Università degli Studi di Ferrara

Department of Engineering Ferrara

Esercitazioni di

FONDAMENTI DI INFORMATICA MODULO B

ESERCITAZIONE 2

Azzolini Damiano: damiano.azzolini@unife.it
Bertagnon Alessandro: alessandro.bertagnon@unife.it

Esercizio 1. Inserimento ordinato

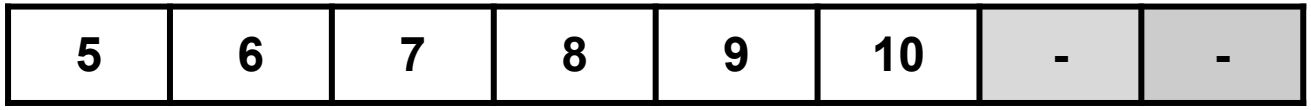
- Si scriva un programma in C che legge da tastiera una sequenza di numeri interi, uno alla volta, e li inserisce in un vettore ordinati in senso **non decrescente** preservando l'ordinamento del vettore ad ogni passo di inserimento. Il vettore ha dimensione fisica DIM e dimensione logica N.

- L'inserimento termina quando l'utente inserisce 0 o quando l'array è pieno (ovvero quando sono stati già inseriti DIM elementi).

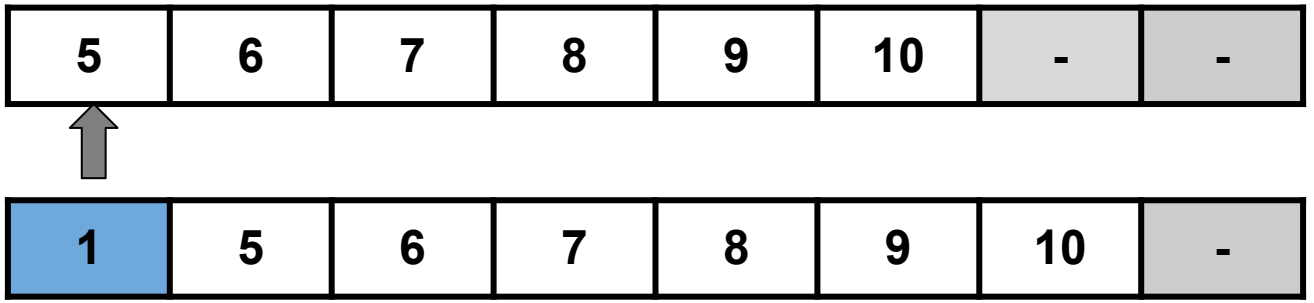
- Per inserire ciascun nuovo valore nel vettore si utilizzi la funzione **`insert`** (da definire nel file **insert.c** come indicato nelle slide successive).

Esercizio 1. Inserimento ordinato

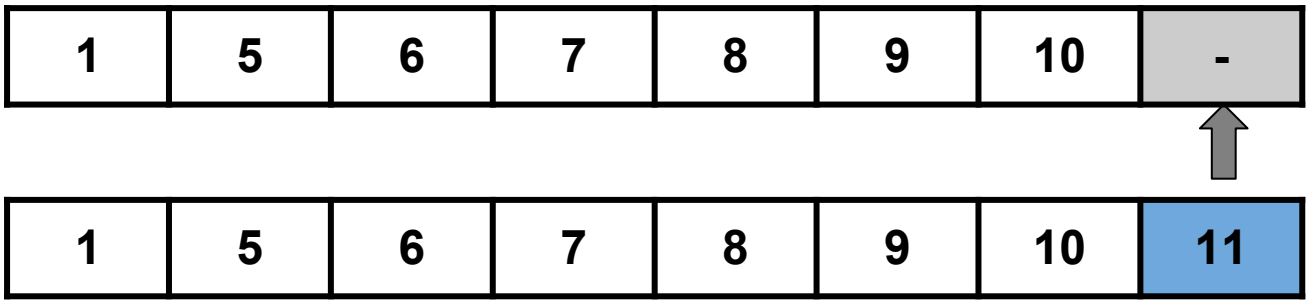
Esempio N = 6, DIM = 8:



- Inserisco $newItem_0 = 1$



- Inserisco $newItem_1 = 11$



- Inserisco $newItem_3 = 13$, cosa succede?

Esercizio 1. Inserimento ordinato

Interfaccia (**insert.h**):

```
typedef enum {FALSE, TRUE} boolean;

boolean insert(int v[], int newItem, int DIM, int *N);
```

- `int v[]` è l'array in cui voglio inserire il nuovo valore
- `int newItem` è il nuovo valore da inserire
- `int DIM` è la dimensione fisica dell'array v[]
- `int *N` è la dimensione logica dell'array v[]

Il valore di ritorno di `insert` indica se l'inserimento è stato effettuato correttamente.

Quando restituire ***false***?

Esercizio 2. Ordinamento di un array: qsort

- Dato un vettore di reali di dimensione DIM, si inseriscano k elementi presi da tastiera (k <= DIM), si utilizzi 0 per terminare l'inserimento.

- Ordinare gli elementi del vettore sfruttando la funzione di libreria **qsort** (**parametrica nella funzione di confronto**), definita nella libreria **stdlib.h**.

- Si stampi infine il vettore ordinato.

- Si strutturi il programma su più file. In particolare implementare la funzione di confronto da passare alla qsort in un file diverso da quello che implementa la funzione main.

Esercizio 2. Ordinamento di un array: qsort

Funzione generica di libreria (stdlib.h):

```
void qsort(void *base, size_t n, size_t width,
int(*fcmp)(const void *el1, const void *el2));
```

- **base** punta al primo elemento del vettore da ordinare.
- **n** è il numero di elementi del vettore (size_t è un tipo definito in stdlib.h, in genere è un unsigned long).
- **width** è la dimensione di ogni elemento del vettore in byte.
- **fcmp** è la funzione di confronto che prende come parametri due puntatori ai due elementi del vettore da confrontare e deve restituire:
  - un intero < 0 se *el1 < *el2
  - un intero > 0 se *el1 > *el2
  - 0 se *el1 == *el2

Consultare l'help di Visual Studio per ulteriori informazioni sull'uso della funzione qsort.

Esercizio 3. Ordinamento di un array : qsort

Si scriva un programma in C, simile al precedente, che però sia in grado di ordinare un vettore di elementi di tipo generico (ELEMENT).

Per permettere tale funzionalità viene richiesto di realizzare apposite librerie per rendere il programma il più modulare possibile. Supponiamo di voler fornire supporto per dati di tipo intero e dati di tipo float, a tal fine si definisca:

- un file di libreria (*element_float.c* e *element_float.h*) per gestire elementi di tipo float;
- un file di libreria (*element_int.c* e *element_int.h*) per gestire elementi di tipo intero.

Esercizio 3. Ordinamento di un array: qsort

Ciascun file di libreria dovrà per prima cosa definire il tipo di dato ELEMENT nel seguente modo:

typedef float ELEMENT (per il file *element_float.h*)

typedef int ELEMENT (per il file *element_int.h*)

e le corrispondenti funzioni:

1. **`int compare(const void *el1, const void *el2)`** da usare con la funzione qsort;
2. **`ELEMENT insert()`** quando invocata richiede all'utente di inserire da tastiera un dato di tipo ELEMENT e lo restituisce;
3. **`void print(ELEMENT el)`** stampa a video l'elemento **`el`** di tipo ELEMENT passato come parametro.

Esercizio 3. Ordinamento di un array : qsort

- Il file contenente la funzione main includerà **UNA** sola tra le due librerie appena definite.
- Il main dovrà dichiarare un vettore di dimensione DIM di dati di tipo ELEMENT, prendere in ingresso da tastiera i valori di tipo ELEMENT tramite la funzione **`insert`** (dopo ogni inserimento il programma dovrà chiedere se si vuole inserire un altro elemento o fermarsi), inserendoli nel vettore.
- Ordinare il vettore di elementi sfruttando la funzione qsort e stampare il vettore ordinato.
- Scambiando la libreria inclusa (**element_float** con **element_int**), la funzione main funzionerà ugualmente senza necessità di modificare alcuna sua parte.

Esercizio 3. Ordinamento di un array : qsort

main.c + element_int.h + element_int.c

```
Inserisci un numero intero: 5
Inserisci un altro elemento? (y/n) : y
Inserisci un numero intero: 2
Inserisci un altro elemento? (y/n) : y
Inserisci un numero intero: 8
Inserisci un altro elemento? (y/n) : n
Sono stati inseriti 3 elementi su 10 disponibili: 2 5 8
```

main.c + element_float.h + element_float.c

```
Inserisci un numero reale: 7.1
Inserisci un altro elemento? (y/n) : y
Inserisci un numero reale: 5.4
Inserisci un altro elemento? (y/n) : y
Inserisci un numero reale: 2.2
Inserisci un altro elemento? (y/n) : n
Sono stati inseriti 3 elementi su 10 disponibili: 2.200000
5.400000 7.100000
```

Il file main.c è lo stesso in entrambi le esecuzioni (non è necessario fare modifiche)