A2: III

Design Document

CSC 411 - Noah Daniels

Marceline Kelly

October 2, 2023

Design Checklist

1. **What is the abstract thing you are trying to represent?**

   I am trying to represent a two-dimensional array or, in other words, a container of containers of values.

2. **What functions will you offer, and what are the contracts of that those functions must meet?**

   This 2D array will offer the following methods:

   - `from_single_value(value: T, width: usize, height: usize)` constructs an array of two given dimensions then sets each element to a predefined value.
   - `from_row_major(vec: Vec<T>, width: usize)` constructs an array from a one-dimensional, row-major vector.
   - `from_col_major(vec: Vec<T>, height: usize)` constructs an array from a one-dimensional, column-major vector.
   - `at(&self, row: usize, col: usize) -> T` accesses individual elements.
   - `iter_row_major(&self) -> Array2Iter<'_, T>` returns a row-major iterator of the array.
   - `iter_col_major(&self) -> Array2Iter<'_, T>` returns a column-major iterator of the array.

3. **What examples do you have of what the functions are supposed to do?**

   ```
   let vec = vec![1,2,3,4,5,6];
   let array2 = Array2::from_row_major(vec, 3);
   // array2 now contains {{1, 2, 3}, {4, 5, 6}}

   let val = array2.at(1, 1); // val == 5

   for value in array2.iter_col_major() {
       print!("{value} ");
   }
   // prints "1 4 2 5 3 6"
   ```

4. **What representation will you use, and what invariants will it satisfy?**

   `Array2` will be built upon a vector of vectors (i.e. `Vec<Vec<T>>`). It will satisfy the following invariants:

   - Any instance of `Array2` with type `T` will have a concrete width and height, each greater than zero. Each element will be a value of type `T`.
   - Row-major and column-major iterators may be requested from an `Array2` regardless of the type of `Vec` used to initialize the array (or the underlying implementation).
   - Requesting the value at coordinates `(x, y)` will produce the value at the `y`th element of the `x`th `Vec` within the root `Vec`. `x` and `y` are both zero-indexed.

5. **When a representation satisfies all invariants, what abstract thing from step 1 does it represent?**

   Nested vectors function as a "container of containers." Each of these sub-containers must contain some value, hence a "container of containers of values."

6. **What test cases have you devised?**

- A `Array2` built from the row-major `Vec [1, 2, 3, 4]` should return the same `Vec` when its row-major iterator is `collect`ed.
- A `Array2` built from the column-major `Vec [1, 2, 3, 4]` should return the same `Vec` when its column-major iterator is `collect`ed.
- a `Array2` built from the row-major `Vec [1, 2, 3, 4]` should be equivalent to one built from the column-major `Vec [1, 3, 2, 4]`

7. **What programming idioms will you need?**

- Creating polymorphic `struct`s using generic types
- Generating iterators from collection types