Programmazione Funzionale e Parallela (A.A. 2015-2016)
Corso di Laurea in Ingegneria Informatica e Automatica
Sapienza Università di Roma

A

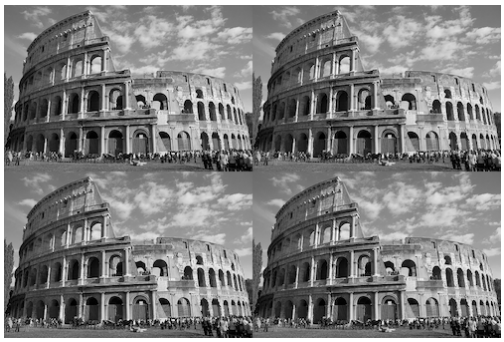
Esame del 10/02/2015 – Durata 1h 30' (solo esonerati)

Inserire nome, cognome e matricola nel file `studente.txt`.

Esercizio 1 (Filtri grafici mediante OpenCL)

Lo scopo dell'esercizio è quello di scrivere un modulo C basato su OpenCL che, data in input un'immagine a 256 toni di grigio di dimensione $w \times h$, crei una nuova immagine di dimensioni $2w \times 2h$, come nell'esempio sotto.

(a) Immagine originale



(b) Immagine dopo creazione poster

Si completi nel file `poster/poster.c` la funzione `poster` con il seguente prototipo:

```
void poster(unsigned char* in, int w, int h,
            unsigned char** out, int* ow, int* oh,
            clut_device* dev, double* td)
```

dove:

- `in`: puntatore a un buffer di dimensione `w*h*sizeof(unsigned char)` byte che contiene l'immagine di input in formato row-major[1];
- `w`: larghezza di `in` in pixel (numero di colonne della matrice di pixel);
- `h`: altezza di `in` in pixel (numero di righe della matrice di pixel);
- `out`: puntatore a puntatore a buffer di dimensione `2w*2h*sizeof(unsigned char)` byte che deve contenere l'immagine di output in formato row-major; **il buffer deve essere allocato nella funzione poster**;
- `ow`: puntatore a `int` in cui scrivere la larghezza di `out` in pixel;
- `oh`: puntatore a `int` in cui scrivere l'altezza di `out` in pixel.

Per compilare usare il comando `make`. Per effettuare un test usare `make test`. Verrà prodotta l'immagine di output `colosseo-poster.pgm`.

---

[1] Cioè con le righe disposte consecutivamente in memoria.

Esercizio 2 (Ricerca delle occorrenze di una stringa in un testo mediante pthread)

Lo scopo dell'esercizio è quello di scrivere una funzione C che cerca tutte le occorrenze di una stringa in un testo in parallelo usando pthread. Si vada nella directory di lavoro `findstr` e si definisca nel file `findstr.c` la funzione `findstr` con il seguente prototipo:

```
void findstr(const char* pattern, const char* text, char* out, int n)
```

dove:

- `pattern`: stringa da cercare;
- `text`: array di `n` caratteri (non terminato con '\0') che rappresenta il testo in cui cercare la stringa;
- `out`: vettore di output di dimensione `n` (si veda sotto come calcolarlo);
- `n`: numero di caratteri del testo.

La funzione deve inizializzare `out` in modo che, per ogni `i` in [0, `n`-1], `out[i]` vale 1 se la stringa `pattern` compare in `text` a partire dall'indice `i`, e 0 altrimenti. La soluzione deve usare almeno 2 thread.

Per compilare usare il comando `make`. Per effettuare dei test usare `make test1` e `make test2`.