Worksheet 23 - 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

Collaborators

Marc, Kelby, James, Tom, Sean, Joseph

Worksheet 23: Introduction to the Iterator

In this assignment, we implement four different functions which control the behavior of an iterator used to iterate through the values stored in a dynamic array. More specifically, we implement the `DynArrIteratorInit`, `DynArrIteratorHasNext`, `DynArrIteratorNext`, and `DynArrIteratorRemove` functions. These functions are described in detail below and are accompanied by comments where needed.

```
/*
     This function initializes an iterator for a dynamic array.  Such an iterator consists
     of a pointer to a dynamic array and the current index of the iterator.  The iterator's
     dynamic array pointer is set to point to the passed dynamic array pointer.  Likewise,
     the current index is set to zero.
*/
void DynArrIteratorInit(struct DynArr *v, struct DynArrIterator *itr) {
     itr->v = v;
     itr->currentIndex = 0;
```

```
}


/*
     This function returns an integer value representing whether the iterator has a next
     value.  If the function returns 1, then the iterator has a next value.  If the function
     returns 0, then the iterator does not have a next value.  The function arrives at the
     returned value by comparing its current index to the size of the dynamic array.
*/
int DynArrIteratorHasNext(struct DynArrIterator *itr) {
     return itr->currentIndex < sizeDynArr(itr->v);
}

/*
     This function first checks to make sure that the iterator has a next value.  If this check
     passes, then the function returns the value at the current index and increments the value
     of the current index.
*/
TYPE DynArrIteratorNext(struct DynArrIterator *itr) {
     assert(DynArrIteratorHasNext(itr));
     return itr->v->data[itr->currentIndex++];
}

/*
     This function first makes sure that the current index is within the bounds of the
     array; this check is accomplished through an assertion statement.  Assuming that this is
     true, the value at the current index is removed and current index is decremented by one.
*/
void DynArIteratorRemove(struct DynArrIterator *itr) {
     removeAtDynArr(itr->v, itr->currentIndex);
     if (itr->currentIndex > 0) {
```

```
            itr->currentIndex--;
        }
}
```

Piazza Discussion

https://piazza.com/class/ib2kus4hsie528?cid=173