Ci sono operazioni e compiti che si ripetono spesso e per i quali si posso utilizzare delle implementazioni più o meno comuni e molto spesso ottimizzate. Illustreremo esempi di ordinamento, ricerca massimi e minimi, e ricerca di un elemento all' interno di un vettore col ritorno dell'indice.

Ordinamento

Premettiamo che bubble sort non è l'algoritmo da usare per ordinare un array di interi, è solo un esempio accademico; si usa invece un'implementazione di sorting già ottimizzata, contenuta nella libreria `stdlib.h`, ad esempio `qsort`; vanno quindi forniti:

- il puntatore all'array
- il numero di elementi nell'array
- la dimensione di un elemento dell'array
- un funzione di comparazione

```
1  #include <stdlib.h>
2
3  int compare_int( const void* a, const void* b)
4  {
5       int int_a = * ( (int*) a );
6       int int_b = * ( (int*) b );
7
8       if ( int_a == int_b ) return 0;
9       else if ( int_a < int_b ) return -1;
10      else return 1;
11 }
12
13 const size_t num_elem = 10;
14 int elements[num_elem] = { 5, 4, 2, 7, 9, 1, 0, 3, 8, 4 };
15 qsort( elements, num_elem, sizeof(int), compare_int );
```

La funzione di comparazione deve essere implementata esternamente da `qsort` e deve seguire delle regole precise: - prendere in argomento due puntatori a void da comparare (che saranno due elementi del vettore) che nel nostro caso specifico diventano degli `int`, se fosse stato un altro tipo, anche `float` al posto degli int avremmo messo `float`;
- restituire un valore maggiore, uguale o minore di zero per i rispettivi casi delle righe 6-8;
Per invertire l'ordine basta scambiare i return -1 e 1, oppure il < col >. Nel caso di array di stringhe, per ordinarle in ordine alfabetico basta ricordare che ogni tipo `char` viene associato ad un decimale della tabella ASCII, quindi possiamo riutilizzare `qsort`:

```
1  #include <stdlib.h>
2  #include <stdio.h>
3  #include <string.h>
4
5  static int compare_string (const void * a, const void * b) {
6    return strcmp (*(const char **) a, *(const char **) b);
7  }
8
9  int main() {
10   const char *vett[] = {"Antartide", "Giada", "Zoo"};
11   int n = sizeof(vett)/sizeof(vett[0]);
12
13 qsort (vett, n, sizeof (const char *), compare_string);
14
15 return 0; }
```

Bisogna fare attenzione a come è dichiarato l'array, in futuro, nella maggior parte dei casi, opereremo su un puntatore alla struttura allocata in memoria nella forma di `*vett`, perciò descriviamo in questo modo il sort di stringhe.

Massimo e minimo

L'implementazione più immediata è una scansione di tutti gli elementi del vettore e segnarsi l'elemento più grande/piccolo di tutti conservando l'indice.
Per salvare più indici per "massimi doppioni" serve un'implementazione un po' più avanzata che richiede un vettore parallelo di egual dimensione per segnarsi tutti gli eventuali indici in quel vettore.

```c
#include <stdio.h>

int main() {
  int dim = 10;
  int array[10] = {1,5,73,38,46,5,83,6,13,45};
  int max, i, indice = 1;

  max = array[0];

  for (i = 1; i < dim; i++)
  {
    if (array[i] > max)
    {
       max  = array[i];
       indice = i+1;
    }
  }

  printf("Posizione del massimo %d ed il suo valore %d.\n", indice, max);
  return 0; }
```

Naturalmente, come nell'esempio precedente, basta sfruttare la simmetria della soluzione per ottenere il minimo invece che il massimo.

Trova stringa

Per trovare un stringa facciamo uso della funzione `strcmp()` di `string.h` che ci restituisce 0 se le due stringhe coincidono, altrimenti un numero corrispondente a quanti caratteri è più grande o più piccola l'una dall'altra. Iteriamo sul vettore e interrmopiamo il ciclo quando trovata ritornando l'indice.
Di seguito un'esempio di riferimento:

```c
#include <string.h>
#include <stdio.h>
#include <stdlib.h>

int str_in(char *mia_str,const char *lista_str[], size_t num_str)
{
    for ( int i = 0; i < num_str; i++ )
        if (strcmp(mia_str, lista_str[i]) == 0 )
            return i;

    return -1;
}

int main() {

  const char *vett[] = {"Ciao", "Andiamo", "Zorro"};
    int num_str = 3;
  char str_mia[5] = "Zorro";

    int ris = str_in(str_mia,vett,num_str);
    if(ris)
      if(ris!=num_str)
          printf("Trovato in %d", ris);
        else
          printf("non trovata");
    else
      printf("errore");

return 0; }
```