Faysal Khatri

CSC220 -- Project 2

2017-07-02

Design Notes

1. matrix_t is a struct which contains a pointer to the first top-left element of each matrix. The matrix is traversed by other functions starting from that element. The struct also contains the number of rows and columns.
2. I had to use malloc to allocate memory for the struct and for the 2-d array inside the struct.
3. I am getting incompatible pointer type warnings which I have not figured out how to solve. I hope you'll have some mercy.

Sample Output

```
faysal@DESKTOP-AOGE5FF:/mnt/c/Users/faysa/Dropbox/School/CSC220/Project2$ ./project2
Printing matrix at address 0x16ca010:
        5       6
        7       8
Printing matrix at address 0x16ca050:
        5       6
        7       8
        0       0
        0       0
Trying to add matrices...

Matrix dimensions do not match, cannot add. Returning pointer to empty matrix.
Printing matrix at address 0x16ca0a0:
        0       0
        0       0
Trying to transpose...

Printing matrix at address 0x16ca0e0:
        5       7       0       0
        6       8       0       0
```

matrix_t struct

```
typedef struct matrix_t {
      int rows;
      int cols;
      int ** topLeft;
} matrix_t;
```

new_matrix()

```
matrix_t* new_matrix(int rows, int cols) {
 /*
  * new_matrix() returns a pointer to a matrix_t struct
  */


  matrix_t* new_matrix = (matrix_t *) malloc(sizeof(new_matrix));
  int matrix[rows][cols];

  new_matrix->topLeft = (int*) malloc(sizeof(matrix));
  new_matrix->rows = rows;
  new_matrix->cols = cols;

  zero_matrix(new_matrix);

  return new_matrix;

}

void zero_matrix(matrix_t* matrix) {
  int i=0;
  int j=0;
  int (*ptr)[matrix->cols];
  ptr = matrix->topLeft;

  for (i=0; i<matrix->rows; i++) {
        for (j=0; j<matrix->cols; j++) {
              ptr[i][j]=0;
        }
  }
}
```

add_matrices()

```
matrix_t * add_matrices(const matrix_t * matrix1, const matrix_t * matrix2) {
      int numRows = matrix1->rows;
      int numCols = matrix1->cols;

      matrix_t* result = new_matrix(numRows, numCols);

      int (*rptr)[numCols];
      int (*ptr1)[numCols];
      int (*ptr2)[numCols];
      int i;
      int j;


      if ( (matrix1->rows != matrix2->rows) || (matrix1->cols != matrix2->cols) ) {
            printf("Matrix dimensions do not match, cannot add. Returning pointer to
empty matrix.\n");
            return result;
      }

    rptr = result->topLeft;
      ptr1 = matrix1->topLeft;
      ptr2 = matrix2->topLeft;

      for (i=0; i<numRows; i++) {
        for (j=0; j<numCols; j++) {
              rptr[i][j] = ptr1[i][j] + ptr2[i][j];
        }
    }

      return result;
}
```