Lab-2: 1-D Life

Introduction

This is a second on-going project we will also refine over the semester to explore the concept of a container class.

The application we will be doing is to implement a simulation of 1-Dimensional Life using the abstraction of a Linear Container.

1-Dimensional Life

The Game Of Life is a 2 dimensional cellular automata invented by J.H. Conway in 1970. We will use a 1 dimensional variation of this game as an application for our Linear Container.

In this variation, the "universe" consists of an infinite 1D container of "cells". Each cell may be in one of two states: Dead or Alive. All cells change state simultaneously from one "generation" to the next based on their current state, and the current state of their four surrounding neighbors (2 to their left and 2 to their right) according to the following rules:

- a Dead cell with either two or three living neighbors will become Alive in the next generation.
- a Live cell dies if if has zero, one, or three living neighbors.

For example:

$$\ldots DLDLD \ldots$$
$$\ldots DDLDD \ldots$$
$$\ldots DDDDD \ldots$$

or:

$$\ldots DDLLDD \ldots$$
$$\ldots DLDDLD \ldots$$
$$\ldots DDLLDD \ldots$$

Try this one:

$$\ldots DLDLLLDD \ldots$$

or this one:

$$\ldots DDLLLLLDD \ldots$$

Linear Container

A Linear Container is an ordered collection of items (of arbitrary type, but we will look at a Linear Container where all elements are of a single, predefined type). While the ordering of elements is arbitrary (determined and enforced by the class user), elements are identified by their Position in the Linear Container.

A Linear Container is an abstraction that can have multiple implementations.

Operations provided by a Linear Container

Simple Operations

- **Create a Linear Container**: Create an initialized empty container.
- **Clear a Linear Container**: Remove all entries of an existing Linear Container
- **Test if container empty**
- **Test if a container is full**
- **Find the size of a container**: Determine the number of elements in the container
- **Add an element**: Add a new element to the end of the container.
- **Traverse a container**: For each element of the container, perform an operation.

More Complicated Operations

- **Insert a new element in a container**: Add a new element to the container at a specified position
- **Delete an element form a container**: Delete and return the element from the container at a specified position.
- **Retrieve an element**: Return the element in the container at a specified position, without removing it.
- **Replace an element**: Replace the element at a specified position with a given element.

Subtask 1: Linear Container as an Array

DUE: at the beginning of Week 7 lab.

Your Job

An array implementation of linear container

- common.h
- common.c
- entry.h
- container.h
- container.cc
- testcontainer.cc
- makefile

Copy my files to your account (I would suggest that you create a new directory for these). You can use the "Save Link" feature of your browser to do this, but then you would need to transfer the files to wiliki. As an alternate, you can run a command like the following on wiliki:

```
wget http://www2.hawaii.edu/~ruizhang/ee205/Labs/Container/Array/common.h
```

to transfer the file to the current directory. Complete the implementation of the Array based Container class.

Some questions to think about:

- How efficient is this?
- Under what conditions?
- How can we improve it?

Your team can demo your code to me during next weeks lab or the beginning of lab the following week.

Subtask 2: Linear Container as a Linked List

DUE: at the beginning of Week 7 lab.

Your Job

An linked list implementation of linear container

- common.h
- common.c
- entry.h
- listnode.h
- container.h
- container.cc
- testcontainer.cc
- makefile

Copy my files to your account (I would suggest that you create a new directory for these). You can use the "Save Link" feature of your browser to do this, but then you would need to transfer the files to wiliki. As an alternate, you can run a command like the following on wiliki:

```
wget http://www2.hawaii.edu/~ruizhang/ee205/Labs/Container/List/common.h
```

to transfer the file to the current directory. Make the test driver and test it. You should be able to identify the results that are correct and those not yet right. Study the code provided and make sure you understand how it works (DRAW the picture). Complete the implementation of the List based Container class (implement the remove operation). Your team can demo your code to me at the end of the lab session today, or the beginning of lab next week.

Your team can demo your code to me during next weeks lab or the beginning of lab the following week.