Worksheet 14 - Group 1

Worksheet Group 1 Members

Marc Clinedinst: clinedim@onid.oregonstate.edu
Kelby Faessler: faesslek@onid.oregonstate.edu
James Fitzwater: fitzwatj@onid.oregonstate.edu
Tom Gariepy: gariepyt@onid.oregonstate.edu
Sean Reilly: reillys@onid.oregonstate.edu
Joseph Struth: struthj@onid.oregonstate.edu

Worksheet 14: Introduction to the Dynamic Array

In this worksheet, our aim is to familiarize ourselves with the functions that control the behavior of a Dynamic Array. A Dynamic Array automatically resizes each time its capacity is reached; it does so by doubling its current capacity. Many of the functions needed for the Dynamic Array were provided in the worksheet; we will define only those functions that were unimplemented in the worksheet.

```
/*
    This function performs the key action that is needed for a
    dynamic array to function properly--it doubles its current
    size.  In order to do this, the function creates a new local
    DynArray and initializes it with a capacity that is twice the
    size of the original dynamic array.  It then copies the values
    from the existing array into the new version of the array,
    keeping track of its size.  It frees the memory from the
    current
    dynamic arrays, and then uses the new dynamic array moving
    forward.
*/
void _setCapacityDynArr(struct DynArr *v, int newCap) {
    struct DynArr new;
    initDynArr(&new, newCap);
    int count;

    for (count = 0; count < v->size; count++) {
        new.data[count] = v->data[count];
        new.size++;
    }

    freeDynArr(v);
```

```
     *v = new;
}

/*
     This function essentially performs the index operation that
     would normally happen with an array.  It checks that the
     requested index is greater than or equal to zero and less
     than the current size of the dynamic array. So long as the
     index is valid, it returns the value stored at that index.
*/
TYPE getDynArr(struct DynArr *v, int pos) {
     assert((pos >= 0) && (pos < sizeDynArr(v)));
     return v->data[pos];
}

/*
     This function allows the user to update the value stored at
     the given index within a dynamic array.  The function first
     checks that the requested index is greater than or equal to
     zero and less than the current size of the dynamic array. So
     long as the index is valid, it updates the value stored at the
     index with the given value.
*/
void putDynArr(struct DynArr *v, int pos, TYPE val) {
     assert((pos >= 0) && (pos < sizeDynArr(v)));
     v->data[pos] = val;
}

/*
     This function swaps the values stored at two different indexes
     within a dynamic array. It first checks that the provided
     indexes are valid--that is, that they are greater than or equal
     to zero and less than the size of the dynamic array.  If this
     is
     true, then the values are swapped.
*/
void swapDynArr (struct DynArr *v, int i, int j) {
     assert((i >= 0) && (i < sizeDynArr(v)));
     assert((j >= 0) && (j < sizeDynArr(v)));
     int temp = v->data[i];
     v->data[i] = v->data[j];
     v->data[j] = temp;
}
```

```
/*
     This function removes a value from the dynamic array at a given
     index.  The function first checks to make sure that the index
     is greater than or equal to 0 and less than the size of the
     array.  If this is true, then the values starting at index + 1
     are shifted to left and the size of the dynamic array is
     decremented.
*/
void removeAtDynArr (struct DynArr *v, int index) {
     assert((index >= 0) && (index < sizeDynArr(v)));

     for (; index < (sizeDynArr(v) - 1); index++) {
          v->data[index] = v->data[index + 1];
     }

     v->size--;
}
```

This worksheet also asked us to consider ways to hide the details of the structure of the dynamic array from the user. The easiest way to do this would be to add a type definition to the .h file where the other prototyping features appear, and then add the actual structure definition to the .c implementation file. This would support encapsulation.

Group Minutes

Our Minutes are available on Piazza at the following URL:

https://piazza.com/class/ib2kus4hsie528?cid=72