6 - Liste

La lista è una sequenza finita (che può essere vuota) di elementi dello stesso tipo
A differenza di un'insieme (concetto che vedremo più in la), in una lista un elemento può apparire più volte ma in posizioni diverse.

Attenzione: la lista è una struttura dinamica, non ha dimensione prefissata.
Una lista si indica con la notazione:

$$i =< a_1, a_2, \ldots, a_n > n \geq_0$$

La posizione (astratta, che può non essere un intero) di un elemento si ottiene con: `pos(i)`.
Il valore di un elemento invece con: `a(i)`.

Accesso

Ad un lista si può accedere solo dal primo elemento della sequenza, per poi scandire tutti gli elementi prima di arrivare a quello desiderato.

La lunghezza indica gli elementi della lista.
Si dice **sottolista** una sequenza di elementi adiacenti nella lista. La lista vuota è sottolista di qualsiasi lista.

Operatori

- crealista : () $\rightarrow$ lista
  - POST: `l'= <>`
- listavuota : (lista) $\rightarrow$ boolean
  - POST: `b = true se l = <> altrimenti b = false`
- leggilista: (posizione, lista) $\rightarrow$ tipoelem
  - PRE: `p = pos(i) 1<=i<=n`
  - POST: `a = a(i)`
- scrivilista: (tipoelem, posizione, lista) $\rightarrow$ lista
  - PRE: `p = pos(i) 1<=i<=n`
  - POST: `l' = <a1, a2, ..., ai-1, a, ai+1, ..., an>`
- primolista: (lista) $\rightarrow$ posizione
  - PRE: `p = pos(1)`
- finelista: (posizione, lista) $\rightarrow$ boolean
  - PRE: `p = pos(i) 1<=i<=n+1`
  - POST: `b = true se p = pos(n+1) b = false altrimenti`
- succlista: (posizione, lista) $\rightarrow$ posizione
  - PRE: `p = pos(i) 1<=i<=n`
  - POST: `q = pos(i+1)`

- predlista: (posizione, lista) → posizione
  - PRE: `p = pos(i) 2<=i<=n+1`
  - POST: `q = pos(i-1)`
- inslista: (tipoelem, posizione, lista) → lista
  - PRE: `p = pos(i) 1<=n<=n+1`
  - POST: `l' = <a1, a2, … , ai-1, a, ai , ai+1, … , an>, se 1 ≤ i ≤ n l' = <a1, a2, … ,an, a> , se i = n+1`
- canclista: (posizione, lista) → lista
  - PRE: `p = pos(i) 1<=i<=n`
  - POST: : l' = <a1, a2, … , ai-1, ai+1, … , an> ``

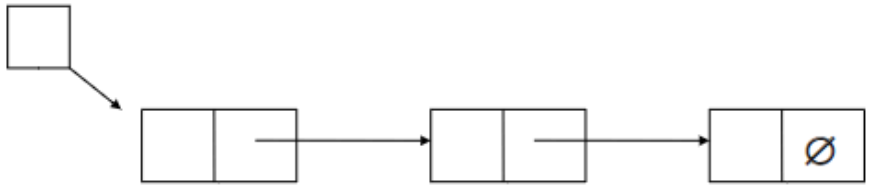
Rappresentazioni

Per rappresentare una lista abbiamo due metodi:

- **rappresentazione sequenziale**
- **rappresentazione collegata**

Per la rappresentazione sequenziale useremo un **vettore**. Questa rappresentazione consente di realizzare molto semplicemente alcune delle operazioni definite per la lista. Tuttavia riscontra problemi durante l'inserzione e la rimozione di componenti, che hanno una complessità computazionale lineare in quanto andranno spostati degli elementi per riottenere l'equivalenza tra la sequenza e il vettore.

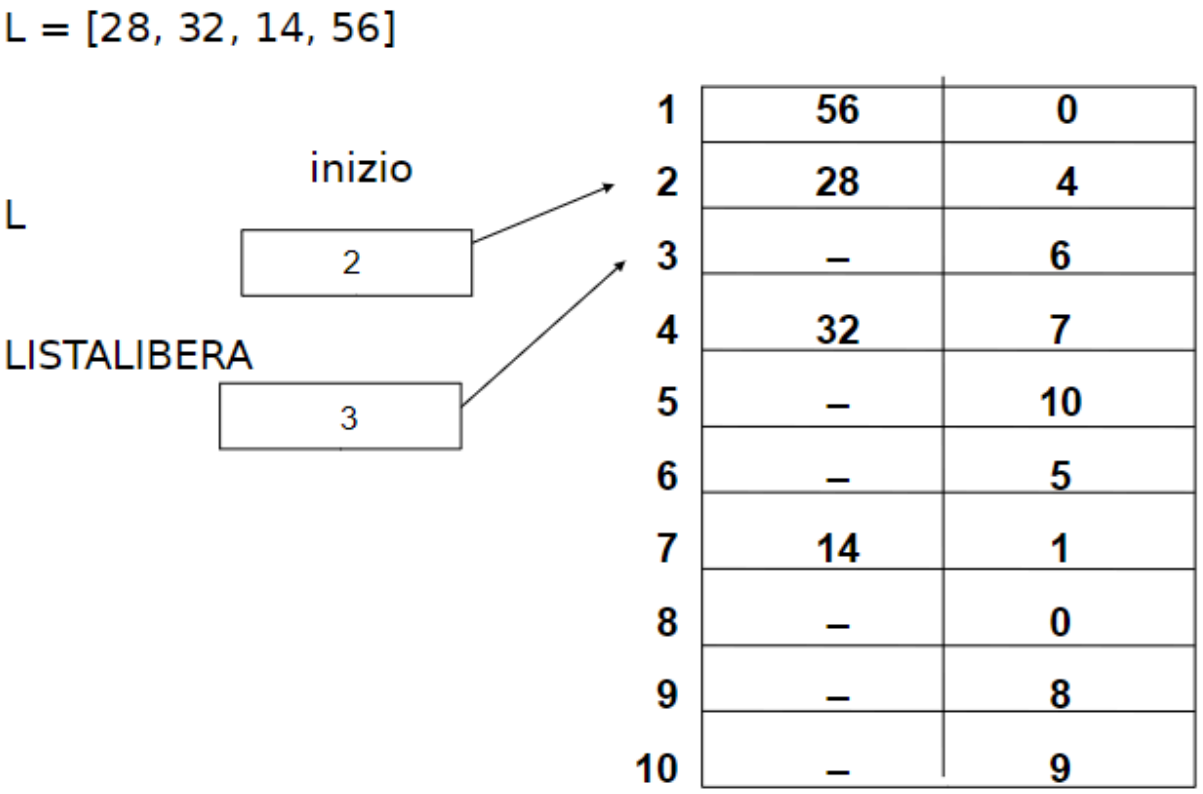
La rappresentazione collegata, invece, prevede che ogni elemento abbia un'informazione per recuperare la posizione dell'elemento successivo.



Si può notare che si usa un riferimento al primo elemento della lista (riferimento iniziale) e un simbolo speciale ∅ come riferimento associato all'ultimo nodo (nel caso la lista sia vuota, tale simbolo compare direttamente nel riferimento iniziale).

Una realizzazione di tale rappresentazione è a realizzazione con cursori, in cui viene utilizzato un vettore per l'implementazione della lista.
I riferimenti si realizzano tramite cursori, ovvero variabili il cui valore è interpretato come indice di un vettore.
Si definisce un vettore spazio che:

- può contenere più liste, ognuna individuata da un proprio cursore iniziale
- contiene tutte le celle libere, organizzate in una lista aggiuntiva, detta “listalibera”
  La listalibera rappresenta un serbatoio da cui prelevare componenti libere dell’array e in cui riversare le componenti dell’array che non sono piu’ utilizzate per la lista.

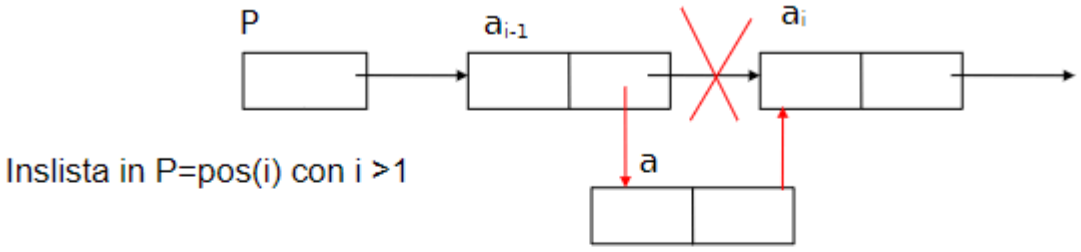


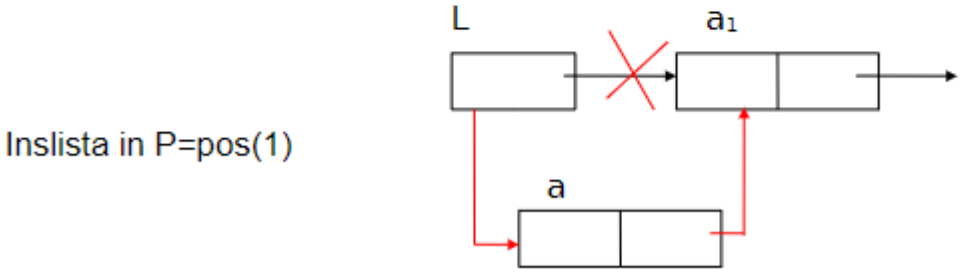
La listalibera dunque permette di evitare lo shift degli elementi presenti ogni qualvolta si effettua un inserimento o una eliminazione. Rimangono gli svantaggi connessi all’uso dell’array: la dimensione dell’array rappresenta un limite alla crescita della lista e la quantità in memoria utilizzata non dipende dalla lunghezza effettiva della lista. Inoltre, rispetto alla rappresentazione sequenziale, vi è un’ulteriore occupazione di memoria, vista la necessità di memorizzare i riferimenti.

Un'altra possibile realizzazione di una lista è quella mediante l’uso congiunto del tipo puntatore e del tipo record.
Le operazioni usualmente disponibili su una variabile di tipo puntatore p sono:

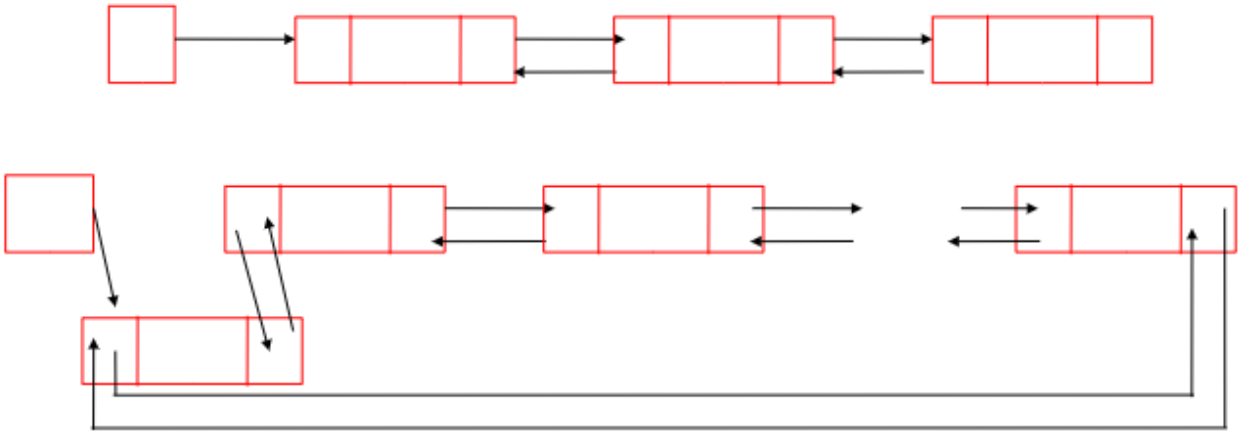
- l’accesso alla locazione il cui indirizzo è memorizzato in p ;
- la richiesta di una nuova locazione di memoria e la memorizzazione dell’indirizzo in p (new);
- il rilascio della locazione di memoria il cui indirizzo è memorizzato in p (delete) ;
  La prima cella è indirizzata da una variabile di tipo puntatore, mentre l’ultima cella punta a un valore convenzionale NULL.

Esempi di operazioni:





Una variante di tale rappresentazione è quella a doppi puntatori o simmetrica in cui ogni elemento contiene, oltre al puntatore al nodo successivo, anche il puntatore al precedente.



Con questa rappresentazione si ha il vantaggio di:

- poter scandire la lista in entrambe le direzioni
- poter individuare facilmente l'elemento che precede
- poter realizzare le operazioni di inserimento senza dover usare variabili aggiuntive

ATTENZIONE: su una lista è sconsigliata la ricerca dicotomica, in quanto non c'è una complessità computazionale costante a causa della impossibilità di accedere direttamente ad un componente.

Ordinamento di una lista

Per effettuare l'ordinamento di una lista useremo il natural merge sort.

Data una lista $l =< a_1 a_2, \ldots a_n >$, diremo che una sottosequenza <a_i, a_{i+1} . . . a_k> costituisce una catena (o run) se accade che:

- $a_{i-1} > a_i$ OR $i = 1$
- $a_j \leq a_{j+1}$ per ogni $j = i, i+1, \ldots, k-1$
- $a_k > a_{k+1}$ OR $k = n$

Ciò significa che una lista è un conseguirsi di catene.

Le catene permettono di fondere due sequenze di n catene in una singola sequenza di n catene, ottenendo una nuova lista con un numero dimezzato di catene. Grazie a ciò, dopo al massimo $\log_2(n)$ passi otterremo una lista completamente ordinata.

- Esempio:
  - l : 82 16 14 15 84 25 77 13 75 4
  - a : 82 14 15 84 13 75
  - b : 16 25 77 4

Fondendo le catene distribuite si può creare una nuova lista l il cui grado di ordinamento è maggiore. L'algoritmo di ordinamento alterna fasi di distribuzione a fasi di fusione, fino a quando si ottiene una lista con un'unica catena

Come primo passo fondiamo le catene consecutive:

l = 16 25 77 82 4 14 15 84 13 75

Ci si ferma quando la distribuzione viene messo tutto ordinatamente nella lista "unica":

l = 4 13 14 15 16 25 75 77 82 84

ATTENZIONE: manca un passaggio

```
a = 16 25 77 82  13 75
b = 4 14 15 84
l = 4 14 15 16 25 77 82 84 13 75
```

Operatori

E' possibile fare l'overloading di alcuni operatori, e quindi scrivere delle funzioni che riscrivono il comportamento di certi operatori. Tali funzioni devono essere del tipo `operator + simbolo` (esempio: operator+, operator*).

- Operatori sovraccaricabili
  - +, -, *, /, %, ^, &, |
  - ~, !, =, <, >, +=, -=, *=
  - /=, %=, ...
  - &&, ||, ==, --, ++, ...
  - [], (), new, delete, ...
- Operatori non sovraccaricabili
  - ., .*, ::, ?:, sizeof

Tuttavia ci sono delle restrizioni:

- Non è possibile cambiare il numero di operatori su cui un operatore agisce (quindi il + dovrà sempre lavorare su due elementi, come ++ su un singolo elemento).
- La precedenza con cui vengono eseguite rimane invariato (ad esempio * e / verranno sempre eseguite prima di + e -).
- Non si possono creare nuovi operatori.
- Per i tipi primitivi non è possibile effettuare l'overloading (esempio: non è possibile modificare la somma di due interi).

Ereditarietà e classi astratte

Una classe B (classe derivata) può derivare da una classe A (classe base). Ogni classe derivata può accedere a tutti i campi della base definiti protected.

Chiamasi classe astratta una classe che contiene metodi senza implementazione (funzioni virtuali).

```
virtual int funzioneVirtuale(int x);
```

Le classi con solo funzioni implementate sono dette classi effettive, e sono le uniche classi che è possibile istanziare.
In pratica useremo le classi astratte per definire le funzioni disponibili al posto del file header (.h).
Tutte le classi derivate da una classe astratta sono delle classi astratte e quindi non possono essere istanziate a meno che non implementino tutte le funzioni virtuali della classe base.
Possiamo usare i template per implementare liste di ogni tipologia di cui abbiamo bisogno.

```
template< class T >
     class Lista{
```

```
        public:
        ...
        private:
    ...
    }
```

ATTENZIONE: La definizione e l'implementazione di un template devono risiedere nello stesso file.

Un modo per implementare la lista è usare l'array doubling, ovvero un vettore che quando diventa pieno, raddoppia la sua dimensione massima. Per evitare sprechi di spazio, qualora il numero di elementi presenti nel vettore scenda al di sotto di un quarto della dimensione corrente, la sua dimesione verrà, invece dimezzata.