Fondamenti di Informatica
Esempio programmazione 4

Istruzioni per lo svolgimento e la consegna

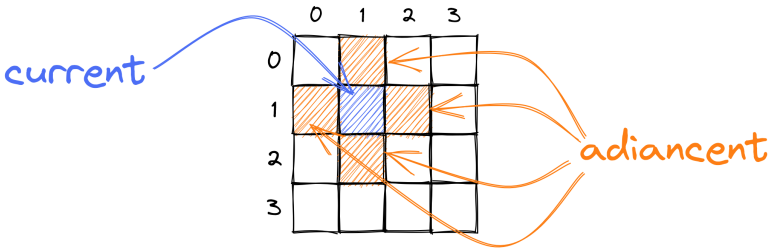
- La prima operazione da effettuare è modificare il file `studente.txt` (presente nella directory dove avete trovato questo `pdf`) inserendo il proprio nome e cognome e numero di matricola. Utilizzare un semplice editor di testo, salvando il file senza modificarne il nome.
- Nella stessa directory sono presenti i file necessari allo svolgimento dell'esercizio. **Per ogni esercizio dovrà essere modificato solamente il file `.c` corrispondente. Non devono essere modificati né spostati o eliminati i rimanenti file, pena la valutazione negativa dell'assegnamento**. Nel file `.c` dovranno essere mantenuti tutti e soli gli output a schermo forniti, in modo da mantenere la corrispondenza con l'output di esempio. Inoltre, è possibile definire funzioni ausiliarie all'interno del sorgente `.c`.
- Per compilare e generare l'eseguibile, da terminale entrate nella directory dove avete trovato questo `pdf` e lanciate il comando `make nome_esercizio`. Il `nome_esercizio` corrisponde al nome del sorgente, privato dell'estensione `.c`. Verrà generato l'eseguibile `nome_esercizio` che, lanciato da terminale (`./nome_esercizio`), vi permetterà di provare il vostro programma.
- Lanciando invece il comando (`./self_evaluation nome_esercizio`) eseguirete in maniera automatica alcuni test per verificare le soluzioni che avete implementato. I test sono studiati per verificare anche i casi particolari, in modo da gestire quelli che possono essere errori comuni in fase di implementazione. **Tenete presente che il correttore funziona solo all'interno di una distribuzione Linux a 64 bit (ad esempio, le macchine messe a disposizione nel laboratorio)**.
- La procedura di consegna dovrà iniziare lanciando il programma `./prepara_consegna.sh` presente nella directory dove avete trovato il presente `pdf`. Una volta lanciato, esso genererà un archivio di nome `consegna.tar.gz`: tale file sarà **l'unico che dovrà essere inviato** attraverso il sito `https://stem.elearning.unipd.it` per consegnare il vostro elaborato, seguendo anche le istruzioni che saranno fornite in aula dai docenti.

Considerate 2 aspetti:

- Se ci sono errori in compilazione/esecuzione, c'è qualcosa che rende errata/incompleta la vostra implementazione;
- Se non ci sono errori in compilazione/esecuzione, verificate che i risultati siano corretti (in alcuni casi è molto semplice fare il calcolo anche a mente).

1 Minimi locali (`local_minima.c`)

Data una matrice quadrata, definiamo un elemento della matrice come un minimo locale se i 4 elementi a lui adiacenti (a sinistra o a destra nella stessa riga, e sopra e sotto nella stessa colonna: vedi figura) o non sono presenti, o sono maggiori o uguali all'elemento considerato.



Sviluppare e testare la seguente funzione C:

```c
// Calcolare la mappa dei minimi loali della matrice in input
void local_minima(int *matrix, int* local_map, int size);
```

in modo che crei una mappa dei minimi locali della matrice input.
Tale mappa è anche essa una matrice, delle stesse dimensioni della matrice in input, e presenta un 1 se l'elemento corrispondente è un minimo locale, uno 0 altrimenti.
Entrambe le matrici sono preallocate.

Il programma di test è già implementato e compilato come file oggetto `local_minima_main.obj`. Per riuscire ad utilizzare usare il comando `gcc`, è necessario linkare il file come segue:

```
gcc -o local_minima local_minima.c local_minima_main.obj
```

Esempio 1

Initial matrix:

```
|    0    1 |
|    1    0 |
```

Local map:

```
|    1    0 |
|    0    1 |
```

Esempio 2

Initial matrix:

```
|    1    2    2 |
|    3    2    2 |
|    0   -1    5 |
```

Local map:

```
|    1    0    1 |
|    0    0    1 |
|    0    1    0 |
```

2 Lettera più frequente (`most_frequent.c`)

Sviluppare e testare la seguente funzione C

```c
// Trova la lettera maggiormente presente nella stringa
// in input e ne conta le occorrenze
char most_frequent(char *str, int *count);
```

che trova la lettera più presente in una stringa e ne conta le occorrenze sapendo che:

- `str` è un array di caratteri che contiene la stringa su cui effetturare il conteggio, tale stringa è composta esclusivamente da caratteri maiuscoli e spazi;
- `count` è un puntatore ad un intero preallocato che dovrà contenere il numero di occorrenze del carattere più frequente;
- la funzione dovrà ritornare la lettera che appare di più all'interno della stringa, gli spazi sono esclusi dal conteggio;
- se c'è più di una lettera con la stessa frequenza, ritornare la prima in ordine alfabetico;
- Se la stringa ha lunghezza 0 restituire `'A'` con `count` pari a 0.

Il programma di test è già implementato e compilato come file oggetto `most_frequent_main.obj`. Per riuscire ad utilizzare usare il comando `gcc`, è necessario linkare il file come segue:

```
gcc -o most_frequent most_frequent.c most_frequent_main.obj
```

Esempio 1

Stringa in input:

```
AAA
```

Output:

```
The most frequent letter is:
A
counting up to:
3
```

Esempio 2

Stringa in input:

```
THE MOST FREQUENT LETTER IS NO ONE
```

Output:

```
The most frequent letter is:
E
counting up to:
6
```