Array e record

Le strutture di dati si classificano in

- **lineari** (gli elementi sono in sequenza, cioè formano una **lista lineare**)
  - sequenzialità delle locazioni di memoria: **array**
  - relazione lineare tra gli elenti consiste in indici: **liste concatenate**
- **non lineari**
  - **alberi**
  - **grafi**

Strutture lineari, sia array o lista concatenata, comprendono:

- Inserimento
- Cancellazione
- Attraversamento
- Ricerca (lineare, binaria)
- Riordino
- Fusione

La scelta della struttura dipende dalla frequenza con cui si eseguono le diverse operazioni (poche per gli array, molte per le liste concatenate).

Array lineari

Per array lineare si intende la lista di un numero finito n di elementi omogenei per i quali:

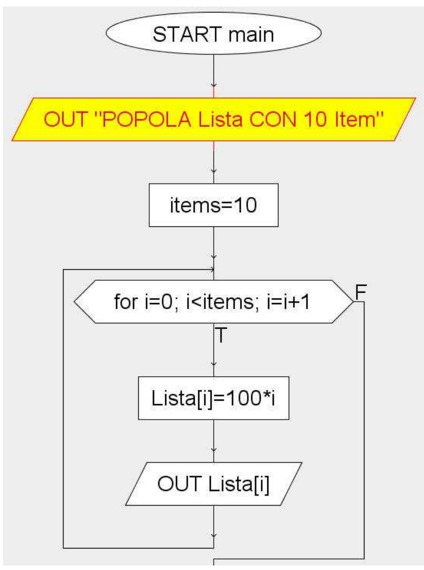
- Il riferimento ai vari elementi si fa con un indice
- Gli elementi sono immagazzinati in locazioni di memoria successive

La lunghezza o dimensioni dell'array vale

Lunghezza = Limite Superiore - Limite Inferiore + 1

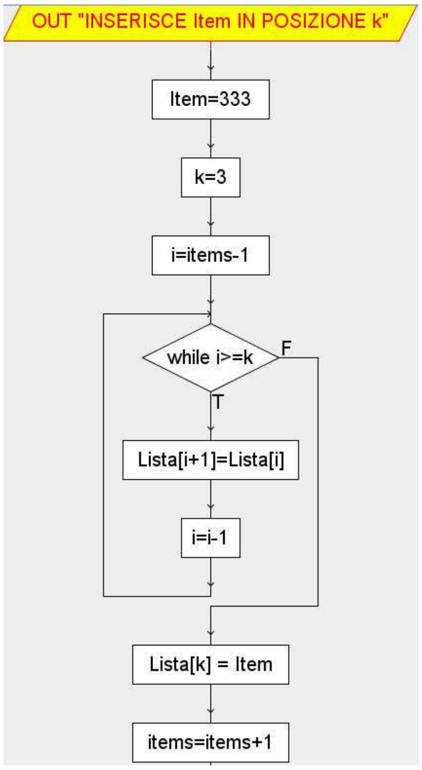
Gli elemti si possono caratterizzare con un indice:

$A_1, A_2, A_3, ..., A_n$; A(1), A(2), ..., A(n); Lista[1], Lista [2], ..., Lista [n]



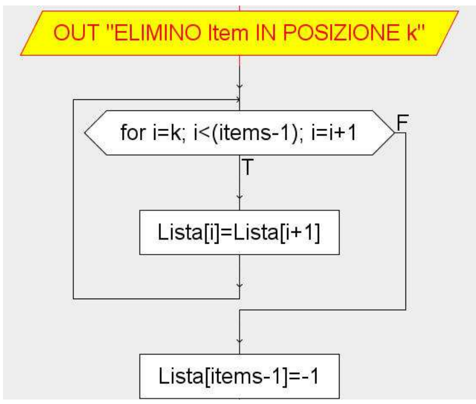
Inserimento e cancellazione

Dato un array lineare Lista di n=10 elementi l'algoritmo seguente inserisce un elemento in Lista nella posizione k =3<= n:



1. Inizializzo contatore: i = n
2. Ripeto i passi 3 e 4 while i >= k
3. Muovo l'elemento iesimo: Lista [i+1] = Lista [i]
4. Decremento contatore i--
5. Fine loop
6. Inserisco in Lista [k] = Item
7. Reset n++
8. Fine

Dato un array lineare Lista di n elementi l'algoritmo seguente elimina un elemento in Lista nella posizione k=3<= n:

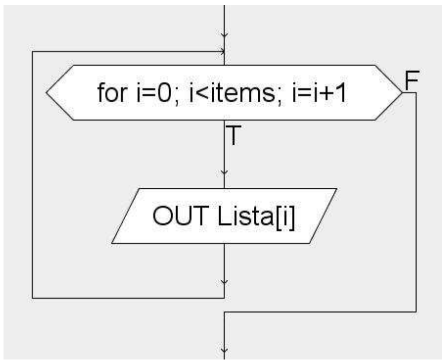


1. Conservo Item = Lista[k]
2. Ripeto per i = k fino a n-1
3. Muovo l'elemento i+1: Lista[i] = Lista[i+1]
4. Fine loop
5. Reset n--
6. Fine

Attraversamento degli array lineari

Dato un array lineare A gli algoritmi seguenti di attraversano A applicando un PROCESSO:

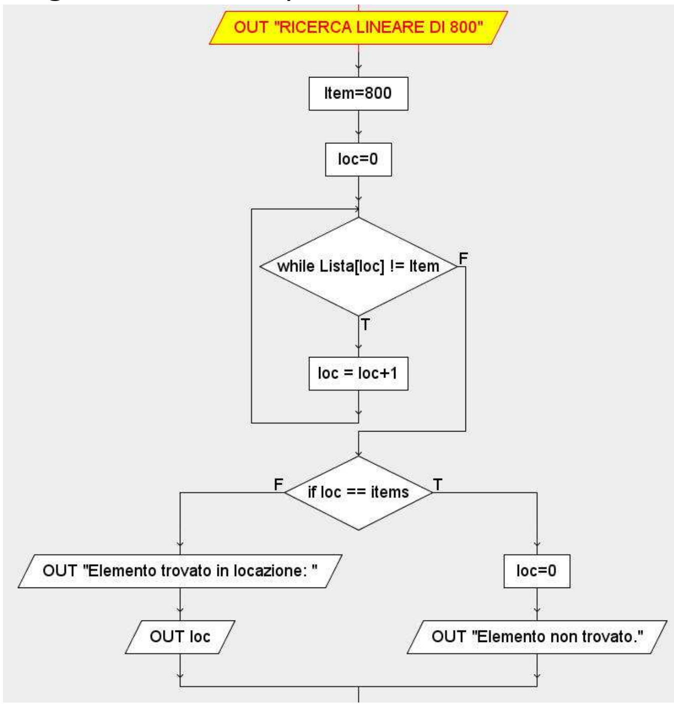
1. Inizializzo contatore: k = LI
2. Ripeto i passi 3 e 4 while k <= LS
3. Visito elemento e PROCESSO A[k]
4. Incremento contatore k++
5. Fine loop
6. Fine



1. Ripeto per k = LI fino a LS
2. Visito elemento e PROCESSO A[k]
3. Fine loop
4. Fine

Ricerca lineare

Dato un array lineare A di n elementi l'algoritmo seguente trova la poszione loc di Item in A:

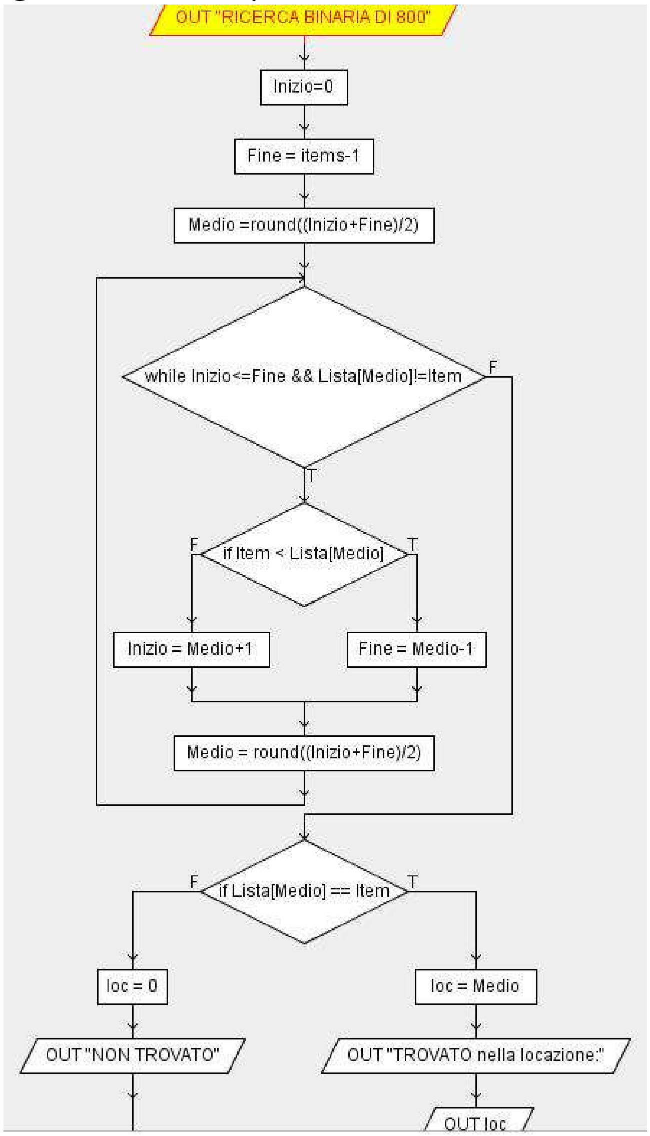


1. Conservo A[n+1] = Item
2. Inizializzo loc = 1
3. Ripeto while A[loc] != Item
4.   loc = loc + 1
5. Fine loop
6. Trovato? if (loc == n+1) then loc = 0
7. Fine

Complessità $f(n) = (n + 1)/2$

Ricerca binaria

Dato un array lineare A di n elementi l'algoritmo seguente trova la poszione loc di Item in A:



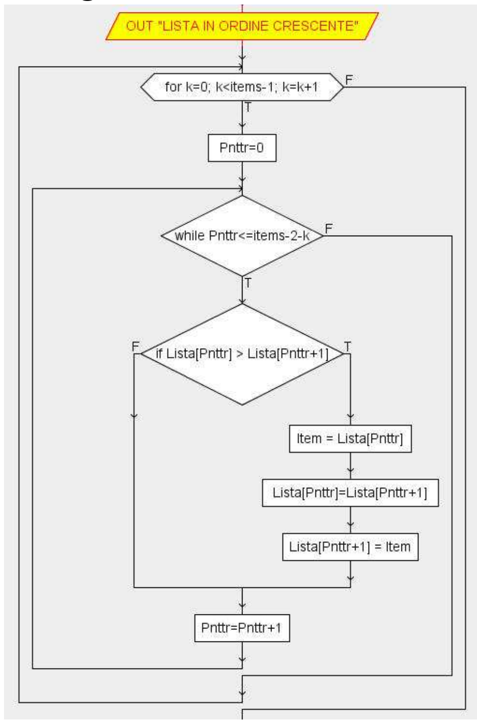
1. Inizializzo: Inizio = LI; Fine = LS; Medio = int (Inizio + Fine)/2
2. Ripeto passi da 3 a 8 while Inizio <= Fine AND A[Medio] != Item
3.   if (Item < A[Medio] then
4.   Fine = Medio - 1
5.   else
6.   Inizio = Medio + 1
7.   FineIF
8.   Medio = int (Inizio + Fine)/2
9. FineLoop
10.   if (A[Medio] == Item) then
11.   loc = Medio
12.   else
13.   loc = 0
14.   FineIF
15. Esci

Complessità $f(n) = (\log_2 n) + 1$

Riordino bubble

Dato un array lineare A di n elementi l'algoritmo seguente ordina gli elementi di A:

1. Ripeto per k = 1 fino a n-1
2.   Inizializzo Pnttr = 1
3.   Ripeto while Pnttr <= n-k
4.   if (A(Pnttr) > A(Pnttr+1) then
5.   Scambio A(Pnttr) con A(Pnttr+1)
6.   FineIf
7.   Pnttr = Pnttr+1
8.   FineWhile
9. FineLoop
10. Esci

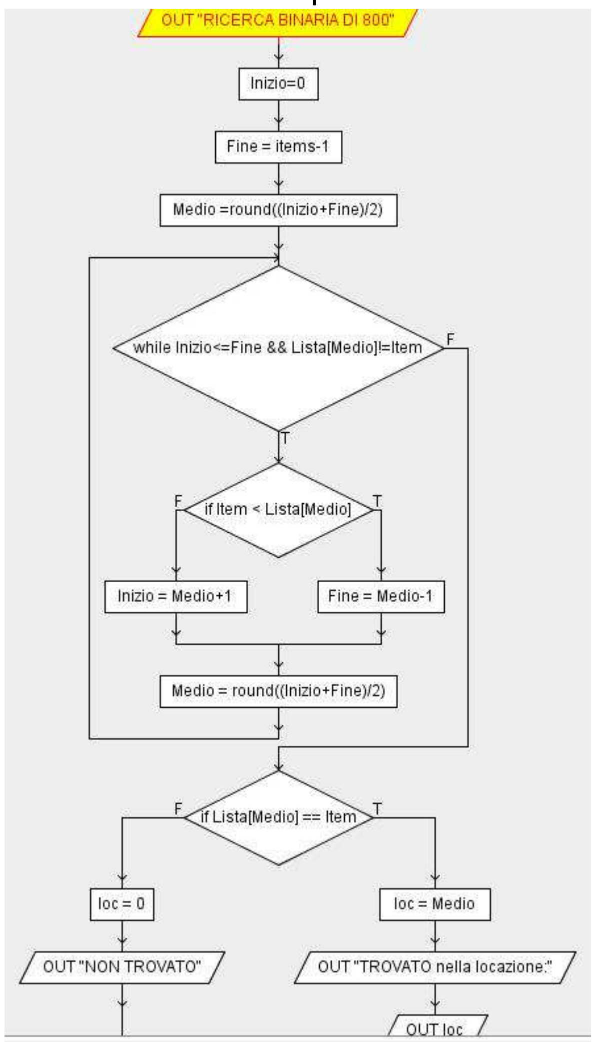


Complessità $f(n) = n^2$

Fusione

Siano A e B degli array ordinati di r ed s elementi rispettivamente. Questo algoritmo fonde A e B in C di n = r + s. Siano A[nA] ed B[nB] i più piccoli elementi non ancora posti in C.

1. nA = 1; nB = 1; Pnttr = 1
2. Ripeto while nA <= r AND nB <= s
3.   if (A(nA) > B(nB)) then
4.   C(Pnttr) = A(nA)
5.   Pnttr = Pnttr +1; nA = nA +1
6.   else
7.   C(Pnttr) = B(nB)
8.   Pnttr = Pnttr +1; nB = nB +1
9.   EndIf
10. FineWile
11. // Assegna a C gli elementi che restano
12. if (nA > r) then
13.   Ripeto per k = 0 fino a s -nB
14.   C[Pnttr+k] = B[nB+k]
15.   FineLoop
16. else
17.   Ripeto per k = 0 fino a r -nA
18.   C[Pnttr+k] = A[nA+k]
19.   FineLoop
20. FineIf
21. Esci



Complessità $f(n) = n$