5.2 Ricerca completa

Un primo algoritmo per determinare se un valore è presente all'interno di un array, applicabile anche a sequenze non ordinate, è quello comunemente detto di *ricerca completa*, che opera una scansione sequenziale degli elementi del vettore confrontandoli con il valore ricercato. Nel momento in cui tale verifica dà esito positivo la scansione ha termine e viene restituito l'indice dell'elemento all'interno dell'array stesso.
Per determinare che il valore non è presente, il procedimento (Listato 5.1) deve controllare uno a uno tutti gli elementi fino all'ultimo, prima di poter sentenziare il fallimento della ricerca. L'array che conterrà la sequenza è `vet` formato da `MAX_ELE` elementi.

```
/* Ricerca sequenziale di un valore nel vettore */

#include <stdio.h>
#define MAX_ELE 1000   /* massimo numero di elementi */

main()
{
char vet[MAX_ELE];
int i, n;
char c;

/* Immissione lunghezza della sequenza */
do {
  printf("\nNumero elementi: ");
  scanf("%d", &n);
}
while(n<1 || n>MAX_ELE);

/* Immissione elementi della sequenza */
for(i=0; i<n; i++) {
  printf("\nImmettere carattere n.%d: ",i);
  scanf("%1s", &vet[i]);
}

printf("Elemento da ricercare: ");
scanf("%1s", &c);
```

```
/* Ricerca sequenziale */
i = 0;
while(c!=vet[i] && i<n-1)  ++i;
if(c==vet[i])
  printf("\nElemento %c presente in posizione %d\n",c,i);
else
  printf("\nElemento non presente!\n");
}
```

Listato 5.1 Ricerca completa

Il programma presenta le solite fasi di richiesta e relativa immissione del numero degli elementi della sequenza e dei valori che la compongono. Successivamente l'utente inserisce il carattere da ricercare, che viene memorizzato nella variabile `c`. La ricerca parte dal primo elemento dell'array (quello con indice zero) e prosegue fintantoché il confronto fra `c` e `vet[i]` dà esito negativo e contemporaneamente `i` è minore di `n-1`:

```
while(c!=vet[i] && i<n-1)  ++i;
```

Il corpo del ciclo è costituito dal semplice incremento di `i`. L'iterazione termina in tre casi:

1. 1. `c` è uguale a `vet[i]` e `i` è minore di `n-1`;
2. 2. `c` è diverso da `vet[i]` e `i` è uguale a `n-1`;
3. 3. `c` è uguale a `vet[i]` e `i` è uguale a `n-1`.

In ogni caso `i` ha un valore minore o uguale a `n-1`, è dunque all'interno dei limiti di esistenza dell'array. L'`if` successivo determinerà se è terminato perché `c` è risultato essere uguale a `vet[i]`:

```
if(c==vet[i])
```

Esistono molte altre soluzioni al problema. Per esempio si potrebbe adottare un costrutto `while` ancora più sintetico, come il seguente:

```
while(c!=v et[i] && i++<n-1)
        ;
```

dove il corpo del ciclo non è esplicitato in quanto l'incremento di `i` avviene all'interno dell'espressione di controllo. Si noti però che, in questo caso, al termine delle iterazioni `i` ha un valore maggiorato di uno rispetto alla condizione che ha bloccato il ciclo, e di questo bisognerà tener conto nel prosieguo del programma.