ALGORITMI DI BASE - Parte 1: SCANSIONE, SOMMA, RICERCA, MINIMO

Algoritmi di Base per l'Elaborazione di Liste di Dati

Per maturare una propria competenza e abilità nella produzione del software, è necessario conoscere una serie di **Algoritmi di Base** iniziando da quelli mirati all'elaborazione di **Liste di Dati**.

Descriveremo tali algoritmi applicandoli ai **Vettori**, ma essi sono *ugualmente applicabili a qualsiasi struttura o oggetto che gestisca una lista di dati* con accesso tramite *indice* (stringhe, liste tipizzate, controlli con liste, ecc.).

Nell'esposizione indicheremo sempre genericamente con il nome **"Vet"** il vettore che contiene la *Lista di Dati*, con il nome **"K"** (o "K1", "K2") l'*indice dei cicli* e con il nome **"N"** il *numero di dati reali* presenti nel vettore stesso.

Scansione generica di un Vettore

Come già visto nell'uso dei vettori, per effettuare la **Scansione Completa di un Vettore**, si utilizza un ciclo a conteggio:

for (int K = 0; K <= N-1; K++).

L'indice **K**, ad ogni ripetizione, cambia, scorrendo così tutte le **posizioni** del vettore dalla prima (**0**) all'ultima (**N-1**)
Ad ogni passo del ciclo, per agire su un *diverso elemento della lista*, si utilizza la notazione **Vet [K]**.

```
for ( int K = 0;  K <= N-1;  K++)
{  … operazioni su  Vet [ K ] …  }
```

Al primo passo, **K vale 0**, quindi la notazione Vet[K] diventa **Vet[0]** e si accede al primo elemento del vettore.
Al secondo passo, **K vale 1**, e la notazione Vet[K] diventa **Vet[1]** e si accede al secondo elemento; e così via.

Spesso capita di dover effettuare delle **Scansioni Parziali di una Lista**.
Per limitare la scansione solo a una parte della lista si possono **variare il valore iniziale e/o finale dell'indice**:

for (int K = A; K <= B; K++)	*… scandisci dalla posizione A alla posizione B*
for (int K = 0; K <= B; K++)	*… scandisci dall'inizio (posizione 0) alla posizione B*
for (int K = A; K <= N-1; K++)	*… scandisci dalla posizione A alla fine (posizione N-1)*

Oppure potrebbe essere necessario **variare il "passo" del ciclo**:

for (int K = 0; K <= N-1; K+=2)	*… scandisci solo gli elementi di posto PARI (0, 2, 4, …)*
for (int K = 1; K <= N-1; K+=2)	*… scandisci solo gli elementi di posto DISPARI (1, 3, 5, …)*
for (int K = N-1; K >= 0; K--)	*… scandisci "all'indietro" dalla fine (pos. N-1) all'inizio (pos. 0)*

Potrebbe **non essere noto il numero di passi** che il ciclo deve effettuare: in questo caso non è possibile utilizzare un "for" ed è necessario utilizzare un ciclo **while** (o un ciclo **do-while**) che *ripeta "fino al verificarsi di una condizione"*.
La gestione dell'indice, in assenza del *for*, deve essere realizzata con apposite istruzioni:

```
int K = 0;
while  ( …condizione di permanenza… )
{
  … operazioni su  Vet [ K ] …
  K++;
}
```

```
int K = 0;
do
{
  … operazioni su  Vet [ K ] …
  K++;
} while ( …condizione di permanenza … )
```

Si noti come l'incremento del contatore K sia situato sempre come *ultima operazione del ciclo*: in tal modo, il primo passo del ciclo stesso, viene eseguito interamente con K = 0, agendo, quindi, sul primo elemento della lista. Inoltre le condizioni vengono valutate immediatamente dopo l'incremento stesso.

Somma dei Dati di un Vettore

Per effettuare la **"Somma dei Dati in un Vettore"**, si utilizza una **variabile accumulatore** che chiameremo "**Somma**" e imposteremo inizialmente al valore **0**.

Si effettua una *scansione completa* della lista e, ad ogni passo, si "aggiunge" a Somma il dato in posizione K.

```
double Somma = 0;                  … dichiara e azzera Somma (i dati sono numeri decimali, di tipo double)
for ( int K = 0;  K <= N-1;  K++)  … scansione completa della lista
{    Somma = Somma + Vet[K];   }   … operazione di "accumulo" del dato Vet[K] in Somma …
```

Al termine della scansione, in Somma è presente la **somma di tutti i dati del vettore**.
Si noti che **se la Lista è Vuota** (ossia N = 0), *il for non effettua nessun passo* e, in Somma, rimane il valore 0.

L'algoritmo è identico anche se i dati sono di tipo *string*. In tal caso, si dichiara Somma di tipo *string*, inizializzandola a *stringa vuota* (Somma = " ") e l'operatore "+" applica un **concatenamento**:

```
string  Somma = " ";
for ( int K = 0;  K <= N-1;  K++)
{    Somma = Somma + Vet[K];   }
```

Il risultato è un'unica stringa Somma che contiene **tutte le stringhe del vettore accodate** una dopo l'altra.

Ricerca della Posizione di un Dato in un Vettore (Ricerca Sequenziale)

Per "**Ricerca di un Dato in un Vettore**", si intende la determinazione della **Posizione** del dato nella lista stessa.
Se **il Dato non è presente**, si restituisce, come risultato, **un valore speciale pari a "-1"**.

Il dato di cui cercare la posizione, è contenuto in una variabile di nome **DatoDaCercare**.

E' necessario utilizzare una variabile di nome **Posizione**, destinata a contenere il risultato. Essa viene *inizializzata al valore speciale "-1"*: se il dato non viene trovato, Posizione resta inalterata e, al termine, contiene ancora "-1".

Si effettua una *scansione completa* della lista e, ad ogni passo, **si confronta il Dato del Vettore indicato da K con il Dato da Cercare**. In caso di uguaglianza, significa che, in posizione K, è presente il valore cercato e, quindi, *si memorizza K nella variabile Posizione*:

```
int Posizione = -1;                          … dichiara Posizione e inizializzala a "-1"
for ( int K = 0;  K <= N-1;  K++)            … scansione completa della lista
{
  if  ( Vet[K] == DatoDaCercare )            … ad ogni passo, confronta Vet[K] con il dato da cercare
  {    Posizione = K;   }                    … se sono uguali, "salva" in Posizione il valore di K
}
```

Questa tecnica di ricerca è detta **Ricerca Sequenziale** poiché *scandisce sequenzialmente*, dal primo all'ultimo, tutti gli elementi della Lista, allo scopo di individuare quello cercato.

Nel caso particolare in cui, il dato da cercare esista più volte nella lista (ossia il dato ha più occorrenze), questo algoritmo restituisce la **posizione dell'ultima occorrenza**.

```
int Posizione = -1;
for ( int K = 0;  K <= N-1;  K++)
{
   if  ( Vet[K] == DatoDaCercare )
   {
        Posizione = K;
        break;
   }
}
```

L'algoritmo precedente può essere *ottimizzato*, notando che **la ricerca si può interrompere appena si trova il dato**.

In tal modo, si evita di proseguire inutilmente la scansione fino al termine della lista, che, in caso di dati numerosi, può essere anche molto onerosa.

Per "interrompere" il ciclo si usa l'**istruzione break**.

Nel caso particolare in cui, il dato da cercare esista più volte nella lista (ossia il dato ha più occorrenze), questo algoritmo restituisce la **posizione della prima occorrenza**.

Ricerca della Posizione del Minimo (o del Massimo) in un Vettore

Per "**Ricerca del Minimo in un Vettore**", si intende la determinazione della **Posizione** del dato con il **Valore più Piccolo** fra tutti quelli della lista stessa.

E' necessario utilizzare una variabile di nome **PosMin**, destinata a contenere il risultato, ossia la *Posizione del Minimo*. Essa viene *inizializzata al valore "***0***"*: quindi si ipotizza, inizialmente, che *il minimo si trovi in prima posizione*.

Si noti che, se *PosMin* indica la posizione del minimo, allora con **Vet [PosMin]** si accede al minimo vero e proprio.

Si effettua una *scansione completa* della lista e, ad ogni passo, *si confronta il dato in esame* (**Vet [K]**) *con il minimo fino ad ora individuato* (**Vet [PosMin]**). In caso il dato esaminato risulti più piccolo del minimo, **allora è esso stesso il minimo**: si procede allora aggiornando *PosMin*, memorizzandovi l'indice contenuto in *K*.

```
int PosMin = 0;                    … dichiara PosMin e inizializzala a "0"
for ( int K = 1;  K <= N-1;  K++)  … scansione del Vettore dalla seconda posizione in poi
{
   if  ( Vet[K] < Vet[PosMin] )    … se il dato in esame è minore del minimo trovato finora …
   {   PosMin = K;   }             … allora K è la posizione del nuovo minimo
}
```

Si noti che **la scansione inizia dalla posizione 1** (secondo elemento): se K iniziasse da 0, il primo confronto sarebbe inutile, in quanto anche *PosMin* è inizialmente 0 … *Vet[K]* e *Vet[PosMin]* indicherebbero lo stesso elemento!

Il risultato così ottenuto in *PosMin* indica la *posizione del minimo*. Volendo ottenere **il valore vero e proprio del minimo** è sufficiente "accedere" all'elemento, scrivendo **Vet [PosMin]**.

L'algoritmo per la **Ricerca del Massimo in un Vettore** è del tutto analogo a quello del minimo: è sufficiente usare **l'operatore "maggiore"** anziché l'operatore "minore" nell'if e denominare la **variabile PosMax** anziché PosMin.