Group 20

Edward Francis
Danielle Goodman
Justin Kruse
Charlotte Murphy
Garret Sweetwood
Woohyuk Yang

Worksheet 0: Building a Simple ADT Using an Array

In Preparation: Read about basic ADTs.

In this worksheet we will construct a simple BAG and STACK abstraction on top of an array. Assume we have the following interface file (arrayBagStack.h) :

```
# ifndef ArrayBagStack
# define ArrayBagStack

# define TYPE int
# define EQ(a, b) (a == b)

struct arrayBagStack {
TYPE data [100];
int count;
};

void initArray(struct arrayBagStack * b);
void addArray (struct arrayBagStack * b, TYPE v);
int containsArray (struct arrayBagStack * b, TYPE v);
void removeArray (struct arrayBagStack * b, TYPE v);
int sizeArray (struct arrayBagStack * b);

void pushArray (struct arrayBagStack * b, TYPE v);
TYPE topArray (struct arrayBagStack * b);
void popArray (struct arrayBagStack * b);
int isEmptyArray (struct arrayBagStack * b);
# endif
```

Your job, for this worksheet, is to provide implementations for all these operations.

```
void initArray (struct arrayBagStack * b){

	assert(b != 0);
	 b->count = 0;

}


/* Bag Interface Functions */

void addArray (struct arrayBagStack * b, TYPE v) {

	if(b->count < 100) {			//The array should be array full at 100 elements
		/*Insert v into the count position in data array. Increment count*/
		b->data[b->count] = v;
		b->count++;
	}
}


int containsArray (struct arrayBagStack * b, TYPE v) {

	/*Set flag for finding v in data array. 0 = false, 1 = true*/
	int vFound = 0;

	/*Loop through array and find a match to v*/
	int i;
	for(i = 0; i < b->count; i ++){
		if(b->data[i] == v) {
			vFound = 1;			//match to v found, set flag to true
			i = b->count;
		}
	}

	return vFound;					//return flag
}


void removeArray (struct arrayBagStack * b, TYPE v) {
```

```
	/*Set flag for finding v in data array. 0 = false, 1 = true*/
	int vFound = 0;
	int vElement;			//position in array where v is found

	int i;

	/*Loop through array and find a match to v*/
	for(i = 0; i < b->count; i ++) {
		if(b->data[i] == v) {
			vFound = 1;			//match to v found, set flag to true
			vElement = i;			//set position of v
			i = b->count;			//exit loop by setting i to count
		}
	}

	/*if v is in data array, shift all elements from v's positions by 1 */
	if(vFound) {
		for(i = vElement; i < b->count - 1; i++) {
			b->data[i] = b->data[i + 1];
		}
		b->count --;				//decrease the count in bag
		b->data[count] = 0;
	}
}


int sizeArray (struct arrayBagStack * b) {
	return b->count;			//return the count in bag
}


 /* Stack Interface  Functions */

void pushArray (struct arrayBagStack * b, TYPE v) {

	if(b->count < 100) {			//Do push if array is full
		/*Insert v into the count position in data array. Increment count*/
		b->data[b->count] = v;
		b->count++;
	}
}
```

```
TYPE topArray (struct arrayBagStack * b) {

        if(b->count != 0) {                         //make sure there is at least one element
                return b->data[b->count - 1];       //return the top element in data array
        }
        else {
                return 0;
        }
}




void popArray (struct arrayBagStack * b) {

        if(b->count != 0) {                         //must be at least 1 element to pop
                /*Decrease bag count by 1. Set popped element to 0*/
                b->count--;
                b->data[b->count] = 0;
        }
}


int isEmptyArray (struct arrayBagStack * b) {

        if(b->count == 0) {
                return 1;       //return true if stack is empty
        }
        else {
                return 0;       //return false if stack is not empty
        }
}
```

A Better Solution…

This solution has one problem. The arrayBagStack structure is in the .h file and therefore exposed to the users of the data structure. How can we get around this problem? Think about it…we'll return to this question soon.