Laboratorio 2

ASD 2022/2023

Nella prima parte del laboratorio vi viene richiesto di implementare in maniera ricorsiva l'algoritmo QuickSort con partizionamento in place e considerando due diverse strategie per la scelta del pivot:

- **Scelta banale**: il pivot è sempre il primo elemento della porzione di array che stiamo considerando
- **Scelta casuale**: la posizione del pivot viene scelta casualmente tra quelle disponibili nella porzione di array che stiamo considerando.

Nella seconda parte del laboratorio, vi viene richiesto di condurre un'analisi sui tempi di esecuzione di QuickSort e degli altri algoritmi di ordinamento visti a lezione (e la cui implementazione vi viene fornita).

Se avete dubbi sul funzionamento di QuickSort e/o su come organizzare la vostra soluzione vi invitiamo a riguardare con attenzione le slide di teoria.

Come primo passo per svolgere il laboratorio dovete scaricare il file .zip che trovate su AulaWeb nella sezione relativa al Laboratorio 2. All'interno di questo file .zip (da 'unzippare' (usando il comando unzip se state usando un sistema operativo linux) troverete diversi file:

- **Labo2_main.cpp**. Contiene il main del vostro programma. Al momento dell'esecuzione, il programma chiede di inserire il nome di un file di input (quello da cui caricare la sequenza di valori con cui riempire l'array, ne trovate diversi disponibili nel materiale) e di un file di output (su cui verranno salvati i risultati dell'esecuzione). Fatto questo, il programma prevede l'applicazione dei diversi algoritmi di ordinamento all'array, stampando a video il risultato dell'ordinamento ed il tempo necessario per l'esecuzione espresso in millisecondi (ms). È possibile passare da un ordinamento al successivo premendo un tasto qualsiasi.
  NOTA: alla prima esecuzione, QuickSort non avrà alcun effetto sull'array in quanto l'implementazione è mancante.
  COME AL SOLITO NON VI SERVE MODIFICARE QUESTO FILE.
- **labo2_aux.cpp e .h.** Contengono delle funzioni di utilità per leggere da file il contenuto di un array e stamparlo a video.
  NON VI SERVE MODIFICARE QUESTI FILE.
- **labo2_aux.cpp e .h.** Contengono le implementazioni delle funzioni di ordinamento, comprese quelle che dovete implementare, i cui prototipi sono i seguenti:

```
void quickSortTrivial(vector<int>& v)
{
     /* Implementare quickSort banale con partizione in place */
}
```

```
void quickSortRandom(vector<int>& v)
{
 /* Implementare quickSort randomizzato con partizione in place */
}
```

Nota importante: l'implementazione della procedura per ottenere la partizione in place deve rispecchiare quella discussa a lezione con la Prof. Mascardi!

- **Diversi file .txt**, che contengono sequenze di numeri interi di diversa lunghezza e possono essere utilizzati come file di input. Alcuni di essi contengono sequenze di valori già ordinate che vi facilitano nella simulazione del caso peggiore.
- **tempi-di-esecuzione-algo-ordinamento.xls**. È un file excel che vi consente di tenere traccia dei tempi di esecuzione dei diversi algoritmi. In particolare, vi viene richieste di riportare sul file, nelle colonne appropriate, i tempi di tre diverse esecuzioni. La media dei tempi delle tre esecuzioni, che dovete considerare nella successiva analisi, viene calcolata in automatico.

E' importante, durante la fase di implementazione, seguire tutti i consigli che vi sono stati spiegati e che sono contenuti nelle slide "Sarebbe bello che ..." della lezione del 8-3-2023. In particolare, è importante compilare spesso il codice (meglio compilare una volta in più che una in meno ...) e iniziare la codifica delle funzioni che possono essere testate, seguendo un ordine che vi permetta sempre di verificare la correttezza di quello che state implementando.
Durante la preparazione della traccia i docenti hanno cercato di seguire i consigli di "buona programmazione": come vedete, il codice che vi è stato fornito è stato annotato con commenti utili (in particolare quelli che spiegano la semantica delle operazioni da implementare) ed è stato indentato in modo che sia facilmente comprensibile.
In linux è possibile anche utilizzare il comando indent che effettua l'indentazione di un programma C/C++ in automatico. Il suo utilizzo è molto semplice: il comando

```
indent -linux sorgente.cpp
```

indenta il codice contenuto nel file sorgente.cpp (attenzione però che per poter applicare questo comando è necessario che il file sia corretto sintatticamente).
E' richiesto inoltre di testare in maniera approfondita il codice. Per questo vi potranno essere utili i file di input forniti.