Resizable Int Arrays – basics

This exercise will introduce an easy user-defined data structure, the resizable integer array (RIA), and have you develop functions to manipulate variables of the RIA type. Together, the type definition and the set of functions working on it, constitute an abstract data type which you can reuse in other programs. You can see it as an elementary precursor of object oriented programming data and method encapsulation in classes.

- **ria.h** will contain the headers (aka declarations) of our functions.
- **ria.c** will contain the definitions of all the functions we will implement.
- **tests.c** will contain the main function used to test our functions.

User-defined data structure

As you already know by now, arrays are not really flexible. To remedy this, we are going to define a resizable *int* array (RIA) data structure which will hold a pointer to a dynamically allocated *int* array and a field storing its size.

```
struct ria_s {
    int * ptr;
    int size;
};
typedef struct ria_s ria_t;
```

Work to do

The following functions need to be implemented and tested;

ria_t* ria_create (int size , int defaultval) ;

- Dynamically allocate a *ria_t* data structure
- Return NULL if *malloc* failed to allocate this memory
- Use *ria_init* to proceed with the initialization of the newly allocated data structure

void ria_init (ria_t* this , int size , int val) ;

- Dynamically allocate and array of *size* integers
- Returns immediately is *this* is NULL
- Set all integers in this array to *val*
- Set the *ptr* field of the data structure pointed by *this* to the dynamical integer array's address and its *size* field to the parameter *size*

void ria_destroy (ria_t** pthis) ;

- Deallocate the dynamical array of integer
- Deallocate the data structure itself
- Set the pointer which address was used as parameter to NULL
- Handle all error situation you can anticipate

void ria_setValue (ria_t* this , int index , int value) ;

- This function set to *value* the dynamical integer array element located at index value *index* in the RIA data structure pointed to by *this*.

```
int     ria_getValue    ( ria_t* this , int index ) ;
```

- This function returns the value of the dynamical integer array element located at index value *index* in the RIA data structure pointed to by *this*.

Testing

- Update *tests.c* to test your functions and make sure that all the error cases are handled properly.