*Università di Ferrara*
*Dipartimento di Ingegneria*

## Esercitazioni di

# FONDAMENTI DI INFORMATICA MODULO B

## ESERCITAZIONE 2

Tutor

Arnaud Nguembang Fadja: ngmrnd@unife.it
Damiano Azzolini: damiano.azzolini@student.unife.it

# Esercizio 1. Inserimento ordinato

- Dato un vettore di dimensione DIM contenente N numeri interi (0 ≤ N ≤ DIM), ordinati in senso non decrescente, si inseriscano k interi, letti uno alla volta da tastiera, preservando l'ordinamento del vettore ad ogni passo di inserimento.

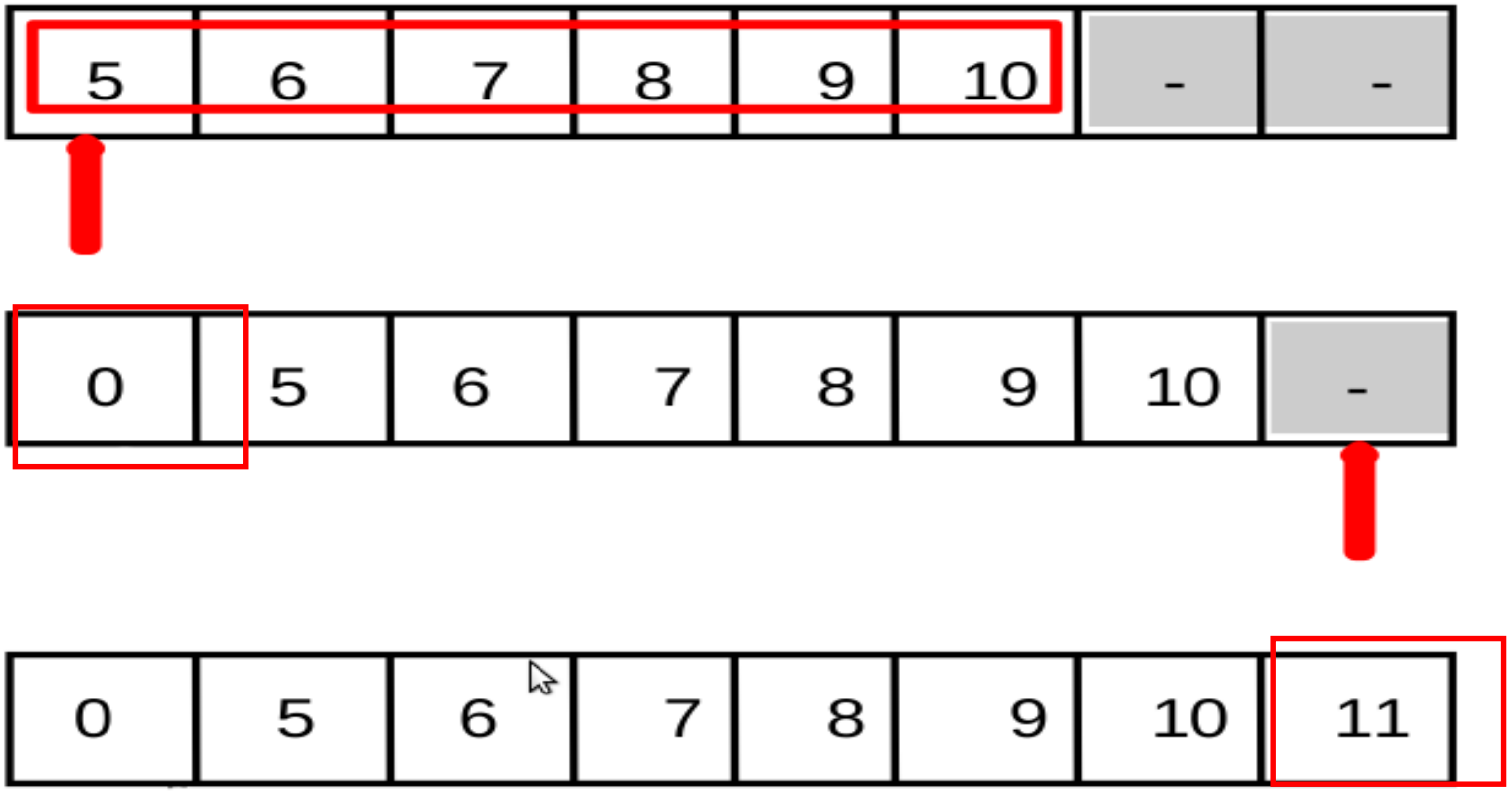
- Se l'array è pieno non vengono aggiunti nuovi elementi.

- Strutturare l'esercizio su più file come richiesto nelle slide successive.

# Esercizio 1. Inserimento ordinato

**Esempio N = 6, k = 2:**

- Vengono inseriti: $newItem_0 = 0$, $newItem_1 = 11$
- ($newItem_3 = 13$, cosa succede?)



# Esercizio 1. Inserimento ordinato

Interfaccia (**insert.h**):

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

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

Il valore di ritorno di ***insert*** indica se l'inserimento è stato effettuato. 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));
```

Consultare l'help di Visual Studio per le informazioni sugli argomenti richiesti dalla funzione.

# Esercizio 3. Ordinamento di un array: qsort

Si modifichi il programma appena creato in modo tale da operare su un vettore di elementi di tipo generico. A tal fine si crei:

- Un file di libreria (*element_float.c* e *element_float.h*) in cui si definisce il tipo di dato ELEMENT:

**typedef float ELEMENT**

e le corrispondenti funzioni:

1. **int compare(const void *el1, const void *el2)** da usare con la funzione qsort,
2. **ELEMENT insert()** prende in input un dato di tipo ELEMENT e lo restituisce,
3. **void print(ELEMENT el)** stampa a video **el**

per gestire gli elementi di tipo float.

# Esercizio 3. Ordinamento di un array : qsort

- Un secondo file di libreria (*element_int.c* e rispettivo *element_int.h*) in cui si definisce il tipo di dato ELEMENT nel seguente modo

**typedef int ELEMENT**

e le corrispondenti funzioni:

1. **int compare(const void *el1, const void *el2)**
2. **ELEMENT insert()**
3. **void print(ELEMENT el)**

per gestire gli elementi di tipo int.

# Esercizio 3. Ordinamento di un array : qsort

- Un terzo file contenente la funzione main che 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 (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.**