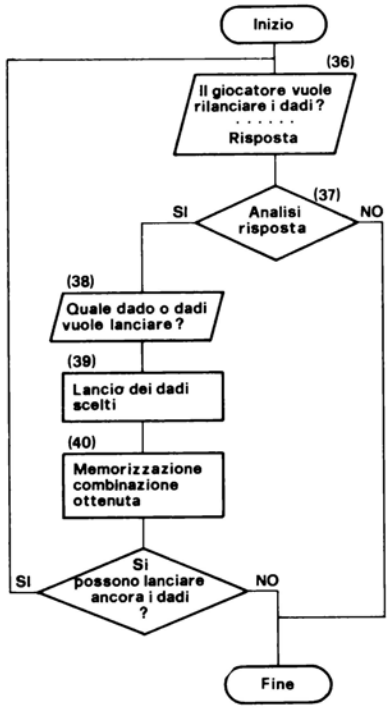
L'operazione successiva consiste nel memorizzare e classificare la combinazione definitiva ottenuta prima di passare al giocatore successivo.



L'ultimo giocatore gioca (22)
Figura 34

Lo sviluppo delle funzioni rappresentate nei blocchi (22) e (23) differisce da quello sopra esposto nei seguenti due punti:
- il blocco (30) viene sostituito, in entrambi i casi, da quello rappresentante la strategia appropriata: blocchi (33) e (34);
- poiché, dopo il turno del primo giocatore, il numero di lanci permessi non è fisso, occorre inserire un controllo (32) che permetta di saltare la parte relativa ai tentativi successivi nel caso in cui questi non possono essere effettuati.

Il diagramma a blocchi presentato in **figura 34** rappresenta lo sviluppo del gioco dell'ultimo giocatore. Nel caso di un giocatore intermedio, basta sostituire il blocco (33) con il blocco (34).



Gioco sotto il controllo umano (29)
Figura 35

Vista la similarità di questo caso con il precedente, non presentiamo il relativo diagramma a blocchi.
Passiamo invece ad esaminare l'interazione tra il calcolatore ed un giocatore in carne ed ossa. (blocchi (16), (25), (29)).

La funzione espressa dal blocco (16) può essere scomposta nelle tre funzioni più semplici (24), (25), (26). Pertanto l'operatore umano interviene solo nell'esecuzione dei blocchi (25) e (29).
Nel caso del blocco (25) viene dato un segnale di partenza premendo sulla tastiera un tasto convenzionale: ad esempio la barra spaziatrice.
Si possono dare significati particolari alla battitura di certi tasti: se viene battuto il carattere R, visualizza ...
Nel caso del blocco (29) invece, verrà chiesto al giocatore umano se vuole lanciare nuovamente i dadi ed in caso di risposta affermativa, quali di essi devono essere rilanciati. In base a queste informazioni verrà poi generata la nuova combinazione.

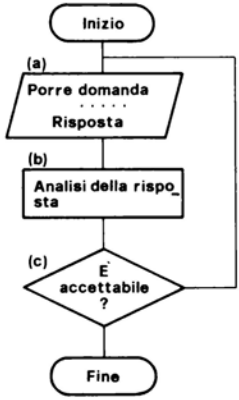


Figura 36

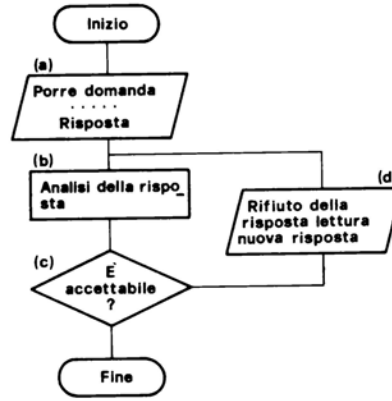


Figura 37

Tutto quanto sopra detto viene chiaramente illustrato nella **figura 35**.
I blocchi dal (36) al (41) rappresentano funzioni così semplici che ogni ulteriore commento appare superfluo.

Nota: il diagramma a blocchi della figura 35 non può essere formato seguendo le regole di scomposizione esposte in precedenza (figure dalla 8 alla 13) e pertanto non può essere codificato direttamente in PASCAL (senza istruzione GOTO). Per arrivare ad una struttura "pura" si potrebbe creare un blocco, nel ramo relativo alla risposta NO alla domanda (36), indicante che il lancio effettuato è l'ultimo possibile. Riunendo i due rami SI e NO del test (37) prima del blocco (41) risolveremmo il nostro problema.

Ma, dato che noi intendiamo effettuare la codifica in Basic, possiamo utilizzare il diagramma a blocchi così come è stato presentato.

Soffermiamoci ora più a lungo sui blocchi (36) e (37) da una parte e (38) e (39) dall'altra.

In ognuna di queste coppie di blocchi viene posta all'uomo una domanda sul video e questi risponde battendo delle parole sulla tastiera.

Bisogna prevedere che l'uomo nel far ciò può commettere degli errori. In questo caso occorrerà che il calcolatore, constatata la presenza dell'errore, riproponga la domanda.

Lo schema proposto nella **figura 36** permette di riformulare la domanda iniziale quando la risposta ottenuta risulta non accettabile: ad esempio quando il giocatore chiede, avendo ottenuto la combinazione 653, di lanciare di nuovo il dado con valore 4.

Esiste anche un modo più sofisticato per risolvere il problema (vedi **figura 37**) e cioé quello di prevedere una nuova richiesta da parte del calcolatore (d). Sviluppando ulteriormente questo caso si possono addirittura prevedere nel blocco (d) diversi messaggi per i vari tipi di errore.

Più in generale si può dire che devono essere previsti controlli di questo genere ogni qualvolta intervenga direttamente l'uomo.

Nel nostro caso bisogna prevedere uno sviluppo di questo tipo per i blocchi (36) e (38) da cui si esce solo in caso di risposta accettabile.

All'interno dei blocchi (36) e (38) inizia pertanto l'analisi della risposta che trova poi la sua logica conclusione nei blocchi successivi.

Il caso più complesso è quello rappresentato dai blocchi (38) e (39): bisogna infatti rilanciare i dadi seguendo le direttive date dall'uomo ed eventualmente effettuare più lanci.

La figura 38 illustra lo sviluppo dettagliato del blocco (38). Il livello di precisione in questo diagramma è molto elevato e non occorrerà perciò scomporlo ulteriormente. Per questo avremmo potuto evitare di contrassegnare anche questi blocchi con i numeri di riferimento, riportandone solo l'elenco. Tuttavia abbiamo preferito mantenerli per facilitare il commento.

Con i blocchi da (42) a (46) si chiede al giocatore umano quale è il primo dado che vuole rigiocare: questo risponde, ad esempio indicandone il valore. Biso-

gna quindi analizzare la risposta per accertarne la validità e concludere catalogando il dado prescelto.

Come vengono svolte queste funzioni? Lo vedremo al momento della codifica.

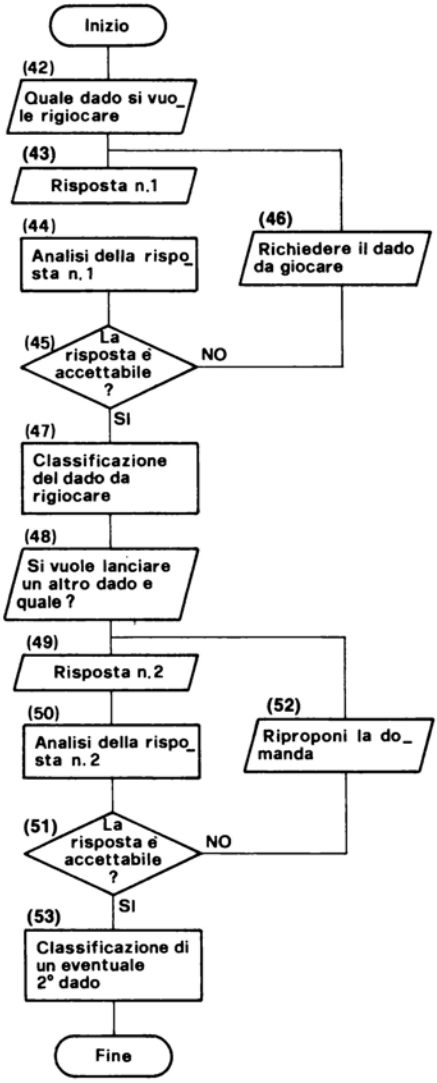


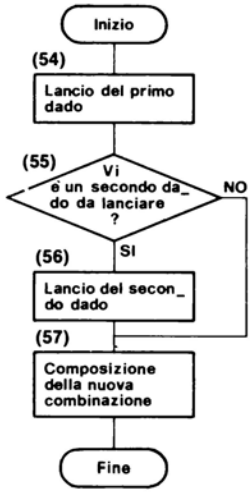
Figura 38

38 Quali dadi vuoi rigiocare

Per la scelta del secondo dado le cose si svolgeranno in modo del tutto analogo; va solo osservato che in questo caso il giocatore potrebbe non avere più altri lanci da effettuare. La risposta sarà pertanto accettata sia se indicherà il valore

di uno dei due dadi rimasti sia se darà l'indicazione che non ci sono dadi da rigiocare, ad esempio scrivendo il valore 0 per il nuovo dado. Infatti non esistono dadi con queto valore ed inoltre si è soliti collegare il valore 0 ad una risposta negativa.

Poiché in questo caso sarà accettabile quindi anche il valore 0 abbiamo dovuto contrassegnare in modo diverso i vari blocchi che eseguono la funzione di controllo. Nei due casi i blocchi (44) o (50).



(39) Lancio dei dadi prescelti

Figura 39

I blocchi (47) e (53) prevedono una classificazione dei dadi da rigiocare. In pratica i valori dei dadi interessati vengono contrassegnati nella combinazione e trattati a parte: entreremo nel dettaglio di queste operazioni in fase di codifica. Comunque in un modo o nell'altro bisognerà distinguere i dadi che verranno rigiocati da quelli che invece non lo saranno e che verranno poi utilizzati per comporre la nuova combinazione.

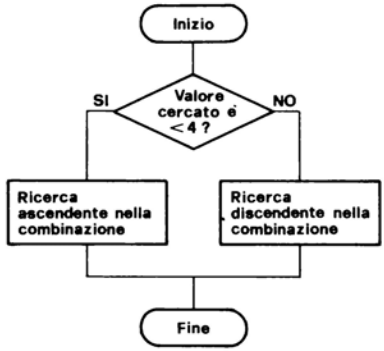
La figura 39 illustra il diagramma a blocchi relativo alla scomposizione del blocco (39). Anche in questo caso il livello di dettaglio raggiunto è più che sufficiente. I blocchi (54) e (56) rappresentano il lancio di un dado e fanno riferimento al sottoprogramma (11). Il test del blocco (55) permette di saltare la fase (56) nel caso in cui si voglia rigiocare un solo dado. Il blocco (57) rappresenta la nuova formulazione della combinazione ottenuta unendo il valore dei dadi non rigiocati al nuovo valore assunto dai dadi rigiocati.

Abbiamo così completamente dettagliato il blocco (29) rappresentante lo svolgimento del gioco sotto controllo del giocatore umano, il quale può

decidere arbitrariamente la strategia da seguire.
Bisogna ora esaminare i blocchi (30), (33) e (34), che descrivono un lavoro simile a quello fatto dall'uomo, in cui però tutto avviene in modo automatico. Occorre quindi scrivere un programma che permetta di automatizzare le varie strategie.

Esaminandole una per una si nota che per poterle seguire bisogna essere capaci di trovare:

- se la combinazione ottenuta con il lancio effettuato è superiore ad un certo valore: sappiamo che questa verifica sarà facile, poiché per ogni combinazione



Trovare se esiste almeno un dado con un valore dato

Figura 40

è stata memorizzata in una opportuna locazione di memoria la posizione nella "scala gerarchica" (più è basso il numero rappresentante la posizione all'interno della scala più è alto il valore della combinazione corrispondente);
- se la combinazione contiene una certa cifra: per effettuare questa verifica occorre creare un sottoprogramma capace di confrontare le varie cifre che compongono la combinazione con il valore richiesto.

Una prima soluzione a cui si può pensare è quella che fa riferimento al modo in cui vengono memorizzate le combinazioni. Se si memorizzano i valori ottenuti dal lancio dei tre dadi già sotto forma di combinazione, cioé ordinati in modo non decrescente, si può operare nel modo seguente: se si vuole verificare la presenza del valore 1, è logico esaminare l'ultima cifra della combinazione poiché se questa è maggiore di 1 è evidente che la combinazione in esame non contiene nessuna cifra uguale a 1.
Generalizzando questo ragionamento, possiamo pensare di verificare la pre-

senza di valori alti iniziando l'esame dalla prima cifra della combinazione, mentre per la ricerca di valori bassi si inizierà dall'ultima cifra.

La figura 40 illustra il diagramma a blocchi che realizza questo tipo di comportamento.

La figura 41 mostra poi il diagramma a blocchi dettagliato relativo alla ricerca di tipo ascendente di un valore inferiore a 4.
(ramo a sinistra del diagramma a blocchi della figura 40).

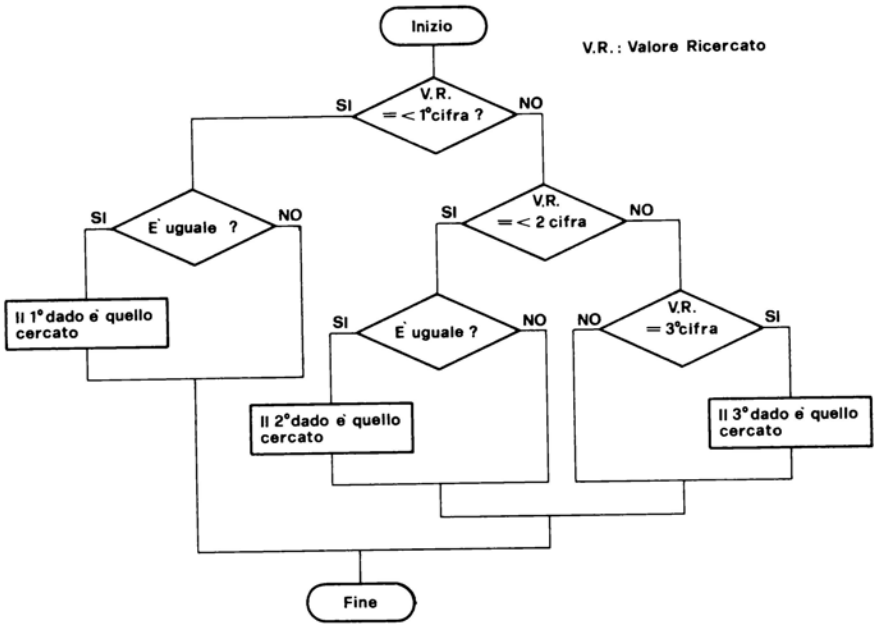


Figura 41 Ricerca ascendente di un valore nella combinazione

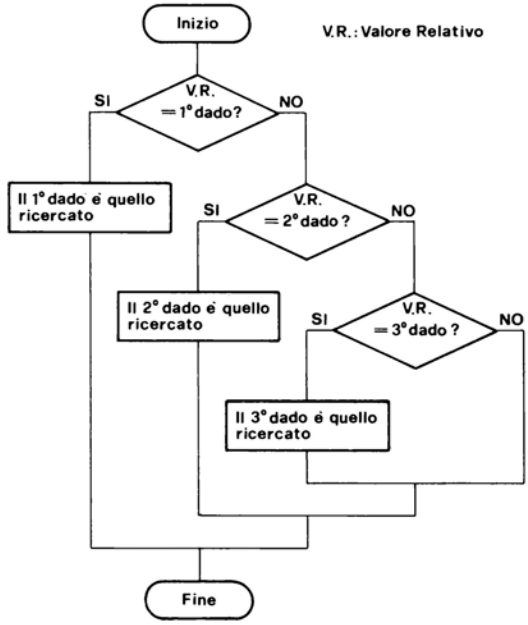
La figura 42 invece illustra il metodo generale di ricerca valido per qualsiasi valore. Dato il limitato numero di confronti (solo 3) questo metodo appare più economico del precedente.

Utilizzeremo la soluzione illustrata nella figura 42 per generare il sottoprogramma per la ricerca di un certo valore nella combinazione. Indicheremo tale sottoprogramma con la notazione (58).

Bisogna infine essere capaci di verificare se la combinazione contiene una coppia di valori dati. Questo problema è molto simile a quello appena trattato.

Proviamo a seguire un ragionamento analogo al precedente: bisogna innanzitutto esaminare il valore del primo dado e vedere se è uguale a uno dei due valori ricercati; in caso di risposta affermativa si continuerà solo la ricerca del valore rimasto, mentre in caso negativo occorrerà proseguire la ricerca per entrambi. Questo metodo è descritto nel diagramma a blocchi della **figura 43**.

Si potrebbe semplificare questo diagramma sostituendo tutta la parte di destra con due tests: uno che confronti VR1 con il secondo dado e VR2 con il terzo e l'altro che confronti VR2 con il secondo dado e VR1 con il terzo. Possiamo comunque osservare che ogni test deve essere ripetuto due volte: questo perché non si conosce a priori l'ordine dei valori ricercati.
Ora, poiché si sa che i valori dei dadi che compongono la combinazione sono ordinati in modo decrescente, si potrebbero ordinare nello stesso modo anche i due valori da ricercare.

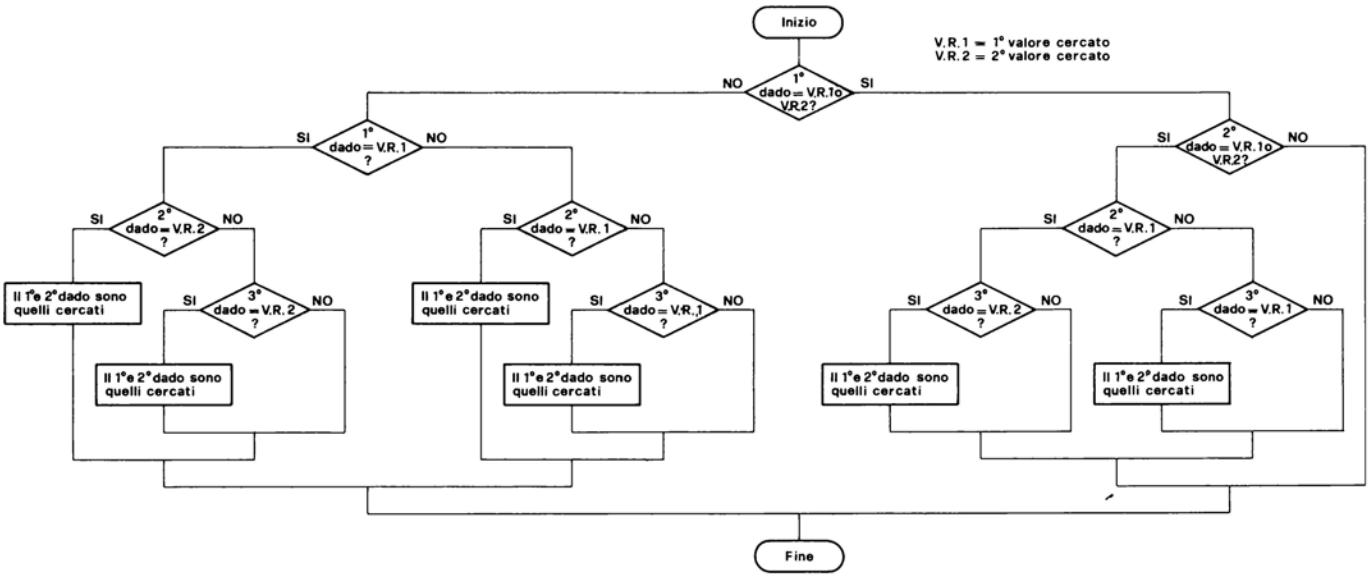


Trovare, se esiste, un dado avente un valore dato (58)

Figura 42

Con questo accorgimento la verifica si può fare con soli tre tests, come illustrato nella **figura 44.**

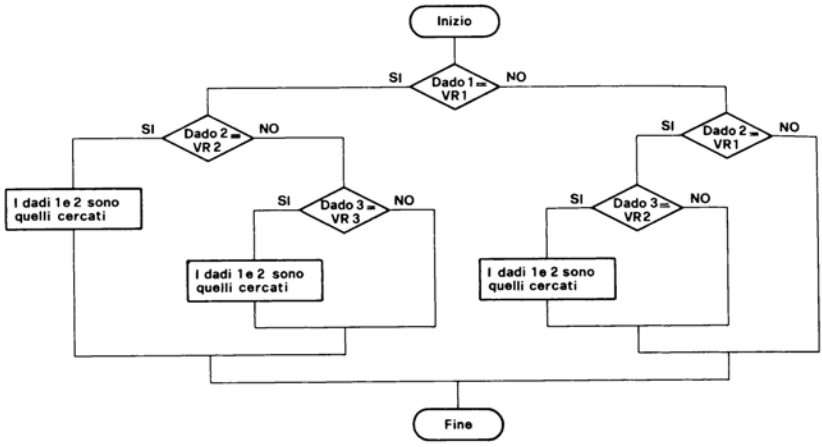
Il metodo che si basa sull'ordinamento preventivo dei valori, mentre rendeva più pesante il programma per la ricerca di un solo valore nella combinazione, diviene invece più conveniente nel caso in cui i valori da ricercare sono due.



Trovare, se esistono, due valori assegnati nella combinazione

Figura 43

Facciamo ora alcune considerazioni sul diagramma a blocchi della figura 44: il valore VR1 deve essere ricercato solo tra i primi due dadi. Infatti VR2, che per ipotesi è minore di VR1, può comparire nella combinazione solo in una posizione successiva a quella occupata da VR1. (Se si inizia il controllo del valore più alto della combinazione, questa osservazione rimane valida anche



(59) Trovare, se esistono, due valori assegnati in una combinazione

Figura 44

nel caso in cui VR1 = VR2). Quindi se il controllo sui primi due valori della combinazine ha dato esito negativo è inutile andare ad esaminare anche il terzo valore. Le tre soluzioni possibili per il nostro problema sono infatti: DA1=VR1 e DA2=VR2; oppure DA1=VR1 e DA3=VR2; o ancora DA2=VR1 e DA3=VR2. Si potrebbe pertanto risolvere il problema effettuando solo i tests relativi a questi tre casi.

Esiste un'altra soluzione: esaminare l'elenco delle combinazioni per valori decrescenti di cui abbiamo parlato spesso. Notiamo ad esempio che la coppia 65 compare in 654 (13ª posizione), in 665 (17ª posizione), e nella serie da 655 a 651 (dal 22° al 27° posto). Questo metodo risulta efficace solo quando i valori ricercati sono alti.

Non conviene comunque decidere ora quale soluzione adottare: la scelta definitiva verrà fatta solo al momento della codifica.

Tutto questo ragionamento ci mostra da una parte l'importanza di trovare più soluzioni alternative e dall'altra l'impossibilità di capire a priori quale sia la soluzione migliore.

Abbiamo ormai tutti gli elementi per stendere i diagrammi a blocchi relativi alle varie strategie.

Per scomporre i blocchi (30), (33) e (34) sarà sufficiente seguire le regole indicate all'inizio del capitolo.
La figura 45 illustra lo schema generale da seguire per scomporre i blocchi (30), (33) e (34).

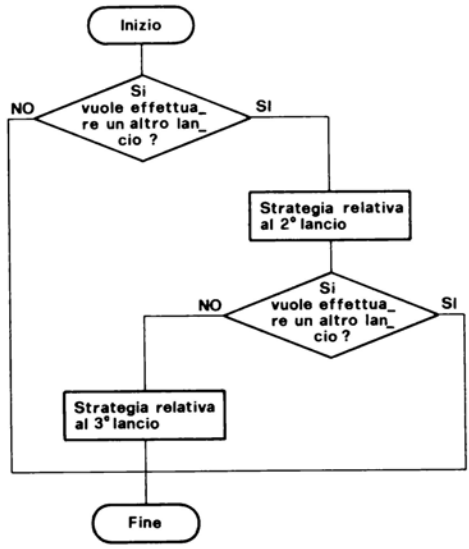


Figura 45

Lo sviluppo della strategia del primo giocatore è illustrato nelle figure (46) e (47). In tali diagrammi alcuni tests sono stati contrassegnati con delle lettere che ci serviranno per identificarli in fase di commento.

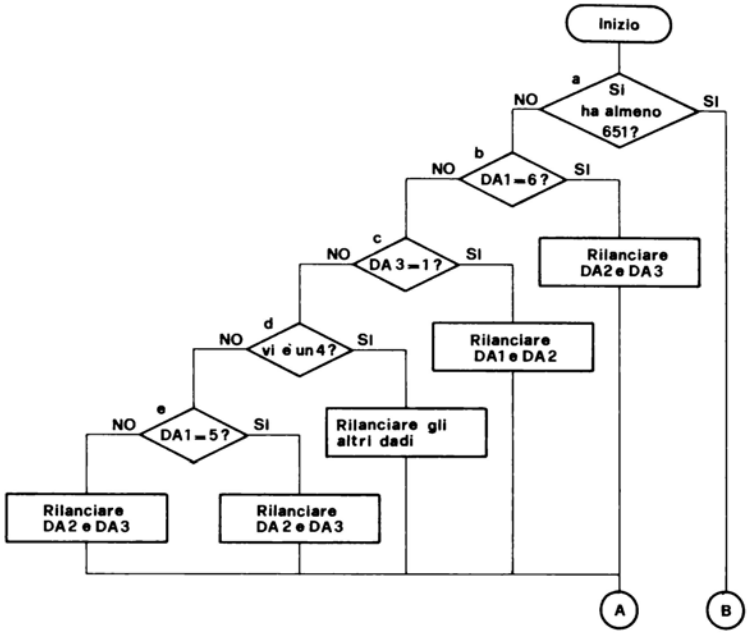
Osserviamo subito che questi diagrammi sono costuiti da una serie di tests in cascata che però non possono essere sostituiti da un unico blocco CASE poiché non sono indipendenti tra loro.

Si può notare che nel diagramma a blocchi della **figura 46**, dopo il blocco (e) non viene effettuato alcun test per la ricerca del valore 3. Esso sarebbe infatti inutile in quanto, essendo stati a questo punto già eliminati tutti i valori superiori a 651 e tutti quelli comprendenti 1 o 4 o 5 o 6, rimangono solò i valori 332 e 322 e pertanto si ha la sicurezza che DA1 = 3. Dovranno pertanto essere rigiocati i dadi DA2 e DA3.

Precisato questo si può osservare che anche il test (e) risulta inutile in quanto prevede le stesse operazioni in entrambi i rami. Tale test è stato comunque lasciato per chiarezza d'esposizione, mentre è stato eliminato nel diagramma relativo alla fase successiva (**figura 47**).

I tests (b) e (c) vengono effettuati su un solo dado in quanto si sa che nelle combinazioni il valore 6 può essere assunto solo dal primo dado ed il valore 1 solo dall'ultimo. Per i valori intermedi è invece necessario fare ricorso al sottoprogramma di ricerca (58).

Come si vede, anche in programmazione, come nella maggior parte delle discipline, vale il principio: "Perché non complicare le cose semplici?".



(61) Il gioco del 1° giocatore dopo un lancio

Figura 46

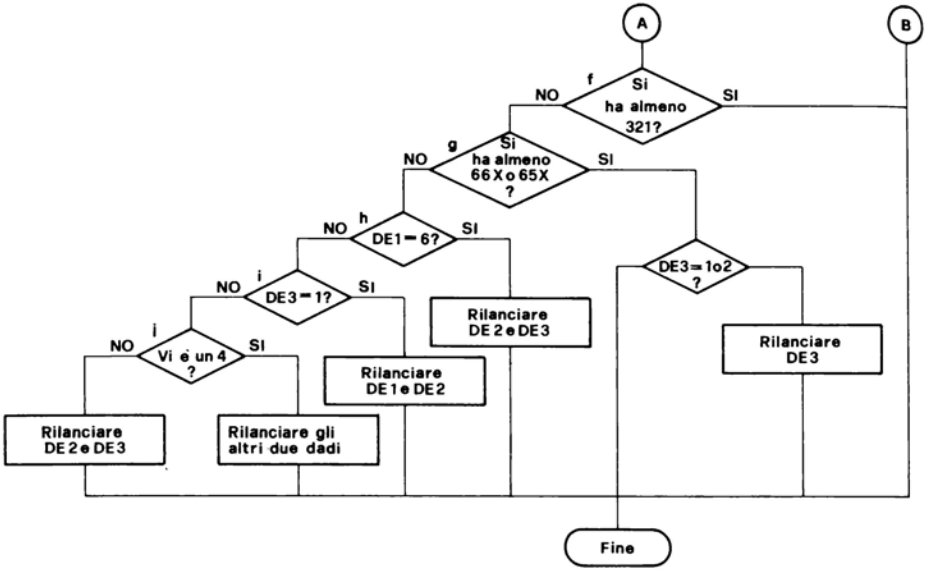
Abbiamo così sviluppato completamente la strategia del primo giocatore.

Cominciando ad analizzare le altre strategie (sia del giocatore intermedio che dell'ultimo) ci si accorge che occorre per prima cosa esaminare la posizione in classifica del giocatore: è l'ultimo? (caso evidentemente critico) oppure è primo? (caso fortunato, anche se questa situazione può essere solo temporanea, se non si è gli ultimi a giocare).

La classifica che si può stilare ha comunque validità solo per i giocatori che hanno già lanciato i dadi.

Per redigere questa classifica parziale si può creare una tabella che contenga qualche elemento caratterizzante quei giocatori che hanno già effettuato il loro gioco: per ottenere ciò potremo pertanto usare i risultati ottenuti dai vari partecipanti durante il giro.

Viceversa, per sapere subito chi non ha ancora giocato, è sufficiente inserire nella posizione corrispondente in tabella un valore impossibile per una combinazione: ad esempio un numero negativo.



(62) Il gioco del 1° giocatore dopo due lanci

Figura 47

Ogni-qualvolta un giocatore lancia i dadi, viene generata la combinazione corrispondente che viene inserita nella tabella, dopo essere stata classificata tra i lanci già giocati.

In pratica bisogna controllare:

- se il risultato in esame è il migliore o il peggiore tra quelli ottenuti dai vari giocatori;
  in una di queste due ipotesi, se vi sono altri giocatori che hanno ottenuto lo stesso risultato. (Ciò potrebbe in seguito portare ad uno spareggio).

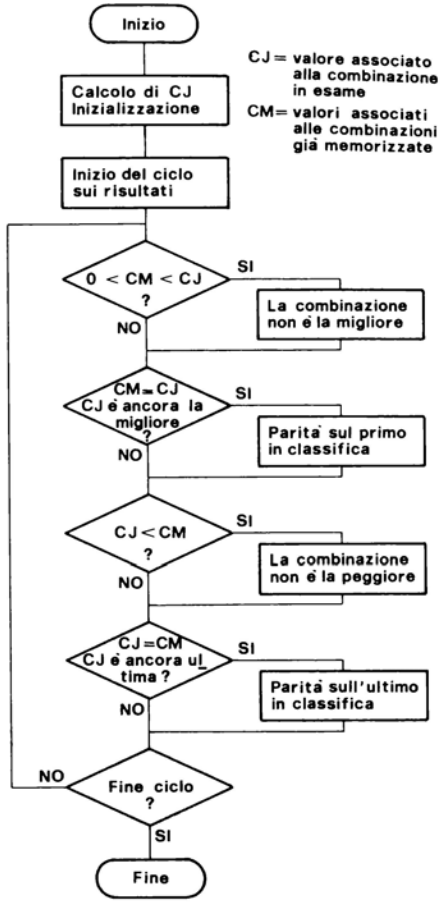
Occorre inoltre specificare che, nel caso in cui vi siano solo due giocatori, se questi hanno ottenuto lo stesso risultato tale risultato è contemporaneamente il migliore ed il peggiore e pertanto con un solo spareggio verranno designati sia il vincente che il perdente.

Una soluzione relativa al problema della classifica è illustrata nella **figura 48**. Il confronto tra i vari risultati è ottenuto con una serie di confronti tra il risultato in esame ed ognuno dei risultati ottenuti dagli altri giocatori, i cui valori sono contenuti nella tabella, escludendo ovviamente i valori simbolici. (Un valore negativo in tabella significa che quel giocatore non ha ancora giocato).

La combinazione in esame può classificarsi prima (la migliore), ultima o risultare in una posizione intermedia. La situazione di parità, nel caso un altro giocatore abbia ottenuto la stessa combinazione, verrà segnalata solo nel caso in cui si verifichi per il primo o l'ultimo posto in classifica.

È evidente che la definizione del primo e l'ultimo classificati può essere fatta solo dopo aver esaminato tutti i risultati ottenuti dai singoli giocatori. Per poter invece affermare che una combinazione non è vincente è sufficiente trovarne nella tabella una migliore.

Quando la combinazione in esame risulta vincente, nella classifica parziale, tale situazione verrà segnalata nel corso del ciclo di controlli finché sussiste, in quanto in caso contrario è sufficiente sapere che la combinazione non risulta vincente.



Posizione in classifica di un giocatore (63)

Figura 48

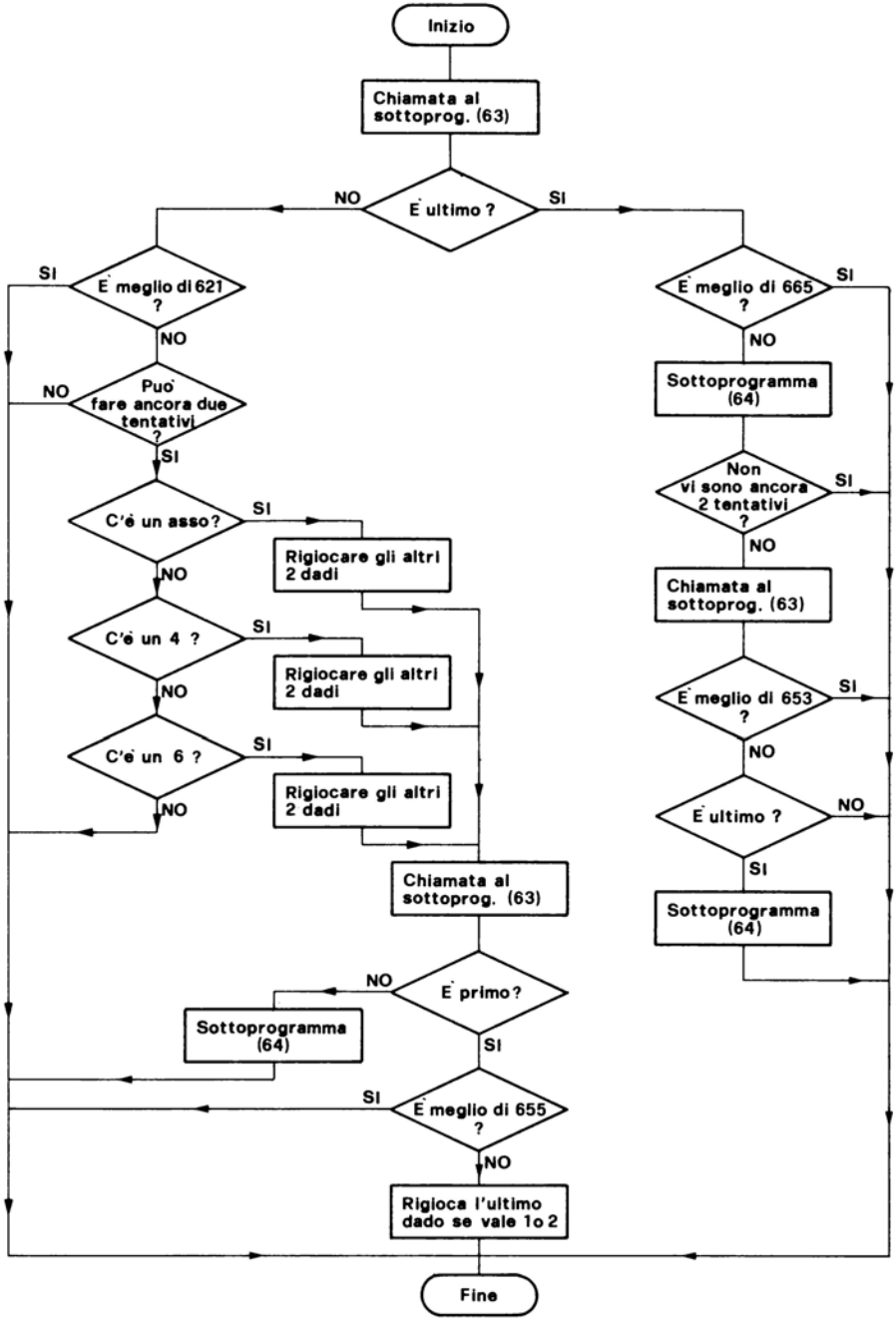
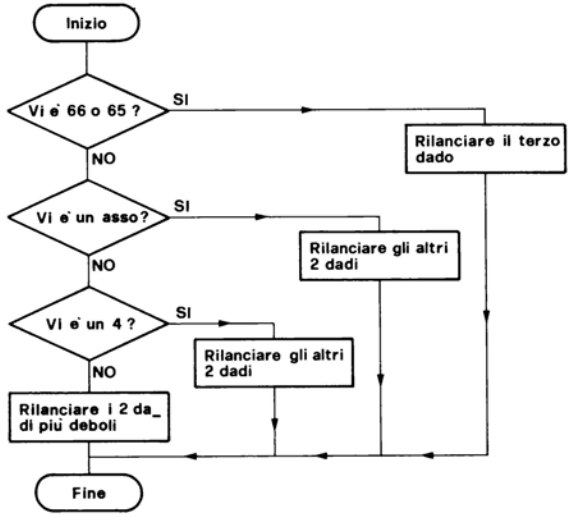


Figura 49 **Prosecuzione del gioco secondo la strategia del giocatore intermedio**

Un ragionamento del tutto analogo viene fatto nel caso in cui la combinazione in esame risulta ultima in classifica.
In fase di codifica per segnalare il fatto che un risultato risulta essere o non essere primo o ultimo in classifica verranno utilizzate due variabili inizializzate appositamente; il loro valore verrà poi modificato con l'evolversi della posizione in classifica della combinazione in esame. Ad esempio, il contenuto dell variabile per il controllo sulla prima posizione in classifica verrà modificato per indicare che la combinazione in esame non è più prima in classifica o è prima ex-aequo.



Strategia per il giocatore intermedio che si trova ultimo in classifica (64)
Figura 50

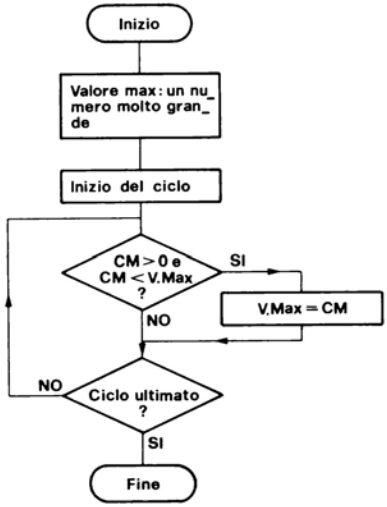
Tutto quanto sopra esposto permette di formulare completamente il diagramma a blocchi relativo alla strategia di un giocatore intermedio. Tale diagramma è presentato nella **figura 49**.

Guardando questa figura, colpisce in primo luogo la sua forma longinea, in cui diversi rami dei tests portano direttamente al blocco successivo. Ci è sembrato opportuno presentare alcuni esempi di questo tipo di stesura di un diagramma in quanto questo metodo di strutturazione viene utilizzato spesso dai programmatori.

È evidente che una costruzione di questo genere non permette di mettere sufficientemente in risalto le strutture logiche da noi presentate nel capitolo dedicato all'analisi.

I richiami al sottoprogramma (63) permettono di determinare se una combinazione si trova al primo posto o all'ultimo o in una posizione intermedia nella classifica provvisoria.

Abbiamo poi considerato il blocco (64) come sottoprogramma in quanto viene utilizzato in diversi casi.
La figura 50 illustra lo sviluppo della funzione (64).
A questo proposito bisogna osservare che la strategia di gioco del caso in cui nella combinazione ottenuta vi sia un 4 varia con la combinazione stessa: è diverso il modo di comportarsi di fronte a un 554 o a un 431.
Anche se in seguito bisognerà tener conto di queste differenze, non abbiamo ritenuto opportuno analizzarle a questo livello di analisi né dettagliare ulteriormente il diagramma a blocchi.
Non ci resta ora che affrontare l'analisi della strategia relativa all'ultimo giocatore.
Nella stesura di questa strategia abbiamo espanso a piacere il numero dei casi particolari, per mostrare come è possibile scomporre un problema quando la logica diventa troppo complessa.



Ricerca del miglior risultato ottenuto (65)
Figura 51

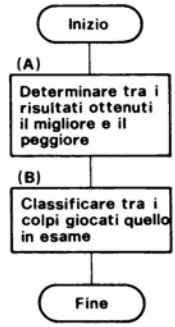
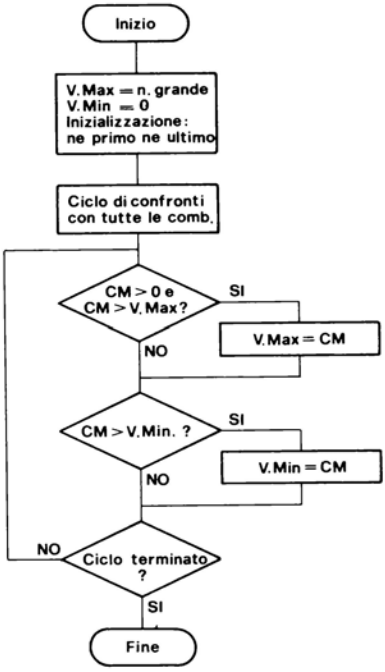


Figura 52

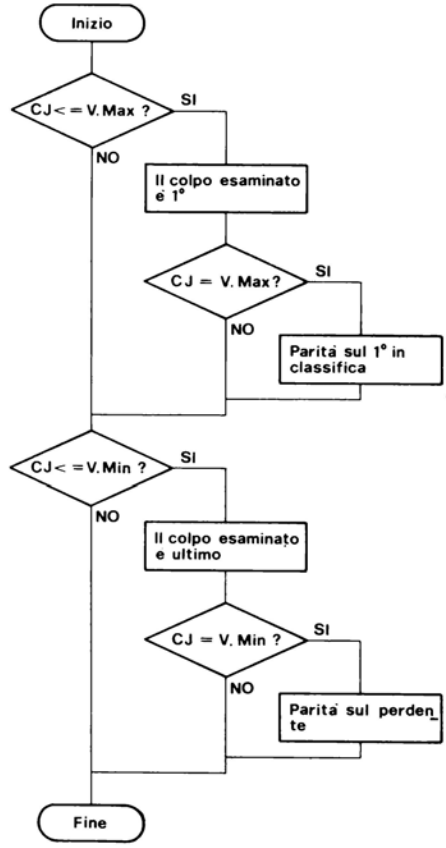
Riflettendo sulle regole di questa strategia ci si accorge che è comunque necessario sapere quali sono le due combinazioni che sono prima e ultima in classifica tra quelle ottenute dagli altri giocatori. Per poter stilare questa classifica potremo utilizzare una tabella contenente i valori relativi alle combinazioni ottenute nel corso del gioco.
Ricordiamo infatti che ad ogni combinazione si può associare un numero rappresentativo della sua posizione "gerarchica" e del suo valore in gettoni.

Per trovare il valore più piccolo in questa tabella, che corrisponde al miglior risultato ottenuto, occorre iniziare il ciclo dei confronti con un valore molto grande e comunque maggiore di quello relativo al peggior risultato conseguibile (ricordiamo che al "pessimo" 221 corrisponde il valore 651). Si prosegue poi la serie di confronti scorrendo tutta la tabella e tenendo in evidenza, per i



A) Determinare la migliore e la peggiore combinazione tra quelle dei vari giocatori

Figura 53

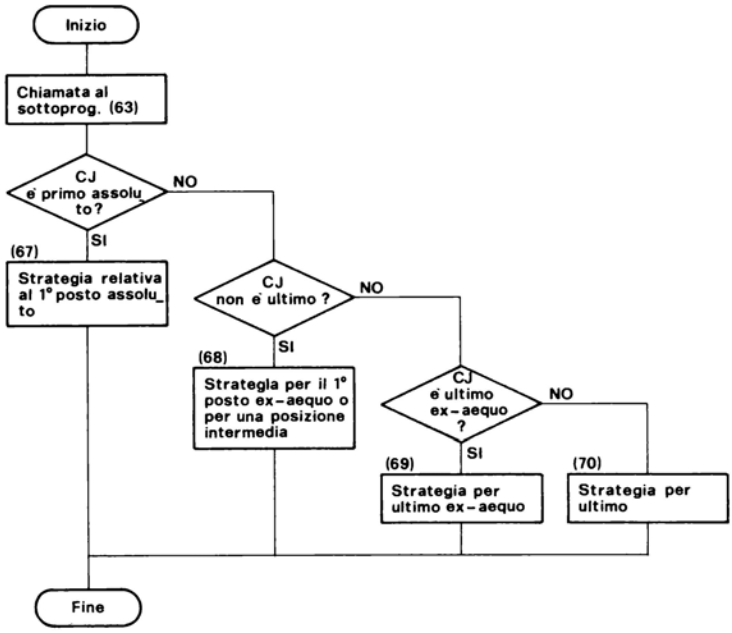


B) Classificare la combinazione in esame tra quelle ottenute dai vari giocatori

Figura 54

confronti successivi, il valore che al confronto attuale risulta minore. Alla fine del ciclo il valore che rimane in evidenza sarà quello corrispondente al miglior risultato ottenuto.

La figura 51 illustra il diagramma a blocchi relativo al sottoprogramma per la ricerca del valore corrispondente alla combinazione prima in classifica (65).



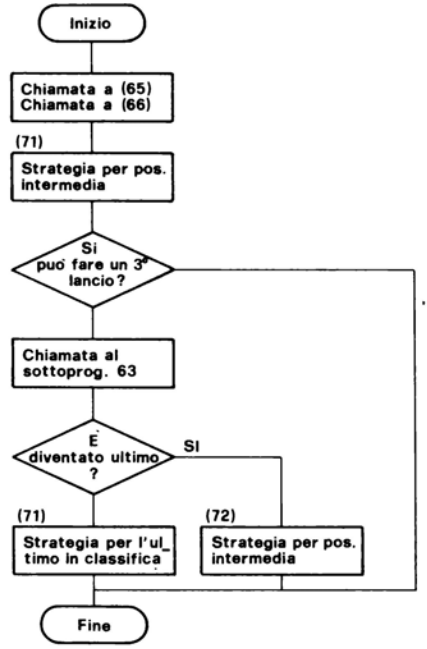
Strategia per l'ultimo giocatore (33)
Figura 55

Il sottoprogramma per la ricerca del valore relativo alla combinazione peggiore, identificato come (66), è molto simile a questo per cui non abbiamo ritenuto opportuno proporne il diagramma a blocchi.

Si può notare che anche il sottoprogramma (63) è molto simile ai due precedenti: tutti e tre contribuiscono ad ottenere la posizione in classifica della combinazione in esame rispetto ai risultati ottenuti dagli altri giocatori.

Si potrebbe quindi pensare di unificare questi sottoprogrammi in uno solo. Lo sviluppo di questo nuovo metodo è illustrato nelle **figure 52, 53, 54** in cui il lettore potrà facilmente riconoscere le funzioni costituenti i sottoprogrammi (63), (65) e (66).
In seguito non adotteremo questa soluzione, che non è comunque da scartare:

Strategia per il 1° assoluto in classifica (67)
Figura 56

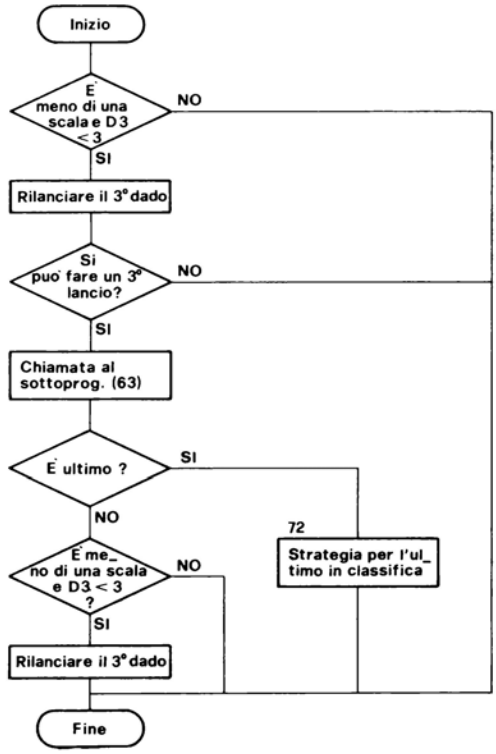


Strategia per il 1° ex-aequo o per una posizione intermedia
Figura 57

infatti anche se meno pratica quando si vuole conoscere solo la posizione relativa in classifica (primo, ultimo) della combinazione in esame, essa ci permette però di ottenere più velocemente delle informazioni suppletive, come

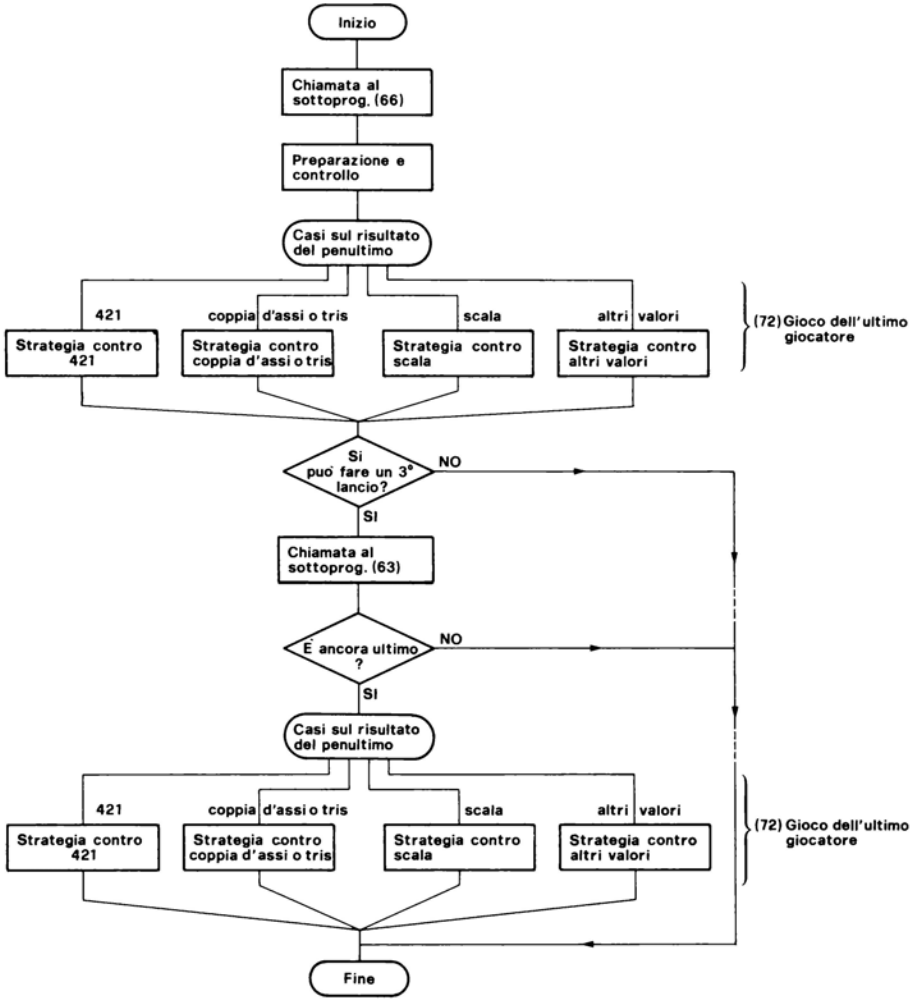
ad esempio il valore delle combinazioni prima ed ultima nella classifica parziale.

Nota: Osserviamo a questo proposito che con un metodo molto simile si poteva ottenere l'esatta posizione in classifica (primo, secondo, terzo ...) della



Strategia per l'ultimo in classifica ex-aequo

Figura 58

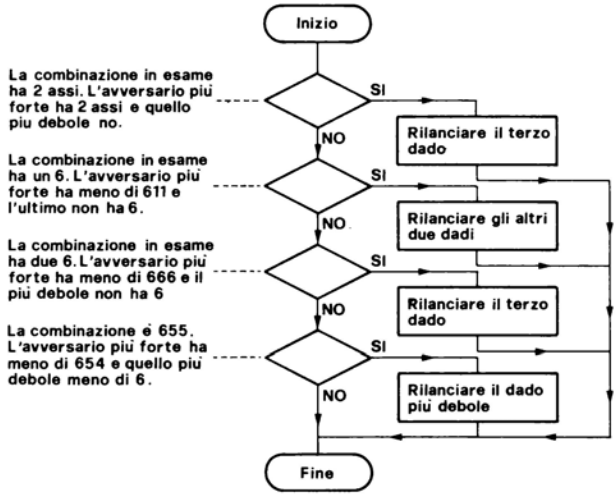


Strategia dell'ultimo giocatore (70)

Figura 59

combinazione in esame, ad esempio contando il numero delle combinazioni migliori.
Definiti tutti gli elementi necessari, possiamo ora passare ad esaminare nei dettagli la "strategia dell'ultimo giocatore".

Il diagrama a blocchi presentato nella **figura 55** illustra la struttura generale di tale strategia, che dovrà però essere ulteriormente dettagliata tenendo conto della particolare posizione occupata in classifica dall'ultimo giocatore.



Strategia per una posizione intermedia
Figura 60

Le varie "sottostrategie" relative alle situazioni che si possono verificare sono descritte graficamente nelle **figure dalla 56 alla 59**.
Hanno particolare importanza i casi in cui il giocatore si ritrova ultimo in classifica (72) o in una posizione intermedia (71).
Il blocco (72) è stato direttamente esaminato nello sviluppo presentato nella figura 59.
La funzione del blocco (71) invece è completamente rappresentata nella figura 60. Il diagramma a blocchi della **figura 60** non è altro però che un modo diverso

per definire le condizioni della strategia di gioco. Tenendo conto di ciò appare superfluo elencare anche i diagrammi a blocchi che illustrano gli ulteriori casi di questa strategia.
Con questa ultima sottostrategia abbiamo seguito completamente i giochi condotti dal giocatore-calcolatore: infatti abbiamo dettagliato le tre strategie che esso deve seguire nei vari casi come giocatore.
Ci resta ora da analizzare solo il blocco (1) di inizializzazione.

Inizializzazione (1)
Figura 61

Ci resta ora da analizzare solo il blocco (1) di inizializzazione.
Durante l'inizializzazione si devono fornire al calcolatore tutti i dati relativi al gioco: sia quelli variabili, come il numero di giocatori, il loro nome, il numero di gettoni del piatto, ..., sia quelli costanti, come l'elenco dei valori relativi a ciascuna combinazione, che devono essere inseriti nell'apposita tabella.
Il diagramma a blocchi che rappresenta questa funzione è descritto nella figura 61.

LA CODIFICA

Terminata l'analisi possiamo affrontare la fase di codifica. Avendo ben dettagliato gli algoritmi e gli elementi che si devono usare non sorgono problemi particolari:

Di seguito diamo l'elenco delle principali variabili utilizzate per la codifica:

CB: Vettore di 56 elementi, con indice variabile tra 0 e 55: contiene il "valore" di ogni combinazione calcolato tramite la funzione F (I, J, K).

NJ: Numero dei giocatori.

NM$: Vettore alfanumerico di NJ elementi, con indice variabile tra 1 e NJ, contiene i nomi dei giocatori; il giocatore calcolatore verrà chiamato "AUTOMA".

DA, DB, DC: Vettori di NJ elementi ciascuno, con indice variabile tra 1 ed NJ; contengono i valori assunti dai tre dadi nella combinazione definitiva ottenuta dai vari giocatori in un giro.

D1, D2, D3: Variabili che contengono i valori assunti dai tre dadi nella combinazione provvisoria relativa ad un giocatore. Tali valori vengono cambiati ogni qual volta il giocatore richiede, potendo, di fare un nuovo tentativo.

D9: Vettore di NJ elementi, con indice variabile da 1 ad NJ, contenente i codici dei giocatori che devono effettuare un turno supplementare di spareggio.

CL: Vettore di NJ elementi, con indice variabile tra 1 ed NJ, che serve per memorizzare i valori delle combinazioni relative ad un giro. Usato nei sottoprogrammi (63), (65), (66), corrisponde al CM usato nei diagrammi a blocchi.

JT: Vettore di NJ elementi, con indice variabile tra 1 ed NJ, contenente il numero di gettoni posseduti da ciascun giocatore.

PT: Contiene il numero di gettoni del piatto e viene usato solo in fase di carico.

JP: Numero di gettoni attribuiti al perdente alla fine di un giro in base alla combinazione vincente.

JJ: Codice del giocatore che è di turno. Alla fine di un giro questa variabile conterrà il codice del perdente che giocherà per primo nel giro successivo.

NK: Numero dei tentativi concessi: tale valore (1, 2, 3) viene definito durante il

gioco del primo giocatore e rappresenta il numero massimo di lanci che gli altri giocatori possono effettuare.

GN: Questa variabile, che assume solo i valori 0 ed 1, in quest'ultimo caso indica che la partita è terminata e che c'é un vincitore (nel vettore JT vi sarà un elemento nullo).

JX: Codice del giocatore vincente in un giro.

JN: Codice del giocatore perdente in un giro.

VX, VN: Variabili che servono ad individuare il vincente ed il perdente o chi si è piazzato meglio o peggio nella classifica parziale.

RX, RN: Variabili di controllo che servono a segnalare la necessità di uno spareggio per determinare il vincitore (RX) od il perdente (RN) secondo le seguenti convenzioni: se assumono valore 0 non vi è spareggio, se invece assumono valore 1 vi è spareggio (Nel caso particolare in cui i giocatori in una partita sono solo due ed è necessario uno spareggio, la variabile RN assume valore 2).

FC: Valore di una combinazione: Viene usato in fase di scarico in modo sistematico. (Corrisponde al CJ dei diagrammi a blocchi).

MX, MN: Variabili utilizzate nei sottoprogrammi per la ricerca della posizione relativa di un giocatore intermedio nella classifica parziale secondo la seguente convenzione:

MX = 0 : 1ª posizione; MX = 5 : 1° ex-aequo ;MX = 10 : non primo
MN = 0 : ultimo ; MN = 5 : ultimo ex-aequo;MN = 10 : non ultimo

I vari casi che si possono verificare sono riassunti nella seguente tabella:

	ultimo	ultimo ex-aequo	né primo né ultimo	primo ex-aequo	primo
MX	10	10	10	5	0
MN	0	5	10	10	10

Il caso MX = 5 e MN = 5 si può verificare solo quando vi e parità in una partita con due soli giocatori.

NN: Numero di cifre comuni tra una combinazione e la combinazione campione 421 (vedere strategia contro il 421)

RD: Numero di tentativi effettuti da un giocatore umano

WX: Variabile che viene usata in fase di spareggio per il primo posto (il nome WX è stato utilizzato per non generare confusione con la variabile VX che serve per la distribuzione dei gettoni)

I, J, W: Indici per i cicli. W viene utilizzato solo nei cicli di temporizzazione, che possono essere inseriti in qualsiasi punto del programma senza turbare l'andamento dei cicli dipendenti da altri indici.

421

```
10 CLEAR 200
20 DIM CB(55)
30 GOSUB 1000: REM INIZIALIZZAZIONE
40 GOSUB 100   REM IL CARICO
50 GOSUB 300   REM LO SCARICO
90 END
95 REM
96 REM ------------------------------------------------
97 REM
100 FOR I=1 TO NJ : REM IL CARICO
110 IF NM$(JJ)<>"AUTOMA" THEN GOSUB 5900
115 PRINT"IL GIOCATORE ";NM$(JJ);" NUMERO ";JJ;" GIOCA"
120 GOSUB 5800:DA(JJ)=D1:DB(JJ)=D2:DC(JJ)=D3:JJ=JJ+1
125 IF JJ>NJ THEN JJ=1
130 GOSUB 9950:PRINT
135 NEXT I
140 GOSUB 700
150 IF JP=7 THEN JP=PT-1:PT=1:GOTO 180
160 IF JP<=JT THEN PT=PT-JP:GOTO 180
170 IF JP>PT THEN JP=PT:PT=0
180 PRINT"IL GIOCATORE ";NM$(JN);" NUMERO ";JN;" HA PERSO"
190 PRINT"EGLI PRENDE ";JP;" GETTONI DAL PIATTO":
200 JT(JN)=JT(JN)+JP
210 PRINT"ADESSO HA ";JT(JN);" GETTONI"
220 JJ=JN:GOSUB 950
230 IF PT>0 THEN 100
290 RETURN
291 REM
292 REM ------------------------------------------------
293 REM
300  GN=0  :      REM      LO SCARICO
310 NK=0:GOSUB 960:IF NM$(JJ)<>"AUTOMA" THEN GOSUB 5900:REM 1 GIOC
320 PRINT"IL GIOCATORE ";NM$(JJ);" NUMERO ";JJ;" GIOCA"
330 GOSUB 5800:IF NM$(JJ)<>"AUTOMA" THEN GOSUB 5500
340 IF NM$(JJ)="AUTOMA" THEN GOSUB 2000
350 DA(JJ)=D1:DB(JJ)=D2:DC(JJ)=D3:JJ=JJ+1:IF JJ>NJ THEN JJ=1
360 GOSUB 9400:GOSUB 9950:IF NJ=2 THEN 440
370 FOR I=2 TO NJ-1
```

```
371 IF NM$(JJ)<>"AUTOMA" THEN GOSUB 5900:REM GIOCATORI INTERMEDI
380 PRINT"IL GIOCATORE ";NM$(JJ);" NUMERO ";JJ;" GIOCA"
385 GOSUB 5800:GOSUB 9950
390 IF NK=1 THEN GOTO 420
400 IF NM$(JJ)<>"AUTOMA" THEN GOSUB 5500
410 IF NM$(JJ)="AUTOMA" THEN GOSUB 3000
420 DA(JJ)=D1:DB(JJ)=D2:DC(JJ)=D3:JJ=JJ+1:IF JJ>NJ THEN JJ=1
430 GOSUB 9400
435 NEXT I
440 IF NM$(JJ)<>"AUTOMA" THEN GOSUB 5900: REM ULTIMO GIOCATORE
450 PRINT"IL GIOCATORE ";NM$(JJ);" NUMERO ";JJ;" GIOCA"
455 GOSUB 5800:GOSUB 9950
460 IF NK=1 THEN 490
470 IF NM$(JJ)<>"AUTOMA" THEN GOSUB 5500
480 IF NM$(JJ)="AUTOMA" THEN GOSUB 4000
490 DA(JJ)=D1:DB(JJ)=D2:DC(JJ)=D3
500 GOSUB 700: REM CLASSIFICA
510 IF JP>=JT(JX) THEN JP=JT(JX):GN=1
520 PRINT" IL GIOCATORE ";NM$(JX);" NUMERO ";JX;" HA VINTO"
530 PRINT"CEDE ";JP;" GETTONI AL GIOCATORE "NM$(JN);" NUMERO";JN
540 JT(JX)=JT(JX)-JP:JT(JN)=JT(JN)+JP
550 GOSUB 900:GOSUB 9950
560 JJ=JN:IF GN=0 THEN 310
570 PRINT:PRINT"IL GIOCATORE "NM$(JX);" NUMERO ";JX
580 PRINT" HA VINTO LA PARTITA"
590 RETURN
591 REM
592 REM ------------------------------------------------------
593 REM
700 D1=DA(1):D2=DB(1):D3=DC(1):GOSUB 9000:  REM  CLASSIFICAZIONE
710 VX=FC:VN=FC:RX=0:RN=0:JN=1:JX=1
720 FOR I=2 TO NJ
730 D1=DA(I):D2=DB(I):D3=DC(I):GOSUB 9000
740 IF FC<VX THEN VX=FC:JX=I:RX=0:GOTO 760
750 IF FC=VX THEN RX=1
760 IF VN<FC THEN VN=FC:JN=I:GOTO 780
770 IF VN=FC THEN RN=1
780 NEXT I
785 IF VN=VX THEN RN=2
790 IF RN=2 THEN GOSUB 9100:RN=0:GOTO 700: 2 SOLI GIOC.  E PARI
800 IF RX=1 THEN GOSUB 9100: SPAREGGIO TRA VINCENTI
810 IF RN=1 THEN GOSUB 9200: SPAREGGIO TRA PERDENTI
820 JP=INT(VX/10):JP=VX-10*JP
890 RETURN
891 REM
892 REM ------------------------------------------------------
893 REM
900 GOSUB 9950:GOSUB 9950:CLS:PRINT"RICAPITOLAZIONE"
910 FOR I=1 TO NJ
915 PRINT"IL GIOCATORE ";NM$(I);" NUMERO ";I;" HA ";JT(I);"GET"
920 NEXT I
930 PRINT:GOSUB 9950
940 RETURN
950 GOSUB 900:"NEL PIATTO RESTANO ";PT;" GETTONI":RETURN
960 FOR I=1 TO NJ:CL(I)=-1:NEXT
970 RETURN : REM PER UN ALTRO GIRO
1000 CLS:PRINT TAB(15)"GIOCO DEL 421":PRINT:PRINT:REM INIZIALIZZ
1010 INPUT"IL N DI GIOCATORI AL VOSTRO TAVOLO E' ";NJ
1020 IF NJ<2 OR NJ>20 THEN PRINT"INSERIRE UN NUMERO TRA 2 E 20"
1025 GOTO 1010
```

```
1030 DIM NM$(NJ),JT(NJ),DA(NJ),DB(NJ),DC(NJ),D9(NJ),CL(NJ)
1040 PRINT"INSERIRE I NOMI DEI GIOCATORI"
1050 PRINT"PER IL CALCOLATORE INSERIRE LA SIGLA AUTOMA"
1060 FOR I=1 TO NJ:PRINT"NOME GIOC.  NUM.  ";I:INPUT NM$(I):NEXT
1080 PRINT:PRINT"ESTRAZIONE A SORTE DEL GIOCATORE CHE INIZIA"
1090 RANDOM:JJ=RND(NJ):PRINT"E' IL GIOC.  NUMERO ";JJ,NM$(JJ)
1100 FOR I=1 TO 55:READ CB(I):NEXT
1110 PT=11:PRINT"VOLETE FISSARE IL NUMERO DI GETTONI DA DARE IN"
1120 PRINT"PARTENZA A CIASCUN GIOCATORE?"
1130 INPUT"SE BATTETE 0 VIENE ESTRATTO A SORTE";JP
1140 IF JP=0 THEN JP=RND(20)
1150 FOR I=1 TO NJ:JT(I)=JP:NEXT
1160 PRINT"CIASCUN GIOCATORE HA ";JP;" GETTONI"
1170 FOR I=1 TO NJ :REM COMPATTAMENTO NOMI
1180 IF RIGTH$(NM$(I),1)<>" " THEN GOTO 1190
1185 NM$(I)=LEFT$(MN$(I),(LEN(NM$(I))-1):GOTO 1180
1190 NEXT I
1195 RETURN
2000 NK=1: REM STRATEGIA DEL PRIMO GIOCATORE
2010 GOSUB 9000
2020 IF FC<255 THEN RETURN
2030 GOSUB 2500
2040 NK=2:GOSUB 9000
2050 IF FC<164 THEN RETURN
2060 IF FC<255 AND D3<3 THEN NK=3:GOSUB 2600:RETURN
2070 IF FC<255 THEN RETURN
2080 GOSUB NK=3   GOSUB 2500
2090 RETURN
2500 IF D1=6 THEN GOSUB 2650:RETURN
2510 IF D3=1 THEN D3=D1:D1=1:GOSUB 2650:RETURN
2520 IF D1=4 THEN GOSUB 2650:RETURN
2530 IF D2=4 THEN D2=D1:D1=4:GOSUB 2650:RETURN
2540 IF D3=4 THEN D3=D1:D1=4:GOSUB 2650:RETURN
2550 GOSUB 2650
2590 RETURN
2591 REM
2592 REM ------------------------------------------------------

2593 REM
2600 PRINT"IL GIOCATORE AUTOMA ";JJ;" RIGIOCA IL DADO ";D3
2610 D3=RND(6)
2620 PRINT"NUOVO VALORE DEL DADO ";D3
2630 GOSUB 5820
2640 RETURN
2641 REM

2642 REM ------------------------------------------------------
2643 REM
2650 PRINT"IL GIOCAT.  AUTOMA ";JJ;" RIGIOCA I 2 DADI "D2,D3
2660 D2=RND(6):D3=RND(6)
2670 PRINT"NUOVI VALORI DEI DUE DADI ";D2,D3
2680 GOSUB 5820
2690 RETURN
2691 REM

2692 REM ------------------------------------------------------
2693 REM
3000 GOSUB 3900: REM STRATEGIA DEL GIOCATORE INTERMEDIO
3030 IF MN<7 THEN 3200
3040 IF FC<342 THEN RETURN
3050 IF NK<3 THEN RETURN
3060 IF D3=1 THEN D3=D1:D1=1:GOSUB 2650:GOTO 3120
```

```
3070 IF D1=4 THEN GOSUB 2650:GOTO 3120
3080 IF D2=4 THEN D2=D1:D1=4:GOSUB 2650:GOTO 3120
3090 IF D3=4 THEN D3=D1:D1=4:GOSUB 2650:GOTO 3120
3100 IF D1=6 THEN GOSUB 2650
3110 IF NK<3 THEN RETURN
3120 GOSUB 3900
3140 IF MN<7 THEN GOSUB 3500: RETURN
3150 IF FC<212 THEN RETURN
3160 IF D3=1 OR D3=2 THEN GOSUB 2600
3170 RETURN
3200 IF FC<163 THEN RETURN
3210 GOSUB 3500
3220 IF NK=2 THEN RETURN
3230 GOSUB 3900
3240 IF FC<232 THEN RETURN
3250 IF MN=10 THEN RETURN
3260 GOSUB 3500
3270 RETURN
3500 IF D1=6  AND (D2=6 OR D2=5) THEN GOSUB 2600:RETURN
3510 IF D3=1 THEN D3=D1:D1=1:GOSUB 2650:RETURN
3520 IF D1=4 THEN GOSUB 2650:RETURN
3530 IF D2=4 THEN D2=D1:D1=4:GOSUB 2650:RETURN
3540 IF D3=4 THEN D3=D1:D1=4:GOSUB 2650:RETURN
3550 GOSUB 2650
3560 RETURN
3900 GOSUB 9000:REM PIAZZAMENTO PROVVISORIO
3910 MX=0:MN=0
3920 FOR J=1 TO NJ
3930 IF CL(J)>0 AND CL(J)<FC THEN MX=10
3940 IF CL(J)=FC AND MX=0 THEN MX=5
3950 IF CL(J)>FC THEN MN=10
3960 IF CL(J)=FC AND MN=0 THEN MN=5
3970 NEXT J
3980 RETURN
3990 REM
3991 REM ------------------------------------------------

3992 REM
4000 GOSUB 3900: REM STRATEGIA ULTIMO GIOCATORE
4030 IF MX>0 THEN 4100
4040 GOSUB 4900
4050 IF (((FC>65 AND FC<>123) OR FC<20) OR VX<65) THEN RETURN
4060 IF D1<3 THEN D3=D1:D1=1:GOSUB 2600
4070 IF NK=2 THEN RETURN
4080 IF D1<3 THEN D3=D1:D1=1:GOSUB 2600
4090 RETURN
4100 IF MN<7 THEN GOTO 4200
4110 GOSUB 4900:GOSUB 4950
4120 GOSUB 4800
4130 IF NK=2 THEN RETURN
4140 GOSUB 3900
4160 IF MN=10 AND MX=10 THEN GOSUB 4800
4190 RETURN
4200 IF MN=0 THEN GOSUB 4900:GOSUB 4950:GOTO 4300
4210 IF FC<200 AND D3<3 THEN GOSUB 2600:GOTO 4230
4220 GOTO 4300
4230 IF NK=2 THEN RETURN
4250 GOSUB 3900
4260 IF MN=0 THEN 4300
4280 IF FC>165 AND D3<3 THEN GOSUB 2600:RETURN
4300 IF VX>20 THEN 4370
```

```
4310 GOSUB 4650
4320 IF NK=2 AND NN=0 THEN RETURN
4330 GOSUB 4600:GOSUB 4650
4340 IF NN=0 OR NN=3 OR NK=2 THEN RETURN
4350 GOSUB 4600
4360 RETURN
4370 IF VX>130 THEN 4440
4380 GOSUB 4550
4390 IF NK=2 THEN RETURN
4400 GOSUB 3900
4410 IF MN=10 THEN RETURN
4420 GOSUB 4550
4430 RETURN
4440 IF VX>170 THEN 4500
4450 GOSUB 5000
4460 IF NK=2 THEN RETURN
4470 GOSUB 3900
4480 IF MN=10 THEN RETURN
4490 GOSUB 5000:RETURN
4500 GOSUB 5100
4510 IF NK=2 THEN RETURN
4520 GOSUB 3900
4530 IF MN=10 THEN RETURN
4540 GOSUB 5100:RETURN
4550 IF D2=1 THEN D3=D1:D1=1:GOSUB 2600:RETURN
4560 IF D2=1 THEN D3=D1:D1=1:GOSUB 2650:RETURN
4570 GOSUB 2650:RETURN
4600 IF NN=0 OR NN=2 THEN GOSUB 2600
4610 IF NN=1 THEN GOSUB 2650
4620 RETURN
4650 NN=0

4660 IF D3=1 THEN NN=1:D3=D1:D1=1
4670 IF D1=4 OR D2=4 OR D3=4 THEN NN=NN+1
4680 ID D1=2 OR D2=2 OR D3=2 THEN NN=NN+1
4690 IF NN=0 OR NN=3 THEN RETURN
4700 IF NN=1 THEN 4750
4710 IF D1=4 OR D1=2 OR D1=1 THEN 4730
4720 DE=D1:D1=D3:D3=DE:RETURN
4730 IF(D2=4 OR D2=2 OR D2=1) AND D1<>D2 THEN RETURN
4740 DE=D2:D2=D3:D3=DE:RETURN
4750 IF D1=4 OR D1=2 OR D1=1 THEN RETURN
4760 IF D2=4 OR D2=2 OR D2=1 THEN DE=D1:D1=D2:D2=DE:RETURN
4770 DE=D1:D1=D3:D3=DE:RETURN
4800 IF(FC=123 OR(FC<65 AND FC>20))THEN 4802
4801 GOTO 4810
4802 IF (VX>20 AND VX<130) AND VN>130 THEN 4805
4803 GOTO 4810
4805 D3=D1:D1=1:GOSUB 2600:RETURN
4810 IF D1=6 AND VX>30 AND VN>345 THEN GOSUB 2650:RETURN
4820 IF D2=6 AND VX>30 AND VN>345 THEN GOSUB 2600:RETURN
4830 IF D1=6 AND D2=5 AND VX>255 THEN GOSUB 2600
4840 RETURN
4900 VX=570
4910 FOR J=1 TO  NJ
4920 IF CL(J)>0 AND CL(J)<VX THEN VX=CL(J)
4930 NEXT J
4940 RETURN
4950 VN=0
4960 FOR J=1 TO NJ
4970 IF CL(J)>VN THEN VN=CL(J): JN=J
```

```
4980 NEXT J
4990 RETURN
5000 IF D3=1 THEN D3=D1:D1=1:GOSUB 2650:RETURN
5010 IF (D2-D3)=1 AND D2<>6 AND D3<>1 AND D2>=DB(JN) THEN 5015
5011 GOTO 5020
5015 GOSUB 2600:RETURN
5020 IF (D1-D2)=1 AND D1<>6 AND D2<>1 AND D1>= DB(JN) THEN 5025
5021 GOTO 5030
5025 GOSUB 2600:RETURN
5030 GOSUB 5100
5090 RETURN
5100 IF D1=6 AND VN>300 THEN GOSUB 2650 : RETURN
5110 IF D3=1 THEN D3=D1:D1=1:GOSUB 2650:RETURN
5120 IF D1=6 THEN GOSUB 2650:RETURN
5130 IF D1=4 THEN GOSUB 2650:RETURN
5140 IF D2=4 THEN D2=D1:D1=4:GOSUB 2650:RETURN
5150 IF D3=4 THEN D3=D1:D1=4:GOSUB 2650:RETURN
5160 GOSUB 2650
5190 RETURN
5191 REM
5192 REM ------------------------------------------------
5193 REM
5500 RD=1 :REM IL GIOCATORE UMANO
5510 GOSUB 5950
5520 IF NK=0 AND B$="N" THEN NK=1:RETURN
5530 IF B$="N" THEN RETURN
5540 RD=RD+1
5560 INPUT"INSERITE IL VALORE DI UN DADO DA RIGIOCARE";DR
5570 IF DR=D1 OR DR=D2 OR DR=D3 THEN 5600
5580 PRINT"NON ESISTE NELLA COMBINAZIONE"
5590 INPUT"INSERIRE UN NUOVO VALORE";DR:GOTO 5570
5600 IF DR=D2 THEN D2=D3:GOTO  5620
5610 IF DR=D1 THEN D1=D3
5620 PRINT"VOLETE GIOCARE UN SECONDO DADO?"
5630 PRINT"SE SI INSERITENE IL VALORE ALTRIMENTI INSERITE 0"
5640 INPUT DR:IF DR=D1 OR DR=D2 OR DR=D3 THEN 5670
5650 PRINT"INSERITE SOLO VALORI DELLA COMBINAZIONE"
5660 PRINT"EVENTUALMENTE UGALE A QUELLO GIA DATO":GOTO 5640
5670 IF DR<>0 THEN GOTO 5700
5680 D3=RND(6)
5690 PRINT"IL NUOVO VALORE DEL DADO E' ";D3:GOTO 5730
5691 REM
5692 REM ------------------------------------------------
5693 REM
5700 IF DR=D1 THEN D1=D2
5710 D2=RND(6): "PRIMO DADO RILANCIATO ";D2
5720 D3=RND(6):PRINT"SECONDO DADO RILANCIATO ";D3
5730 GOSUB 5820
5740 IF NK=RD THEN RETURN
5750 GOSUB 5950
5760 IF NK=0 THEN NK=3
5770 GOTO 5530
5780 REM
5790 REM ------------------------------------------------
5791 REM
5800 D1=RND(6):D2=RND(6):D3=RND(6): REM I TRE DADI
5810 PRINT" I VALORI DEI TRE DADI SONO ";D1,D2,D3
5820 IF D2>D1 THEN DE=D1:D1=D2:D2=DE:REM LA COMBINAZIONE
5830 IF D3>D2 THEN DE=D3:D3=D2:D2=DE
5840 IF D2>D1 THEN DE=D2:D2=D1:D1=DE
```

```
5850 PRINT"E DANNO LA COMBINAZIONE ";D1;D2;D3
5890 RETURN
5891 REM
5892 REM ------------------------------------------------------
5893 REM
5900 PRINT"E' IL TURNO DI ";NM$(JJ);" GIOC. NUMERO ";JJ
5901 REM ---IL GIOCATORE UMANO:SUE ISTRUZIONI-----
5910 PRINT"BATTETE UN TASTO PER GIOCARE"
5920 INPUT A$
5929 RETURN
5950 PRINT"VOLETE RIGIOCARE DEI DATI?"
5960 A$="":INPUT A$
5970 B$=LEFT$(A$,1)
5980 IF B$<>"N" AND B$<>"S" THEN PRINT"SI O NO?": GOTO 5960
5990 RETURN
5991 REM
5992 REM ------------------------------------------------
5993 REM
9000 RR=(D1-1)*D1*(D1+1)/6:RR=RR+D2*(D2-1)/2:RR=56-D3-RR
9010 FC=CB(RR):REM VALORE DELLA COMBINAZIONE
9020 RETURN
9030 REM
9040 REM ------------------------------------------------
9050 REM
9100 D1=DA(JX):D2=DB(JX):D3=DC(JX):JB=JX+1:D9(1)=JX:REM SPAREGGO
9110 GOSUB 9350:GOSUB 9300
9120 WX=999:RX=0
9125 FOR I=1 TO JP
9126 JW=D9(I)
9130 PRINT"IL GIOCATORE NUMERO ";JW;" RIGIOCA"
9140 GOSUB 5800:GOSUB 9000
9150 IF WX>FC THEN WX=FC:JX=JW:GOTO 9170
9153 REM
9160 IF WX=FC THEN RX=1
9170 DA(JW)=D1:DB(JW)=D2:DC(JW)=D3
9171 NEXT I
9180 IF RX>0 THEN 9100
9195 RETURN
9196 REM
9197 REM ----------------------------------------
9198 REM
9200 D1=DA(JN):D2=DB(JN):D3=DC(JN):JB=JN+1:D9(1)=JN:REM SPAREGGO
9210 GOSUB 9350:GOSUB 9300
9220 VN=0:RN=0
9225 FOR I = 1 TO JP
9226 JW=J9(I)
9230 PRINT"IL GIOCATORE NUMERO ";JW;" RIGIOCA"
9240 GOSUB 5800:GOSUB 9000
9250 IF VN<FC THEN VN=FC:JN=JW:GOTO 9270
9260 IF VN=FC THEN RN=1
9270 DA(JW)=D1:DB(JW)=D2:DC(JW)=D3
9271 NEXT I
9280 IF RN>0 THEN 9200
9290 GOSUB 9950
9295 RETURN
9296 REM
9297 REM ------------------------------------------------
9298 REM
9300 PRINT:PRINT"VI E' UNO SPAREGGIO"
```

```
9310 PRINT "TRA I GIOCATORI"
9320 FOR I=1 TO JP:PRINT D9(I):NEXT
9330 PRINT:PRINT:PRINT:RETURN
9335 REM
9336 REM ------------------------------------------------
9337 REM
9350 FOR I=2 TO NJ:D9(I)=0:NEXT   REM RICERCA DI EX-AEQUO
9351 JP=1
9360 FOR I=JB TO NJ:LI=DA(I):LJ=DB(I):LK=DC(I)
9370 IF D1=LI AND D2=LJ AND D3=LK THEN JP=JP+1:D9(JP)=I
9380 NEXT I
9385 RETURN
9386 REM
9387 REM
9400 GOSUB 9000
9410 CL(JJ)=FC:RETURN
9420 REM
9430 REM-------------------------------------------------
9440 REM
9950 FOR W=1 TO 1000:NEXT:RETURN:REM CICLO DI RITARDO
9951 REM
9952 REM ------------------------------------------------
9990 DATA 73,171,181,191,201,211,221,132: REM TABELLA
9991 DATA 231,241,251,261,271,281,291,301:REM DELLE
9992 DATA 311,321,331,341,26,83,351,361   :REM COMBINAZIONI
9993 DATA 371,381,391,142,401,411,421,431:REM ESPRESSE CON
9994 DATA 441,451,461,35,93,471,481,491   :REM I LORO VALORI
9995 DATA 501,152,511,521,17,44,103,531
9996 DATA 541,551,162,53,113,561,62,123
9997 REM ------------------------------------------------
```

Per aiutare il lettore nell'esame del listato riportiamo la seguente tabella di corrispondenza tra diagramma a blocchi e istruzioni.

Blocchi	Linee di programma
1	1000 - 1195
2	100 - 290
3	300 - 590
4	100 - 140
5	150 - 170
6	180 - 220
7	300 -
8	310 - 500
9	510 - 550
10	560
11	La funzione RND (6)
12	5800 - 5810
13	9000 - 9020
14	820
15	nessuna

Blocchi	Linee di programma
16	100 – 116
17	120 - 125
18	130
19	nessuna
20	310 - 360 inizio
21	360 fine
22	440 - 500
23	370 - 430
24	Non è codificato come sottoprogr. (istr. 320, 380, 450)
25	5900 - 5929 e 5950 - 5990
26	Non è codificato come sottoprogr. (Istr. 320, 380, 450)

Blocchi	Linee di programma
27	niente (allineamento di D1, D2, D3)
28	330, 400 e 470
29	5500 - 5770
30	2000 - 2090
31	350 (inizio) il numero di colpi giocati è in NK e vi resta
32	460
33	4000 - 4390
34	3000 - 3170
35	490
36	5950 - 5960
37	5970 - 5980
38	5560 - 5570
39	5680 - 5730
40	nessuna
41	5740 - 5780
42, 43	5560
44, 45	5570
46	5580 - 5590
47	5600 - 5610
48	5620 - 5630

Blocchi	Linee di programma
49, 50, 51	5640
52	5650 - 5660
53	5680 - 5700
54, 55, 56	5680 - 5620 con una riorganizzazione del diagramma a blocchi per rendere più facile la codifica del testo dei messaggi
57	5730
58, 59	Non codificato sotto forma di sottoprogramma
61	2020 - 2030
62	2050 - 2080
63	3910 - 3970
64	3500 - 3650
65	4900 - 4940
66	4950 - 4980
67	4040 - 4090
68	4110 - 4190
69	4210 - 4280
70	4300 - 4390
71	4800 - 4840
72	4320

Le seguenti note completano ciò che si può dire sulla codifica del programma da noi presentata:

- sono stati generati due sottoprogrammi per rigiocare rispettivamente uno e due dadi (dato che secondo le regole non è possibile rigiocarli tutti e tre).

Il primo sottoprogramma, formato dalle istruzioni che vanno dalla 2600 alla 2649, permette di rigiocare il dado D3. Prima di richiamare tale sottoprogram-

ma bisognerà quindi fare in modo che D1 e D2 rappresentino i dadi che si vogliono conservare e D3 invece quello che si vuole rigiocare.

Il secondo sottoprogramma, formato dalle istruzioni che vanno dalla 2650 alla 2699, serve invece per rigiocare i dadi D2 e D3.

- Questa codifica non è certo la migliore che si potesse realizzare. In particolare è appesantita molto dal fatto di aver utilizzato una versione limitata del linguaggio Basic. Il lettore potrà facilmente ottenere un programma migliore utilizzando tutte quelle particolari istruzioni che il linguaggio Basic implementato sul suo calcolatore gli permette.
- Il programma realizzato non identifica un buon giocatore di 421. La colpa non è né del calcolatore né del programmatore, ma di chi ha definito questa strategia di gioco. Per ottenere un buon giocatore di 421 converrà quindi perfezionare questa strategia. I mezzi per far ciò, pur senza essere codificati, sono stati segnalati durante la fase di analisi. Il nostro scopo, del resto, non era quello di realizzare una strategia perfetta, ma di insegnare un metodo di programmazione. Aggiungere un numero considerevole di casi particolari avrebbe portato soltanto ad un notevole appesantimento in fase di codifica senza alcun vantaggio da un punto di vista didattico: al contrario, le note raccolte nella "messa a punto" sul come ottenere una strategia migliore, possono completare la trattazione e nello stesso tempo stimolare il lettore a proseguire il lavoro per proprio conto.

- Questo programma potrebbe "girare" ugualmente e per giunta più velocemente utilizzando delle variabili di tipo intero.
Sul TRS 80 con il comando DEF INT A-Z si possono trasformare tutte le variabili di tipo reale in variabili di tipo intero. Utilizzando questa soluzione risulterebbe inutile l'uso della funzione INT, come invece fatto ad esempio nella riga 820.

Su altri calcolatori si può raggiungere lo stesso scopo facendo seguire tutti i nomi di variabile dal carattere % rappresentativo del tipo "intero". Questo metodo però è piuttosto noioso e comporta un notevole spreco di spazio in memoria. Poiché tuttavia è proprio questo il metodo maggiormente in uso, abbiamo preferito utilizzare delle variabili reali.

LA MESSA A PUNTO

La messa a punto di un programma come questo, già di una certa consistenza, deve essere piuttosto accurata. Ricordiamo che per messa a punto si intende "il raggiungimento di un programma conforme all'obiettivo che ci si era prefissati" e non di "un programma perfetto" che nel nostro caso starebbe a significare un programma che rappresenti un'ottima strategia per il gioco del 421.

La messa a punto si effettua per blocchi contemporaneamente alla codifica. In una prima fase non ho codificato che il "carico", comprendente i seguenti blocchi di istruzioni: 10–299; 700-950; 1000-1199; 5800-5999; 9000-9389; 9950-9996 e mi sono poi occupato della sua messa a punto.

In questa prima fase è stato possibile sistemare definitivamente:

- la determinazione del vincitore e del perdente in un giro

- l'uso dello spareggio

- l'attribuzione dei gettoni

e inoltre la trasformazione dei valori ottenuti per tre dadi in una combinazione del gioco; il calcolo del suo valore; ... . Tutte queste funzioni permettono di avere un certo numero di routines "sicure" che possono essere utilizzate anche nella messa a punto delle altre parti del programma.

Le operazioni di input-output sono piuttosto numerose. È inutile pertanto inserirne delle nuove di controllo! È sufficiente munirsi di carta e penna ed annotarsi i risultati man mano che questi compaiono sul video.

In una fase successiva sarà invece utile generare a priori delle situazioni particolari. Ad esempio con un'istruzione del tipo:

5805 INPUT "D1, D2, D3", D1, D2, D3

si può generare direttamente una particolare combinazione,

Una volta ultimata la messa a punto del "carico" sono passato a controllare la funzione di "scarico".

Lo "scarico" è stato codificato in blocchi di istruzioni progressivi che possono essere individuati dalle loro istruzioni iniziale e finale.

Ad esempio:

```
2000  REM
2999  RETURN
3000  REM
3999  RETURN
4000  REM
4999  RETURN
```

Questa scomposizione ha permesso di evidenziare tutta l'architettura della funzione di "scarico" ed in particolare la parte riguardante il giocatore umano (istruzioni dalla 5500 alla 5799).

Prendendo un insieme di 5 o 6 giocatori di cui uno simulato dal calcolatore (AUTOMA) e gli altri umani (TIZIO, CAIO, ...) la messa a punto è stata facilitata dall'uso dell'istruzione 5805 già citata.

Si sono in seguito controllati separatamente i sottoprogrammi che realizzano le tre diverse strategie: per il primo giocatore (blocco (30)), per il giocatore intermedio (34) e per l'ultimo giocatore (33).

Poi si è fatta una prova di lunga durata: si è impostata una partita assegnando inizialmente 50 gettoni a ciascun giocatore e denominando tutti i giocatori come AUTOMA.

Lanciato il programma ho potuto così dedicarmi ad altre attività: mangiare, ricevere amici, ... Il calcolatore lavorava instancabilmente e si fermava solo quando trovava qualche errore sintattico. In questo modo ho potuto facilmente eliminare tutti gli errori di battitura. (E ve ne erano parecchi!).

La fase successiva è consistita nel rilanciare il programma corretto annotando però i vari risultati, man mano che la partita procedeva.
(Giocando sull'istruzione 9950 si può influire sulla velocità di esecuzione e sulla leggibilità dei risultati).

Durante la messa a punto si sono fatte delle scoperte sorprendenti: non è possibile scrivere 8000 caratteri senza commettere errori (probabilmente non sarei un buon dattilografo!).

In queste ultime fasi della messa a punto non ho fatto ricorso alle istruzioni del tipo illustrato della 5805 in quanto l'esperienza mi ha provato che, essendo stato già controllato tutto in precedenza, sarebbe stato inutile.

Dopo la messa a punto del programma si può cercare di ottimizzare la strategia di gioco, utilizzando ad esempio più giocatori automatici che utilizzino strategie diverse, detti AUTOMA, AUTOMA1, AUTOMA2, AUTOMA3....

Si potrebbe poi scrivere una strategia diversa per ogni giocatore (che coraggio!) e farli giocare tutti una stessa partita, assegnando in partenza un elevato numero di gettoni (da 50 a 100) in modo da rendere poco influente sul risultato finale la fase di carico. Annotando i risultati (quale giocatore ha vinto, il punteggio finale ottenuto da ciascuno dei giocatori, ...) relativo a diverse partite si potrà alla fine estrarre una buona strategia. (occorre comunque anche un minimo di ragionamento per scegliere la strategia migliore!).

In pratica nel programma il test (28) sarebbe rimpiazzato con un test a più uscite. Inoltre in questo nuovo test non deve comparire il ramo relativo ad un giocatore umano.

CONCLUSIONE

In programmi di questo genere i vantaggi del metodo di programmazione da noi proposto sono più percepibili. È stato infatti necessario fare uno sforzo notevole prima di poter scrivere anche una sola riga di programma. Ma questo sforzo si è dimostrato benefico al momento della messa a punto, facilitata moltissimo da una codifica coerente. (Un errore di battitura nei DATA, ad esempio, è stato scoperto molto rapidamente!).

La realizzazione di routines, come quella per l'attribuzione dei gettoni o per gli spareggi, messe a punto su un programma semplice come il "carico", ci ha aiutato durante la messa a punto della parte relativa allo "scarico", molto più complessa della precedente.

Infine, questo metodo permette di migliorare in seguito il programma, dato che la sua modularità permetterebbe di provare facilmente molte strategie diverse (almeno quante ce ne può fornire la nostra immaginazione), purché si abbia il coraggio di tradurle in programma!!!

CAPITOLO 8

LA CONTABILITA' PERSONALE

Questo esempio è stato introdotto per illustrare un programma di tipo gestionale. È un tipo di programma che può venir realizzato su un gran numero di calcolatori. Questo tipo di programma è inoltre facilmente realizzabile su dei piccoli Personal Computers, mentre ad esempio sugli stessi elaboratori un programma di tipo scientifico sarebbe eseguito con grande lentezza. Per un programma di tipo scientifico di una certa mole si dovrebbe utilizzare un elaboratore di tipo diverso, che generalmente ha un prezzo più elevato.

Il tipo di contabilità da noi proposta è poi piuttosto elementare in modo da permettere un corretto approccio ai problemi di tipo gestionale pur mantenendo il programma di dimensioni tali da poter essere convenientemente usato su un Personal Computer.

Per impostare il problema abbiamo ritenuto opportuno riportare brani di un dialogo avvenuto tra **un utente (U)** e il **realizzatore del programma (R)**.

L'utente è colui che ha una certa necessità e vorrebbe risolvere i suoi problemi con l'uso di un elaboratore.

Il realizzatore invece deve capire ciò che l'utente spera di ottenere, per poterne tradurre adeguatamente i bisogni, e realizzare un prodotto che funzioni in modo conveniente.

Alla fine di ogni passo verrà effettuato un **bilancio**, facendo il punto del dialogo, riassumendone le parti meno importanti e commentando per il lettore i passi salienti.

Il dialogo proposto si ispira alle esperienze personali che l'autore ha avuto in questo campo.

U : ha espresso il desiderio di utilizzare un Personal Computer per la sua contabilità personale. Vediamo il seguito del dialogo:

R : *Che cosa volete fare?*

U : Sapere quale è la mia situazione finanziaria

R : *Ma cosa volete controllare?*

U : I miei diversi conti correnti bancari e postali, i libretti di risparmio in tutte le loro forme

R : *Ma vi è veramente utile seguire i vari movimenti sui libretti di risparmio, visto che dopo ogni operazione vi viene evidenziato il saldo?*

U : Oh! Sì... No...

R : *Non avreste un'altra idea?*

U : Ma sì! Quello che mi interessa è di non rischiare di emettere degli assegni scoperti; per questo mi serve sapere quando occorre effettuare dei giri da un conto all'altro od eventualmente chiedere un prestito alla banca per un acquisto importante.

R : *Per conoscere sempre il totale disponibile occorre però tenere in debito conto le varie tratte emesse e le fatture periodiche che vengono pagate ed addebitate automaticamente.*

U : È ciò che sto dicendo dall'inizio! Voglio avere tutti gli elementi per poter conoscere l'evolversi della mia situazione finanziaria.

Questo è lo stesso tipo di dialogo a cui si assiste spesso: negli scambi verbali tra i due interlocutori, le stesse parole non esprimono esattamente lo stesso concetto. Se il programmatore ha un minimo di esperienza, sa che deve andare sistematicamente a fondo delle cose facendosi spiegare tutti i punti dubbi.

L'utente ha spesso l'impressione di avere a che fare con una persona che "vuol tagliare in quattro un capello", e tende ad innervosirsi.

Lo stesso fenomeno può verificarsi anche quando le due parti sono fatte da una stessa persona:
- da una parte la persona che svolge il proprio lavoro abituale, a suo agio e tranquillamente,
- dall'altra la stessa persona che si accosta per la prima volta all'informatica e non sa bene come operare, tendendo così a spazientirsi e ad entrare in "ebollizione".

Ma ritorniamo al dialogo:

R : *Ripeto le cose per precisarle:*
- *bisogna tenere aggiornato un certo numero di conti correnti e verificare dopo ogni operazione le reali disponibilità,*
- *bisogna anche tener conto a dedurre l'importo delle varie cambiali e pagamenti periodici che vengono fatti automaticamente dalla banca ed ad debitati sui vari conti*

U : Esatto! È proprio così.

R : *Quali sono i conti che si devono considerare?*

U : I diversi conti correnti bancari, i conti correnti postali, i libretti di risparmio e le carte di credito. Queste ultime infatti creano numerose difficoltà se non vengono utilizzate in modo appropriato!

R : *Esaminiamo ora ogni singolo tipo di conto: i conti correnti bancari e postali funzionano praticamente nello stesso modo. Il denaro arriva dall'esterno, sotto forma di bonifico o mediante dei versamenti di assegni e contanti effettuati allo sportello. Per noi tutto ciò rappresenterà del denaro che viene dall'esterno.*

U : Ma no! Non sempre ... Io faccio anche dei trasferimenti tra un conto e l'altro e non arriva niente dall'esterno.

R : *Volete allora che questi trasferimenti vengano considerati come una unica operazione, lasciando al programma il compito di elaborarla e trasformarla in due operazioni distinte?*

U : Oh! Se fosse possibile ... Quanto a me preferirei un'unica operazione, perchè mi capita spesso di accreditare l'importo su un conto dimenticandomi però di detrarlo sull'altro, e quando arrivano gli estratti conto dalla banca è spesso troppo tardi!!! ...

R : *Per i libretti di risparmio le cose sono praticamente uguali: si possono depositare assegni, fare giri, depositare o prelevare denaro contante. Solo non si possono emettere assegni.*

U : Si, è così.

R : *Per le carte di credito il discorso è più complesso?*

U : Oh sì! A tal punto che ormai non ne faccio più uso. Le adopero solo per i pagamenti in contanti che si fanno a fine mese, o nel mese successivo alla emissione della fattura.

R : *Vediamo di chiarire. Come si può sapere se il pagamento viene effettuato a fine mese o nel mese successivo?*

U : Beh, in genere una fattura emessa ad esempio prima del 10 di settembre viene saldata al 20 di settembre mentre quelle emesse tra l'11 di settembre ed il 10 di ottobre, vengono saldate al 20 di ottobre, e così via ...

R : *Questo è chiaro. Quindi bisogna sapere ogni volta che si usa la carta di credito la data e l'importo dell'operazione. Vi servono altre cose oltre alla tenuta dei conti?*

U : Attenzione! Per tenuta dei conti io intendo dire anche che devo poter esaminare i vari estratti conto della banca. E questo è terribile: infatti alcuni assegni emessi molto tempo prima a volte non sono ancora stati registrati mentre lo sono stati altri più recenti.
Vi sono poi dei prelevamenti fissi che vengono registrati nell'estratto conto anche se io non ho ancora ricevuto il denaro ...
Verificare questi estratti conto, anche solo per essere sicuri dei conti tenuti personalmente, diventa un incubo: a volte si riscontrano differenze di 10000 o 100.000 lire ...

R : *Si commettono facilmente infatti degli errori di riporto. Per verificare i conti vi proporrei di registrare le entrate e uscite relative ai vari conti. Cosa preferite? Che vi programmi la macchina affinché vi elenchi tutti gli assegni e tutte le operazioni di cui non abbiamo tenuto conto nell'analisi precedente, indicandovi se sono già state registrate o no; oppure volete trascrivere tramite la tastiera l'estratto conto ricevuto dalla banca, segnalando numero di assegno, importo, ...*

U : Non mi va bene nessuna delle due soluzioni. Ma dovendo scegliere preferirei la prima.

R : *Bene! Allora il calcolatore, a richiesta, vi fornirà il saldo dei vari conti. Se questo saldo quadra con quello fornito dalla banca siamo a posto.*

U : Ma se non quadra, non voglio dovere effettuare nuovamente il controllo manualmente.

R : *Dobbiamo creare un programma che parta dagli elementi aggiornati con l'ultimo estratto conto e poi, per ogni operazione, controlli l'esattezza degli importi, che voi spunterete sul vostro conto. Questo permette di trovare eventualmente delle operazioni registrate in modo errato o addirittura non registrate.*

U : Speriamo che non sia troppo complicato.

R : *Esaminiamo ora gli addebiti fatti direttamente dalla banca sui vari conti: Vi sono addebiti di importo vario, ma noto, come quelli effettuati per il pagamento delle bollette del telefono, del gas, ...*
Vi propongo di considerarli alla stregua degli assegni (anche se non sono numerati).

U : Si; va bene. Ma deve essere facile ...

R : *Terrò conto di questo desiderio imperativo di semplicità. Per gli addebiti automatici e regolari sarà ancora più semplice: basterà infatti definire all'inizio l'importo, la periodicità, la data di inizio, la durata, ... e tutto verrà effettuato automaticamente.*

U : E si potrà in questo modo evitare di essere scoperti quando si fanno dei prelievi?

R : *Includeremo nel programma alcuni elementi che permettano di verificare le conseguenze di un eventuale prelievo. Questo dovrebbe facilitare il tipo di diagnosi da lei richiesto.*

Prima di proseguire è utile trarre le conclusioni su ciò che dobbiamo sapere per portare a buon fine la tenuta dei conti.
In primo luogo è indispensabile poter *identificare i vari conti* su cui bisogna lavorare: identificare significa "riconoscere facilmente" quando si lavora con un calcolatore. Si possono utilizzare a questo scopo i numeri di conto, se si amano i numeri, o dei codici alfabetici: C.C. per un conto bancario, CCP per un conto postale, e così via, se si preferisce qualcosa di più semplice. (Ovviamente questo secondo metodo è efficace se si ha un solo conto per ogni tipo).

Bisogna ora caratterizzare le operazioni che si possono eseguire su un conto:

Addebito o accredito: Queste operazioni possono venire caratterizzate da una lettera alfabetica, da un numero con segno o da qualcos'altro.

Importo: Data la precisione necessaria occorre utilizzare un numero in virgola mobile, eventualmente in doppia precisione.

Descrizione: Ogni operazione deve essere caratterizzata da un nome riconoscibile dall'utente. Per esempio:

LUCE 1/TR
GAS GEN/FEB
STIP. APRILE

Negli esempi sono stati utilizzati per la descrizione al più 10 caratteri. Sarebbe possibile utilizzarne anche di più (15 o 20) ma si correrebbe il rischio di non avere poi una chiara visualizzazione di tutta l'operazione, date le limitate dimensioni delle linee sul video.

Data: Occorrono per lo meno 6 caratteri, utilizzando ad esempio la forma

11 12 79

per indicare il giorno 11 dicembre 1979. Se nel corso dell'elaborazione si devono fare delle ricerche e degli ordinamenti sulle date, conviene allora memorizzarle nella forma 79 12 11 che permette un veloce ordinamento. Con un semplice sottoprogramma si potrà, in fase di stampa, ritrasformarle nella forma usuale.

I caratteri 79 12 11 possono anche essere considerati nella forma numerica 791211 e venire quindi trattati come un numero reale. Potremo utilizzare in seguito quest'altra forma.

Questi sono gli elementi indispensabili per registrare correttamente un'operazione contabile. Può essere utile però aggiungere a questi altri elementi, che facilitino poi il controllo di un estratto conto, come ad esempio:

Data dell'operazione: È la data in cui l'operazione viene realmente effettuata. Essa viene utilizzata per effettuare diverse operazioni sugli estratti conto (mensili o altro). Per utilizzarla nel programma dovremo memorizzarla con lo stesso metodo utilizzato per le altre date.

Data di registrazione: È la data in cui viene registrata l'operazione sul conto ed inviata la relativa comunicazione al cliente.

Quest'ultimo elemento può facilitare notevolmente il controllo degli estratti conto periodici. Si potrebbe eventualmente mantenere solo questo elemento eliminando la data dell'operazione. Una scelta definitiva su questo argomento verrà fatta in sede di analisi.

Esaminiamo ora *i giri da un conto ad un altro*: in realtà questa operazione può essere fatta in due modi: o emettendo un assegno di un conto e versandolo poi allo sportello sull'altro conto, oppure dando direttaente alla banca l'ordine di effettuare lo spostamento di capitale.

L'esperienza dimostra che nel primo caso viene registrato prima il versamento e poi viene addebitato l'importo dell'assegno sull'altro conto, mentre nel secondo caso avviene l'inverso. Inoltre le date delle due operazioni risultano

essere diverse se i due conti sono aperti presso due banche diverse, mentre in genere coincidono se i conti sono aperti nella stessa banca.

Per non appesantire troppo la trattazione noi supporremo che in ogni caso le registrazioni coincidano con la data in cui viene effettuta la operazione.

Se per l'effettuazione dei giri tra conti si sceglie la seconda via, converrà prevedere nel programma un'operazione del tipo: "giro dal conto A al conto B", che, con un'unica scrittura da parte dell'utente, permetta di effettuare automaticamente le due registrazioni di addebito e di accredito rispettivamente sui conti A e B.

Questa soluzione, come si è detto, permette all'utente di trasmettere una sola volta al calcolatore le informazioni relative all'operazione in oggetto.

Le operazioni effettuate con l'uso delle carte di credito dovranno essere registrate su un conto speciale, sul quale verranno fatti di volta in volta gli opportuni addebiti.

Va notato che comunque che se si vogliono fare delle previsioni a breve termine, sarà opportuno considerare questi addebiti non dilazionati nel tempo, come sono in realtà, ma da effettuare immediatamente.

Ci restano ora da esaminare i cosiddetti *addebiti periodici*, che in genere corrispondono al rimborso rateizzato di un prestito. Queste operazioni sono caratterizzate oltre che dalla data di inizio anche dalla cadenza dei pagamenti.

Per esempio se per rimborsare una certa somma, i pagamenti vengono fatti al 7 di ogni mese, converrà definire:

La cadenza di pagamento: in questo caso mensile, ma potrebbe essere benissimo trimestrale o semestrale, ...

La scadenza nel singolo intervallo di tempo: il calcolo di questa data appare piuttosto complesso. Infatti nel caso di pagamenti trimestrali, la data di scadenza potrebbe essere il settimo giono del secondo mese del trimestre. Vi è comunque una soluzione alternativa: si può memorizzare la data di scadenza della prima rata e poi, tramite la nota periodicità, ottenere da questa le scadenze successive.

La data finale: è utile segnalare questa data, in cui si ha l'estinzione del debito, e per farlo si potrebbe ad esempio utilizzare il metodo già esposto dei 6 caratteri, anche se in questo caso sorgerebbero dei problemi per rappresentare date successive al 31/12/1999.

Questo elemento ci potrà servire sia per gestire in modo automatico gli addebiti relativi a questi pagamenti, sia per effettuare delle corrette previsioni sulla disponibilità finanziaria.

Prima di proseguire nella trattazione, vorremmo cominciare ad introdurre un grosso problema; quello del *controllo sulla validità e la correttezza dei dati trasmessi al calcolatore.*

Come abbiamo già più volte avuto occasione di ripetere, errare è umano, e per questo conviene premunirsi, per quanto possibile, contro eventuali errori, effettuando i dovuti controlli sui dati.

Esaminiamo ora, caso per caso, ciò che si può fare.

Codici di conto: una tabella prestabilita, che può essere modificata solo tramite delle operazioni speciali, permette di conoscere tutti i codici utilizzati per identificare i vari conti. Ogni qualvolta si inserisce un nuovo codice, questo, prima di venire accettato, viene confrontato con quelli già presenti nella lista.

Data: si possono effettuare su questo tipo di dato diversi controlli. Il numero rappresentante il mese, ad esempio, non può essere maggiore di 12 (si potrebbe addirittura scrivere il nome del mese in lettere ed effettuare poi il controllo sui caratteri, ma sarebbe piuttosto pesante). Per il giorno si può ricorrere ad un controllo molto raffinato che tenga conto del particolare mese a cui ci si riferisce: aprile ha 30 giorni, febbraio 28 (o 29 se l'anno è bisestile),...; oppure ad un controllo più grossolano: il numero dei giorni di un mese non supera mai 31.

Numero di un assegno: poiché ogni assegno appartiene ad un libretto di assegni, per rendere possibile un controllo basta introdurre come dati, per ogni conto, il numero del primo e dell'ultimo assegno di ogni libretto. Si può inoltre controllare che ogni numero di assegno venga utilizzato una sola volta.

Vi sono comunque dei dati sui quali è impossibile prevedere un controllo automatico. Tra questi il più importante è l'importo relativo ad una operazione. Poiché è bene effettuare un controllo su questo dato, si potrebbe richiedere che esso venga registrato più volte in momenti diversi.

Altri dati, come la descrizione dell'operazione, pur non potendo essere controllati, risultano però meno importanti.

Resta comunque un fatto essenziale: su ogni dato bisogna effettuare tutti i possibili controlli.

Ritorniamo ora al nostro dialogo:

U : Vi ho chiesto di tenermi aggiornata la contabilità, di aiutarmi a controllare gli estratti conto, ma vorrei anche avere un consuntivo delle spese sostenute.

R : *Vi sono due possibilità:*

- *una descrizione esauriente che per ogni addebito chiarisca a cosa si riferisce*

- *un codice contabile per ogni tipo di operazione. Potreste gestirvi questo codice come volete ma un mio consiglio è questo:*

  contrassegnare con un codice da 1 a 10 le spese correnti (vitto, vestiario, ...)

  da 11 a 20 le spese per la casa (affitto, spese condominiali, ...)

  da 21 a 30 le spese di trasporto

  da 81 a 90 le entrate (stipendi, premi di produzione, affitti attivi, ...)

  da 91 a 99 tutte le altre voci (trasferimenti di capitale, prestiti, donazioni, ...)

U : Quest'ultimo sistema mi piace. Ma come posso utilizzarlo?

R : *Basta inserire il codice insieme agli altri dati relativi all'operazione da registrare.*

U : Utilizzando questo metodo posso anche conoscere l'ammontare delle spese di un certo tipo effettuate in un mese?

R : *Sì, si può avere anche questo.*

La realizzazione

Per questo esempio non dettaglieremo tutta l'analisi, presentando i diagrammi a blocchi relativi ai vari algoritmi, ma ci limiteremo ad una esposizione di tipo generale, presentando poi il programma con la relativa documentazione.

I files

In problemi di questo tipo vengono utilizzati dei dati, come il totale disponibile su un conto, l'importo di un assegno emesso ma non ancora addebitato, ... che

devono rimanere memorizzati anche spegnendo il calcolatore, in quanto saranno utilizzati anche in momenti successivi.

Vi è un solo modo per ottenere ciò: copiare alla fine della giornata tutti i dati, o almeno quelli utili, su un file esterno in modo da poterli ritrovare il giorno successivo.

Come abbiamo detto nella prefazione, disponiamo di un registratore a cassetta come unico supporto esterno. Va notato comunque che la gestione dei files sarebbe notevolmente semplificata utilizzando altri supporti come floppy disck o altro.

Esaminiamo comunque le conseguenze derivanti dal ftto di utilizzare un unico registratore a cassetta:

- *non si possono leggere contemporaneamente i dati da una cassetta e scrivere i risultati su un'altra*: al più, prima si leggono tutti i dati da una cassetta e poi si scrivono i risultati su un'altra (o sulla stessa).

- *i dati di una cassetta possono essere letti solo nell'ordine in cui sono stati scritti:* non è possibile, dopo aver letto un certo numero di dati, rileggere un dato già letto facendo riavvolgere automaticamente il nastro (questa operazione è invece possibile su alcune unità nastro in funzione su grossi calcolatori).

- se si cerca un particolare dato, occorre *iniziare la ricerca dall'inizio del nastro e proseguire leggendo dato dopo dato, finché si trova quello ricercato.*

Da queste semplici osservazioni possiamo dedurre le seguenti regole pratiche per l'uso di una memoria a cassette magnetiche:

- *si legge dalla cassetta* un file e *si conservano in memoria tutte le parti utili,*

- *si copia il contenuto di un file presente in memoria*, completamente o in parte, *su una o più cassette.*

Pertanto durante l'esecuzione del programma in memoria deve essere presente il file completo su cui si intende operare. Anzi, vedremo che in pratica dovranno essere presenti in memoria tutti i files su cui si intende agire, e non uno solo per volta.

I diversi files utilizzati

Prendiamo ora in esame il problema relativo alla creazione dei files. Questa è senza dubbio la parte più significativa dell'esempio proposto.

Iniziamo dal **file degli assegni**, nel quale verranno registrati tutti i dati relativi a ciascun assegno emesso.

Viene spontaneo in primo luogo chiedersi se conviene creare un solo file o se invece sarebbe più utile crearne uno per ogni conto. Poiché l'uso di più files, anche se di ridotte dimensioni, comporta maggiori difficoltà pratiche nella gestione delle cassette magnetiche, ho optato per la prima soluzione. Utilizzeremo pertanto un solo file per tutti i conti.

Vediamo ora come deve essere organizzato il file.

Ciascun record (così si chiama un elemento di un file) deve contenere tutti i dati relativi ad un assegno ed in particolare:

- il *conto* su cui è stato emesso l'assegno,
- il *numero dell'assegno* (unico elemento utilizzato negli estratti conto),
- l'*importo* dell'assegno.

A questi tre dati essenziali possono essere aggiunti, per utilità:

- la *causale* dell'emissione,
- la *data* di emissione,
- un *codice contabile* (come definito nell'ultima parte del dialogo).

Come abbiamo detto ogni assegno deve riferirsi ad un particolare conto esistente. Per essere sicuri che, quando si registra un assegno, la definizione del conto sia corretta (si potrebbe battere erroneamente CPP e CPC anziché CCP) è necessario avere un **file dei conti** su cui effettuare le varie registrazioni.

Come deve essere strutturato un record di questo nuovo file? Deve contenere:

La *denominazione del conto*, che sarà generalmente data da una sigla: CCP, BNL, CI, ...

- l'*importo* disponibile sul conto dopo l'ultima operazione effettuata: questo dato, tenuto sempre aggiornato, permetterà di evitare l'emissione di assegni scoperti.

A questi dati essenziali se ne possono aggiungere altri che facilitano il controllo degli estratti conto (e che rendano la tenuta del conto sul Personal Computer conforme a quella effettuata dalla banca):

la *data dell'ultimo estratto conto* controllato,

l'*importo disponibile* alla fine di tale estratto conto.

Abbiamo poi accennato alla necessità di effettuare un controllo sul numero degli assegni. Per far ciò bisogna creare un file detto dei libretti di assegni che per ogni libretto registri in un record:

- il *numero del conto* a cui il libretto si riferisce,
- il *numero del primo assegno,*
- il *numero dell'ultimo assegno.*

Infine, se si vogliono utilizzare dei codici contabili, occorrerà predisporre una *lista dei codici contabili* ammessi.

Tutti questi files servono per il trattamento degli assegni. A questi va poi aggiunto:

Il file dei pagamenti rateali che deve contenere in ciascun record:

- il *conto* sul quale è addebitato il pagamento,
- la *periodicità* del pagamento (espressa in mesi),
- la *data dell'ultimo pagamento* effettuato,
- l'*importo* della rata,
- la *causale* del pagamento,
- la *data di chiusura del pagamento,*
- il *codice contabile.*

Per la registrazione di queste operazioni non occorrono altri files.

Ma come si può fare una registrazione di questo tipo in modo automatico? Bisognerà creare una "operazione" apposita, analoga a tante altre create in altri casi (giri-conto, abbuoni, bonifici, ...) che venga registrata solo alla scadenza di una tratta (o più in generale di un pagamento periodico).
Queste operazioni diverse, utilizzate in particolari casi, saranno elencate in un file, detto appunto *file delle operazioni contabili.*
Alcune di queste possono essere generate direttamente sulla tastiera, altre in modo indiretto: pagamenti periodici, giri da conto a conto, ...
Un'operazione viene tradotta in un record comprendente:

- il *conto* a cui si riferisce,
- la *data* in cui deve essere effettuata,
- l'*importo,*
- la *causale,*
- il *codice contabile.*

Abbiamo così descritto tutti i files che verranno utilizzati nel programma.

Per tradurre tutto in Basic occorre innanzitutto notare che tale linguaggio non ammette il tipo di variabile "record" e pertanto un record non può essere trattato come un dato singolo.

Perciò, invece di creare in memoria un vettore di "record", dovremo creare diversi vettori contenenti ciascuno un elemento (campo) del record (cosa che del resto viene fatta in modo automatico da quei compilatori come COBOL, PASCAL, PL/1 che riconoscono i records come variabili).

Le variabili utilizzate nel programma, relative ai vari files, sono:

File dei conti

Dimensione massima	D1
Indice massimo generato	C1
Nome	NB$ (C1)
Importo dopo l'ultima operazione	MF (C1)
Data dell'ultimo E. C.	DB (C1)
Importo sull'ultimo E. C.	MB (C1)

File dei libretti di assegni

Dimensione	D2
Indice massimo	C2
Codice di conto	BQ% (C2)
Numero del primo assegno	DQ% (C2)
Numero dell'ultimo assegno	FQ% (C2)

File dei codici contabili

Dimensione	D3
Indice massimo	C3
Codice contabile	CD% (C3)

File degli assegni

Dimensione	D4
Indice massimo	C4
Codice di conto	BC% (C4)
Numero dell'assegno	NC% (C4)
Causale dell'assegno	IC$ (C4)

Importo dell'assegno	MC (C4)
Data dell'assegno	DC (C4)
Codice contabile	CC% (C4)

File delle operazioni

Dimensione	D5
Indice massimo	C5
Codice di conto	BP% (C5)
Causale	IP$ (C5)
Importo	MP (C5)
Data	DP (C5)
Codice	CP% (C5)

File dei pagamenti periodici

Dimensione	D6
Indice massimo	C6
Codice di conto	BV% (C6)
Causale	IV$ (C6)
Importo	MV (C6)
Periodicità	PV% (C6)
Data dell'ultimo pagamento	DV (C6)
Data finale	LV (C6)
Codice contabile	CV% (C6)

OSSERVAZIONI

- I nomi dei conti figurano solo in NB$. In tutti gli altri files i conti vengono individuati tramite un codice che non è altro che il valore dell'indice relativo alla posizione del conto nel file dei conti.

- I codici contabili vengono assegnati con le seguenti regole: ogni codice è un numero intero compreso tra 1 e 99. A questo codice si deve aggiungere 200 nel caso in cui l'operazione sia un addebito (con questa convenzione si utilizzano solo importi positivi). Si è utilizzata questa soluzione perché permette un controllo più rigoroso ed evita delle ambiguità dovute al segno degli importi. (L'utente al momento di battere l'importo non deve più chiedersi quale sia il tipo di operazione a cui esso è associato ...). Al codice si deve invece aggiungere 1000 nel caso in cui l'operazione contabile sia già stata registrata su un estratto conto (in tal caso non deve essere conservata in memoria). Infine, se si tratta di un addebito differito (ad esempio proveniente da un'operazione effettuata con le carte di credito) si deve togliere al codice 200.

Le altre variabili utilizzate sono essenzialmente di lavoro:

Le variabili alfanumeriche **A$, B$, C$.**
A$ è usata nel modulo relativo al trattamento delle causali delle varie operazioni

Alcune variabili che possono assumere solo valori interi ma che sono considerate reali, non essendo seguite dal carattere %:
N1, N2, ..., N6

- N1 viene utilizzata nella ricerca del codice di conto,
- N2 viene utilizzata per il numero dell'assegno (ridotto alle ultime quattro cifre di destra),
- N3 viene utilizzata per il codice contabile.

I, TR

- I è un indice di ciclo,
- TR è una variabile di controllo utilizzata nelle ricerche. Vale 0 finché non si è trovato l'elemento cercato, poi vale 1.

J1, J2, J3

- vengono utilizzate per le date.

Alcune variabili reali che per coerenza dovrebbero però essere in doppia precisione:
M1, M2, M3, M4

- M1 è utilizzata per l'importo relativo ad una operazione,
- M2 per le date (qui non serve la doppia precisione).

Per presentare la documentazione ho utilizzato un criterio sistematico piuttosto classico, dando nell'ordine, per ogni parte del programma:

- Titolo del lavoro,
- Variabili utilizzate (e files),
- Sottoprogrammi o moduli utilizzati,
- Diagramma a blocchi con spiegazioni e note,
- Codifica.

Per poter comprendere una documentazione di questo tipo occorre prima avere un'idea dell'organizzazione generale del programma. Esso è formato da routines di programma e routines di servizio. Le routines di programma sono:

- una routine che serve per *effettuare le registrazioni degli assegni* e addebitare i relativi importi sui vari conti.

- una routine per il *trattamento dei giri-conto* che effettua gli opportuni addebiti e accredito sui vari conti.

- una routine per il *trattamento dei pagamenti rateali* che li registra sul file delle operazioni diverse senza fare operazioni sui conti (al più esegue un solo addebito).

- una routine che effettua la *registrazione delle operazioni diverse* sull'opportuno file e i movimenti sui vari conti (tranne che per gli addebiti differiti).

- una routine per *effettuare gli addebiti relativi ai pagamenti rateali*. Poiché questa routine visualizza anche la situazione contabile dei vari conti, si consiglia di utilizzarla per prima.

- una routine per la *verifica degli estratti conto bancari*.

- una routine per la *conservazione dei files su cassetta* (la lettura viene invece fatta automaticamente).

- una routine per l'*aggiornamento dei files* dopo il controllo di un estratto conto (vengono eliminate tutte le operazioni già registrate su tale conto).

- una routine per le *statistiche*: ad esempio segnalare l'ammontare di tutte le operazioni aventi lo steso codice contabile effettuate in un certo mese.

Più avanti verrà descritto l'uso di queste routines.

Vi sono poi nel programma anche delle parti che abbiamo chiamato funzioni di servizio e di cui parleremo tra breve.

Note

- Abbiamo definito routines di programma quelle parti di programma che servono per effettuare delle operazioni contabili come registrazioni, movimenti di conto, ...

- Abbiamo poi definito routines di servizio quelle parti di programma che effettuano delle operazioni più strettamente legate al calcolatore: copiatura di files da una cassetta ad un'altra, controllo sui dati, ...

Le routines di servizio

1 - *lettura e scrittura su files*

Vengono utilizzate tre routines di servizio:

- Routine per la lettura della cassetta (linee 9000 - 9199)
- Routine per la scrittura di files completi su cassetta (9600 - 9699)
- Routine per la scrittura di files aggiornati (9700 - 9930)

La routine di lettura richiama il modulo di dimensionamento per i vari vettori (inizio alla riga 9400).

Dati utilizzati: tutti i files

Variabili: **I** per tutte e tre le routines
N1, N2, N3 solo per la terza (scrittura di files aggiornati).

È utile notare a questo punto che le dimensioni massime per i files sono fisse (rispettivamente D1, D2, ..., D6).

Per attenuare questa rigidità in fase di lettura, per predisporre i vettori in memoria si possono modificare questi valori massimi in modo da adattarsi alla situazione reale (linee 9200 - 9260).

Quando un file sta espandendosi troppo, cioé quando si cerca di inserire in esso più dati di quanti ne possa contenere, vi è un sottoprogramma che blocca queste operazioni e copia il file su cassetta. Questo sottoprogramma (9600 - 9699) viene richiamato automaticamente.

La situazione anormale viene segnalata all'utente che, volendo, può modificare la dimensione massima e rileggere nuovamente il file da cassetta per continuare l'elaborazione interrotta. Va notato che la trascrizione del file su cassetta non distrugge i dati in memoria (mentre lo fa una eventuale lettura da nastro). Quindi, prima di rileggere il file, si può fare un controllo sullo spazio a disposizione in memoria per vedere se è possibile aumentare le dimensioni dei vettori relativi al file che ha generato il problema. Se la memoria è completamente occupata, per aumentare tale dimensionamento occorrerà ovviamente diminuire quello relativo a qualche altro file, oppure stralciare dal file alcune registrazioni non indispensabili e aggiornare il file su cassetta (in questo modo solo alcune registrazioni vengono copiate sul file e nessuna viene però cancellata dalla memoria).

Anche in fase di lettura, come abbiamo detto, è possibile modificare il dimensionamento con l'ausilio di un sottoprogramma di controllo.

Nota: Questi problemi sono strettamente legati all'uso del nastro magnetico. Ma occorre sapere che problemi analoghi esistono anche per i dischi: ogni supporto fisico esterno ha una dimensione finita e pertanto bisogna sempre prevedere la situazione limite, in cui non si ha più spazio disponibile.

```
9000 INPUT#-1,D1,D2,D3,D4,D5,D6:REM LETTURA DI TUTTI I FILES
9010 DM=D1+1:PRINT"IL N.MASSIMO DI CONTI UTILIZZABILE E' ";DM
9020 GOSUB 9200:D1=DM-1
9030 DM=D2+1:PRINT"IL N.MASSIMO DI LIBRETTI E' ";DM
9040 GOSUB 9200:D2=DM-1
9050 DM=D3+1:PRINT"IL N.MASSIMO DI CODICI E' ";DM
9060 GOSUB 9200:D3=DM-1
9070 DM=D4+1:PRINT"IL N.MASSIMO DI ASSEGNI E'";DM
9080 GOSUB 9200:D4=DM-1
9090 DM=D5+1:PRINT"IL N.  DI REGISTRAZIONI E' ";DM
9100 GOSUB 9200:D5=DM-1
9110 DM=D6+1:PRINT"IL N.  DI PAGAMENTI RATEALI E' ";DM
9120 GOSUB 9200: D6=DM-1
9130 GOSUB 9400
9140 INPUT#-1,C1
9141 FOR I=0 TO C1
9142 INPUT#-1,NB$(I),MB(I),DB(I),MF(I)
9143 NEXT I
9150 INPUT#-1,C2
9151 FOR I=0 TO C2
9152 INPUT#-1,B0%(I),D0%(I),F0%(I)
9153 NEXT I
9160 INPUT#-1,C3
9161 FOR I =0 TO C3
9162 INPUT#-1,CD%(I)
9163 NEXT I
9170 INPUT#1,C4
9171 FOR I=0 TO C4
9172 INPUT#-1,BC%(I),NC%(I),IC$(I),MC(I),CC%(I),DC(I)
9173 NEXT I
9180 INPUT#1,C5
9181 FOR I=0 TO C5
9182 INPUT#-1,BP%(I),IP$(I),MP(I),CP%(I),DP(I)
9183 NEXT I
9190 INPUT#-1,C6
9191 FOR I=0 TO C6
9192 INPUT#-1,PV%(I),BV%(I),IV$(I),MV(I),DV(I),LV(I),CV%(I)
9193 NEXT I
9195 LC=1:RETURN
9197 REM

9198 REM ------------------------------------------------
9199 REM
9200 INPUT "VOLETE MODIFICARE IL DIMENSIONAMENTO ";A$
9210 IF LEFT$(A$,1)="N" THEN RETURN
9220 INPUT "INSERIRE IL NUOVO VALORE ";I
9230 IF I>DM THEN DM=I: RETURN
9240 INPUT"VOLETE RIDURRE LA DIMENSIONE? ";A$
9250 IF LEFT$(A$,1)="S" THEN DM=I: RETURN
9260 GOTO 9220
9261 REM

9262 REM ------------------------------------------------
9263 REM
9400 DIM NB$(D1),MB(D1),MF(D1),DB(D1): REM DIMENSIONAMENTO
9410 DIM B0%(D2),D0%(D2),F0%(D2)
9420 DIM CD%(D3)
9430 DIM BB%(D4),NC%(D4),IC$(D4),MC(D4),CC%(D4),DC(D4)
9440 DIM BP%(D5),IP$(D5),MP(D5),CP%(D5),DP(D5)
9450 DIM PV%(D6),BV%(D6),IV$(D6),MV(D6),DV(D6),LV(D6),CV%(D6)
9490 RETURN
```

```
9498 REM ------------------------------------------------
9499 REM
9500 PRINT"SALVATAGGIO SU CASSETTA":GOSUB 8950
9510 PRINT#-1,D1,D2,D3,D4,D5,D6
9520 PRINT#-1,C1
9521 FOR I=0 TO C1
9522 PRINT#-1,NB$(I),MB(I),DB(I),MF(I)
9523 NEXT I
9530 PRINT#-1,C2
9531 FOR I=0 TO C2
9532 PRINT#-1,B0%(I),D0%(I),F0%(I)
9533 NEXT I
9540 PRINT#-1,C3
9541 FOR I=0 TO C3
9542 PRINT#-1,CD%(I)
9543 NEXT I
9550 PRINT#-1,C4
9551 FOR I=0 TO C4
9552 PRINT#-1,BC%(I),NC%(I),IC$(I),MC(I),CC%(I),DC(I)
9553 NEXT I
9560 PRINT#-1,C5
9561 FOR I=0 TO C5
9562 PRINT#-1,BP%(I),IP$(I),MP(I),CP%(I),DP(I)
9563 NEXT I
9570 PRINT#-1,C6
9571 FOR I=0 TO C6
9572 PRINT#-1,PV%(I),BV%(I),IV$(I),MV(I),DV(I),LV(I),CV%(I)
9573 NEXT I
9595 RETURN
9597 REM

9598 REM ------------------------------------------------------
9599 REM
9600 PRINT"ARRESTO A RAGGIUNGIMENTO SATURAZIONE"
9610 IF C1>D1 THEN C1=D1
9620 IF C2>D2 THEN C2=D2
9630 IF C3>D3 THEN C3=D3
9640 IF C4>D4 THEN C4=D4
9650 IF C5>D5 THEN C5=D5
9660 IF C6>D6 THEN C6=D6
9670 GOSUB 9500
9680 RETURN
9697 REM
9698 REM ------------------------------------------------
9699 REM
9700 PRINT"STRALCIO DEI FILES SU CASSETTA":GOSUB 8950
9710 PRINT#-1,D1,D2,D3,D4,D5,D6
9720 PRINT#-1,C1
9721 FOR I=0 TO C1
9722 PRINT#-1,NB$(I),MB(I),DB(I),MF(I)
9723 NEXT I
9730 PRINT#-1,C2
9731 FOR I=0 TO C2
9732 PRINT#-1,B0%(I),D0%(I),F0%(I)
9733 NEXT I
9740 PRINT#-1,C3
9741 FOR I=0 TO C3
9742 PRINT#-1,CD%(I)
9743 NEXT I
9750 N1=0
9751 FOR I=0 TO C4: CONTA LE REGISTRAZ.  DA STRALCIARE
```

```
9760 IF CC%(I)>1000 AND CC%(I)<4800 THEN N1=N1+1
9770 NEXT I
9780 TR=C4-N1:PRINT#-1,TR
9781 FOR I=0 TO C4: REM COPIA LE REGISTRAZIONI UTILI
9782 K=CC%(I)
9790 IF K>=1000 AND K<=4800 THEN 9800
9795 PRINT#-1,BC%(I),NC%(I),IC$(I),MC(I),CC%(I),DC(I)
9800 NEXT I
9810 N2=0
9815 FOR I=0 TO C5: REM CONTA LE REGISTRAZIONI DA STRALCIARE
9820 IF CP%(I)>1000 AND CP%(I)<4800 THEN N2=N2+1
9830 NEXT I
9840 PRINT#-1,(C5-N2)
9841 FOR I=0 TO C5: REM RICOPIA I DATI UTILI
9842 K=CP%(I)
9850 IF K>=1000 AND K<=4800 THEN 9860
9855 PRINT#-1,BP%(I),IP$(I),MP(I),CP%(I),DP(I)
9860 NEXT I
9870 N3=0
9875 FOR I=0 TO C6   REM STRALCIO
9880 IF CV%(I)>1000 THEN N3=N3+1
9890 NEXT I
9900 PRINT#-1,(C6-N3)
9905 FOR I=0 TO C6   REM SALVATAGGIO
9910 IF CV%(I)>1000 THEN 9920
9915 PRINT#-1,PV%(I),BV%(I),IV$(I),MV(I),DV(I),LV(I),CV%(I)
9920 NEXT I
9930 RETURN
9931 REM
9932 REM ------------------------------------------------
```

2 - **Routines di controllo sui vari elementi che compongono una registrazione**

Sono state realizzate varie routines che controllano ciascuna un elemento ed in particolare:

a. **Nome del conto:** controlla che il nome del conto battuto sulla tastiera esista sulla lista dei conti.

   Utilizza: **A$,** letto da tastiera, **NB$, I, TR.**

   Fornisce: **NI,** codice del conto.

   Richiama: Il sottoprogramma per la generazione di un nuovo conto (2300)

   Nota: la routine cerca il conto in NB$ in modo sequenziale. Se non lo trova chiede chiarimenti all'operatore e se richiesto aggiunge il nome nell'elenco NB$ (utilizzando il sottoprogramma che inizia a 2300).

   Istruzioni 2400 - 2490

b. **Numero dell'assegno:** controlla che il numero fornito appartenga ad un libretto di assegni registrato sul conto in oggetto

Utilizza: **A$, N2** (per il numero dell'assegno)
N1 (codice del conto)
BQ%, DQ%, FQ%

Nota: Controlla il numero di assegno ed eventualmente registra un nuovo libretto mediante un opportuno sottoprogramma (1750).

Istruzioni 1650 - 1740.

c. **Codice contabile:** controlla che il codice inserito sia accettabile e appartenga alla lista dei codici conosciuti.

Utilizza: **N3; CD%**

Nota: Se il codice non è conosciuto, viene aggiunto, ovviamente a richiesta, alla lista (sottoprograma per la creazione di un codice (2150)).

Istruzioni 2000 - 2090

d. **Data:** controlla sommariamente la correttezza del giorno (accetta il 31 febbraio), del mese e dell'anno.

Utilizza: **J1, J2, J3**

Fornisce: **M2**

Istruzioni 2200 - 2280

e. **La periodicità dei pagamenti:** riconosce una delle parole chiave ammesse tramite un controllo sui primi due caratteri e la trasforma in codice numerico.

Utilizza: **B$, C$, TR.**

Fornisce: **N2** (durata in mesi).

Istruzioni 3600 - 3730.

f. **Data di inizio e di fine della rateizzazione:** raccoglie le date e le dispone in memoria a beneficio del programma.

Utilizza: **J1, J2, J3, M2.**

Fornisce: **M3** data del primo pagamento
M4 data dell'ultimo pagamento.

Richiama: la routine per il controllo delle date (2200)

Istruzioni 3750 - 3780

```
2400 INPUT "DENOMINAZIONE CONTO RICHIESTO";A$:TR=0
2410 FOR I=0 TO C1
2415 IF A$=NB$(I) THEN N1=I:TR=1
2420 NEXT I
2430 IF TR=1 THEN RETURN
2440 PRINT A$;" NOME NON NOTO"
2450 INPUT"E' IL NOME DI UN NUOVI CONTO?" B$
2460 IF LEFT$(B$,1)="N" THEN 2400
2470 GOSUB 2300
2480 N1=C1
2490 RETURN
2491 REM
2492 REM ----------
2493 REM
1600 GOSUB 1900:D$=A$:RETURN
1610 GOSUB 2200:DA=M2:RETURN
1611 REM
1612 REM ----------
1613 REM
1650 INPUT"NUMERO DELL'ASSEGNO ";B$
1660 N2=VAL(RIGTH$(B$,4)):TR=0
1670 FOR I=0 TO C2
1680 IF B0%(I)=N1 AND N2<=F0%(I) AND N2>=D0%(I) THEN TR=1
1690 NEXT I
1700 IF TR=1 THEN RETURN
1710 PRINT"L'ASSEGNO NON APPARTIENE AI LIBRETTI REGISTRATI"
1720 INPUT"SI TRATTA DI UN NUOVO LIBRETTO? "B$
1730 IF LEFT$(B$,1)="N" THEN 1650
1740 GOSUB 1750:RETURN
1741 REM
1742 REM ----------
1743 REM
2000 INPUT"CODICE CONTABILE";N3:N3=INT(N3+.1)
2010 IF N3<1 OR N3>99 THEN PRINT "CODICE ERRATO":GOTO 2000
2020 TR=0
2025 FOR I=0 TO C3
2030 IF N3=CD%(I) THEN TR=1
2040 NEXT I
2050 IF TR=1 THEN RETURN
2060 PRINT"QUESTO CODICE NON ESISTE"
2070 INPUT"SI TRATTA DI UN NUOVO CODICE?";B$
2080 IF LEFT$(B$,1)="S" THEN GOSUB 2150:RETURN
2090 GOTO 2000
2091 REM
2092 REM ----------
```

```
2200 INPUT"DATA DI EMISSIONE:GIORNO,MESE,ANNO";J1,J2,J3
2210 GOSUB 2250
2220 M2=10000*J3+100*J2+J1
2230 RETURN
2250 IF J1>=1 AND J1<=31 THEN 2260
2251 PRINT"GIORNO ERRATO":INPUT"RISCRIVERE,PREGO";J1:GOTO 2250
2260 IF J2>=1 AND J2<=12 THEN 2270
2261 PRINT"MESE ERRATO":INPUT"RISCRIVERE PREGO";J2:GOTO 2260
2270 IF J3>=78 AND J3<=99 THEN RETURN
2271 PRINT"ANNO ERRATO":INPUT"RISCRIVERE PREGO";J3:GOTO 2270
2278 REM
2279 REM ------------------------------------------------------
2280 REM
3600 PRINT"PER DESCRIVERE LA PERIODICITA' DELLA RATEAZIONE"
3610 PRINT"INDICARE SE ANNUALE,SEMESTRALE,TRIMESTRALE"
3620 PRINT"BIMESTRALE,MENSILE"
3630 PRINT"INSERIRE I PRIMI DUE CARATTERI DELLA COMBINAZIONE"
3640 INPUT B$
3650 C$=LEFT$(B$,2):N2=0
3660 IF C$="AN" THEN N2=12
3670 IF C$="SE" THEN N2=6
3680 IF C$="TR" THEN N2=3
3690 IF C$="BI" THEN N2=2
3700 IF C$="ME" THEN N2=1
3710 IF N2<>0 THEN RETURN
3720 PRINT"I PRIMI DUE CARATTERI DEL CODICE NON RAPPRESENTANO"
3730 PRINT"ALCUNA DELLE COMBINAZIONI DI RATEAZIONE NOTA"
3740 PRINT"RIPETERE,PREGO":GOTO 3600
3741 REM
3742 REM ------------------------------------------------------
3743 REM
3750 INPUT"SCADENZA PRIMA RATA:GIORNO,MESE,ANNO";J1,J2,J3
3760 GOSUB 2210:M3=M2:RETURN
3770 INPUT"DATA ULTIMA RATA:GIORNO,MESE,ANNO":J1,J2,J3
3780 GOSUB 2210:M5=M2:RETURN
3781 REM
3782 REM-- ------------------------------------------------------
```

3 - **Creazione di nuovi elementi**

Per aggiungere nuovi elementi nei files dei conti, dei codici contabili, degli assegni o altro bisogna usare alcune precauzioni:

Sono state pertanto generate apposite routines che permettono di effettuare queste operazioni.

a. **Apertura di un nuovo conto:** verifica che tale operazione sia possibile (problema relativo al dimensionamento del file) e raccoglie tutti i dati necessari per effettuare l'operazione.

   Utilizza: **A$**, trasmesso dal programma, che contiene il nome del nuovo conto.

   C1, NB$, MB, DF, DB per aggiungere un elemento nel file dei conti

   M1: importo disponibile sul conto
   M2: data di apertura del conto

   Fornisce: il file dei conti aggiornato

   Richiama: il programma per il controllo del dimensionamento (9600)

   Istruzioni 2300 - 2390

b. **Registrazione di un nuovo libretto di assegni:** verifica che tale operazione sia possibile e che nel libretto sia compreso l'assegno utilizzato dal programma chiamante.

   Utilizza: **N2**, trasmesso dal programma, che contiene il numero dell'assegno usato dal programma stesso.

   B$, C$, N4, N5 per i numeri del primo ed ultimo assegno in caratteri e in cifre.
   C2, BQ%, DQ%, FQ%, per aggiornare il file

   Richiama: il sottoprogramma per il controllo del dimensionamento (9600).

   Istruzioni 1750 - 1830

c. **Generazione di un nuovo codice contabile:** esegue l'operazione dopo aver verificato che essa sia possibile.

Utilizza: **N3** nuovo codice
C3 e **CD%** per aggiornare il file

Richiama: il sottoprogramma per il controllo del dimensionamento (9600)

Istruzioni 2150 - 2180

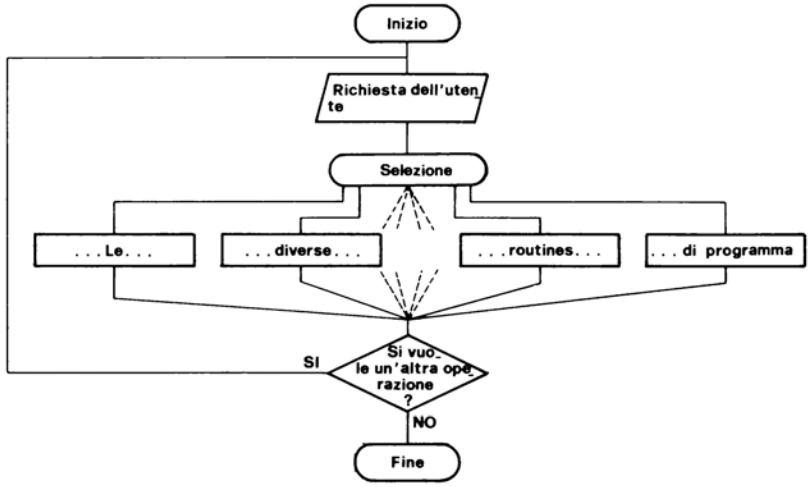
```
2300 C1=C1+1
2310 IF C1>D1 THEN PRINT"PIU' CONTI DEL PREVISTO":GOSUB 9600
2320 INPUT "SALDO DEL CONTO";M1
2330 INPUT"DATA APERTURA DEL CONTO:GIORNO,MESE,ANNO";J1,J2,J3
2340 GOSUB 2250: M2=10000*J3+100*J2+J1
2350 NB$(C1)=A$
2360 MB(C1)=M1
2370 MF(C1)=M1
2380 DB(C1)=M2
2390 RETURN
2392 REM ------------------------------------------------------------


1750 C2=C2+1
1760 IF C2>D2 THEN PRINT"TROPPI LIBRETTI": GOSUB 9600
1770 PRINT"INSERIRE IL NUMERO DEL PRIMO E ULTIMO ASSEGNO"
1780 INPUT B$,C$
1790 N4=VAL(RIGTH$(B$,4)):N5=VAL(RIGTH$(C$,4))
1800 IF N2>=N4 AND N2<=N5 THEN B0%(C2)=N1
1801 D0%(C2)=N4: F0%(C2)=N5
1802 RETURN
1810 PRINT "L'ASSEGNO NON APPARTIENE AL LIBRETTO"
1820 PRINT "PER FAVORE RIBATTERE I DATI"
1830 GOTO 1770
1831 REM
1832 REM ------------------------------------------------


2150 C3=C3+1
2160 IF C3>D3 THEN PRINT "PIU' CODICI DEL PREVISTO":GOSUB 9600
2170 CD%(C3)=N3
2180 RETURN
2181 REM
2182 REM ------------------------------------------------------------
```

Il selezionatore di routines

È rappresentato dalla parte iniziale del programma, che seleziona le diverse routines da eseguire su richiesta dell'utente.



L'organigramma generale

Figura 62

Utilizza: **A$, TR** e le varie routines di programma

Nota: Se la memoria centrale è piuttosto limitata, si potrà richiedere un solo tipo di operazione per volta, cosa che comporta numerose manipolazioni da parte dell'utente. Si potrebbe però almeno dividere il programma in due parti: una per le registrazioni e gli aggiornamenti contabili e l'altra per le statistiche, i controlli degli estratti conto e le eliminazioni di quelle scritture ormai superflue.

```
10 CLEAR 650: CLS: LC=0:TR=0
20 PRINT "LISTA DELLE OPERAZIONI PERMESSE"
30 PRINT "PER AGGIUNGERE UN ASSEGNO BATTERE 'CH'"
40 PRINT "PER AGGIUNGERE UN GIRO-CONTO BATTERE 'VR'"
50 PRINT "PER AGGIUNGERE UN PAGAMENTO RATEALE BATTERE 'VS'"
60 PRINT "PER VERIFICARE UN ESTRATTO CONTO BATTERE 'CP'"
70 PRINT "PER AGGIORNARE UN CONTO BATTERE 'MAJ'"
80 PRINT "PER SALVARE SU CASSETTA BATTERE 'SV'"
90 PRINT "PER AGGIUNGERE DELLE REGISTRAZIONI BATTERE 'OP'"
100 PRINT"PER AGGIORNARE UN FILE BATTERE 'EL'"
110 PRINT"PER LE STATISTICHE BATTERE 'ST'"
120 PRINT:PRINT"PER FINE LAVORO BATTERE 'FIN'"
130 PRINT: INPUT "COSA VOLETE FARE"; A$
131 REM
132 REM ------------------------------------------------------
300 IF A$="CH" THEN TR=1
310 IF A$="VR" THEN TR=2
320 IF A$="VS" THEN TR=3
330 IF A$="CP" THEN TR=4
340 IF A$="MAJ"THEN TR=5
350 IF A$="OP" THEN TR=6
360 IF A$="SV" THEN TR=7
370 IF A$="EL" THEN TR=8
380 IF A$="ST" THEN TR=9
390 IF A$="FIN"THEN TR=10
500 IF TR=0 THEN PRINT A$;"NON CONOSCIUTO":PRINT:GOTO 30
520 ON TR GOSUB 1000,4000,3000,6000,5000,5500,9500,9700,7000,550
530 INPUT "VOLETE EFFETTUARE ALTRE OPERAZIONI?"; A$
540 IF LEFT$(A$,1)="S" THEN PRINT: GOTO 30
550 END
```

Gli assegni: registrazione ed addebito

Questa routine realizza la registrazione relativa all'emissione di assegni e ne addebita l'importo sul relativo conto. Ciascuno dei dati utilizzati viene prima controllato tramite le relative routines di controllo.

Dopo il calcolo del saldo viene visualizzata la situazione e viene chiesta all'operatore una conferma dell'operazione. Viene così data all'utente la possibilità di modificare i dati introdotti. (Dopo l'esame dela situazione potrebbe

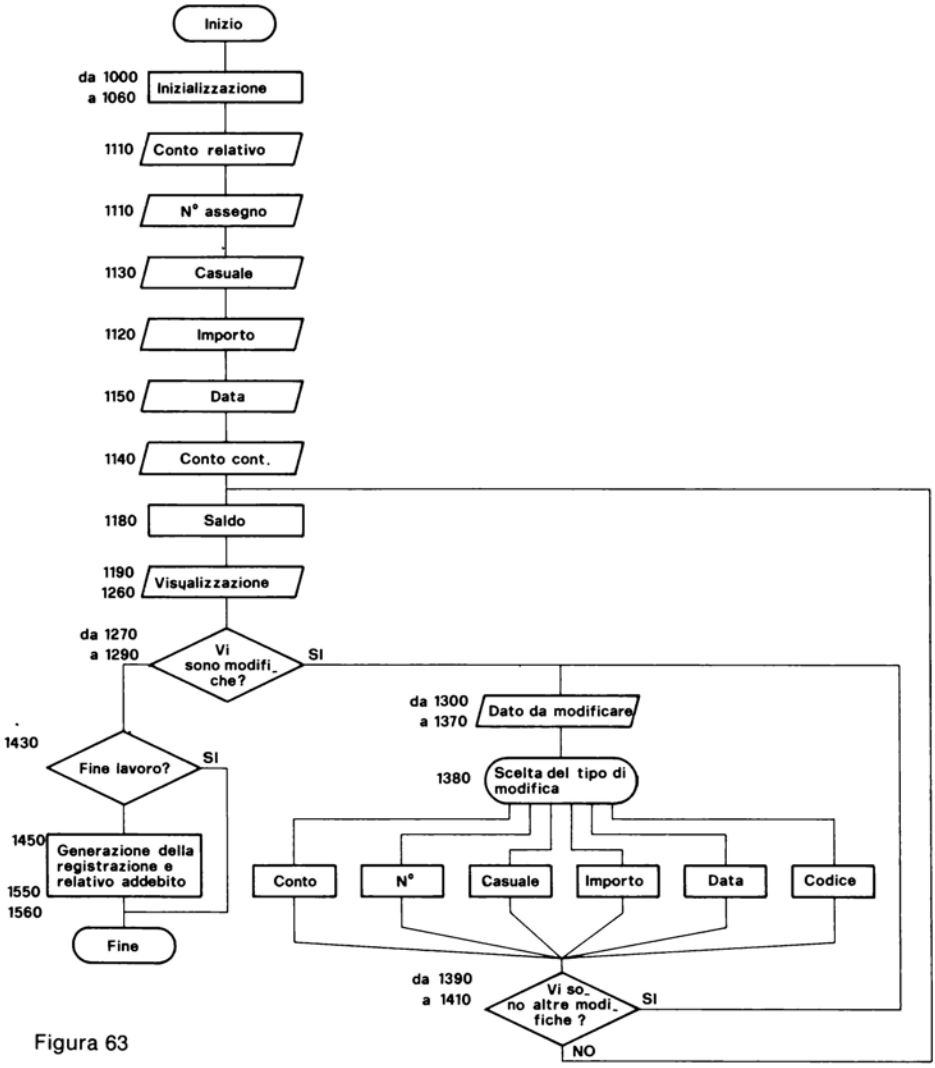


Figura 63

anche annullare tutta l'operazione). L'operazione contabile viene registrata sugli opportuni files solo dopo la conferma definitiva.

Utilizza:

- i files

- dei conti
- dei libretti di assegni
- dei codici contabili
- degli assegni

Utilizza inoltre delle variabili di lavoro e le seguenti **routines di servizio**:
Routine di lettura di cassetta (eventualmente)
Routines di controllo (eccezion fatta per quella relativa ai pagamenti rateali).

Nota: questa routine è stata concepita in modo tale da poter essere utilizzata anche per l'inizializzazione dei vari files.

In questo modo si è potuto evitare di creare una apposita routine per inizializzare i files quando si utilizza il programma per la prima volta.

```
1000 C1=-1:C2=-1:C3=-1:C4=-1:C5=-1:C6=-1
1010 IF LC<>0 THEN GOTO 1100
1020 INPUT"BISOGNA LEGGERE DA CASSETTA?";B$
1030 IF LEFT$(B$,1)="S" THEN GOSUB 9000:GOTO 1100
1040 GOSUB 9300
1050 NB$(0)="":BQ%(0)=-1:CD%(0)=-1:DQ%(0)=-1:FQ%(0)=-1
1060 PRINT:PRINT
1100 GOSUB 2400   REM ACQUISIZIONE CODICE CONTO
1110 GOSUB 1650   REM ACQUISIZIONE NUMERO ASSEGNO
1120 GOSUB 1850   REM ACQUISIZIONE IMPORTO
1130 GOSUB 1600   REM ACQUISIZIONE CAUSALE
1140 GOSUB 2000 : REM ACQUISIZIONE CODICE CONTABILE
1150 GOSUB 1610 : REM ACQUISIZIONE DATA
1180 M3=MF(N1)-M1
1181 REM
1182 REM ------------------------------------------------
1183 REM
1190 CLS:PRINT"RICAPITOLAZIONE"
1200 PRINT"CONTO ";TAB(25)NB$(N1)
1210 PRINT"NUMERO ASSEGNO";TAB(25) N2
1220 PRINT"CAUSALE";TAB(25) D$
1230 PRINT"IMPORTO DELL'ASSEGNO";TAB(25) M1
1240 PRINT"CODICE CONTABILE";TAB(25) N3
1250 PRINT"DATA D'EMISSIONE";TAB(25) DA
1260 PRINT"SALDO DEL CONTO";TAB(25) M3
1261 REM
1262 REM ------------------------------------------------
1263 REM
1270 PRINT:PRINT"VOLETE MODIFICARE QUALCHE DATO?"
1280 INPUT B$
1290 IF LEFT$(B$,1)="N" THEN 1430
1291 REM
1292 REM ------------------------------------------------
1293 REM
1300 PRINT"SE VOLETE MODIFICARE";TAB(25)"BATTETE"
1310 PRINT TAB(3) "IL CONTO";TAB(25)"0"
1320 PRINT TAB(3)"IL N DI ASSEGNO";TAB(25)"1"
1330 PRINT TAB(10)"LA CAUSALE";TAB(25)"2"
1340 PRINT TAB(11)"L'IMPORTO";TAB(25)"3"
1350 PRINT TAB(4)"IL CODICE CONTABILE";TAB(25)"4"
1360 PRINT TAB(15)"LA DATA";TAB(25)"5"
1370 PRINT:INPUT"VOSTRA DECISIONE";N4
1371 REM
1372 REM ------------------------------------------------
1373 REM
1380 ON N4+1 GOSUB 2400,1650,1600,1850,2000,1610
1390 INPUT"VOLETE FARE ALTRE MODIFICHE?",B$
1400 IF LEFT$(B$,1)="S" THEN 1300
1410 GOTO 1180
1411 REM
1412 REM ------------------------------------------------
1413 REM
1430 INPUT"VOLETE ANNULLARE L'OPERAZIONE?";B$
1440 IF LEFT$(B$,1)="S" THEN 1560
1450 C4=C4+1
1460 IF C4>D4 THEN PRINT"TROPPI ASSEGNI":GOSUB 9600
1470 BC%(C4)=N1: REM    CREAZIONE
1480 NC%(C4)=N2
1490 IC$(C4)=A$
```

```
1500 MC(C4)=M1
1510 CC%(C4)=N3
1520 DC(C4)=DA
1530 MF(N1)=M3
1540 INPUT"ALTRI ASSEGNI DA REGISTRARE?";B$
1550 IF LEFT$(B$,1)="S" THEN 1100
1560 RETURN
1561 REM
1562 REM ------------------------------------------------
1563 REM
1850 INPUT"IMPORTO DELL'OPERAZIONE";M1
1860 IF M1<=0 THEN PRINT"IMPORTO ERRATO":GOTO 1850
1870 RETURN
1871 REM
1872 REM ------------------------------------------------
1873 REM
1900 INPUT"CAUSALE";A$
1910 IF LEN(A$)<16 THEN RETURN
1920 A$=LEFT$(A$,15)
1930 PRINT"LA CASUALE VIENE CONSIDERATA ";A$
1940 INPUT"VA BENE?";B$
1950 IF B$="NO" THEN 1900
1960 RETURN
9299 REM ------------------------------------------------------
9300 INPUT"NUMERO DI CONTI PREVISTI ";N1:D1=INT(ABS(N1))-1          1))-1
9310 INPUT"NUMERO DI LIBRETTI ASS. PREVISTI ";N1:D2=INT(ABS(N1))-1
9320 INPUT"NUMERO DI CODICI PREVISTI ";N1:D3=INT(ABS(N1))-1        I))-1
9330 INPUT"NUMERO DI PAGAM. RATEALI ";N1:D4=INT(ABS(N1))-1
9340 INPUT"NUMERO DI ASSEGNI PREVISTI";N1:D5=INT(ABS(N1))-1
9350 INPUT"NUMERO DI OP. DIVERSE PREVISTO ";N1:D6=INT(ABS(N1))-1
9360 GOSUB 9400
9370 LC=1
9380 RETURN
9381 REM ------------------------- ----------------------------
```

Routine per il trattamento delle operazioni di giro da conto a conto

Partendo dai dati forniti per l'operazione di giro, la routine effettua due operazioni contabili distinte: una di accredito su un conto e una di addebito sull'altro.

Per questa routine è presentato solo un diagramma a blocchi di tipo generale, in quanto le specifiche sono del tutto simili a quelle presentate per la routine precedente.

Utilizza:	**I files**	**Le routines di controllo**
	- dei conti - dei codici contabili - delle operazioni diverse	tutte salvo quelle relative agli assegni e ai pagamenti rateali.

Prima dell'accettazione definitiva e della registrazione dell'operazione, è prevista la visualizzazione dei saldi che si avrebbero sui due conti dopo l'operazione stessa. La registrazione e la relativa modifica dei files interessati avviene solo dopo la conferma da parte dell'utente.

Istruzioni: 4000 - 4570

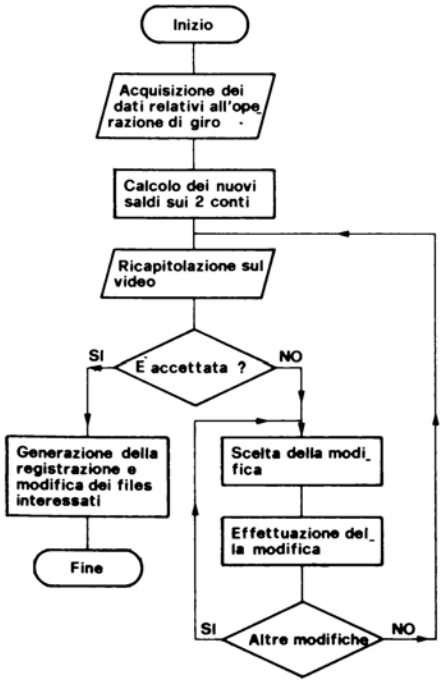


Figura 64

```
4000 IF LC=0 THEN PRINT"LETTURA DA CASSETTA":GOSUB 9000
4010 PRINT
4020 INPUT"IL CONTO DEBITORE";A$
4030 TR=0:GOSUB 2410:N2=N1
4040 INPUT"IL CONTO CREDITORE";A$
4050 TR=0:GOSUB 2410: REM ACQUISIZIONE DATI
4060 GOSUB 1600
4070 GOSUB 1850
4080 GOSUB 2000
4090 GOSUB 4750   REM FINE ACQUISIZIONE DATI
4091 REM
4092 REM-----------------------------------------------
4093 REM
4100 CLS:PRINT"RICAPITOLAZIONE"
4110 PRINT"CONTO DEBITORE";NB$(N2)
4120 PRINT"CONTO CREDITORE";NB$(N1)
4130 PRINT"CAUSALE ";D$
4140 PRINT"IMPORTO ";M1
4150 PRINT"DATA ";M2
4160 PRINT"CODICE CONTABILE ";N3
4170 M3=MF(N2)-M1
4180 M4=MF(N1)+M1
4190 PRINT"SALDO DEL CONTO ";NB$(N2);" =";M3
4200 PRINT"SALDO DEL CONTO ";NB$(N1);" =";M4
4201 REM
4202 REM ------------------------------------------------
4203 REM
4210 PRINT:INPUT"VOLETE MODIFICARE DEI DATI?";B$
4220 IF LEFT$(B$,1)="N" THEN 4380
4230 PRINT"SE VOLETE CAMBIARE";TAB(30) "BATTETE"
4240 PRINT TAB(3)"IL CONTO DEBITORE";TAB(30)"1"
4250 PRINT TAB(3)"IL CONTO CREDITORE";TAB(30)"2"
4260 PRINT TAB(3)"L'IMPORTO";TAB(30)"3"
4270 PRINT TAB(3)"LA CAUSALE";TAB(30)"4"
4280 PRINT TAB(3)"LA DATA";TAB(30)"5"
4290 PRINT TAB(3)"IL CODICE";TAB(30)"6"
4300 PRINT:INPUT"FATE LA SCELTA";N4
4310 ON N4 GOSUB 4600,4650,4700,1600,4750,2000
4320 GOTO 4100
4321 REM
4322 REM -----------------------------------------------
4323 REM
4380 INPUT"VOLETE ANNULLARE L'OPERAZIONE?";B$
4390 IF LEFT$(B$,1)="S" THEN RETURN
4400 C5=C5+1
4410 IF C5>D5 THEN PRINT"TROPPE OPERAZIONI":C5=C5-2:GOSUB 9600
4420 BP%(C5)=N2: REM REGISTRAZIONE
4430 IP$(C5)=D$
4440 MP(C5)=M1
4450 CP%(C5)=N3+200
4460 DP(C5)=M2
4470 BP%(C5-1)=N1
4480 IP$(C5-1)=D$
4490 MP(C5-1)=M1
4500 CP%(C5-1)=N3
4510 DP(C5-1)=M2
4520 MF(N2)=M3
4530 MF(N1)=M4
4550 INPUT"ALTRI GIRO-CONTI?";A$
4560 IF LEFT$(A$,1)="S" THEN 4000
```

```
4570 RETURN
4571 REM
4572 REM ------------------------------------------
4573 REM
4600 N5=N1: REM MODIFICA DEL CONTO DEBITORE
4610 INPUT"IL CONTO DEBITORE";A$
4620 GOSUB 2400
4630 N2=N1:N1=N5
4640 M3=MF(N2)-M1:RETURN
4650 INPUT"IL CONTO CREDITORE";A$:REM MODIFICA CONTO
4660 GOSUB 2400
4670 M4=MF(N1)+M1
4680 RETURN
4700 GOSUB 1850:REM MODIFICA IMPORTO
4710 M3=MF(N2)-M1
4720 M4=MF(N1)+M1
4730 RETURN
4750 INPUT"DATA:GIORNO,MESE,ANNO";J1,J2,J3:REM DATA
4760 GOSUB 2210
4770 RETURN
```

Routine per i pagamenti rateali

Questa routine serve solo per registrare i dati relativi ad un pagamento periodico che verranno poi utilizzati in seguito.

Dopo l'acquisizione dei dati relativi all'operazione viene presentato sul video il riepilogo della situazione.

Solo dopo eventuali modifiche e dopo la conferma definitiva da parte dell'utente si procede alla registrazione sull'apposito file e all'addebito dell'importo della prima rata sul conto interessato.

Utilizza: **I files** **La routine di controllo**

- dei conti
- dei codici contabili
- dei pagamenti rateali
- delle operazioni diverse

con particolare riguardo per quelle relative alla periodicità)

Nota: La data iniziale richiesta è quella relativa alla prima rata. Le scadenze successive vengono desunte matematicamente da questa: 1, 2, 3, ... mesi dopo, a seconda della periodicità.

Istruzioni: 3000 - 3590

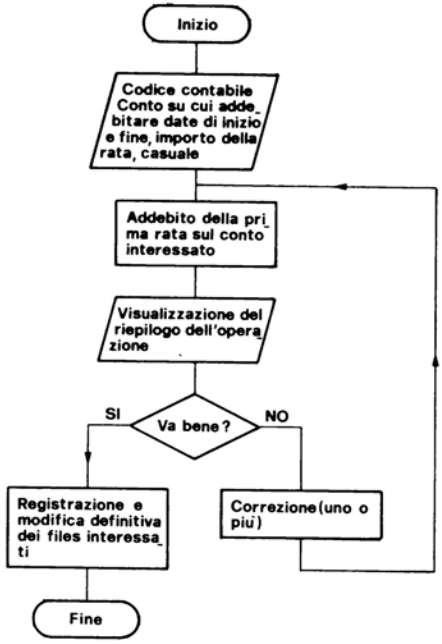


Figura 65

```
3000 IF LC=0 THEN PRINT"LETTURA DA CASSETTA":GOSUB 9000
3010 PRINT
3020 INPUT"IL CONTO SU CUI ADDEBITARE ";A$
3030 TR=0:GOSUB 2410
3040 INPUT"LA CAUSALE";A$:GOSUB 1910:D$=A$
3050 INPUT"L'IMPORTO ";M1
3060 IF M1<=0 THEN PRINT"IMPORTO ERRATO":GOTO 3050
3070 GOSUB 3750: REM DATA PRIMA RATA
3080 GOSUB 3770: REM DATA ULTIMA RATA
3090 GOSUB 3600: REM PERIODICITA'
3120 GOSUB 2000: REM CODICE CONTABILE
3130 M4=MF(N1)-M1
3141 REM
3142 REM ------------------------------------------------
3143 REM
3150 PRINT:PRINT"RICAPITOLAZIONE"
3160 PRINT"CONTO ";NB$(N1)
3170 PRINT"IMPORTO ";M1
3180 PRINT"CAUSALE ";D$
3190 PRINT"DATA PRIMA RATA ";M3
3200 PRINT"DATA ULTIMA RATA ";M5
3210 PRINT"PAGAMENTO OGNI ";N2;" MESI"
3220 PRINT"CODICE CONTABILE ";N3
3230 PRINT"NUOVO SALDO ";M4
3231 REM
3232 REM ------------------------------------------------
3233 REM
3250 PRINT"VOLETE MODIFICARE DEI DATI? ":INPUT B$
3260 IF LEFT$(B$,1)="N" THEN 3450
3270 PRINT"SE VOLETE CAMBIARE";TAB(30)"BATTETE"
3280 PRINT TAB(3)"IL CONTO";TAB(#0)"1"
3290 PRINT TAB(3)"LA CAUSALE";TAB(30)"2"
3300 PRINT TAB(3)"L'IMPORTO";TAB(30)"3"
3310 PRINT TAB(3)"LA PERIODICITA'";TAB(30)"4"
3320 PRINT TAB(3)"LA DATA PRIMA RATA";TAB(30)"5"
3330 PRINT TAB(3)"LA DATA ULTIMA RATA";TAB(30)"6"
3340 PRINT TAB(3)"IL CODICE";TAB(30)"7"
3360 PRINT:INPUT"FATE LA SCELTA";N4
3380 ON N4 GOSUB 2400,1600,1850,3600,3750,3770,2000
3390 INPUT"ALTRE MODIFICHE? ";B$
3400 IF B$="SI" THEN 3270
3410 GOTO 3140
3411 REM
3412 REM ------------------------------------------------
3413 REM
3450 INPUT"VOLETE ANNULLARE L'OPERAZIONE?";B$
3460 IF LEFT$(B$,1)="S" THEN 3590
3470 C5=C5+1
3480 IF C5>D5 THEN PRINT"TROPPE OPERAZIONI":GOSUB 9600
3490 C6=C6+1
3500 IF C6>D6 THEN PRINT"TROPPE RATEAZIONI":GOSUB 9600
3510 BV%(C6)=N1:BP%(C5)=N1: REM REGISTRAZIONE
3520 IV$(C6)=D$:IP$(C5)=D$
3530 MV(C6)=M1:MP(C5)=M1
3540 DV(C6)=M3:DP(C5)=M3
3550 LV(C6)=M5
3560 PV%(C6)=N2
3570 MF(N1)=M4
3580 CV%(C6)=N3:CF%(C5)=N3
3590 RETURN
```

La verifica di un estratto conto

Partendo dal saldo disponibile alla fine dell'estratto conto precedente, la routine esamina tutte le operazioni, sia di addebito che li accredito, effettuate su quel conto, contrassegnando in modo particolare quelle che sono registrate anche sull'estratto conto della banca.

In pratica la routine esamina tutti gli assegni e tutte le altre operazioni relative al conto, non ancora contrassegnate e per ciascuna chiede all'utente;

- se tale operazione figura nell'estratto conto
- se occorre effettuare una correzione

Dopo aver effettuato la correzione, l'operazione viene contrassegnata modificando provvisoriamente il codice contabile (vi si aggiunge 5000).

Una volta fatto ciò, il saldo ottenuto viene confrontato con quello dichiarato sull'estratto conto. Inoltre viene calcolato il saldo effettivo, tenendo conto anche di quelle operazioni non ancora registrate sull'estratto conto.

Se tra il totale della banca è quello calcolato vi è discrepanza si ripete il controllo. Se invece l'estratto conto risulta esatto, si passa dal contrassegno provvisorio a quello definitivo (si aggiunge al codice contabile 1000) aggiornando i vari files.

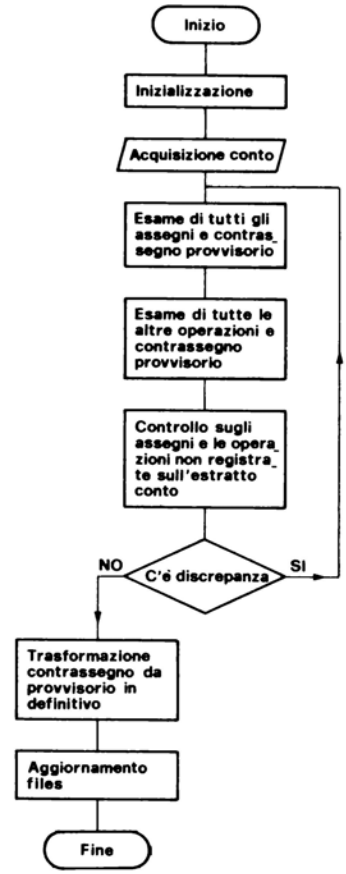


Figura 66

Utilizza:

I files

- dei conti
- degli assegni
- delle operazioni diverse

Le **routines di controllo** per i conti

Aggiorna i files:

DB, MB ed eventualmente **MF, CC, CP**

Nota per l'utente: Se, durante il controllo sugli assegni e sulle altre operazioni, si vuole modificare l'importo relativo, conviene rispondere alla domanda, formulata sul video, "C è sull'estratto conto" con una frase che inizi con la lettera C.

Istruzioni: 6000 - 6810

```
6000 IF LC=0 THEN PRINT"LETTURA DA CASSETTA":GOSUB 9000
6010 INPUT"NUME DEL CONTO DA ESAMINARE ";A$: REM IL CONTO
6020 TR=0
6030 FOR I=0 TO C1.IF NB$(I)=A$ THEN TR=1:N1=I:NEXT
6040 IF TR=0 THEN PRINT"CONTO NON APERTO":GOTO 6010
6050 PRINT"INSERIRE LA DATA DEL CONTROLLO PRECEDENTE":REM LA DATA
6060 INPUT"SOTTO LA FORMA:GIORNO,MESE,ANNO ";J1,J2,J3:REM PRECEDENTE
6070 GOSUB 2250:M1=10000*J3+100*J2+J1
6080 IF M1=DB(N1) THEN 6150
6090 PRINT"LA DATA INSERITA NON E' QUELLA DELL'ULTIMO ESTRATTO"
6100 PRINT"CONTO CONTROLLATO! VE NE SIETE DIMENTICATO QUALCUNO?"
6110 INPUT A$
6120 IF LEFT$(A$,1)="S" THEN 6050
6130 INPUT"VOLETE CONTINUARE UGUALMENTE? ";A$
6140 IF LEFT$(A$,1)="N" THEN RETURN
6150 PRINT"QUAL'E' LA DATA DI QUESTO ESTRATTO CONTO? ":REM DATA
J160 INPUT"INSERIRE GIORNO,MESE,ANNO ";J1,J2,J3:        REM ATTUALE
6170 GOSUB 2250:M2=10000*J3+100*J2+J1
6171 REM
6172 REM    ------------------------------------------------------------
6173 REM
6180 CLS:PRINT"ESAME DEGLI ASSEGNI":PRINT:REM  GLI ASSEGNI
6190 M3=MB(N1)
6200 PRINT"NUMERO";TAB(10)"CAUSALE";TAB(24)"IMPORTO";TAB(38)"CODICE"
6210 FOR I=0 TO C4
6220 IF BC%(I)<>N1 OR (CC%(I)>1000 AND CC%(I)<5000) THEN 6290
6230 PRINT NC%(I);TAB(6)IC$(I);TAB(22)MC(I);TAB(36)CC%(I)
6240 INPUT"C'E' SULL'ESTRATTO CONTO? ";B$
6250 IF LEFT$(B$,1)<>"S" THEN 6260
6255 M3=M3-MC(I):IF CC%(I)<5000 THEN CC%(I)=CC%(I)+5000
6260 IF LEFT$(B$,1)<>"C" THEN 6290
6270 INPUT "IMPORTO MODIFICATO ";M1
6280 M3=M3-M1:MF(N1)=MF(N1)+MC(I)-M1:MC(I)=M1
6285 IF CC%(I)<5000 THEN CC%(I)=CC%(I)+5000
6290 NEXT I
6291 REM
6292 REM ------------------------------------------------------
6293 REM
6300 PRINT"ESAME DELLE ALTRE OPERAZIONI"
6310 PRINT"CAUSALE";TAB(16)"IMPORTO";TAB(26)"CODICE"
6320 FOR I=0 TO C5
6325 K=CP%(I):TR=0
6330 IF BP%(I)<>N1 OR K<0 OR (K>800 AND K<4800) THEN 6440
6340 M4=MP(I):IF K>5200 OR (K>200 AND K<300) THEN K=K-200:M4=-M4:TR=1
6350 PRINT IP$(I);TAB(16)MP(I);TAB(27)K:IF TR=0 THEN PRINT " DEB"
6360 IF TR=1 THEN PRINT " CRED"
6370 INPUT"C'E' SULL'ESTRATTO CONTO ";B$
6380 IF LEFT$(B$,1)<>"S" THEN 6390
6385 M3=M3-M4:IF K<5000 THEN CP%(I)=CP%(I)+5000
6390 IF LEFT$(B$,1)<>"C" THEN 6440
6400 "IMPORTO CORRETTO ";M1:MP(I)=ABS(M1)
6410 INPUT"E' IN DEPOSITO O UN PRELIEVO";B$
6420 IF LEFT$(B$,2)="PR" THEN CP%(I)=K+200
6430 IF CP%(I)<5000 THEN CP%(I)=CP%(I)+5000
6440 NEXT I
6441 REM
6442 REM ------------------------------------------------------
6443 REM
6500 INPUT"SALDO DELL'ESTRATTO CONTO? ";M4:REM SALDO
```

```
6510 IF ABS(M3-M4)<.005 THEN 6700
6520 PRINT"IL SALDO CONTABILE NON COINCIDE"
6530 PRINT"VI SIETE DIMENTICATO QUALCHE ASSEGNO O OPERAZIONE?"
6540 PRINT"SE SI EFFETTUARE L'AGGIORNAMENTO USCENDO DAL PROGRAMMA"
6550 PRINT"E RICHIAMANDO POI LA ROUTINE NECESSARIA"
6560 INPUT"VOLETE USCIRE DAL PROGRAMMA? ";B$
6570 IF LEFT$(B$,1)="N" THEN GOTO 6130
6580 PRINT"ANNULLATO IL CONTROLLO DELL'ESTRATTO CONTO"
6590 RETURN
6600 REM
6610 REM ------------------------------------------------------------
6620 REM
6700 PRINT"L'ESTRATTO CONTO E' CORRETTO"
6710 MB(N1)=M4:DB(N1)=M2
6720 FOR I=0 TO C4
6730 IF CC%(I)>5000 THEN CC%(I)=CC%(I)-4000
6740 NEXT I
6750 FOR I=0 TO C5
6760 IF CP%(I)>5000 THEN CP%(I)=CP%(I)-4000
6770 NEXT I
6780 PRINT"IL CONTROLLO E' ULTIMATO E I DATI REGISTRATI"
6790 PRINT:INPUT"VOLETE CONTROLLARE ALTRI E.C.? ";A$
6800 IF LEFT$(A$,1)="S" THEN 6010
6810 RETURN
```

Aggiornamento dei conti

La routine, partendo dalla data corrente, esamina i files relativi ai pagamenti rateali e alle operazioni con le carte di credito ed effettua sui vari conti gli addebiti che non sono ancora stati registrati.

In pratica, dopo l'acquisizione della data corrente, viene effettuato un ciclo di controllo sui pagamenti rateali. Tenendo conto della periodicità di ogni singola rateizzazione, si controlla se nel periodo intercorso dalla data in cui è stata fatta l'ultima registrazione e quella corrente c'é qualche scadenza. In caso affermativo si procede alla registrazione dell'avvenuto pagamento e all'addebito dell'importo sul relativo conto.

Un procedimento analogo viene utilizzato per gli addebiti differiti (carte di credito).

Alla fine del processo vengono visualizzati i nuovi saldi contabili dei vari conti.

Osservazione: Questa routine può anche essere utilizzata per visualizzare solo i saldi relativi ai vari conti.

Utilizza: **I files**
- dei conti
- dei pagamenti rateali
- delle operazioni diverse

Istruzioni: 5000 - 5440

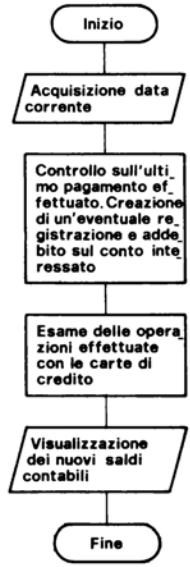


Figura 67

```
5000 IF LC=0 THEN PRINT "LETTURA DA CASSETTA": GOSUB 9000
5010 PRINT:PRINT"INSERIRE LA DATA PER L'AGGIORNAMENTO DEL CONTO"
5020 INPUT"SOTTO LA FORMA:GIORNO,MESE,ANNO"; J1, J2, J3
5030 GOSUB 2250: M1=10000*J3+100*J2+J1
5040 FOR I=0 TO C6:    REM        PAGAMENTI RATEALI
5050 M2=DV(I): J=PV%(I)
5060 L=INT(M2/10000):M3=M2 -10000*L
5070 M3=M3+100*J
5080 IF M3<=1250 THEN 5090
5081 M3=M3-1200: L=L+1:GOTO 5080
5090 M2=M3+10000*L
5100 IF M2>M1 THEN 5250
5110 IF M2>LV(I) THEN 5250
5120 C5=C5+1
5130 IF C5>D5 THEN PRINT"TROPPE OPERAZIONI":GOSUB 9600
5131 REM
5132 REM ------------------------------------------------
```

```
5133 REM
5140 K=BV%(I):  REM      REGISTRAZIONE DI OPERAZIONE
5150 BP%(C5)=K
5160 IP$(C5)=IV$(I)
5170 MP(C5)=MV(I)
5180 CP%(C5)=CV%(I)
5190 DP(C5)=M2
5200 DV(I)=M2
5210 MF(K)=MF(K)-MP(C5)
5220 PRINT"IL CONTO ";NB$(K);" HA IL SALDO ";MF(K)
5230 GOTO 5070
5250 NEXT I
5251 REM
5252 REM ------------------------------------------------
5253 REM
5260 FOR I=0 TO C5:  REM  CARTE DI CREDITO
5270 IF CP%(I)>0 THEN 5330
5280 IF DP(I)>M1 THEN 5330
5290 CP%(I)=CP%(I)+200
5300 K=BP%(I)
5310 MF(K)=MF(K)-MP(I)
5320 PRINT "DOPO L'OPERAZIONE ";IP$(I);" IL CONTO ";NB$(K)
5325 PRINT "HA IL SALDO DI LIRE ";MF(K)
5330 NEXT I
5331 REM
5332 REM ------------------------------------------------
5333 REM
5340 REM  STAMPA DEI RISULTATI DELL'OPERAZIONE
5400 CLS:PRINT "SALDI DEI VARI CONTI"
5410 FOR I=0 TO C1
5420 PRINT"PER IL CONTO ";NB$(I);TAB(20) " IL SALDO E' ";MF(I)
5430 NEXT I
5440 RETURN
```

Registrazione di particolari operazioni contabili

Questa routine permette di registrare le più svariate operazioni ed effettuare gli opportuni movimenti in conto.

Questa routine è molto simile a quella presentata per la registrazione degli assegni. Le uniche differenze consistono nel fatto che non si controllano numeri di assegni, che le operazioni possono comportare sia addebiti che accrediti e che inoltre è previsto un trattamento speciale nel caso di operazioni relative a carte di credito, per differirne l'addebito.

Dopo l'acquisizione di tutti i dati necessari:

- il codice di conto
- la data
- l'importo (da addebitare o accreditare)
- la causale
- i codici contabili particolari (ad esempio per le carte di credito).

l'operazione viene ripresentata globalmente all'utente, a cui viene richiesta una conferma. Ovviamente la registrazione definitiva avviene solo dopo tale conferma.

Utilizza:

I files:

- dei conti
- delle operazioni diverse

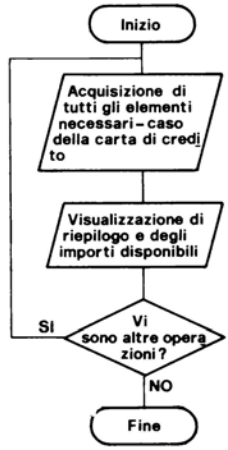


Figura 68

Le routines di controllo:

. sui codici di conto
. sulle date
. sui codici contabili

Variabili utilizzate: A$, B$, N1, N2, N3, M1, M2, M3

Istruzioni: 5500 - 5999

```
5500 IF LC=0 THEN PRINT"LETTURA DA CASSETTA":GOSUB 9000
5510 INPUT"DATA ODIERNA:GIORNO,MESE,ANNO";J1,J2,J3
5520 GOSUB 2210:DT=M2
5530 PRINT"ELEMENTI DELL'OPERAZIONE"
5540 GOSUB 2400:REM IL CONTO
5550 GOSUB 1850: REM L'IMPORTO
5560 M1=ABS(M1):K=1
5570 INPUT"ACCREDITO?";B$
5580 IF LEFT$(B$,1)="S" THEN K=-1
5590 GOSUB 1900   REM CAUSALE
5600 GOSUB 2000 : REM CODICE CONTABILE
5610 INPUT"DATA OPERAZIONE"J1,J2,J3: GOSUB 2210
5620 INPUT"OPERAZIONE CON CARTA DI CREDITO?";B$
5630 IF LEFT$(B$,1)="N" THEN 5670
5640 IF J1>10 THEN J2=J2+1:IF J2>12 THEN J2=1:J3=J3+1
5650 K=0:J1=20:M2=10000*J3+100*J2+J1
5660 IF M2<=DT THEN K=1
5661 REM
5662 REM ------------------------------------------------
5663 REM
5670 CLS:PRINT"RICAPITOLAZIONE":M3=MF(N1)-K*M1
5680 PRINT"CONTO ";NB$(N1)
5690 PRINT"CAUSALE ";A$
5700 PRINT"IMPORTO ";M1
5710 PRINT"CODICE CONTABILE ";N3
5720 PRINT"DATA"; M2
5730 IF K<>0 THEN PRINT "SALDO DEL CONTO ";M3
5740 IF K=0 THEN PRINT"L'ADDEBITO SARA' FATTO IN ALTRA DATA"
5741 REM
5742 REM ------------------------------------------------
5743 REM
5750 INPUT"VA BENE?";B$
5760 IF LEFT$(B$,1)="S" THEN RETURN
5770 INPUT"VOLETE ANNULLARE L'OPERAZIONE?";B$
5780 IF LEFT$(B$,1)="S" THEN RETURN
5790 GOTO 5540
5791 REM
5792 REM ------------------------------------------------
5793 REM
5900 C5=C5+1
5910 IF C5>D5 THEN PRINT"TROPPE OPERAZIONI":GOSUB 9600
5920 BP%(C5)=N1:IP$(C5)=A$: REM REGISTRAZIONE
5930 MP(C5)=M1:DP(C5)=M2
5940 IF K=0 THEN N3=N3-200
5950 IF K=-1 N3=N3+200
5960 CP%(C5)=N3
5970 IF K<>0 THEN MF(N1)=M3
5980 INPUT"ALTRE OPERAZIONI?";B$
5990 IF LEFT$(B$,1)="S" THEN 5540
5999 RETURN
```

Le statistiche

Questa routine controlla le operazioni effettuate durante un certo mese e fornisce il totale di quelle registrate con un certo codice contabile.

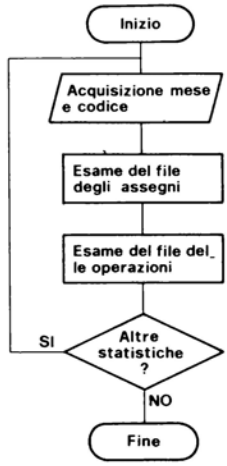


Figura 69

Dopo aver acquisito il mese e il codice rispetto ai quali effettuate la statistica, la routine esamina tutte le operazioni contrassegnate con quel codice contabile ed effettuare in quel mese (considerando il codice contabile nella sua forma originaria) e ne fornisce un bilancio.

Utilizza:

I files:

- degli assegni
- delle operazioni diverse

Non utilizza alcuna routine di servizio.

Nota:

Il risultato di queste statistiche può essere falsato da eventuali aggiornamenti dei files effettuati nel frattempo. Si potrebbe eventualmente completare questa routine con l'uso di files "cronologici" che conservano tutti i dati eventualmente stralciati dai files di lavoro in fase di aggiornamento.

```
7000 IF LC=0 THEN PRINT"LETTURA DA CASSETTA":GOSUB 9000
7010 PRINT:PRINT
7020 PRINT"INSERIRE IL MESE E L'ANNO (DUE CIFRE) PER CUI"
7030 PRINT"SI VUOLE LA STATISTICA":INPUT J2,J3
7040 J1=1:GOSUB 2210
7050 M1=M2+30
7060 PRINT"INSERIRE IL CODICE CONTABILE RISPETTO A CUI SI"
7070 PRINT"VUOLE LA STATISTICA":INPUT N1
7080 IF N1<>INT(N1) THEN PRINT"CODICE ERRATO":GOTO 7060
7090 M3=0
7100 PRINT"STATISTICA SU ";N1;" NEL MESE ";J2
7101 REM
7102 REM ------------------------------------------------
7103 REM
7110 PRINT"STATISTICA SUGLI ASSEGNI"
7120 PRINT"CONTO";TAB(10)"CAUSALE";TAB(25)"IMPORTO"
7125 PRINT TAB(35)"DATA";TAB(45)"NUMERO"
7130 FOR I=0 TO C4
7140 DA=DC(I):K=CC%(I)
7150 IF K>5000 THEN K=K-5000
7160 IF K>1000 THEN K=K-1000
7170 IF K<>N1 OR DA<M2 OR DA>M1 THEN 7200
7180 GOSUB 7500:M3=M3+MC(I)
7190 PRINT NB$(BC%(I));TAB(7)IC$(I);TAB(25)M(I)
7199 PRINT TAB(35)J1;"/";J2;"/";J3;TAB(47)NC%(I)
7200 NEXT I
7201 REM
7202 REM ------------------------------------------------
7203 REM
7210 PRINT:PRINT"STATISTICA SULLE OPERAZIONI DIVERSE"
7220 PRINT"CONTO";TAB(10)"CAUSALE";TAB(25)"IMPORTO";TAB(35)"DATA"
7230 FOR I=1 TO C5
7240 DA=DP(I):K=CP%(I):M4=MP(I)
7250 IF K>4800 THEN K=K-5000
7260 IF K>800 THEN K=K-1000
7270 IF K>200 THEN K=K-200:M4=-M4
7280 IF K<>N1 THEN 7310
7290 M3=M3+M4
7300 PRINT NB$(BP%(I));TAB(7)IP$(I);TAB(25)MP(I);TAB(35)J1;J2;J3
7310 NEXT I
7320 A$="ADDEBITO":IF M3<0 THEN A$="ACCREDITO":M3=-M3
7330 PRINT TAB(15)"TOTALE";TAB(30)M3;"    ";A$
7340 INPUT"ALTRE STATISTICHE?";B$
7350 IF B$="SI" THEN GOTO 7010
7360 RETURN
7361 REM
7362 REM ------------------------------------------------
7363 REM
7500 J3=INT(DA/10000):DB=DA-10000*J3
7510 J2=INT(DB/100):J1=DB-100*J2
7520 RETURN
```

Conservazione e stralcio delle informazioni contenute nei files

Una routine permette di ricopiare i files presenti in memoria su una cassetta magnetica, affinché possono essere nuovamente utilizzati in futuro. Questa routine consiste in pratica in un'istruzione di richiamo diretto al sottoprograma per la scrittura di un file completo su cassetta.

La routine di aggiornamento permette invece di salvare su nastro solo quelle registrazioni che non sono ancora state trovate su alcun estratto conto bancario. Essa è molto utile in quanto permette di guadagnare molto spazio in memoria. Praticamente consiste in un richiamo al sottoprogramma per la scrittura su cassetta dei files aggiornati e stralciati.

Un modo per utilizzare razionalmente la routine di controllo degli estratti conto e dell'aggiornamento dei files sarebbe quello di effettuare queste operazioni ad intervalli regolari (ad esempio tutti i mesi o ogni trimestre), dopo aver fatto le varie statistiche relative a quel periodo.

Un'altra soluzione, come abbiamo già visto, consiste nel creare dei files "cronologici" in cui conservare quelle registrazioni che sono state stralciate in fase di aggiornamento dei files. La cosa è del resto piuttosto agevole in quanto l'aggiornamento dei files su cassetta non comporta la variazione di quelli giacenti nella memoria centrale.

In questo caso occorrerebbe modificare la routine per le statistiche in modo che controlli anche le registrazioni contenute in questi nuovo files. Tale controllo avviene trasportando in memoria un record per volta e percorrendo così tutto il file. Queto metodo ricalca quello che viene utilizzato sistematicamente quando si hanno a disposizione o dei dischi o per lo meno due registratori a cassetta.

La messa a punto del programma

La messa a punto è stata effettuata essenzialmente facendo "girare" sul calcolatore le diverse routines e controllando la correttezza dei risultati conseguiti di volta in volta.

In questo caso, più che in altri, è indispensabile preparare un esempio completo, **già svolto manualmente**, che comprenda tutti i tipi di operazioni contabili, e provare poi il programma mediante questo esempio, confrontando poi i risultati ottenuti automaticamente con quelli ottenuti a mano.

Tale controlo non deve essere effettuato solo sui risultati finali, ma anche su quelli intermedi.

Poiché per ogni operazione viene visualizzata una "ricapitolazione" prima che vengano effettuate le dovute registrazioni sui files, il programma fornisce già di per sé una buona dose di dati per la messa a punto. Questo limita in modo considerevole il numero di stampe di controllo ausiliarie da aggiungere al programma in fase di messa a punto. ,

L'unico punto che non viene mai controllato già all'interno del programma è quello relativo alla scrittura e lettura di files su cassetta. Per controlare queste funzioni si può utilizzare un programma "di servizio" che visualizzi il contenuto completo di una cassetta. (Un programma siffatto è stato da noi approntato e la sua codifica presentata alla fine del presente paragrafo. Istruzioni: 20000 - 20510).

```
20000 CLEAR 650:CLS:PRINT"LETTURA DI UNA CASSETTA"
20010 INPUT#-1,D1,D2,D3,D4,D5,D6
20020 PRINT"NUMERO DI CONTI";D1
20030 PRINT"NUMERO MAX LIBRETTI ASSEGNI ";D2
20040 PRINT"NUMERO MAX DI CODICI CONTABILI ";D3
20050 PRINT"NUMERO MAX DI ASSEGNI ";D4
20060 PRINT"NUMERO MAX DI OPERAZIONI ";D5
20070 PRINT"NUMERO MAX DI PAGAM. RATEALI ";D6
20080 GOSUB 9400
20090 INPUT#-1,C1
20100 PRINT"CONTI APERTI ";C1:REM I CONTI
20110 PRINT"CONTO";TAB(9)"SALDO PREC";TAB(27)"DATA";TAB(44)"SALD"
20120 FOR I=0 TO C1
20130 INPUT#-1,NB$(I),MB(I),DB(I),MF(I)
20140 PRINT NB$(I);TAB(12)MB(I);TAB(30)DB(I);TAB(44)MF(I)
20150 NEXT I
20160 INPUT#-1,C2
20170 PRINT"N LIBRETTI ASSEGNI ";C2 : REM I LIBRETTI
20180 PRINT"CONTO";TAB(15)"PRIMO ASS.";TAB(25)"ULTIMO ASS"
20190 FOR I=0 TO C2
20200 INPUT#-1,B0%(I),D0%(I),F0%(I)
20210 PRINT NB$(B0%(I));TAB(15)D0%(I);TAB(30)F0%(I)
20220 NEXT I
20230 INPUT#-1,C3:REM I CODICI
20240 PRINT"NUMERO CODICI ";C3
20250 PRINT"CODICI:"
20260 FOR I=1 TO C3
20270 INPUT#-1,CD%(I)
20280 PRINT CD%(I)
20290 NEXT I
20300 INPUT#-1,C4
20310 PRINT"NUMERO DI ASSEGNI ";C4 :REM GLI ASSEGNI
20320 PRINT"N CONTO","CONTO","N. AS","CAUS","IMPORTO","COD","DAT"
20330 FOR I=0 TO C4
20340 INPUT#-1,BC%(I),NC%(I),IC$(I),MC(I),CC%(I),DC(I)
20350 PRINT BC%(I),NB$(BC%(I)),NC%(I),IC$(I),MC(I),CC%(I),DC(I)
20360 NEXT I
20370 INPUT#-1,C5: REM LE OPERAZIONI
20380 PRINT"N OPERAZIONI";C5
```

```
20390 PRINT"N CONTO","CAUSALE","IMPORTO","CODICE","DATA"
20400 FOR I=0 TO C5
20410 INPUT#-1,BP%(I),IP$(I),MP(I),CP%(I),DP(I)
20420 PRINT BP%(I),IP$(I),MP(I),CP%(I),DP(I)
20430 NEXT I
20440 INPUT#-1,C6: REM PAGAMENTI RATEALI
20450 PRINT"N VERSAMENTI ";C6
20460 PRINT"CONTO","CAUSALE","PER.","IMPORTO","1VERS","FIN","COD"
20470 FOR I=0 TO C6
20480 INPUT#-1,PV(I),BV%(I),IV$(I),MV(I),DV(I),LV(I),CV%(I)
20490 PRINT BV%(I),IV$(I),PV(I),MV(I),DV(I),LV(I),CV%(I)
20500 NEXT I
20510 END
```

Istruzioni per l'utilizzo del programma:

Alla prima utilizzazione:

Quando il programma viene utilizzato per la prima volta non esiste su cassetta alcun dato in quanto non sono mai state fatte delle registrazioni.

Bisogna dunque partire richiamando necessariamente la routine per il trattamento degli assegni in quanto è l'unica che permette di effettuare delle modifiche sui files dei conti, dei libretti degli assegni, dei codici contabili e degli assegni.

Per evitare successivi problemi converrà predisporre anche la registrazione di un caso di pagamento rateizzato, in quanto ciò permette di inizializzare i files delle operazioni diverse e dei pagamenti rateali.

Una volta registrata almeno un'operazione per ogni file, si può considerare terminata la fase di inizializzazione e si può pertanto conservare il tutto su cassetta.

Utilizzazioni successive:

Una volta inizializzati tutti i files le varie routines possono essere chiamate in un ordine qualsiasi.

Si raccomanda comunque di effettuare per prima cosa l'aggiornamento dei conti per quanto concerne i pagamenti rateali, in quanto potrebbero questi modificare i saldi contabili. Dopo di che si può procedere a piacere nella registrazione delle varie operazioni.

Tenendo conto della capacità di memoria del calcolatore, bisogna stabilire la frequenza con cui devono essere effettuate le statistiche e l'aggiornamento dei files su cassetta, stralciando le registrazioni ormai inutili. Il programma per le statistiche da noi presentato prevede per queste operazioni un-a cadenza mensile e plurimensile.

Infine, soprattutto all'inizio, conviene far ricorso spesso al programma che visualizza il contenuto delle cassette in modo da poter controllare l'andamento del lavoro.

Osservazione finale:

Abbiamo realizzato e messo a punto il programma in modo modulare.

È evidente che questa struttura permette di inserire facilmente nel programma delle altre routines supplementari.

Se si desidera modificare la forma in qualche tipo di registrazione, lo si può fare abbastanza facilmente. Infatti, per i numerosi controlli che sono stati previsti prima dell'accettazione di una operazione contabile, ciascun tipo di registrazione è completamente trattato in una parte ben definita del programma.

Durante la trattazione sono state suggerite diverse idee per effettuare delle variazioni o delle aggiunte. Il lettore ne avrà comunque delle altre e dovrebbe essere in grado, seguendo il metodo da noi proposto, di realizzarle agevolmente.

Note:

Questo programma è particolarmente lungo, anche a causa delle numerose frasi esplicative che sono state inserite nelle operazioni di input output, e non può essere utilizzato su un Personal Computer con una memoria di soli 16K. Occorrerà in questo caso suddividere il programma in due programmi distinti, come indicato nel paragrafo relativo al selezionatore.

In punti diversi del programma, algoritmi molto simili tra loro sono stati codificati in modo diverso per mostrare al lettore come una stessa funzione possa essere codificata in più modi.

CONCLUSIONE

Eccoci arrivati alla fine dei tre esempi, scelti in campi diversi con lunghezza crescente in fase di codifica.

Certo numerosi problemi non sono emersi durante la trattazione di questi esempi.

È stato invece possibile illustrare come, con il metodo da noi proposto, si può realizzare la segmentazione di programmi di dimensione troppo elevata per poter essere utilizzati su dei Personal C-omputers. Incoraggerei il lettore ad utilizzare questo metodo sia per programmi di piccole dimensioni che per programmi di grosse dimensioni, perché solo così imparerà ad usarlo spontaneamente.

Certo, e l'abbiamo già detto all'inizio, l'uso di questo metodo presuppone una certa predisposizione, metodicità e precisione. Ma la programmazione non tollera né pasticcioni né imprecisi. E non li tollera neppure per piccoli programmi!

Insisterò ancora sulla necessità di esaminare ciò che è già stato realizzato riguardo ad un certo problema: è meglio utilizzare un buon programma, anche se non dà tutto ciò che si vorrebbe, piuttosto che lanciarsi in una nuova realizzazione tanto più fortunosa in quanto deve prevedere anche "tutto ciò che nessun altro ha previsto".

Come per ogni altra disciplina, bisogna procedere per piccoli passi, cominciando da semplici programmi per arrivare a programmi più complessi e studiare attentamente i programmi presentati su riviste e su testi specializzati.

GLI Z-BOOKS

Z-80
Programmazione in linguaggio Assembly

Il libro esamina il linguaggio assembly.
Spiega la programmazione in linguaggio assembly, descrive le funzioni di assembler e le istruzioni assembly, tratta i concetti di sviluppo del software di base.
Esamina esempi di programmazione, da un semplice ciclo di caricamento della memoria a un completo progetto di programma.
Offre, ed è questa la grande originalità del volume, gli strumenti di debugging, la relativa procedura di base, i tipi più comuni di errori, nonché alcuni esempi di debugging di programmi. Fornisce, inoltre, esempi di programmi pratici.

Sommario
Introduzione alla programmazione in linguaggio assembly - Assemblatori - Set d'istruzioni per il linguaggio assembly - Semplici programmi - Semplici cicli di programma - Dati codificati come caratteri - Conversione del codice - Problemi aritmetici - Tabelle e liste - Subroutine - Ingresso/Uscita - Interrupt - Definizione del problema e progetto del programma - Debugging e testing - Documentazione e riprogetto - Progetto campione.

Pagg. 650 — Formato 14,5 x 21
Prezzo L. 29.500 — Codice 326P

IMPARIAMO A PROGRAMMARE IN BASIC CON LO ZX80

Non è un semplice manuale operativo (architettura, set d'istruzioni, caratteristiche del linguaggio BASIC, ...), ma partendo da premesse di carattere generale (fondamenti sulla struttura hardware, nozioni di base di programmazione,...) il libro pone l'accento, con esempi chiari ed efficaci, sull'uso pratico di un personal computer, travalicando gli scopi a prima vista limitati al sistema ZX80.

Sommario
Struttura del calcolatore - Il programma - Installazione del calcolatore ZX80 - La tastiera dello ZX80 - Il linguaggio BASIC per il calcolatore ZX80 - I sottoprogrammi - Le funzioni implementate dal BASIC - Norme operative - Errori segnalati dal sistema - Utilizzo della memoria RAM - Il linguaggio macchina - Caratteri del sistema - Variabili del sistema - Scheda BASIC ZX80.

Pagg. 95 — Formato 14,5 x 21
Prezzo L. 4.500 — Codice 317B

LA PROGRAMMAZIONE DELLO Z8000

Il libro descrive in dettaglio l'architettura ed il funzionamento dello Z8000 microprocessore a sedici bit che costituisce da solo un completo calcolatore e la sua famiglia di dispositivi di supporto. Fornisce una introduzione alla programmazione in linguaggio macchina. Presenta molti esempi di programmi al fine di illustrare i principi e le tecniche essenziali. Fa vedere come possono essere implementati con la programmazione importanti principi di ingegnerizzazione del software come la semplicità, la chiarezza dei commenti, la modularità, ecc..

Sommario
Concetti base - Architettura circuitale dello Z8000 - Introduzione alle tecniche di programmazione - Modi di indirizzamento dello Z8000 - Tecniche per la gestione dell'ingresso/uscita - Componenti periferici dello Z8000 - Esempi di programmi di utilità - Tecniche avanzate di programmazione - L'ambiente di sviluppo dei programmi.

Pagg. 302 — Formato 14,5 x 21,5
Prezzo L. 22.000 — Codice 321D

PROGRAMMAZIONE DELLO Z80 E PROGETTAZIONE LOGICA

Il libro descrive l'implementazione della logica sequenziale e combinatoria con l'uso del linguaggio assembly all'interno di un sistema a microcomputer basato sullo Z80. Lo scopo è quello di insegnare ai progettisti logici come la programmazione abbia trovato uno scopo nuovo nel progetto logico.
Vengono simulate sequenze logiche digitali, poi illustrate alcune efficienti soluzioni per spiegare l'uso corretto dei microcomputer. Un capitolo, infine, contiene il set completo di istruzioni dello Z80.

Sommario
Introduzione - Linguaggio assembly e logica digitale - Una simulazione diretta della logica digitale - Un semplice programma - Prospettiva del programmatore - Set di istruzioni - Alcune subroutine impiegate comunemente - Codici di caratteri ASCII.

Pagg. 400 — Formato 14,5 x 21
Prezzo L. 19.000 — Codice 324P

NANOBOOK Z80 Vol. 1
Tecniche di programmazione

Questo volume è dedicato al software dello Z80 con particolare riguardo alla programmazione in linguaggio macchina ed in linguaggio assembler. L'approfondimento di ogni argomento svolto prescindendo da una preparazione specifica su computer, tecniche di programmazione ed elettronica digitale, avviene con la sperimentazione, il che permette non solo di verificare l'esattezza degli esercizi svolti, tendenti a prevenire ogni possibile dubbio del lettore, ma anche di evidenziarne gli errori per una corretta rianalisi.

Sommario
Codici digitali - Introduzione alla programmazione dei microcomputer - Alcune istruzioni del microcomputer Z80 - Il Nanocomputer NZ80 e il Nanocomputer Super NBZ80S - Alcuni semplici programmi del Microcomputer Z80 - Registri, Memorie e trasferimenti dati - Metodi di indirizzamento dello Z80 - Salti, chiamate e ritorni - Istruzioni logiche - Manipolazione dei bit, istruzioni di rotazione e shift - Istruzioni aritmetiche e di ricerca dei blocchi - Come utilizzare la SGS-Ates Z80 CPU programming reference card - Calcolo dei tempi di esecuzione - Precauzione da adottare nel maneggiare dispositivi MOS - Elenco degli indirizzi assoluti delle locazioni indicate con notazione simbolica nel testo.

Pagg. 240 — Formato 14,5 x 21
Prezzo L. 15.000 — Codice 310P

NANOBOOK Z80 Vol. 3
Tecniche dell'interfacciamento

Il libro continua la trattazione dello Z80 iniziata con il volume 1, di cui peraltro si presuppone una certa familiarità, introduce ai problemi ed alle tecniche di interfacciamento con gli elementi CPU, PIO e CTC della stessa famiglia.
Viene mantenuto l'approccio pragmatico e sperimentale del volume precedente per cui al lettore non si richiede quasi nessuna conoscenza tecnica ma solamente capacità logiche.

Sommario
Interfacciamento dello Z80 - La scheda per esperimenti NZ80 dell'NBZ80S - Generazione degli impulsi di sincronizzazione: impulsi di selezione dispositivi e indirizzi - Bus, Buffer Tri-state e I/O dello Z80 - L'hardware e il software di sistema del nanocomputer - Tecniche di interruzione - Il dispositivo di ingresso uscita parallelo PIO Z80 - Un tester per circuiti integrati TTL - Il circuito contatempi-contaventi - Documentazione del software utilizzato durante gli esperimenti - Elenco degli indirizzi assoluti delle locazioni indicate con notazione simbolica nel testo - Precauzione da adottare nel maneggiare dispositivi MOS - Schemi elettrici del nanocomputer.

Pagg. 451 — Formato 15 x 21
Prezzo L. 18.000 — Codice 312P

Per ordinare i volumi, utilizzare l'apposita cartolina inserita in ultima pagina

GRUPPO EDITORIALE JACKSON
Divisione Libri

PASCAL

IMPARIAMO IL PASCAL

Compattezza, concisione, chiarezza e notevoli potenzialità scientifiche, oltre a prestarsi ottimamente per calcoli gestionali e ad essere usato anche con i microcomputer, sono le caratteristiche che decretano il successo del PASCAL come linguaggio di programmazione. Non vi era però finora un testo che insegnasse a tutti a programmare in PASCAL: o perché i libri esistenti sono troppo concisi, o troppo semplici, oppure perché richiedono la conoscenza di altri linguaggi di programmazione, o, non ultimo, perché in inglese. Questo libro colma queste lacune [illegible] portandolo "a [illegible]". I capitoli sono il [illegible] organici, in [illegible] la loro consultazione [illegible] semplice ed agevole. [illegible] riassunto di quanto si apprende [illegible] posto all'inizio e non [illegible] capitolo, perché il lettore possa subito avere un metro [illegible] con cui verificare passo [illegible] suo apprendimento. E poi, ci [illegible] consigli, problemi, esercizi affinché il libro sia "usato" e non letto, perché [illegible] sapere come si usa un'istruzione [illegible] che conoscerne le differenze sostanziali tra linguaggio e linguaggio. Con un [illegible] graduale, partendo senza alcuna conoscenza di programmazione, dopo circa due settimane dovreste conoscere abbastanza bene il [illegible]AL. Un buon risultato, no?!

Formato 15 x 21

Codice 501A

SOMMARIO

0. Da non trascurare
1. Come si descrive la sintassi del linguaggio
2. Come si scrive in PASCAL
3. Il programma e le dichiarazioni in generale
4. Le dichiarazioni ed i tipi standard
5. I tipi speciali e subrange
6. Gli statements di assegnazione
7. Gli statements di ripetizione
8. Gli statements logici
9. I dati strutturati - Generalità
10. Il tipo array
11. Il tipo record
12. Il tipo set
13. Il tipo file
14. Il tipo pointer
15. Le procedure e le funzioni
16. Procedure ricorrenti input ed output
17. I diagrammi di struttura

GRUPPO EDITORIALE JACKSON DIVISIONE LIBRI

Usare il sistema operativo CP/M

IL LIBRO

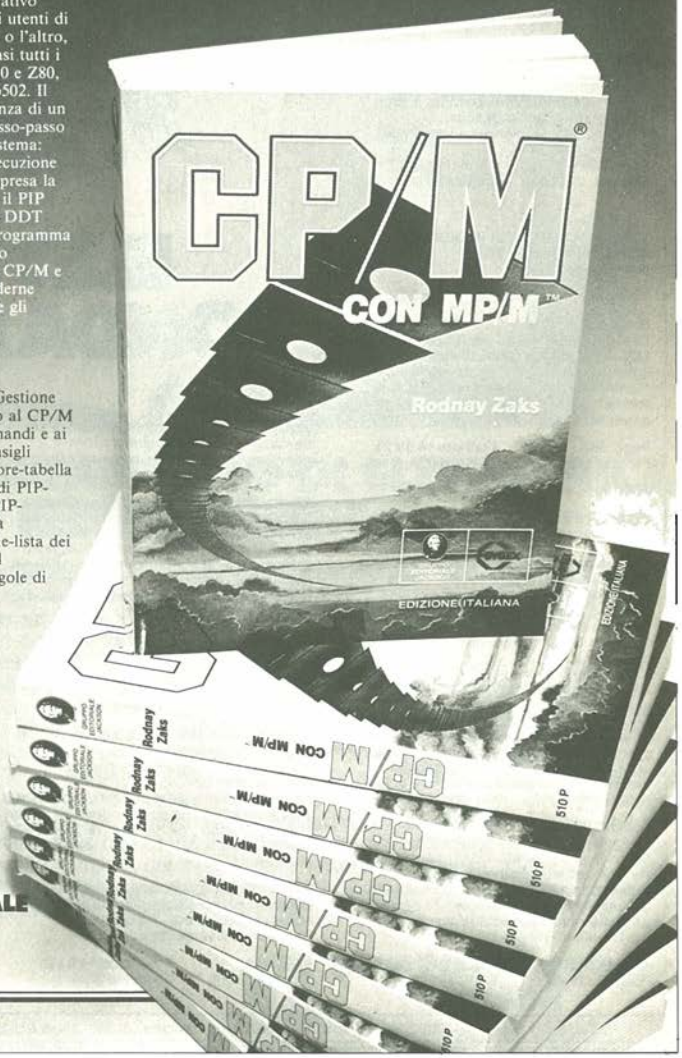
Il sistema operativo CP/M è stato progettato per rendere semplice l'uso di un microcomputer. Questo libro vi renderà semplice l'uso del CP/M. (Le versioni esaminate del CP/M sono il CP/M 1.4-il CP/M 2.2. e il nuovo sistema operativo multiutente MP/M) La maggior parte di utenti di microcomputer dovrà, infatti, un giorno o l'altro, fare ricorso al CP/M, disponibile su quasi tutti i computer basati sui microprocessori 8080 e Z80, come pure su certi sistemi utilizzanti il 6502. Il libro, senza presupporre alcuna conoscenza di un calcolatore, inizia con la descrizione, passo-passo delle procedure di inizializzazione del sistema: accensione, inserimento dei dischetti, esecuzione delle più comuni operazioni su file, compresa la duplicazione dei dischetti. Prosegue con il PIP (programma di trasferimento dei file), il DDT (programma di messa a punto) e ED (programma editor). Per entrare sempre più, fornendo numerosi consigli pratici, all'interno del CP/M e delle sue operazioni, al fine di comprenderne appieno le risorse ed eventualmente dare gli strumenti per successive modifiche.

SOMMARIO

Introduzione al CP/M e all'MP/M-Le caratteristiche del CP/M e dell'MP/M-Gestione dei file con PIP-L'uso dell'editor-Dentro al CP/M e all'MP/M-Guida di riferimento ai comandi e ai programmi del CP/M e dell'MP/M-Consigli pratici-Il futuro-messaggi comuni di errore-tabella di controllo di ED-nomi dei dispositivi di PIP-riassunti dei comandi-parole chiave di PIP-parametri di PIP-tasti di controllo per la digitazione dei comandi-tipi di estensione-lista dei materiali-organizzazione della stanza del calcolatore-verifiche in caso di errore-regole di base per la localizzazione dei guasti.

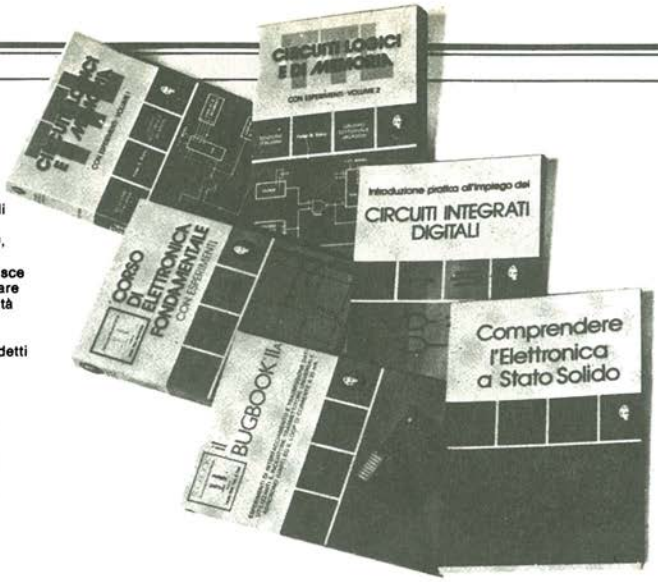
Pagg. 320 Cod. 510P

L.22.000



Per ordinare i volumi, utilizzare l'apposita cartolina inserita in ultima pagina

GRUPPO EDITORIALE JACKSON
Divisione Libri



ELETTRONICA FONDAMENTALE

CIRCUITI LOGICI E DI MEMORIA CON ESPERIMENTI VOL. 1
(già Bugbook I)

Un approccio diretto al mondo dell'elettronica digitale.
Da subito si fa la conoscenza con i chip di circuiti integrati, vengono introdotti i concetti di switch logici, indicatori a LED, generatori di impulsi e display. Il libro unitamente al vol. 2 con il quale costituisce un corso completo, insegna come utilizzare questi elementi ed in più offre la possibilità di effettuare 90 esperimenti dalla complessità crescente, basati sul collegamento tra i circuiti integrati e suddetti componenti.

Sommario
Il sistema di breadboarding con gli outboards LR - Il "gating" di un segnale digitale - Tabelle della verità - Alcuni esperimenti particolari che utilizzano un four-decade counter-Decoder, demultiplexer, multiplexer e sequencer.

Pagg. 350 Formato 15 x 21
Prezzo L. 22.000 Codice 001A

CIRCUITI LOGICI E DI MEMORIA CON ESPERIMENTI VOL. 2
(già Bugbook II)

Completa la trattazione del volume 1.

Sommario
Diodi ad emissione di luce (LED) e display a LED - Bus: stadi di uscita tristate ed a collettore aperto - Flip-flop e multivibratori monostabili - Memorie a semiconduttore; RAM e ROM - Registri, contatori, elementi aritmetici e trigger di Schmitt.

Pagg. 350 Formato 14,5 x 21
Prezzo L. 22.000 Codice 002A

CORSO DI ELETTRONICA FONDAMENTALE CON ESPERIMENTI

Testo ormai adottato nelle scuole per l'alto valore didattico, fa "finalmente" capire l'elettronica della teoria atomica ai transistori. Ciascun argomento viene svolto secondo i sui principi base e ne vengono descritte le applicazioni pratiche e i circuiti reali.
La sua caratteristica peculiare, comunque, è la grande chiarezza con cui tutti gli argomenti vengono esposti e gli esperimenti descritti. Si configura, quindi, come vero e proprio corso per l'autodidatta. Il sussidio sperimentale consigliato unitamente alla serie dei componenti per realizzare gli esperimenti, è di costo contenuto e di facile reperibilità.

Sommario
Fondamenti di elettricità - Identificazione di schemi e componenti - Kit per esperimenti di elettronica fondamentale, tester ed oscilloscopi - Legge di Ohm - Circuiti serie - Circuiti parallelo - Circuiti serie e parallelo - Capacità - Bobine, corrente alternata e trasformatori - Diodi - Transistori.

Pagg. 439 Formato 15 x 21
Prezzo L. 15.000 Codice 201A

IL BUGBOOK IIa

Esperimenti di interfacciamento e trasmissione dati utilizzanti il ricevitore/trasmettitore universale asincrono (UART) ed il loop di corrente a 20 mA.

Il testo, parte complementare del "Circuiti logici e di memoria" vol. 2, sviluppa circuiti di comunicazione utilizzabili per trasferire, da pochi metri a molti chilometri (tecniche asincrone seriali) informazioni digitali da un circuito a qualche sistema di ingresso/uscita come ad esempio una teletype usando un circuito integrato LSI a 40 pin.

Pagg. 56 Formato 14,5 x 21
Prezzo L. 4.500 Codice 021A

INTRODUZIONE PRATICA ALL'IMPIEGO DEI CIRCUITI INTEGRATI DIGITALI

Il volume "demistifica" finalmente il circuito integrato digitale permettendo di comprendere il funzionamento al pari di qualsiasi altro circuito.
Le definizioni di base esposte sono comprensibili a tutti e permettono un rapido apprendimendo dei circuiti di base e la realizzazione di circuiti decisamente interessanti.

Generalità sui circuiti integrati logici - Esperimenti con differenti tipi di porte - Materiale necessario - Gli oscillatori - Calcolo e visualizzazione.

Pagg. 112 Formato 14,5 x 21
Prezzo L. 7.000 Codice 203A

COMPRENDERE L'ELETTRONICA A STATO SOLIDO

Il libro, partendo "da zero" consente di comprendere i semiconduttori e come questi funzionano insieme in sistemi elettronici a stato solido. Articolato come corso autodidattico in 12 lezioni, completo di quesiti e di glossari, utilizzando solo semplici nozioni di aritmetica, spiega la teoria e l'uso di diodi, transistori, tiristori, dispositivi elettronici e circuiti integrati bipolari, MOS e lineari.

Sommario
Che cosa fa l'elettricità in ogni sistema elettrico - Funzioni dei circuiti fondamentali nel sistema - Come i circuiti prendono delle decisioni - Relazioni fra semiconduttori e sistemi - I diodi cosa fanno e come funzionano - Prestazioni e caratteristiche dei diodi - I transistori: come funzionano e come sono fatti - Il transistore PNP e le caratteristiche dei transistori - Tiristori ed optoelettronica - Introduzione ai circuiti integrati - Circuiti integrati digitali - MOS e circuiti integrati lineari.

Pagg. 222 Formato 14,5 x 21
Prezzo L. 14.000 Codice 202A

GRUPPO EDITORIALE JACKSON
Divisone Libri

Per ordinare i volumi, utilizzare l'apposita cartolina inserita in ultima pagina

l'ultima novità editoriale

in offerta speciale di lancio a L. 109.000...

CORSO PROGRAMMATO DI ELETTRONICA ED ELETTROTECNICA

40 volumi
2700 pagine

Cod. 099A
L. 109.000

Il corso articolato in 40 fascicoli per complessive 2700 pagine, permette in modo rapido e conciso l'apprendimento dei concetti fondamentali di elettrotecnica ed elettronica di base, dalla teoria atomica all'elaborazione dei segnali digitali.

La grande originalità dell'opera, non risiede solo nella semplicità con cui gli argomenti vengono trattati, anche i più difficili, non solo nella struttura delle oltre 1000 lezioni incentrate su continue domande e risposte, esercizi, test, al fine di permettere la costante valutazione del grado di apprendimento raggiunto, ma soprattutto nella possibilità di crearsi in modo organico un corso "ad personam" rispondente alle singole necessità ed obiettivi. Se non avete tempo o non volete dedicare 120 delle vostre ore, anche in modo frammentario, al completamento del corso, potete seguire un programma di minima, sempre con brillanti risultati, con obiettivi, anche parziali, modificabili dinamicamente nel corso delle letture successive. Ogni libro è una monografia esauriente singolarmente consultabile per l'approfondimento di un particolare argomento.

GRUPPO EDITORIALE JACKSON
Divisione Libri

Tagliando ordine **Ilbri Jackson** da inviare a:
Gruppo Editoriale Jackson - Via Rosellini, 12 - 20124 Milano

Nome Cognome

Indirizzo

Cap. **Città**

Codice Fiscale (indispensabile per le aziende)

Inviatemi i seguenti libri:

☐ Pagherò al postino l'importo di L. ..... + L. 1.500 per contributo fisso spese di spedizione

☐ Allego assegno n° ..... di L. .....
(in questo caso la spedizione è gratuita)

Codice Libro	Quantità	Codice Libro	Quantità	Codice Libro	Quantità	Codice Libro	Quantità

Tagliando ordine **Ilbri Jackson** da inviare a:
Gruppo Editoriale Jackson - Via Rosellini, 12 - 20124 Milano

Nome Cognome

Indirizzo

Cap. **Città**

Codice Fiscale (indispensabile per le aziende)

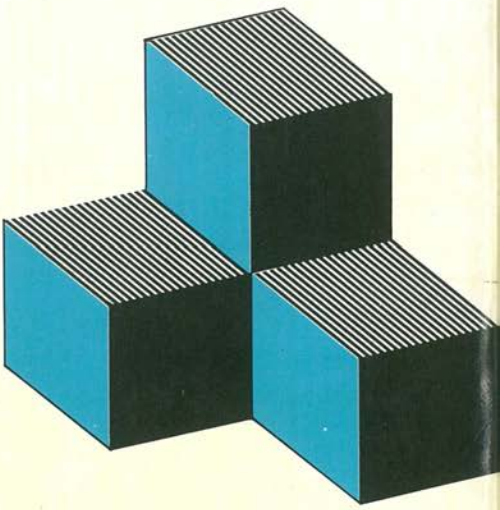
Inviatemi i seguenti libri:

☐ Pagherò al postino l'importo di L. ..... + L. 1.500 per contributo fisso spese di spedizione

☐ Allego assegno n° ..... di L. .....
(in questo caso la spedizione è gratuita)

Codice Libro	Quantità	Codice Libro	Quantità	Codice Libro	Quantità	Codice Libro	Quantità

Realizzare un buon programma oltre che di esperienza, inventiva, competenza abbisogna di metodo. Insegnare un metodo è proprio lo scopo del libro. Un metodo accessibile a tutti, che nasce dall'esperienza dell'autore, dai consigli che si è trovato a dare, da annotazioni prese.
Chi programma, deve passare da un'idea inizialmente vaga alla realizzazione pratica del programma vero e proprio che svolga le funzioni richieste. Deve imparare, perciò, ad enunciare e definire correttamente l'idea iniziale, ad analizzarla, a come trasformarla, e come verificare la correttezza della stessa sino a giungere alla stesura in Basic di un programma ben documentato, leggibile e facilmente modificabile. Vengono esplicitate anche tutte le altre fasi intermedie del lavoro: tutte le vie alternative che via via si presentano e tra cui bisogna scegliere, le eventuali estensioni, i ripensamenti, le prove e le verifiche che occorre fare per ottenere un programma che dia risultati conformi a quanto ci si era proposti. Poichè era necessario appoggiarsi a un linguaggio di programmazione, si è scelto il Basic per la sua larga diffusione e perchè utilizzato dalla maggior parte dei personal computer. I concetti chiaramente esposti e molto dettagliatamente illustrati, comunque, sono utilizzabili indipendentemente dal linguaggio prescelto. I programmi proposti poi, sono stati tutti provati e girano su computer da 4 K a 64 K di memoria.



L. 12.000

Cod. 511A ISBN 88-7056-109-7

GRUPPO EDITORIALE JACKSON

COME PROGRAMMARE

Jean-Claude Barbance

52