TDP: Esercitazione 8

1. Scaricare i file per l'esercitazione

Opzione 1

- Scaricare il file .tgz contenente i file dell'esercitazione (file_per_esercitazione_8.tgz)

- decomprimerlo in una directory tramite il comando

```
$ tar xzvf <nomefile.tgz>
```

Opzione 2

Scaricare lo script download_E8.sh, eseguirlo in un terminale e lavorare nella directory indicata nella finestra del terminale.

2. Modificare il file matrix_ops.c implementando le funzioni indicate in seguito

Lavorare solo nella directory in cui sono stati estratti i file dell'esercitazione.

3. Compilare ed eseguire il programma

comando per la compilazione

```
$ make
```

esecuzione del programma principale

```
$ ./matrix_main
```

Funzioni da implementare

Esercizio 1
/*alloca una matrice di num_rows * num_cols elementi*/
float Matrix_allocArrayOfArray(int num_rows, int num_cols);**

/*dealloca una matrice con num_rows righe
  m: puntatore all'array di righe
  num_rows: numero di righe (lunghezza di m)
*/
void Matrix_freeArrayOfArray(float m, int num_rows);**

/*NOTA

  dopo aver svolto queste due funzioni, modificare Matrix_alloc e Matrix_free in modo da invocare

```
  Matrix_allocArrayOfArray e Matrix_freeArrayOfArray.
*/
```

Esercizio 2

```
/*copia una matrice in memoria
  src: matrice in input
  num_rows: righe
  num_cols: colonne
  restituisce un'array di puntatori alle righe;
*/
float** Matrix_clone(float** src, int num_rows, int num_cols);
```

Esercizio 3

```
/*restuisce una nuova matrice che e' la trasposta della matrice di input*/
float** Matrix_copyTransposed(float** src, int num_rows, int num_cols);
```

Esercizio 4

```
/* scrive in dest (che deve essere preallocato)
   il sottoblocco di src
   che si trova in posizione
     [block_start_row....block_start_row+block_num_rows],
     [block_start_col....block_start_col+block_num_cols]

     num_rows e num_cols sono le dimensioni di src;
     se il blocco copiato eccede le dimensioni di src
     la funzione ritorna un valore negativo (-1).
     in caso contrario ritorna il numero di elementi copiati
 */
int  Matrix_extractBlock(float** dest, float** src,
            int num_rows, int num_cols,
            int block_start_row, int block_start_col,
            int block_num_rows, int block_num_cols);
```

Esercizio 5

```
/*prodotto matrice vettore
  dest: area di memoria preallocata in cui scrivere il risultato
        la dimensione di dest e' pari al numero di righe di m
  m: matrice
  num_rows: numero di righe di m
  num_cols: numero di colonne di m
  v: array di valori da moltiplicare per m;
```

SUGGERIMENTO: usare la Vec_dot

*/

void Matrix_vectorProduct(float* dest, float m, int num_rows, int num_cols, float* v);**

Esercizio 6

/*

scambia la riga r1 ed r2 nella matrice m

*/

void Matrix_exchangeRows(floatm , int r1, int r2);**

Esercizio 7

/*

somma alla riga r1+=r2*scale

*/

void Matrix_sumAndScaleRows(float m, int r1, int r2, int num_cols, float scale);**

Esercizio 8

/*

trova l'indice di riga che contiene il massimo valore (in modulo),

considerando solo la colonna col

*/

int Matrix_findMaxIdxInCol(floatm, int num_rows, int col);**

Ultime modifiche: giovedì, 14 aprile 2016, 09:08