Dynamic Array using C Programming Part 3

Learning step 4 : Insert elements

```c
/*
 * Taking the value to be inserted and position(index) as parameters
 * check if index is in valid range and if size equals capacity of array
 * if size has reached capacity make room just like in append and make sure pointer for data doesn't point to NULL
 * Run a full loop from last element of array till index and shift the elements to right by iterating left
 * update index value which is target and increment the size as new element has been added
 */
void insertAt(DynamicArray *arr, int val, int index){
        if (index < 0 || index > arr->size){
        printf("Operation Failed");
        return;}

        if (arr->size == arr->capacity){
        arr->capacity*=2;
        arr->data = realloc(arr->data, arr->capacity * sizeof(int));
                if (arr->data == NULL){
                        exit(1);
}
}


        for (int i = arr->size -1; i >= index; i --)
        {
                arr->data[i+1] = arr->data[i];
        }
        arr->data[index] = val;
        arr->size++;
}
```

Above is the insert functions that first checks if size of array has reached capacity and then make room for the new element. It later iterates on the array by shifting elements from index to right, finally inserting the target value to the given position. Increment size by 1.

```c
int main(){
        DynamicArray *arr = createArray(2);
        append(arr, 1);
        append(arr, 2);
        append(arr, 3);
        int x = get(arr, 2);
        printf("Got: %d \n", x);
        update(arr, 5, 2);
        removeA(arr, 0);
        insertAt(arr, 1, 1);
        insertAt(arr, 0, 0);
        printf("Final Array: ");
        for (int i = 0; i < arr->size; i++){
         printf("%d ", arr->data[i]);
        }
        printf("\n");
        freeArray(arr);
        return 0;
}
```

```
phenodiss1de@Phen0:~/ccode/DSA-DAY1/C-Project-based-Learning/DSA/Dynamic Array/Day3$ ./insert
Got: 3
Final Array: 0 2 1 5
```

Learning step 5: Linear Search

O(n) complexity depending on element position, it serahces till it finds it through the entire array. PLease note that we can customise it to count occurences and we can also use it for sorted as well unsorted array.

```c
/*
 *Linear search is just a simple search for any passed value it fetches the index of the element
*Typically search starts from the first element and goes on till we find the element
*/

int lSearch(DynamicArray *arr, int val)
{
        for (i=0; i < arr->size; i++)
        {
                if (arr->data[i] == val){
                return i;
                }
        }
        retuen -1;
}
```

```c
int main(){
        DynamicArray *arr = createArray(2);
        append(arr, 1);
        append(arr, 2);
        append(arr, 3);
        int x = get(arr, 2);
        printf("Got: %d \n", x);
        update(arr, 5, 2);
        removeA(arr, 0);
        insertAt(arr, 1, 1);
        insertAt(arr, 0, 0);
        printf("Final Array: ");
        for (int i = 0; i < arr->size; i++){
         printf("%d ", arr->data[i]);
        }
        printf("\n");
        int g = lSearch(arr, 2);
        printf("%d \n", g);
        freeArray(arr);
        return 0;
}
```

```
phenodiss1de@Phen0:~/ccode/DSA-DAY1/C-Project-based-Learning/DSA/Dynamic Array/Day3$ ./insert
Got: 3
Final Array: 0 2 1 5
1
phenodiss1de@Phen0:~/ccode/DSA-DAY1/C-Project-based-Learning/DSA/Dynamic Array/Day3$
```