Dec 2023
Wesley Cooke

Fixing the Arc

When we align WIL with North, there is a sight error that causes arcs between GPS points as seen on page 114 and 115.

To eliminate these arcs, we need to eliminate this error. How do we do this?

~~We~~ We have a heading from our BNO Chip. This heading is good to read at any time and doesn't depend on the speed of WIL.

We also get a heading from the GPS. However, the heading from the GPS is not accurate unless WIL is moving fast enough.

What we might be able to do is compute the average error in the heading that the BNO reports and the heading the GPS reports over the course of some amount of time.

This average difference is hopefully our "error" term that is introduced from not Perfectly aligning WIL with North.

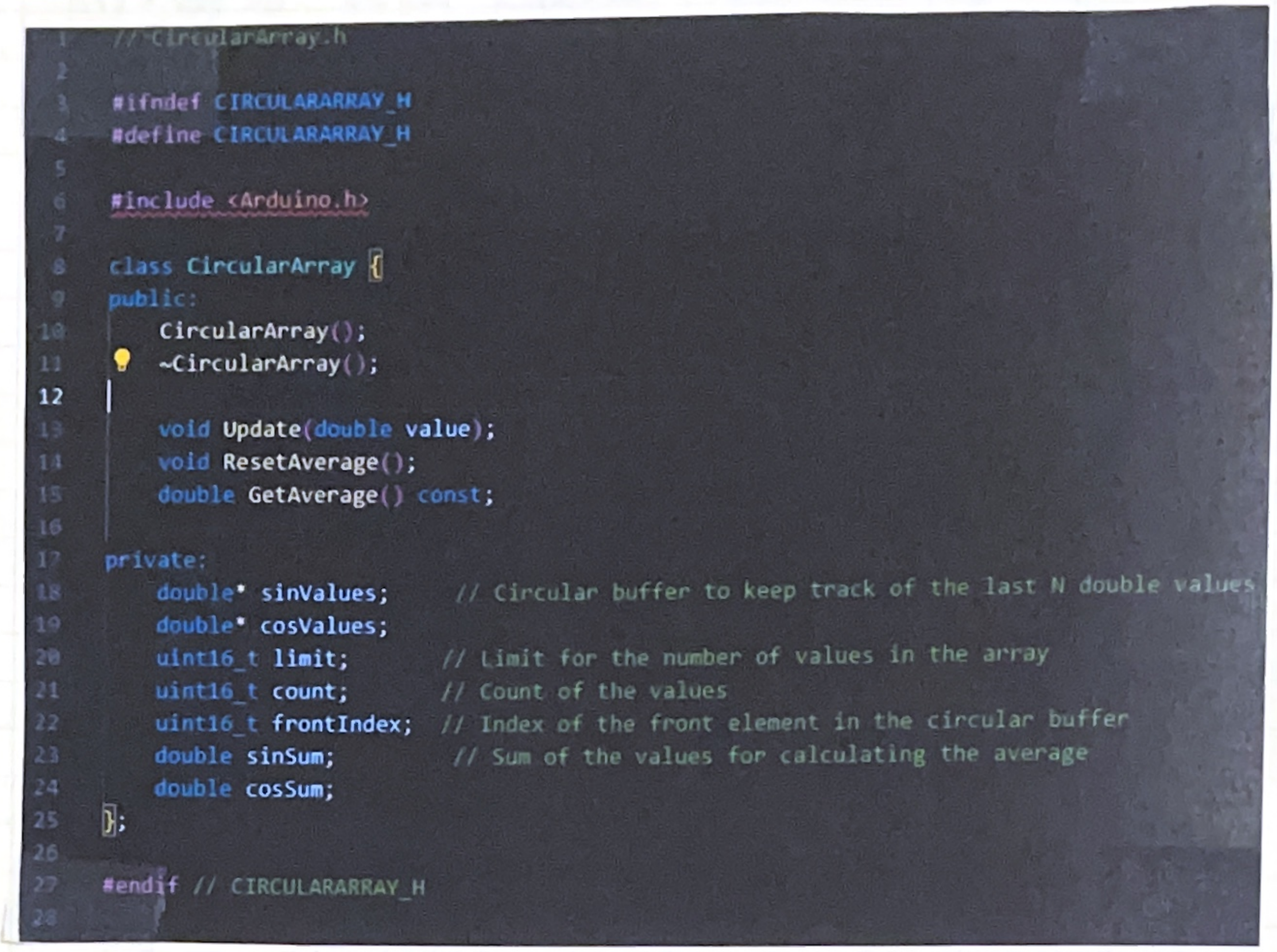
Note that this will ensure we are aligned with True North and not Magnetic north.

Let's write a library that can keep track of this error. Note that the average of 0° and 360° when dealing with headings should be 0°. The average of 0° and 180° should be 90.

* We are also going to use a Circular Array. This means, if we try to add more values than the array can hold, we will wrap around to the start of the array and begin over writing old values.

Fixing the Arc

Circular Array. H



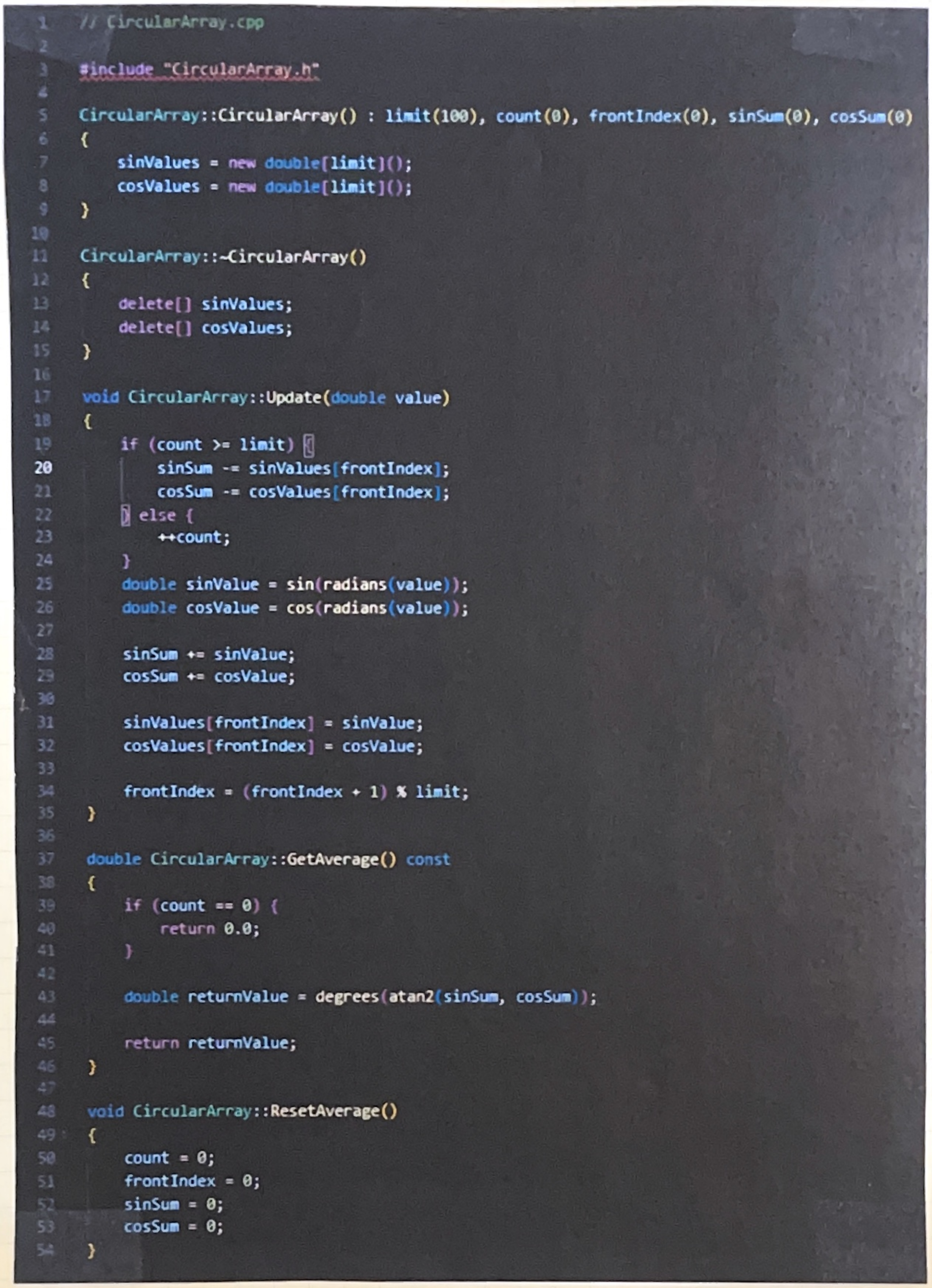
```cpp
// CircularArray.h

#ifndef CIRCULARARRAY_H
#define CIRCULARARRAY_H

#include <Arduino.h>

class CircularArray {
public:
    CircularArray();
    ~CircularArray();

    void Update(double value);
    void ResetAverage();
    double GetAverage() const;

private:
    double* sinValues;      // Circular buffer to keep track of the last N double values
    double* cosValues;
    uint16_t limit;        // Limit for the number of values in the array
    uint16_t count;        // Count of the values
    uint16_t frontIndex;   // Index of the front element in the circular buffer
    double sinSum;          // Sum of the values for calculating the average
    double cosSum;
};

#endif // CIRCULARARRAY_H
```

Here, the ~ Circular Array (); is a de constructor. It tells arduino what to do it this object is deleted.

Dec 2023
Wesley Cooke

Fixing the Arc

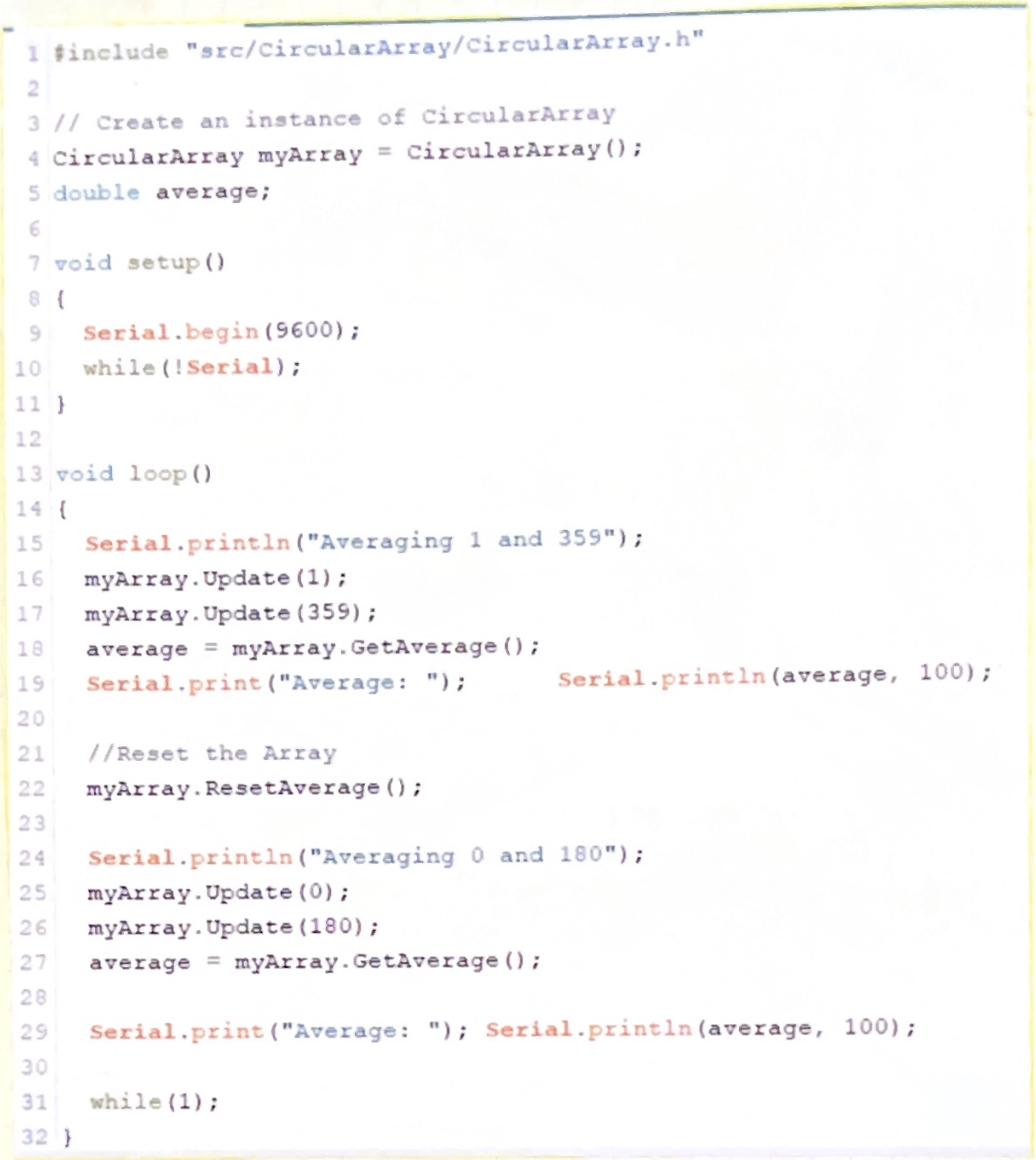
Circular Array.cpp



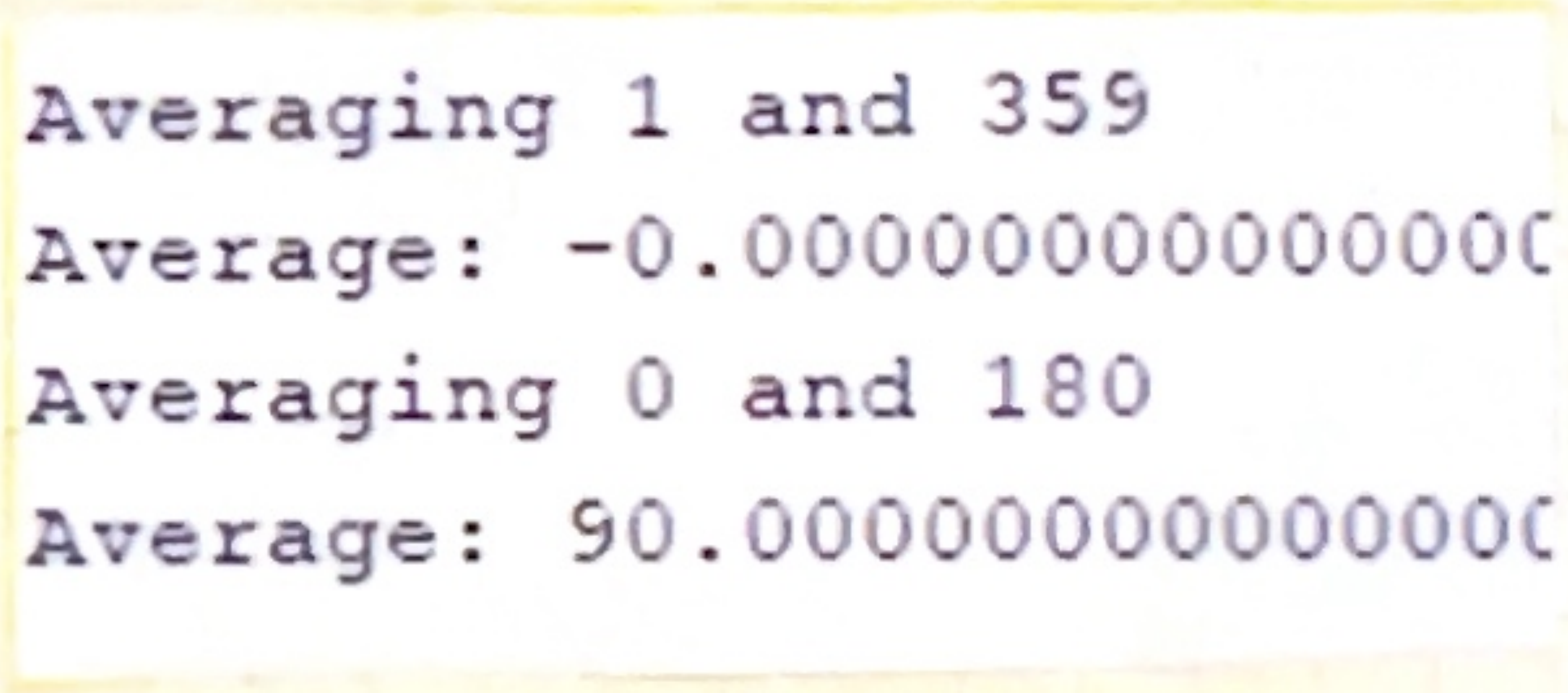
```cpp
// CircularArray.cpp

#include "CircularArray.h"

CircularArray::CircularArray() : limit(100), count(0), frontIndex(0), sinSum(0), cosSum(0)
{
    sinValues = new double[limit]();
    cosValues = new double[limit]();
}

CircularArray::~CircularArray()
{
    delete[] sinValues;
    delete[] cosValues;
}

void CircularArray::Update(double value)
{
    if (count >= limit) {
        sinSum -= sinValues[frontIndex];
        cosSum -= cosValues[frontIndex];
    } else {
        ++count;
    }
    double sinValue = sin(radians(value));
    double cosValue = cos(radians(value));

    sinSum += sinValue;
    cosSum += cosValue;

    sinValues[frontIndex] = sinValue;
    cosValues[frontIndex] = cosValue;

    frontIndex = (frontIndex + 1) % limit;
}

double CircularArray::GetAverage() const
{
    if (count == 0) {
        return 0.0;
    }

    double returnValue = degrees(atan2(sinSum, cosSum));

    return returnValue;
}

void CircularArray::ResetAverage()
{
    count = 0;
    frontIndex = 0;
    sinSum = 0;
    cosSum = 0;
}
```

Fixing the Arc

Example Driver Code:



```cpp
#include "src/CircularArray/CircularArray.h"

// Create an instance of CircularArray
CircularArray myArray = CircularArray();
double average;

void setup()
{
  Serial.begin(9600);
  while(!Serial);
}

void loop()
{
  Serial.println("Averaging 1 and 359");
  myArray.Update(1);
  myArray.Update(359);
  average = myArray.GetAverage();
  Serial.print("Average: ");      Serial.println(average, 100);

  //Reset the Array
  myArray.ResetAverage();

  Serial.println("Averaging 0 and 180");
  myArray.Update(0);
  myArray.Update(180);
  average = myArray.GetAverage();

  Serial.print("Average: "); Serial.println(average, 100);

  while(1);
}
```



```
Averaging 1 and 359
Average: -0.0000000000000
Averaging 0 and 180
Average: 90.0000000000000
```