```c
/*****************************************************************
 *
 *                     uarray2.h
 *
 *     Assignment: iii
 *     Authors:  Jack Burton, Maurice Jang
 *     Date:     Feb 4, 2023
 *
 *    Summary
 *
 *    This file provides the interface for the implementation of
 *    of a 2 dimensional polymorphic unboxed array
 *
 *****************************************************************/
#include <stdio.h>
#include <stdlib.h>



typedef struct UArray2_T *UArray2_T;


/********** UArray2_new ********
 *
 * Allocates, initializes, and returns a polymorphic 2D array of
dimensions
 * width x height, with each element occupying size bytes.
 *
 *
 * Parameters:
 *      int width: Width (i.e. number of columns) of array
 *      int height: Height (i.e. number of rows) of array
 *      int size: Size of each element, in bytes
 *
 * Return: 2D array of dimensions width x height, elements of "size" bytes
 *
 * Expects
 *      width and height to be nonnegative
 *      size to be positive
 * Notes:
```

```
 *      Will CRE if client's arguments do not meet expectations.
 *
 ************************/
UArray2_T UArray2_new(int width, int height, int size);


/********** UArray2_height ********
 *
 * Returns the height of a UArray2 object.
 *
 *
 * Parameters:
 *      UArray2_T array: A Uarray2 object.
 *
 * Return: Height (number of rows) in a UArray2 object.
 *
 * Expects
 *      A valid UArray2 object.
 *
 ************************/
int UArray2_height(UArray2_T array);

/********** UArray2_width ********
 *
 * Returns the width of a UArray2 object.
 *
 *
 * Parameters:
 *      UArray2_T array: A Uarray2 object.
 *
 * Return: Height (number of columns) in a UArray2 object.
 *
 * Expects
 *      A valid UArray2 object.
 *
 ************************/
int UArray2_width(UArray2_T array);

/********** UArray2_size ********
 *
```

```
 * Returns the size (in bytes) of each element of a UArray2 object.
 *
 *
 * Parameters:
 *      UArray2_T array: A Uarray2 object.
 *
 * Return: Size, in bytes, of the elements for array.
 *
 * Expects
 *      A valid UArray2 object.
 *
 ************************/
int UArray2_size(UArray2_T array);

/********** UArray2_at ********
 *
 * Returns a pointer to element at index (row, col)
 *
 *
 * Parameters:
 *      UArray2_T array: A Uarray2 object.
 *      int row: Row index in array
 *      int col: Column index in array
 *
 * Return: Pointer to element at index (row, col)
 *
 * Expects
 *      row and col to be less than the width/height, respectively,
 *      of array.
 * Notes:
 *      Will CRE if client's arguments do not meet expectations.
 *
 ************************/
void *UArray2_at(UArray2_T array, int row, int col);

/********** UArray2_map_col_major ********
 *
 * Traverses UArray2 with rows varying faster than columns, applying
apply()
 * function to each element.
```

```
 *
 *
 * Parameters:
 *      UArray2_T array: A Uarray2 object.
 *      void apply(int i, int j, UArray2_T a, void *p1, void *p2): Pointer
 *      to a function to execute on each element of array.
 *      void *cl: Pointer to a closure function to execute on each
element.
 *
 * Return: None
 *
 * Expects
 *      Valid uarray2 object and apply function pointer matching specified
 *      parameters.
 *
 ************************/
void UArray2_map_col_major(UArray2_T array,
        void apply(int i, int j, UArray2_T a, void *p1, void *p2), void
*cl);

/********** UArray2_map_row_major ********
 *
 * Traverses UArray2 with columns varying faster than rows, applying
apply()
 * function to each element.
 *
 *
 * Parameters:
 *      UArray2_T array: A Uarray2 object.
 *      void apply(int i, int j, UArray2_T a, void *p1, void *p2): Pointer
 *      to a function to execute on each element of array.
 *      void *cl: Pointer to a closure function to execute on each
element.
 *
 * Return: None
 *
 * Expects
 *      Valid uarray2 object and apply function pointer matching specified
 *      parameters.
 *
```

```
 ************************/
void UArray2_map_row_major(UArray2_T array,
        void apply(int i, int j, UArray2_T a, void *p1, void *p2), void
*cl);

/********** UArray2_new ********
 *
 * UArray2 free deallocates and clears *UArray_T. It is a checked runtime
 * error for uarray or *uarray to be null.
 *
 *
 * Parameters:
 *      UArray2_T *array: pointer to a 2D array
 *
 * Return: void

 * Expects
 *      A 2D UArray that is using memory
 * Notes:
 *      will raise checked runtime error if uarray or *uarray are null
 *
 ************************/
void UArray2_free(UArray2_T *array);
```

```
/**************************************************************
 *
 *                     Bit2.h
 *
 *     Assignment: iii
 *     Authors:  Jack Burton, Maurice Jang
 *     Date:     Feb 4, 2023
 *
 *    Summary
 *
```

```
 *    This file provides the interface for the implementation of
 *    of a 2 dimensional unboxed array of bits
 *
 *****************************************************************/
#include <stdio.h>
#include <stdlib.h>



typedef struct Bit2_T *Bit2_T;



Bit2_T Bit2_new(int width, int height);
int Bit2_height(Bit2_T array);
int Bit2_width(Bit2_T array);

int Bit2_put(Bit2_T array, int width, int height, int Bit);
int Bit2_get(Bit2_T array, int width, int height);

void Bit2_map_col_major(Bit2_T array,
        void apply(int i, int j, Bit2_T a, int bit, void *cl), void *cl);
void Bit2_map_row_major(Bit2_T array,
        void apply(int i, int j, Bit2_T a, int bit, void *cl), void *cl);
void Bit2_free(Bit2_T *array);
```