Le strutture a lista

di Anna Pugliese

Eccoci di nuovo a parlare di strutture dati.
Sul numero scorso ci siamo preoccupati essenzialmente di gettare le basi per una trattazione delle strutture informative della memoria principale, evidenziando alcune caratteristiche dei tipi di dato più semplici e del modo in cui essi possono essere memorizzati; per quanto riguarda le strutture dati vere e proprie, ci siamo soffermati adeguatamente sugli array e sui problemi implementativi loro connessi

Un'importante considerazione fatta a suo tempo, che sarà bene richiamare in questa sede, riguarda la differenza esistente tra le strutture informative di base, talora dette interne, e quelle astratte; quest'ultime sono caratterizzate dal fatto che la loro implementazione è realizzata mediante l'utilizzo di strutture interne e procedure di simulazione. Vedremo, a cominciare da questo numero, come avviene questa realizzazione. Tuttavia, per ragioni propedeutiche, ci occuperemo inzialmente di una seconda struttura informativa interna (la prima era l'array): la lista.

Il vantaggio della flessibilità

Il vettore, fra le strutture dati, è il più vicino all'organizzazione fisica della memoria, e perciò stesso il più semplice da implementare. Esso presenta d'altra parte alcune limitazioni, il cui superamento è costituito dalle liste. Vediamone alcune.

Consideriamo un vettore di stringhe, ognuna delle quali rappresenta un nominativo. Ogni nominativo contenuto nel vettore è caratterizzato dalla posizione, all'interno del vettore, che l'elemento che lo contiene occupa. Queste informazioni hanno la forma di un elenco di dati cui è stato assegnato un ordinamento prestabilito. Solitamente, l'inserzione di un nuovo nominativo nel vettore viene effettuata nella posizione successiva a quella dell'ultimo elemento occupato del vettore. Questo tipo di organizzazione presenta una scarsa flessibilità che deriva proprio dal fatto che lo spazio libero è costituito unicamente dalle posizioni del vettore che seguono l'ultimo elemento occupato. Se si dovesse allora presentare la necessità di inserire un nuovo nominativo in una posizione intermedia agli elementi i e i+1, la cosa sarebbe realizzabile soltanto cancellando tutti i nominativi che seguono l'iesimo e riscrivendo ciascuno di essi una posizione più avanti. Analogo è il discorso nel caso in cui sia richiesta la cancellazione di un elemento del vettore che occupa una posizione intermedia: è necessario cancellare tutte le informazioni che seguono quella da rimuovere, e riscriverle ciascuna nella posizione precedente.

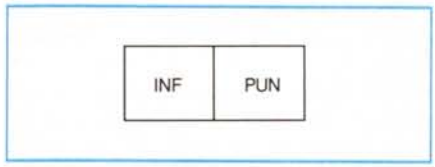


Figura 1 - L'elemento costitutivo delle Liste Semplici.

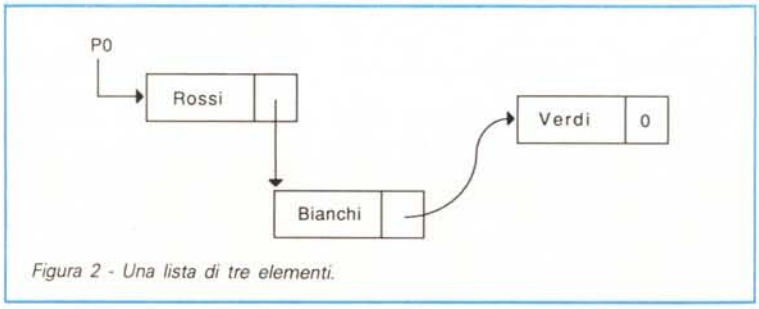


Figura 2 - Una lista di tre elementi.

L'inserzione e l'eliminazione di elementi da un elenco, può essere effettuata in maniera molto più efficiente, se i dati non occupano posizioni prestabilite all'interno dell'elenco, ma sono registrati ognuno su una scheda diversa, in modo che l'elenco sia costituito da un pacco di schede. In tal modo, l'operazione di «spostare» tutti gli elementi seguenti un elemento specificato, può essere realizzata spostando il relativo «pacco di schede».

L'idea che sta alla base delle strutture a lista, è quella appena presentata. Essa è realizzabile solo a condizione di impiegare un trucco che ora vedremo, reso necessario dal fatto che non è possibile «spostare» le celle di memoria di un calcolatore. Il trucco consiste nel ricorrere ad un metodo che permetta di assegnare alle celle di memoria un ordinamento logico arbitrario, diverso dal loro effettivo ordinamento fisico (che è costituito dagli indirizzi).

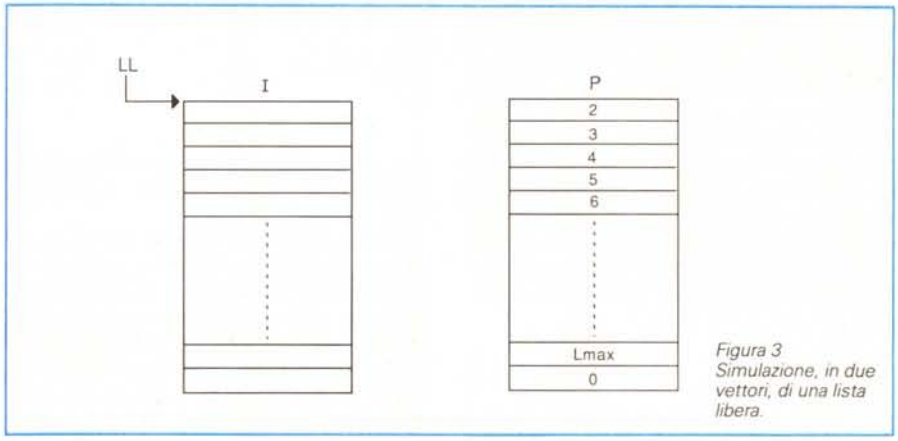


Figura 3
Simulazione, in due vettori, di una lista libera.

Liste semplici

Vediamo allora com'è fatto il generico elemento di una lista semplice.

La figura 1 riporta una schematizzazione del componente elementare di una lista. INF e PUN sono un'abbreviazione per INFORMAZIONE e PUNTATORE, che nel seguito abbrevieremo ulteriormente con I a P. Questo tipo di elemento è proprio delle liste dette «semplici»; liste di tipo diverso differiscono da queste per il numero di puntatori che ogni elemento contiene.

Lo schema in figura dev'essere astrattamente interpretato come una coppia di variabili, dove la prima variabile (INF) è del tipo corrispondente a quello delle informazioni che devono essere contenute nella lista, mentre la seconda ha un tipo che è indipendente da queste informazioni e che, nel caso in cui la lista è effettivamente messa a disposizione dal linguaggio, viene detto: tipo PUNTATORE. In parole semplici, il tipo puntatore può essere considerato come quello i cui valori sono gli indirizzi delle celle di memoria principale. Spieghiamoci meglio.

Eliminiamo una potenziale fonte di ambiguità, supponendo che la struttura a lista della quale stiamo parlando venga fornita direttamente dal linguaggio. Il generico elemento di una lista semplice può essere allora considerato come una coppia di parole, diciamo consecutive, della memoria principale, ed è caratterizzato dall'indirizzo cui esso risiede, che corrisponde all'indirizzo della prima cella di memoria di cui esso è composto.

Osserviamo la figura 2. Essa rappresenta una lista semplice di nominativi, lunga 3, che risiede all'indirizzo iniziale PO. Il meccanismo dei puntatori è visualizzato mediante frecce, con il seguente significato. La freccia che partendo da PO giunge fino al primo elemento della lista, sta a significare che la variabile (di tipo puntatore) PO contiene, come valore, l'indirizzo della prima cella di memoria in cui risiede l'informazione, di tipo stringa, «Rossi»; l'intera struttura contenente questa informazione viene detta INF (PO) o, per abbreviare ancor più, I (PO), e la cosa deve'essere letta come: l'informazione residente all'indirizzo PO. L'elemento contenente come informazione il nominativo «Bianchi», può risiedere in qualunque punto della memoria principale, dal momento che la sua accessibilità è garantita dal fatto che il suo indirizzo è il contenuto della parola di memoria che segue l'informazione «Rossi», vale a dire P(PO) che si legge: il puntatore contenuto nell'elemento della lista residente all'indirizzo PO. Quanto detto è rappresentato in figura 2 dalla freccia che uscendo dalla «zona puntatore» del primo elemento della lista giunge fino al secondo elemento della lista stessa. A sua volta, il secondo elemento della lista «punta» al terzo, grazie al fatto che la sua parte puntatore ne contiene l'indirizzo iniziale; infine la parte puntatore del terzo elemento della lista, non dovendo puntare a nessun altro elemento, contiene un valore puntatore convenzionalmente nullo, che è stato indicato in figura con il simbolo «O».

Lo schema della figura 1 richiama, già a prima vista, un vettore unidimensionale di due elementi. Avendo assunto, tuttavia, che è il linguaggio stesso che mette a disposizione del programmatore la struttura dati Lista Semplice, questa nostra considerazione lascia il tempo che trova, vale a dire che non è un nostro problema come questa struttura dati venga effettivamente implementata, e che, comunque sia, la sua implementazione difficilmente è realizzata sulla base di un vettore, ma ben più probabilmente essa godrà di una tecnica di memorizzazione e gestione adotta-

```
procedure INIT;
Begin
  for k:=1 to Lmax-1 do P(k):=k+1;
  P(Lmax):=0;
  LL:=1
End;
```

Figura 5

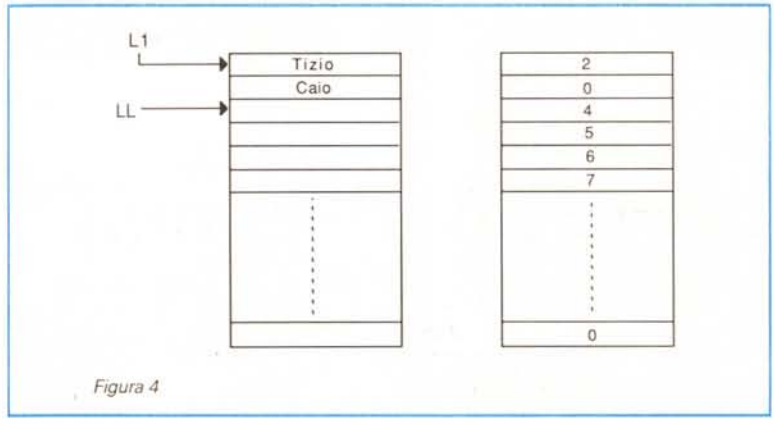


Figura 4

ta proprio allo scopo di gestire nel modo più efficiente tutte le caratteristiche delle liste.

D'altro canto, la maggior parte dei linguaggi di programmazione in voga non fornisce le strutture a lista come predefinite nel linguaggio. Ne consegue che è di grande utilità pratica, nonché didattica, preoccuparsi di trovare strategie di simulazione di tali strutture. È quello che faremo, non prima di aver risottolineato che questo nuovo modo di intendere le liste è in un certo senso alienante nei loro confronti; vale a dire che in tal modo non avremo più a che fare con una struttura informativa inter-

```
procedure CREATE (var L : integer);
Begin
  L:=0
End;
```

Figura 6

na, ma con una struttura informativa astratta. Ma forse questa considerazione è un po' troppo accademica; non perdiamoci più in chiacchiere.

La simulazione delle Liste

Quello che ci proponiamo, è di simulare la struttura dati Lista Semplice facendo uso della struttura informativa interna Vettore. Per far ciò, necessitiamo di illustrare il meccanismo di rappresentazione dei vari elementi della lista nel vettore, e di fornire le procedure attraverso le quali è possibile operare sulla lista stessa. Iniziamo allora scegliendo, fra le tante soluzioni possibili, quella di utilizzare due vettori di uguale e prefissata lunghezza, diciamo Lmax. Chiamiamo «I» il primo e «P» il secondo di tali vettori, e confidiamo sul fatto che a nessuno sia sfuggito che il primo vettore lo destiniamo a contenere le informazioni vere e proprie, mentre il secondo ci servirà per implementare il meccanismo dei puntatori. Veniamo adesso ad un punto molto importante. Quali sono le operazioni che manipolano le liste? Essenzialmente: INSERZIONE e RIMOZIONE di elementi dalla lista, in secondo luogo: spostamento di elementi da una posizione (logica) ad una altra della lista; per far tutto questo necessitiamo, infine, di un operazione di ricerca di elementi della lista che si basa sulla procedura elementare di scansione. Procediamo con ordine.

La figura 3 mostra il contenuto cui vengono inizializzati gli elementi dei due vettori I e P, al fine di realizzare quello

```
procedure INSERT (Name : string ; var L : integer);
var temp:integer;
Begin
  temp:=L;
  L:=LL;
  LL:=P(LL);
  P(L):=temp;
  I(L):=Name;
End;
```

Figura 7a

che è lo strumento essenziale per la simulazione delle strutture a lista: la Lista Libera.

Osservando la figura, notiamo che il vettore delle informazioni non contiene valori iniziali. Tale vettore è costituito da elementi che devono essere del tipo cui devono appartenere gli elementi informativi della lista; nel caso in cui si voglia memorizzare in una lista un elenco di nominativi, avremo che il vettore I può essere definito dalla seguente istruzione di definizione Pascal-like:

I : array [1..Lmax] of string;

per quanto riguarda il vettore P, una opportuna istruzione di definizione per esso può essere:

P : array [1..Lmax] of integer;

quest'ultimo è infatti destinato a contenere informazioni di tipo puntatore che nel nostro caso possono essere considerati valori numerici interi, che rappresentano il valore da dare all'indice dei due vettori per designare, all'interno dei due array, la coppia di elementi che virtualizza l'elemento astratto elementare della lista, cui il puntatore intende riferirsi. Sempre con riferimento alla figura 3, abbiamo allora che I(1)è la parte informativa del primo elemento della lista la cui parte puntatore è P(1); quest'ultimo contiene il valore intero 2 che, interpretato come puntatore, indica che l'elemento seguente ad (I(1),P(1)) è (I(2), P(2)); essendo poi P(2)=3, possiamo raggiungere il terzo elemento della lista che è (I(3),P(3)) che può anche essere indicato con (I(P(2)), P(P(2))) che è chiaramente la stessa cosa. Infine P (Lmax) = 0 sta ad indicare la terminazione della lista, mentre LL=1 (rappresentato ancora con il meccanismo delle frecce in figura) ci dice che la lista libera comincia dall'elemento (I(LL),P(LL)) = (I(1),P(1)).

Ci si chiederà: come è possibile simulare le strutture a lista, con questa megagalattica lista desolatamente vuota? Niente paura. Osserviamo al proposito la figura 4; essa mostra la stessa megagalattica lista che comincia ad assumere dei connotati più consoni a quelli che ci piacerebbe che avesse.

Le frecce riportate in figura 4, stanno ad indicare che LL =3 e che L1=1, vale a dire che è stata creata e successivamente soggetta all'inserzione di due elementi una lista la cui «chiave d'accesso» è L1. Gli elementi inseriti nella lista L1 sono stati prelevati dalla lista libera, che di conseguenza ha subito la rimozione dei suoi primi due elementi. È ovvio a questo punto dove saranno mossi gli eventuali elementi che devono essere eliminati da una lista. Ad esempio, possiamo considerare la figura 3 come il risultato cui si perverrebbe in seguito alla rimozione di due elementi dalla lista L1 indicata in figura 4.

A questo punto, possiamo dedicarci all'implementazione delle operazioni che manipolano le liste che abbiamo appena descritto. Illustreremo, come al solito, gli algoritmi di tali operazioni, facendo uso di un linguaggio non eccessivamente formale, ed ispirato al Pascal.

Abbiamo già visto le istruzioni di definizione dei due vettori I e P; per completezza viene fornita, in figura 5, la procedura di inizializzazione di tali vettori; in tal modo ci troveremo nelle condizioni di avere a disposizione una lista

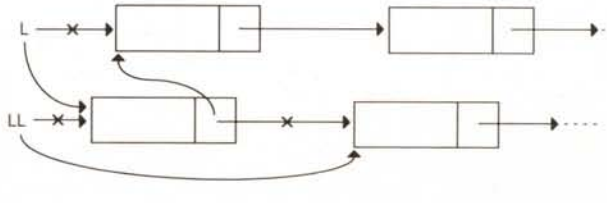


Figura 7b
Inserzione di un nuovo elemento in testa alla lista L.

```
procedure REMOVE (Name:string ; var L:integer);
var P1,P2,temp:integer;
Begin
  P1:=L;
  while P1<>0 do begin
                 if I(P1)=Name then EXIT;
                 P2:=P1;
                 P1:=P(P1);
                 end;
  if P1=0 then NameNotFound
          else if P1=L then begin
                              temp:=LL;
                              LL:=P1;
                              L:=P(L);
                              P(LL):=temp;
                            end
                       else begin
                              temp:=LL;
                              LL:=P1;
                              P(P2):=P(P1);
                              P(P1):=temp;
                            end;
End;
```

Figura 8a

libera: il punto di partenza è stato raggiunto.

Potrebbe sembrare, a prima vista, limitativo il dover stabilire una dimensione massima da dare ai due vettori I e P, e di conseguenza alla lista libera. Tale dimensione Lmax costituisce il massimo numero di elementi che potranno essere utilizzati per tutte le liste coinvolte nell'applicazione in cui verranno impiegate; vale a dire che la sommatoria di tutte le dimensioni delle liste esistenti in un certo istante, nel corso dell'esecuzione del programma, non può mai superare Lmax. Questo fatto non è una limitazione nuova, in quanto qualsiasi istruzione di definizione di una nuova struttura informativa potrebbe sempre risolversi con esito «OUT OF MEMORY». Prescinderemo quindi, nel seguito, da questo aspetto, supponendo un dimensionamento della lista semplice, sufficiente allo scopo.

La prima operazione sulle liste, che andremo a vedere, quella di creazione, la implementeremo in un modo che potrebbe sembrare banale. Tuttavia, riflettendoci sopra, non dovrebbe essere difficile capire cosa realmente implica questa operazione. La figura 6 illustra l'operazione di creazione di una nuova lista. La figura 7 e la figura 8 mostrano le due operazioni fondamentali, INSERZIONE e RIMOZIONE rispettivamente, di elementi in una lista.

Cerchiamo di capire, osservando le figure 7b e 8b, in cosa consistano gli algoritmi delle figure 7a e 8a.

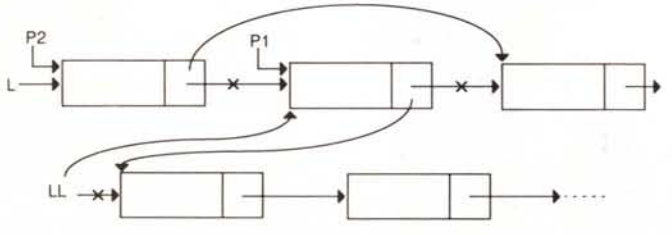
Per quanto riguarda l'operazione di inserimento di un nuovo nominativo in lista (figura 7), abbiamo supposto, per semplicità, che tale inserzione potesse avvenire in un punto qualsiasi della lista, ed abbiamo scelto al proposito l'inizio della lista che, almeno per quanto riguarda le liste semplici, è il punto più facile da raggiungere. È bene sottolineare che altre scelte possono essere maggiormente convenienti; una buona soluzione sarebbe stata quella di inserire il nuovo nominativo in modo da rispettare un ordinamento alfabetico; in tal caso avremmo potuto realizzare una migliore strategia di ricerca nella lista nella procedura di rimozione. Comunque sia, l'elemento da inserire nella lista L dev'essere prelevato dalla lista libera LL. Essendo LL una variabile di sistema, il suo valore è globalmente conosciuto, e non occorre che venga passato fra i parametri. Veniamo all'algoritmo. L'istruzione L:=LL serve per far diventare il primo elemento della lista libera un nuovo primo elemento della lista L; il vecchio valore di L, che rappresenta la vecchia lista, sottolista della nuova, viene salvato in una variabile temporanea. Con LL:=P(LL) si aggiorna l'indirizzo inziale della lista libera, che comincerà dal suo ex secondo elemento, quello, per l'appunto, precedentemente puntato dal primo. Infine il nuovo elemento inserito viene inizializzato al resto della lista L, per quanto riguarda la parte puntatore, ed al nominativo passato come parametro in Name, per la sua parte informativa. La sequenza di istruzioni della figura 7a, provoca la cancellazione delle frecce asteriscate e la creazione delle nuove frecce riportate in figura 7b.

Per l'operazione di rimozione, analoghe considerazioni ci dicono che l'elemento da rimuovere andrà a finire in testa alla lista libera. Proprio di questa operazione, è il meccanismo di ricerca nella lista del nominativo da rimuovere; tale nominativo potrebbe anche non essere contenuto nella lista, nel qual caso l'algoritmo illustrato si appella ad una opportuna procedura di gestione dell'errore che è stata chiamata «NameNotFound». Una volta trovato l'elemento da rimuovere, il gioco è quasi fatto; il quasi si riferisce al fatto che le modifiche da apportare ad L e ad LL presenterebbero serie difficoltà di realizzazione se non si ricorresse allo stratagemma dei due puntatori P1 e P2. Si pensi, al proposito, al fatto che prima di rimuovere un elemento occorre trovarlo, e di conseguenza utilizzare un puntatore per riferirsi ad esso; il secondo puntatore (P2 nel nostro caso) ha lo scopo di puntare al padre dell'elemento cercato, in quanto, in seguito alla sua rimozione, il puntatore di suo padre dovrà puntare a suo figlio (il nipote del padre) e non più ad esso. La figura 8b dovrebbe essere esemplificatoria al riguardo. Infine si noti l'adozione di due diversi algoritmi di rimozione, da utilizzare nel caso in cui l'elemento da rimuovere sia il primo della lista, nel qual caso è necessario aggiornare il puntatore L, oppure no. Il resto dovrebbe essere chiaro, anche se, per chi non è avvezzo a simili cose, richiede un attimo di riflessione.

E con questo possiamo dire di aver trattato sufficientemente l'argomento «liste semplici». Dal prossimo numero ci dedicheremo alla simulazione di strutture informative astratte vere e proprie, quali la coda, la pila, gli alberi e così via; presenteremo implementazioni basate sull'uso di strutture interne diverse, vale a dire implementazioni mediante array ed implementazioni a lista, continuando in questo modo la nostra passeggiata in questo mondo, a mio parere affascinante, delle strutture dati.

A risentirci.

Figura 8b
Rimozione del secondo elemento della lista L.



RICORDI ▷ Archimedes

Buon lavoro, con la potenza del RISC!

▷ **RISC:** è il principio di **Archimedes**, lo straordinario e velocissimo personal computer a 32 bit ▷ Mettetelo alla prova con un foglio elettronico come **SigmaSheet**, 200 volte più rapido dei suoi simili (ricalcola un cash-flow di 32 anni *in meno di 25 secondi*), o con un integrato come **Pipe-dream** (predisposto per comunicare con i portatili della nuova generazione), o con un project-manager versatile come **Logistix**, o con un database come **System Delta Plus** (che può gestire oltre *due miliardi di records*) ▷ Confrontate la potenza dei pacchetti di *grafica*, del software per applicazioni *musicali, didattiche, scientifiche, mediche* ▷ Valutate la facilità con cui sono state sviluppate soluzioni originali e sofisticatissime nei vari linguaggi disponibili (**BBC Basic, Assembly, C, Pascal, Fortran 77, Lisp, Prolog**) ▷ Apprezzate la possibilità di continuare a utilizzare tranquillamente i vostri pacchetti **MS-DOS** preferiti ▷ Mai un computer così nuovo e rivoluzionario ha avuto tanto software così presto ▷ Ed è solo il principio.

DOPPIOUNI

G. RICORDI & C.
Settore Informatico
Via Salomone, 77
20138 MILANO
tel. 02/5082-315

Distributore esclusivo:

Acorn
The choice of experience.
Un'azienda del gruppo Olivetti

Per maggiori informazioni, inviate questo coupon a G. RICORDI & C. Settore Informatico, Via Salomone, 77, 20138 MILANO

Desidero avere maggiori informazioni su Archimedes

Nome:

Cognome:

Qualifica professionale:

Ditta, Ente o Scuola:

Indirizzo: