IL PASCAL

Prima parte

Mi piace sempre fare un po' di storia dell'informatica: credo che si possa capire meglio il significato di certe scelte e di certe strutture se se ne conosce l'evoluzione storica. E posso ammettere che alcune volte i riferimenti al periodo paleolitico del calcolatore siano un po' tirati.

Nel caso del PASCAL, su cui si apre con questa prima parte un ciclo di articoli, i riferimenti storici sono indispensabili: non credo che si possa capire appieno l'importanza di questo linguaggio, né conoscere a fondo la sua struttura, se non si inquadra il periodo storico — una decina di anni fa — in cui il PASCAL fece la sua comparsa.

Non me se ne voglia quindi, se prenderò il discorso un po' alla larga, citando abbondantemente altri linguaggi che all'apparenza non hanno alcuna parentela con il PASCAL. Avremo poi tempo e modo di entrare in tutti i dettagli del linguaggio ma, come ho detto, questa premessa mi sembra importante.

Come nacque il PASCAL

Alla fine degli anni sessanta la situazione del software sembrava abbastanza consolidata: oltre ai vari linguaggi assemblatori e ai linguaggi specializzati come il LISP, il programmatore di computer aveva a sua disposizione tre linguaggi fondamentali, FORTRAN, ALGOL e COBOL, ciascuno orientato ad una diversa problematica.

Il FORTRAN possiede il fascino del classico: è il più antico dei linguaggi ad alto livello, e permette di svolgere calcoli complessi con pochissima spesa in termini di software. Suo figlio naturale è il BASIC, e da questo si può capire la sua struttura, basata su cicli di istruzioni (FOR in BASIC, DO in FORTRAN) e su un uso intensivo dell'istruzione GOTO. Analoga al BASIC, sebbene leggermente più complessa, è la potenza di istruzioni di I/O, che permettono di formattare agilmente le stampe senza dover fare salti mortali.

Ma — ahimé — il FORTRAN è un linguaggio molto rigido: e la giungla di GOTO e l'assoluta impossibilità di gestire stringhe di caratteri o di costruire un puntatore lo rendeva difficile da masticare in quei tempi in cui si iniziava a parlare di programmazione strutturata, ricorsività, insomma di un software a più livelli. A ciò i programmatori rimediarono in qualche modo inventando di sana pianta un nuovo linguaggio.

Infatti l'ALGOL, nella prima versione del '60 e soprattutto nella seconda del '68, risolve buona parte dei problemi lasciati in sospeso dal suo predecessore: finalmente compare una struttura a blocchi (*begin... end*), e la possibilità di definire puntatori e stringhe. Ma il linguaggio è ancora troppo orientato ai calcoli matematici, risente insomma del vecchio pregiudizio secondo cui il computer deve far di conto e basta; così quello che si guadagna da una parte lo si perde dall'altra, e per stampare anche solo una tabella sono dolori.

Con l'introduzione del computer nelle banche e negli uffici anagrafici, si fa sempre più pressante l'esigenza di definire dei dati strutturati, e non necessariamente numerici; nasce così il COBOL, l'unico dei tre ad avere ancora oggi una grande diffusione, che introduce i concetti di *stringa* e di *record* diviso in *campi*. Un bel passo avanti, non c'è che dire: ma stavolta bisogna fare i salti mortali per estrarre una radice quadrata e, con il COBOL, non è neanche concepibile una strutturazione a blocchi.

Ecco dunque la situazione dell'informatica una decina di anni fa: il sogno di tutti era di poter programmare con la potenza di calcolo e di I/O del FORTRAN, la strutturazione a blocchi e i puntatori dell'ALGOL, e i record e le stringhe del COBOL, così come tanti uomini sognano una ragazza con i capelli di Tizia, il corpo di Caia e l'intelligenza di Sempronia...

Fu tenendo ben presente questa esigenze

Caratteristiche del PASCAL

- Forte orientamento verso la programmazione strutturata, sia a livello di istruzioni (blocchi "compound") che a livello di dati (definizioni di tipo strutturato).
- Potenza di calcolo paragonabile e forse superiore agli "specialisti" FORTRAN e BASIC
- Possibilita' di definire e gestire dati non numerici e insiemi di operazione arbitrari (tipo "scalar").
- Ampliamento, rispetto ai linguaggi classici, della potenzialita' delle singole istruzioni (ad esempio istruzioni condizionali con piu' di due alternative).
- Particolare facilita' di definizione e richiamo di funzioni e sottoprogrammi
- Possibilita' di programmare in modo ricorsivo.

che il professor Wirth del Politecnico di Zurigo definì nel 1971 il linguaggio PASCAL: ed aveva ragione, poiché non soltanto il linguaggio stesso ha avuto una diffusione incredibile a tutti i livelli, ma ogni nuovo linguaggio definito negli anni successivi, anche il più specializzato, non ha potuto non farvi riferimento: ho addirittura visto degli articoli scientifici in cui venivano esposti degli algoritmi matematici in forma simile al PASCAL.

Sperando di aver incuriosito abbastanza i lettori, possiamo ora passare a descrivere questo sorprendente linguaggio, iniziando da alcuni concetti di carattere generale e proseguendo con le specifiche definizioni.

Orientamento e struttura generale del PASCAL

Come si è detto, il linguaggio è nato per riassumere in un solo strumento tutte (o quasi) le possibilità del software agli inizi degli anni '70. Le sue caratteristiche fondamentali sono quindi le seguenti:

1) possibilità di strutturare sia i programmi che i dati in modo analogo alle scatole cinesi: il PASCAL è costruito apposta per programmare secondo le regole della progettazione strutturata; ad una elevata potenza e facilità di definire e richiamare sottoprogrammi unisce una universale e flessibilissima struttura a blocchi, fino a sconsigliare l'uso dell'istruzione GOTO, che pur è presente nel linguaggio; quanto ai dati, la struttura *record* permette di costruire "pacchi" di dati anche diversissimi fra loro, ed articolati in più livelli (in teoria infiniti).

TABELLA DI COMPARAZIONE DI ALCUNI LINGUAGGI

	POTENZA DI CALCOLO	STRUTTURAZIONE	GESTIONE DATI NON NUMERICI	POTENZA DI I/O
ASSEMBLER	solo numeri interi	difficile ma possibile	possibile	inesistente
FORTRAN	elevatissima	impossibile	praticamente impossibile	elevata
ALGOL	elevatissima	molto forte	possibile	dipendente dalla macchina, ma non eccelsa
COBOL	scarsa	presente ma molto rigida	raccomandata	elevatissima
PASCAL	elevatissima	elevatissima e raccomandata	possibile e molto facilitata	dipendente dalla macchina, comunque su livelli FORTRAN

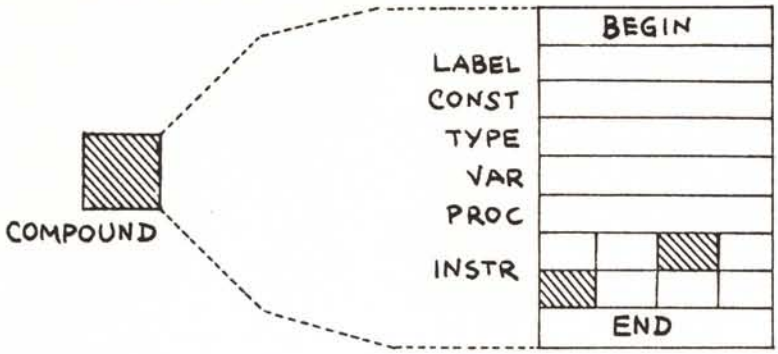
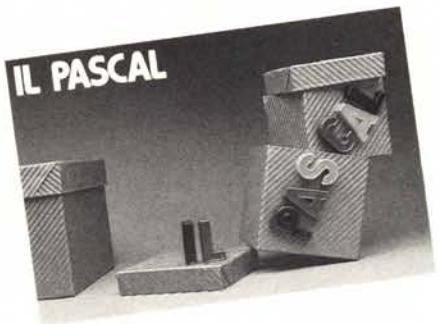


Figura 1 - Rappresentazione grafica dell'istruzione "compound": fra le parole "begin" e "end" si trovano definizioni ed istruzioni, fra cui possono esserci altre "compound".



2) possibilità di definire dati diversi dalle variabili numeriche: qui il professor Wirth si è veramente sbizzarrito, includendo nel linguaggio tutti i tipi di dati possibili, dalle variabili intere e reali ai puntatori, dai files agli insiemi (notevolissima questa ultima facoltà), fino a poter definire, come vedremo, dei tipi di variabili a scelta dell'utente.

3) agilità di programmazione: poiché il linguaggio è orientato alla programmazione strutturata, ecco istruzioni di controllo meno rigide di quelle del FORTRAN, con le quali si possono eseguire cicli di lunghezza variabile, o istruzioni condizionali con più di due alternative.

4) possibilità di programmazione ricorsiva: analogamente al LISP, il linguaggio è in grado di eseguire programmi ricorsivi; la ricorsività è anzi consigliata in quanto uno degli elementi base della programmazione strutturata.

Tenendo conto di questi orientamenti, il generico programma PASCAL è strutturato in questo modo:

begin
definizioni di label (opzionale)
definizioni di costanti "
definizioni di tipo "
definizioni di variabili "
definizioni di sottoprogrammi "
istruzioni separate dal punto e virgola (almeno una)
end

In realtà questa è *una delle istruzioni* del linguaggio, che prende il nome di *compound*, ed ha lo specifico compito di creare un blocco della struttura: ne consegue che un programma PASCAL è formato da *un'unica istruzione compound.*

Una compound può comparire, e di solito ciò accade, nella lista di istruzioni prima dell'*end*, e in questo modo si crea la struttura a blocchi: ogni compound apre una parentesi, in cui possono essere anche definite delle variabili e delle costanti, che hanno valore soltanto fra il *begin* e l'*end.*

Con questa definizione ricorsiva ("il programma PASCAL è formato da una istruzione compound che può contenere delle altre compound") abbiamo definito una struttura a scatole cinesi: anche i sottoprogrammi, come vedremo, sono definiti allo stesso modo; ed è anzi molto facile trasformare una compound in un sottoprogramma.

Le definizioni elencate nella prima parte del blocco compound hanno valore nelle istruzioni del blocco stesso e in tutti i sottoblocchi definiti all'interno di esso, a meno che variabili con lo stesso nome non siano definite in due blocchi a livelli diversi, nel qual caso "vince" la definizione del blocco più interno. Si osservi ad esempio la struttura della fig. 1), e si supponga che nel blocco M sia definita una variabile di nome ALFA. Tutti gli altri blocchi potranno usare questa variabile senza problemi, e faranno sempre riferimento alla stessa area di memoria riservata al livello di M; ma se il blocco B definisce una variabile con lo stesso nome (e può farlo senza che sorgano conflitti) un riferimento ad ALFA nei blocchi B e C indicherà una variabile diversa da quella puntata da un uguale riferimento nei blocchi A, D, E ed F, in quanto la definizione al livello più interno (B) prevale rispetto a quella al livello più esterno (M).

Non si creda comunque che ogni volta che si scrive *begin* si debba per forza sciorinare una caterva di definizioni: nella pratica una compound racchiude il più delle volte soltanto una serie di istruzioni.

Procediamo comunque con ordine e vediamo in dettaglio le singole dichiarazioni.

Le dichiarazioni del PASCAL: label, const, var

Una delle poche cose fisse del PASCAL è l'ordine in cui vengono elencate le definizioni all'interno di un blocco compound. Non è possibile alterare l'ordine della lista (1) esposta sopra, possono solo essere saltate quelle che non sono necessarie: una dichiarazione di tipo dovrà sempre venire prima di una dichiarazione di variabile e mai dopo.

La dichiarazione di label è la più semplice e la meno usata: una label ha infatti senso soltanto se esiste una istruzione GO-TO che vi fa riferimento, e si è già detto che l'uso di questa istruzione è vivamente sconsigliato.

Comunque, una label è sempre e soltanto un numero intero, e viene definita in questo modo:

label 3, 18;

ossia tramite l'identificatore *label* seguito dai numeri che verranno impiegati come etichette nel corso del programma, separati da virgole.

Il punto e virgola conclude ogni dichiarazione.

Sorte migliore incontrano le dichiarazioni di costante. Una costante è in PASCAL un identificatore scelto dall'utente a

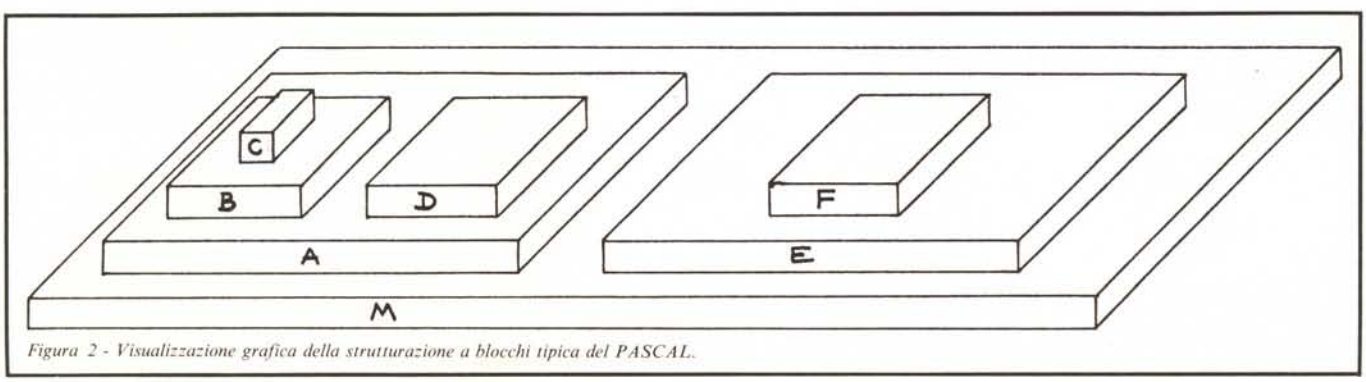


Figura 2 - Visualizzazione grafica della strutturazione a blocchi tipica del PASCAL.

cui viene assegnato un ben preciso valore, che non può essere modificato nel corso del programma.
Esempio:
const uno = 1, pigreco = 3.14159, mionome = 'pietro hasenmajer';

Il segno uguale associa agli identificatori i valori specificati d'ora in poi al posto dei valori potranno essere usati i nomi come nel seguente esempio:
area : = pigreco * sqr (raggio)
(ove la funzione sqr (n) calcola il quadrato del numero n, ed "area" e "raggio" sono variabili reali).

Le *variabili* vengono definite quasi nello stesso modo, associando uno o più identificatori ad un *tipo*. Lasciando per ora in sospeso cosa si intenda per tipo (l'argomento merita un capitolo a parte), l'ossatura della dichiarazione di variabile è la seguente:

```
var id, id..., id : tipo 1;
    id,..., id : tipo 2;
    ...
    id : tipo n;
```

ad esempio:

```
var area, raggio : real;
    n : integer;
```

ove *real* e *integer* sono tipi di genere standard, che non necessitano di una definizione a parte, analoghi a quelli FORTRAN.

Il concetto di tipo e le definizioni elementari

Bene o male il concetto di "tipo" di una variabile è presente in tutti i linguaggi: in BASIC, ad esempio, è specificato da un simbolo posto dopo il nome della variabile, e così tutti sanno che, se A è una variabile di tipo reale, A$ è una variabile di tipo stringa. In FORTRAN possono essere definite variabili di tipo *logico*, che possono assumere soltanto i valori "vero" e "falso"; nei linguaggi ad alto livello, insomma, una dichiarazione di "tipo" — implicita o esplicita che sia — è sempre presente.

La grossa differenza rispetto al PASCAL è che in questi linguaggi i tipi sono *pochi* e *standardizzati*, cioè sono forniti dal compilatore e possono essere usati soltanto nella dichiarazione delle variabili, come nell'ultimo esempio del capitolo precedente; non esiste insomma una esplicita *dichiarazione di tipo*, in cui viene assegnato un nome non alla variabile ma al suo tipo.

In PASCAL invece ciò è possibile: grazie alla varietà di tipi standard e alla possibilità di strutturarli, si può definire un tipo, poi usarlo come *elemento* in una struttura a livello più elevato, infine dichiarare variabili dei tipi definiti.

L'ossatura della dichiarazione di tipo è la seguente:
type nome = f(tipi);
ove f(tipi) indica una "funzione" (secondo modalità che vedremo) o dei tipi standard o di tipi definiti precedentemente con la stessa modalità.

I tipi standard possono essere semplici o strutturati: in questa prima parte analizzeremo unicamente i tipi semplici.

integer e *real* sono i più elementari, e non ricorrono quasi mai da soli nelle dichiarazioni di tipo: è stupido scrivere:

```
type pincopallino = integer;
var alfa : pincopallino;
```

quando si può direttamente scrivere:

```
var alfa : integer;
```

senza perdere in generalità e compattezza.

Integer e *real* compaiono dunque in forma elementare nelle dichiarazioni di variabile, e definiscono rispettivamente variabili intere e reali.

Allo stesso modo viene trattato il tipo *boolean*, che definisce una variabile di tipo logico:

```
var bit : boolean;
```

definisce come logica la variabile "bit": essa potrà assumere un valore *logico* (vero o falso) qualsiasi:

Si potrà ad esempio scrivere:
bit : not (A > 0) and (B = C);
ed usare la variabile nelle istruzioni condizionali:
if bit *then... else...;*
analogamente al FORTRAN.

Le stringhe di caratteri vengono definite tramite il tipo *char*. In realtà questo tipo definisce una variabile come *stringa formata da un solo carattere*: una variabile di tipo char può assumere il valore di un (e un solo) carattere ASCII:

```
var letter : char;
letter := 'A';
```

È dunque incorretto scrivere:
letter := 'Pietro';

Per gestire le stringhe di più di un carattere esistono metodologie particolari, che verranno esaminate nel capitolo riguardante i tipi strutturati.

Caratteristici del PASCAL sono invece i tipi *scalar* e *subrange*.

Per capire il loro significato occorre tenere presente cosa vuol dire dichiarare una variabile come appartenente ad un certo tipo: praticamente si specifica che la variabile in questione potrà assumere un valore compreso in un certo insieme; che potrà essere quello dei numeri interi (da -32767 a +32767) per una variabile intera, oppure quello dei caratteri ASCII per una variabile di tipo char, e così via.

Il tipo *scalar* permette di definire a scelta dell'utente l'insieme in cui la variabile può giostrare, elencandone gli elementi. In PASCAL è possibile avere variabili di questo tipo:

```
type COLORE = (rosso, arancio, giallo,
     verde, blu, indaco, violetto);
var  RAINBOW : COLORE;
```

La variabile RAINBOW potrà ora assumere come valore uno dei sette colori dell'arcobaleno:
RAINBOW := blu;
e potrà essere usata come se fosse una variabile numerica, ad esempio in una istruzione condizionale o ciclica;

```
if   RAINBOW = rosso then...
for  RAINBOW := rosso to violetto...
```

Con il tipo *scalar* si può insomma creare un insieme di dati "astratto" su cui lavorare; le operazioni possibili sono, oltre all'assegnamento, le due funzioni *pred* e *succ*, che indicano rispettivamente l'elemento

precedente e quello *successivo* nella lista di definizione.

Così, nel nostro esempio dei colori, succ (rosso) = arancio, e pred (blu) = verde.

La funzione *ord* stabilisce un ponte fra l'insieme definito in modo scalare e l'insieme dei numeri interi, in quanto indica la *posizione* dell'elemento nella lista di definizione:
ord (rosso) = 1 e ord (blu) = 5

Il tipo *subrange* si appoggia su uno dei tipi standard o su un tipo scalare definito precedentemente, che stabilisce il proprio insieme di variabilità come *sottoinsieme* del tipo richiamato. Una definizione del tipo subrange è strutturata nel seguente modo:

```
type nome = e 1... e 2;
```

dove e1 ed e 2 sono il primo e l'ultimo degli elementi costituenti il sottoinsieme di variabilità.

Vediamo qualche esempio:

```
type giorno = 1... 31;
     lettera = A... Z;
     rossi = rosso... verde;
```

Una variabile definita come appartenente ad uno di questi tipi potrà assumere, nel primo caso, un valore numerico da 1 a 31; nel secondo caso, un valore *char* corrispondente alle sole lettere maiuscole; nel terzo caso, un valore di colore compreso fra il rosso e il verde. Se si tenta di assegnare alla variabile un valore fuori dai limiti, il compilatore segnala un errore.

Si noti che le dichiarazioni di tipo subrange possono essere incluse nelle dichiarazioni di variabile:

```
var day : 1..31;
```

è più immediato che non la sequenza:

```
type giorno = 1..31;
var day : giorno;
```

Il tipo subrange sarà importantissimo nelle dichiarazioni di tipo matriciale, o *array*, che sono la prossima tappa del nostro itinerario attraverso il PASCAL: tuttavia, poiché il tipo array apre un nuovo capitolo (quello dei tipi *strutturati*); preferisco continuare la prossima volta.

Fermiamoci dunque qui, in attesa di una nuova puntata in cui si esamineranno i tipi strutturati e le istruzioni di programma.

Pietro Hasenmajer

IL SUONO, IL COLORE, LA LOGICA

La versione standard del DAI comprende:

- **BASIC semi compilato, molto potente e veloce, in 24 K di ROM.**
- **13 modi grafici, fino a 256 x 336 punti a 16 colori in alta risoluzione (istr. DRAW - DOT - FILL).**
- **Capacità video di 24 linee x 60 colonne (1440 caratteri maiuscoli e minuscoli).**
- **Monitor di linguaggio macchina 8080.**
- **Potente EDITOR residente.**
- **Sintesi musicale: 4 generatori programmabili, con uscite in stereofonia.**
- **Sintesi vocale.**
- **48 K di RAM a disposizione dell'utente.**
- **Interfaccia seriale RS 232 - 2 interfacce per cassette.**
- **Interfaccia parallela (3 porte programmabili).**
- **Interfaccia per TV a colori.**

Numerose opzioni: floppy disks, stampante, processore aritmetico, paddles, ecc.

Per informazioni scrivere a
Casella Postale 10488
20100 Milano

Dimostrazioni e vendita presso

SHOP bit

DAI THE MICROCOMPUTER COMPANY