a cura di Andrea de Prisco

Le strutture informative
Terza parte: stack e queue

di Anna Pugliese

Pila e coda, in inglese rispettivamente «stack» e «queue», potrebbero essere considerate le due strutture più importanti nella programmazione a livello di sistema.
I più noti esempi del loro impiego, sono: la gestione degli ambienti nelle chiamate di sottoprogrammi, per quanto riguarda lo stack; le politiche di scheduling nella gestione delle risorse condivise, per ciò che concerne la queue.
Anche i programmi applicativi fanno spesso uso di queste due strutture. Per poterle utilizzare occorre tuttavia realizzare alcune procedure di simulazione. È quello che faremo in questa puntata di «Appunti di Informatica»

Il punto

Per dare una struttura organica alla trattazione delle diverse strutture dati che ci siamo prefissati di esaminare, è il caso di fare delle considerazioni, del tutto generali, sull'argomento, che ci permettano di tracciare il punto della situazione alla quale siamo giunti nelle due precedenti puntate della rubrica.

I programmi sono degli algoritmi, espressi in un qualche linguaggio, descriventi le azioni da intraprendere per elaborare informazioni.

Le informazioni, mentre da una parte modellano più o meno complesse entità del mondo esterno, dall'altra sono modellate da dati di diversi tipi.

Un tipo di dato consiste in un insieme di valori che il dato può assumere, assieme ad un certo numero di operazioni, vale a dire di trasformazioni elementari cui il valore di un dato può essere sottoposto.

Le operazioni sui dati sono svolte, in ultima analisi, dal processore di un computer, il quale è capace di eseguire, per sua natura, solo operazioni su dati di tipo numerico e dimensione prefissata.

Dai punti precedenti si evince un duplice problema:
1) da una parte l'esigenza di avere a disposizione TIPI DI DATO (quindi un insieme di valori e di trasformazioni su di essi operabili), che permettano l'utilizzo di una più larga tipologia di dati rispetto a quella offerta dalla macchina nuda. E questo è compito dei progettisti di linguaggi ad alto livello.
2) Dall'altra la necessità di organizzare i dati, che modellano le informazioni da elaborare, all'interno di strutture informative, in modo da poter facilmente operare su di essi mediante i tipi di operazioni messe a disposizione dal linguaggio.

È questo secondo problema che resta completamente a carico del pro-

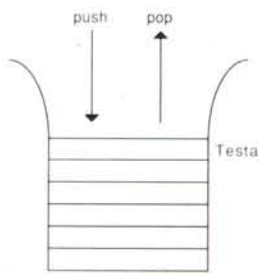


Figura 1 - La pila.

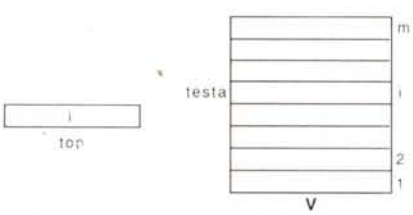


Figura 2a

```
Push(x,P);
begin
top:=top+1;
if top > m then "memory fault";
V[top]:=x;
end;
```

```
Pop(P);
begin
if top = 0 then "error";
top:=top-1
return(V[top+1]);
end;
```

Figura 2b

```
Init(P);
begin
top:=0;
end;
```

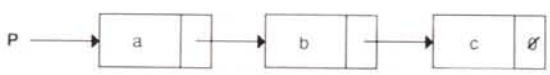


Figura 3a - Allocazione in memoria di una pila, a lista.

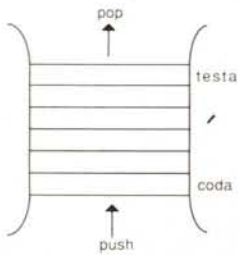


Figura 4 - La coda.

```
Push (x,P);
begin
var new : reference;
new := alloca(length);
INF(new) := x;
PUN(new) := P;
P := new;
end;
```

```
Pop (P);
begin
var value : integer;
var temp : reference;
if P=0 then "error";
temp := P;
value := INF(P);
P:=PUN(P);
dealloca(temp);
return(value);
end;
```

Figura 3b

grammatore di applicazioni: l'organizzazione dei dati. Questo non significa solo scegliere fra le strutture dati messe a disposizione dal linguaggio, ma anche definire nuovi tipi e nuove strutture.

È da quasi un ventennio ormai, che si è avuta un'importante inversione di tendenza da parte dei progettisti di linguaggi, che stanno sempre più sfoltendo, i linguaggi stessi, da tipi e strutture predefiniti, arricchendoli invece di strutture di base e di potenti meccanismi di astrazione sui dati.

Dal nostro punto di vista, che è quello dell'esame delle strutture informative, un'importante considerazione che abbiamo già tracciato per altre vie nelle precedenti puntate, è quella relativa ai due fondamentali meccanismi di strutturazione dei dati:

l'organizzazione SEQUENZIALE, che potremmo anche definire «ad indirizzi relativi»; l'organizzazione A LISTA, che potremmo invece chiamare «ad indirizzi assoluti».

Da questi due tipi di organizzazione dei dati scaturiscono le due strutture informative di base, che abbiamo già visto sotto il nome di «strutture interne», e precisamente: i vettori e le liste semplici. Essi sono solo degli esempi, i più immediati, di applicazione dei due tipi di organizzazione; altri potrebbero essere le matrici e le liste multiple, o le strutture ed i grafi, e così via.

In generale una struttura dati, che è caratterizzata dalle sue proprie specifiche funzionali, può poi essere realizzata mediante una strutturazione ad indirizzi relativi (in cui i vari componenti della struttura sono allocati in posizioni di memoria aventi distanze calcolabili) e/o mediante una strutturazione ad indirizzi assoluti (dove viceversa, i vari elementi occupano posizioni non specificabili a priori) e/o, come non poche volte accade, utilizzando soluzioni intermedie alle due.

La pila

Le caratteristiche funzionali della pila, potrebbero a prima vista sembrare quanto meno strane. In realtà esiste una larga classe di problemi facilmente trattabili a condizione di utilizzare questa particolare struttura dati. Facciamo subito un esempio.

Consideriamo la seguente espressione algebrica:

(2+3)((5−(9/2))+1)/2,

in essa sono presenti 4 parentesi aperte e 4 parentesi chiuse, condizione necessaria per la benfattezza dell'espressione, ma non sufficiente. Infatti, l'espressione:

(2+3))(5−(9/2()+1)/2,

pur avendo un totale di 4 parentesi aperte e 4 parentesi chiuse, non è un espressione benfatta, in quanto la seconda parentesi chiusa chiude un'inesistente parentesi aperta.

Nel caso di una completa meccanizzazione del processo di calcolo dell'espressione, è necessaria una verifica di benfattezza dell'espressione che potrebbe essere fatta con il seguente algoritmo: scandendo l'espressione dalla sinistra verso destra, ogni volta che si incontra una parentesi aperta, la si inserisce dentro una struttura; incontrando una parentesi chiusa, essendo questa da associare all'ultima parentesi aperta incontrata, si estrae dalla struttura l'ultima parentesi aperta inserita. Se non c'è nella struttura, una parentesi aperta da associare a quella chiusa che è stata incontrata, si può concludere la non benfattezza dell'espressione. Se la scansione giunge invece fino alla fine dell'espressione, si può concludere che la stessa è benfatta, a condizione che la struttura sia rimasta vuota, viceversa essa non è benfatta.

La struttura della quale abbiamo parlato è la pila; essa infatti può essere sottoposta ad inserzioni e rimozioni di elementi, mediante le operazioni (vedi figura 1) di PUSH (=spingere) e POP (=tirare).

La legge funzionale cui la pila deve obbedire è quella per cui l'elemento da estrarre dalla pila corrisponde sempre all'ultimo inserito; tale elemento è quello che occupa la posizione che in figura 1 è stata definita TESTA della pila. Così, se abbiamo una pila che contiene come elementi quelli della sequenza Pila=[a,b,c,d], dove «a» è l'elemento di

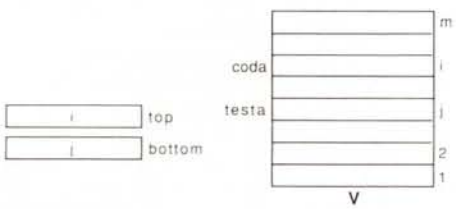


Figura 5a

```
Push(x,C);
begin
if bottom = m then "memory fault";
bottom:=bottom+1;
V[bottom]:=x;
end;
```

```
Pop(C);
begin
if top > bottom then "error";
top:=top+1
return(V[top-1]);
end;
```

```
Init(C);
begin
bottom:=0;
top:=1;
end;
```

Figura 5b

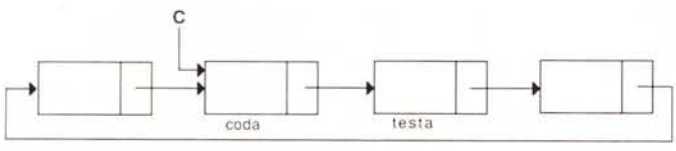


Figura 6a

```
Push (x,C);
begin
var new : reference;
new := alloca(length);
INF(new) := x;
PUN(new) := PUN(C);
PUN(C) := new;
C := new;
end;
```

```
Pop (C);
begin
var value : integer;
var temp : reference;
if C=0 then "error";
temp := PUN(C);
value := INF(temp);
PUN(C):=PUN(PUN(C));
dealloca(temp);
return(value);
end;
```

Figura 6b

TESTA, allora l'operazione

push(x,Pila)

trasforma la pila in una nuova pila Pila1=[x,a,b,c,d]; mentre l'operazione

pop(Pila)

produrrà la nuova pila PILA1=[b,c,d], restituendo, come valore dell'operazione eseguita, l'elemento «a».

Per fare un altro esempio, avremo che

pop(push(x,Pila))=x

e restituisce Pila qualunque sia Pila, mentre

pop(PilaVuota)=<errore>.

La struttura pila, ha un funzionamento comune a tutti quei processi mentali, in cui un problema in corso d'esame, è lasciato temporaneamente aperto per affrontare un sottoproblema nato nell'ambito del primo e che potrebbe a sua volta dare origine a sottoproblemi, e così via.

In ogni fase del processo, l'ultimo problema generato, balza in testa alla pila dei problemi da risolvere, ed ogni problema risolto, fa passare il controllo del processo al problema, nell'ambito del quale, esso è stato generato.

Allocazione in memoria della pila

Data una pila P, la più semplice risoluzione del problema della sua allocazione in memoria, è fornita dall'impiego di un vettore V di m elementi lunghi d, il primo dei quali risiede all'indirizzo base iB. Oltre al vettore V, sarà necessario impiegare anche un contatore TOP, il cui valore corrisponde all'indice dell'elemento che nella pila occupa la posizione di TESTA.

Osserviamo la figura 2a.

Essa rappresenta il vettore V e la variabile TOP, mediante le quali è possibile realizzare la pila P. In figura 2b sono riportate, la procedura push, la funzione pop, e la procedura di inizializzazione della pila.

Allocare in memoria una pila, mediante un vettore, è la soluzione più frequentemente adottata. Tuttavia essa presenta alcuni problemi, quasi sempre trascurabili, che potrebbero farci optare per una soluzione di tipo lista. Il problema fondamentale è la scelta della grandezza m, cioè del massimo numero di elementi che la pila può contenere. Quando ci si trova davanti a pressanti esigenze di risparmio della memoria, e le dimensioni della nostra pila sono difficilmente predicibili, può essere opportuno utilizzare come soluzione l'allocazione a lista degli elementi della pila. La soluzione è illustrata dalla figura 3a, che mostra una pila con 3 elementi («a», «b» e «c»), dei quali quello che occupa la posizione di TESTA è «a», essendo il più facile da raggiungere mediante il puntatore P alla pila.

In particolare, la procedura push(x,P), dichiara una variabile «new» di tipo «reference», vale a dire una variabile adatta a contenere valori di tipo puntatore (indirizzi di memoria); questa variabile viene poi inizializzata da una chiamata alla funzione di sistema «alloca» che restituisce un puntatore ad un'area di memoria lunga «Length». Per quanto riguarda la funzione pop, essa, dopo il doveroso controllo di «pila vuota», dichiara due variabili locali: «value», alla quale abbiamo dato tipo integer supponendo che tali siano gli elementi della lista (ma non vi è nessuna limitazione al riguardo), la quale serve a contenere il valore di TESTA della pila, da restituire all'ambiente che ha chiamato la funzione, e la variabile «temp», che serve a puntare l'elemento della lista da deallocare mediante l'altra funzione di sistema «dealloca». L'implementazione a lista della pila, appare probabilmente più complessa di quella precedente; essa tuttavia presenta notevoli vantaggi in termini di occupazione della memoria della struttura.

Diverso è il discorso per quanto riguarda la struttura dati coda, che, come vedremo, presenta delle caratteristiche che richiedono, per quanto possibile, un'allocazione a lista rispetto a quella mediante vettori.

La coda

La struttura dati coda, presenta caratteristiche funzionali molto più semplici da descrivere rispetto a quelle della pila. Basti pensare al proposito, a quanto accade (o dovrebbe accadere) davanti allo sportello di un ufficio pubblico. Un nuovo arrivato prende la sua posizione in fondo alla coda, ed avanza all'interno della coda stessa, mano mano che le persone in testa vengono servite, fino a giungere al primo elemento della coda, posizione dalla quale può essere estratto. La struttura dati coda è una struttura dinamicamente variabile e costituita da una sequenza ordinata di elementi, come la pila, che però differisce da quest'ultima per il fatto che, mentre l'operazione di rimozione avviene dalla TESTA della coda, quella di inserzione dev'essere fatta in CODA alla coda (scusate il gioco di parole).

L'impiego di una simile struttura dati è diffuso in vari contesti, aventi in comune la necessità di «tamponare» dati che potrebbero essere prodotti ad una velocità maggiore di quella alla quale essi stessi sono consumati.

Una schematizzazione di questa struttura dati è riportata in figura 4, dalla quale è possibile notare la similitudine rispetto alla pila di figura 1.

Allocazione in memoria della coda

È evidente che esiste una forte analogia tra l'allocazione in memoria della coda e quella della pila. Anche nel caso della coda, le soluzioni possibili sono due: l'organizzazione sequenziale, mostrata in figura 5, e l'organizzazione a lista, mostrata in figura 6. Invece di fare una dettagliata descrizione degli algoritmi mostrati in figura 5b e 6b, che dovrebbero essere chiari, a condizione di operare su di essi una dovuta riflessione, cerchiamo di guardare alle differenze esistenti rispetto al caso della struttura dati pila.

Osserviamo la figura 5. Inizialmente V è vuoto, bottom è zero, e top è uno. Se inseriamo tre elementi nella coda, avremo che i due contatori varranno: top=1 e bottom=3, a testimonianza del fatto che gli elementi V[1], V[2] e V[3] (cioè quelli da top a bottom) contengono dati che possono essere estratti mediante operazioni di pag. Se eseguiamo, a questo punto, due estrazioni da C, avremo che in coda rimarrà un solo elemento significativo, quello presente in V[3], e che bottom=top=3 a conferma di quanto detto. A questo punto però, le prime due posizioni di V resteranno per sempre inutilizzabili! In pratica, la zona del vettore V che contiene gli elementi della coda C, tende nel tempo, a migrare verso le posizioni più alte del vettore, provocando un inaccettabile spreco di memoria.

Questo problema potrebbe essere risolto con una complicata gestione dei contatori bottom e top; tuttavia risultati complessivamente più accettabili, sono conseguibili mediante l'implementazione a lista della coda.

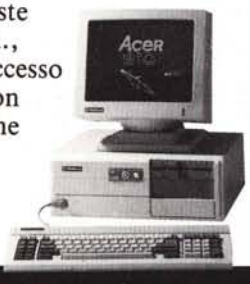
Sfruttando l'organizzazione a lista, sono possibili diverse soluzioni, delle quali quella mostrata in figura 6 dovrebbe essere la più semplice. Essa è basata sull'impiego di una lista circolare, vale a dire di una lista nella quale l'ultimo elemento punta al primo, invece di avere un puntatore nullo. Il puntatore C all'ingresso della lista, è posizionato sull'ultimo elemento della lista (l'ultimo elemento inserito nella coda), poiché da questo elemento è immediato raggiungere l'elemento di testa, che è quello da estrarre quando verrà generata un'operazione di pop.

Dall'algoritmo di pop in figura 6b, si nota che la condizione «struttura vuota», che rappresenta un errore in seguito ad un tentativo di estrazione, è rilevabile allo stesso modo di come accade nel caso della pila, cioè testando sulla condizione di puntatore d'ingresso nullo (C=0).

E questo è tutto. A risentirci. MC

Il successo della gamma Acer in Italia mette d'accordo prezzo e affidabilità.

La S.H.R. con grande orgoglio presenta la sua gamma **Acer**: una vera soluzione ai problemi di scelta dell'utente e un vero, grande successo in tutta Italia. Un successo garantito dalla elasticità di proposte S.H.R., un successo che non può che produrre altro successo.



Acer
La parola per dire valore

Le Soluzioni SHR

L'informatica dal volto umano

Società del Gruppo FERRUZZI

PER ULTERIORI INFORMAZIONI SCRIVETE A: SHR S.R.L. - CASELLA POSTALE 275 - 48100 RAVENNA - TEL. 0544/463200