iii Design Doc – nozden01, bdioni01

Uarray2.h:

void UArray2_map_col_major(UArray2_T arr, void apply(int row, int col, UArray2_T arr, void *elem, void *closure), void *closure);
Description: This function parses through the given 2D-array in column major order and calls the given apply function which is then executed to every element in the array. The apply function takes in the row and column indices of an element of the 2D-array, a pointer to the element, a reference to the 2D-array, and a void pointer closure. The closure argument is a void pointer that will be passed into the apply function.
Expectations: arr is not NULL

void UArray2_map_row_major(UArray2_T arr, void apply(int row, int col, UArray2_T arr, void *elem, void *closure), void *closure);
Description: This function parses through the given 2D-array in row major order and calls the given apply function which is then executed to every element in the array. The apply function takes in the row and column indices of an element of the 2D-array, a pointer to the element, a reference to the 2D-array, and a void pointer closure. The closure argument is a void pointer that will be passed into the apply function.
Expectations: arr is not NULL

void *UArray2_at(UArray2_T arr, int row, int col);
Description: This function accesses an element in the given 2D array represented by the UArray2_T type. It returns a void pointer to the element at the given row and column indices within the array.
Expectations: The row and column indices are in the bounds of the 2D array. So, row must be in the range of [0, height of 2D array), and col must be in range of [0, width of 2D array). Additionally, arr must not be NULL.

UArray2_T UArray2_new(int row, int col, int size);
Description: Creates an uninitialized 2D array with the given row and column dimensions. Allocates the memory for the 2D array corresponding to the given size integer. Returns a UArray2_T pointer to the new 2D array.
Expectations:

- row ≥ 0 & row != NULL
- col ≥ 0 & col != NULL
- size ≥ 0 & size != NULL

int UArray2_width(UArray2_T arr);
Description: Finds and returns the width (an integer) of the given 2D array.
Expectations: arr != NULL

int UArray2_height(UArray2_T arr);
Description: Finds and returns the height (an integer) of the given 2D array.
Expectations: arr != NULL

size_t UArray2_size(UArray2_T arr);
Description: This function finds and returns the size (size_t type) of the pointer to the given 2D array.
Expectations: arr != NULL

void UArray2_free(UArray2_T *arr);
Description: Frees the memory associated with the given 2D array via a pass to the address of a pointer to the 2D array.
Expectations: *arr != NULL

Bit2.h:

int Bit2_width(Bit2_T bitmap);

int Bit2_height(Bit2_T bitmap);

Bit2_T Bit2_new(int row, int col);

int Bit2_put(Bit2_T arr, int row, int col, int marker);

int Bit2_get(Bit2_T arr, int row, int col);

void Bit2_map_col_major(Bit2_T bitmap, void apply(int row, int col, Bit2_T bitmap, int bit, void *closure), void *closure);

void Bit2_map_row_major(Bit2_T bitmap, void apply(int row, int col, Bit2_T bitmap, int bit, void *closure), void *closure);

void Bit2_free(Bit2_T *bitmap);

Bit2_T pbmread(FILE *inputfp);

void pbmwrite(FILE *outputfp, Bit2_T bitmap);