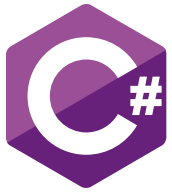
Arrays and collections

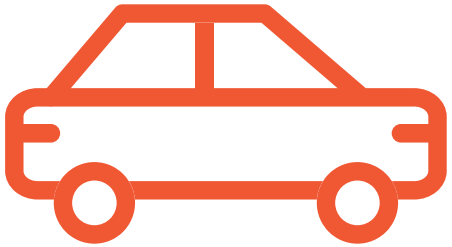
Agenda

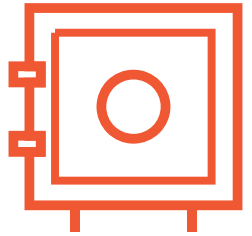
Working with collections

Arrays

Working with collections of objects







A collection is an object that manages groups of related objects

.NET Documentation

Multiple instances

Collection instance

2 types of collection

Arrays

Fixed size

Other Collections

Dynamic size

Elements

An element is one of the reference/value stored in the collection

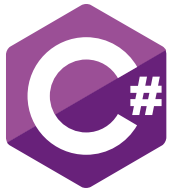
All the elements can be enumerated in an iteration statement

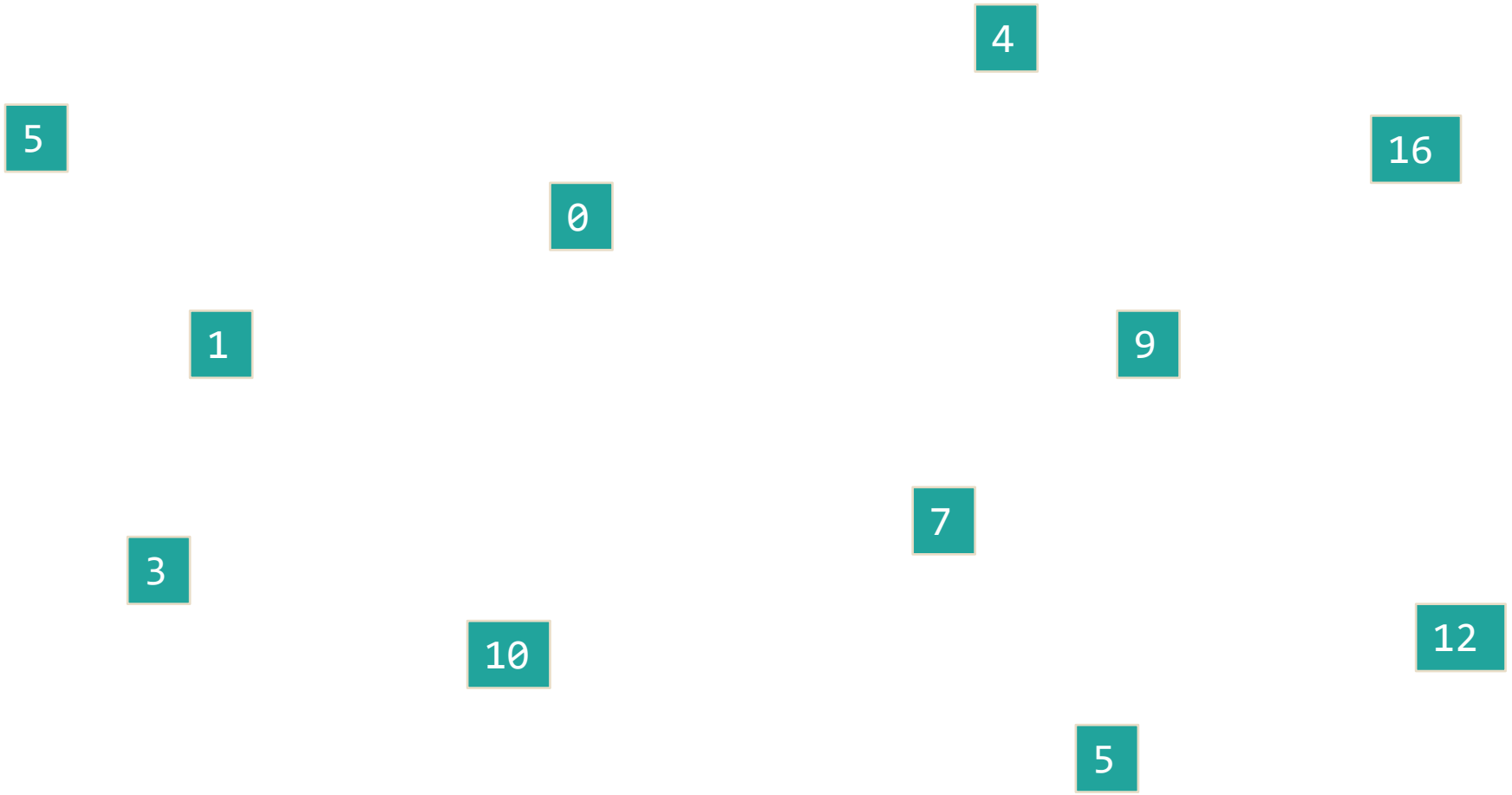
Each element can be retrieved

Indexing

Arrays and some collections are zero based indexing

Arrays




4
5
16
0
1
9
7
3
12
10
5


5
1
0
10
4
7
9
12
3
16

Type of an array

The type of the elements is specified at declaration of the array

An array can be of any type including object and array types

An array is a reference type

Fixed size

Unlike collections arrays are fixed size

Number of elements

The number of elements of an array is specified at the instantiation of the array

Array of string declaration and instantiation

```
string[] stringArray = new string[7];
```

Array of int declaration and instantiation

```
int[] intArray = new int[3];
```

Array of object declaration and instantiation

```
object[] objectArray = new object[3];
```

Number of dimensions

The number of dimensions of an array is specified at the declaration of the array

Multidimensional array declaration

```
int[,] array = new int[4,2];
```

Array initialization

If the array is not initialized, the elements are initialized to their default value

An array can be initialized at instantiation

Array initialization with collection initializer

```
string[] planets = new string[8] {
        "Mercury",
        "Venus",
        "Earth",
        "Mars",
        "Jupiter",
        "Saturn",
        "Uranus",
        "Neptune"
};
```

Array size defined at runtime

```
int randomNumberOfElements = new Random().Next(1,10);
int[] randomlySizedArray = new int[randomNumberOfElements];
```

Array initialization with collection initializer

```
string[] planets = {
        "Mercury",
        "Venus",
        "Earth",
        "Mars",
        "Jupiter",
        "Saturn",
        "Uranus",
        "Neptune"
};
```

Order

Array elements are stored in a definite order

Arrays are indexed

Array indexing is zero based (does not start at 1, starts at 0)

Zero based indexes

Retrieve an element

```
planets[0] → Mercury
```

Modify elements

```
string[] planets = new string[8];
planets[0] = "Mercury";
planets[1] = "Venus";
planets[2] = "Earth";
planets[3] = "Mars";
planets[4] = "Jupiter";
planets[5] = "Saturn";
planets[6] = "Uranus";
planets[7] = "Neptune";
```

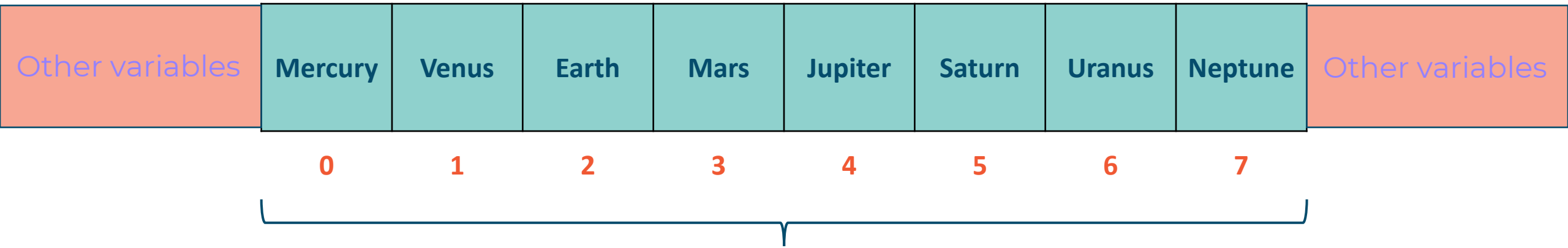
Add/Remove elements

You can’t add or remove elements

Performance

For array, retrieving an element is fast

Fast element access



Adjacent memory blocks

Array class

Arrays are derived from the abstract class `System.Array`

Useful methods

`Clone`
`Copy`
`Clear`
`Reverse`
`Find`
`Sort`

Access elements in a for loop

```
for (int i = 0; i < planets.Length; i++)
{
    Console.WriteLine(planets[i]);
}
// Mercury
// Venus
// Earth
// Mars
// Jupiter
// Saturn
// Uranus
// Neptune
```

Demo

Declare/Instantiate arrays
Modify/retrieve elements
for/foreach

Challenge

A ring buffer or circular buffer is a fixed-size buffer

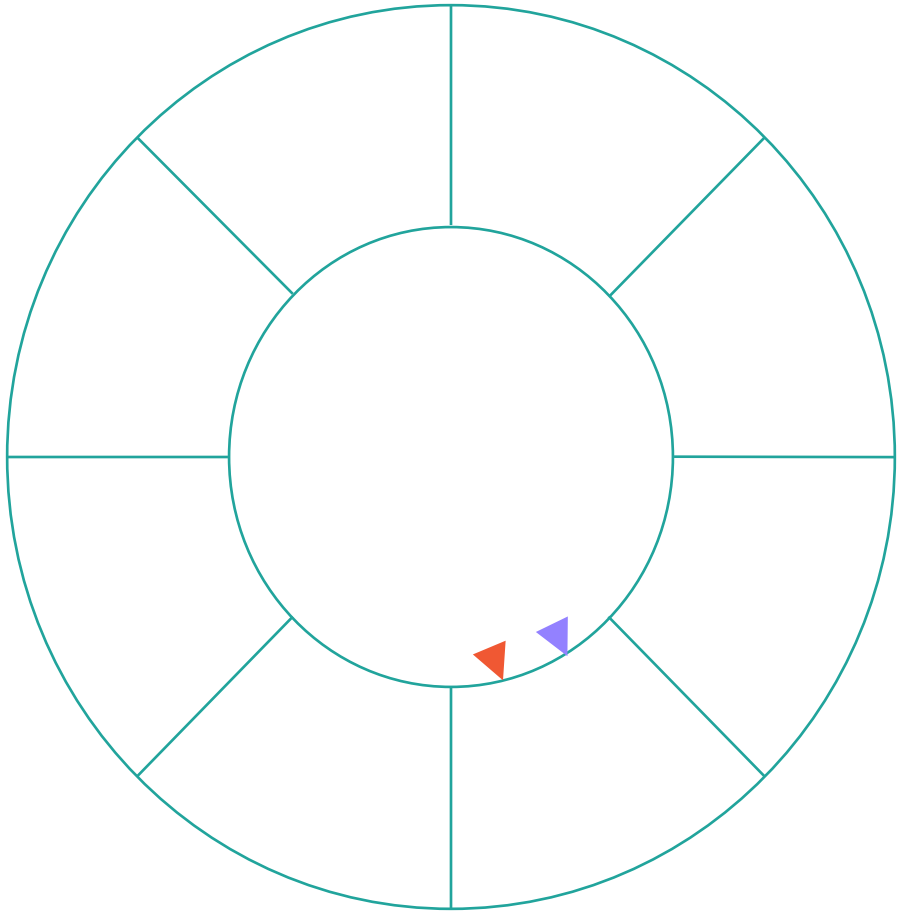
It behaves as if the start of the buffer was connected to the end of the buffer

When instantiated the buffer is empty. The number of elements is a parameter of the ring buffer constructor

The Write method writes an element to the buffer

The Read method reads and removes the oldest element in the buffer

Ring buffer (initial state)

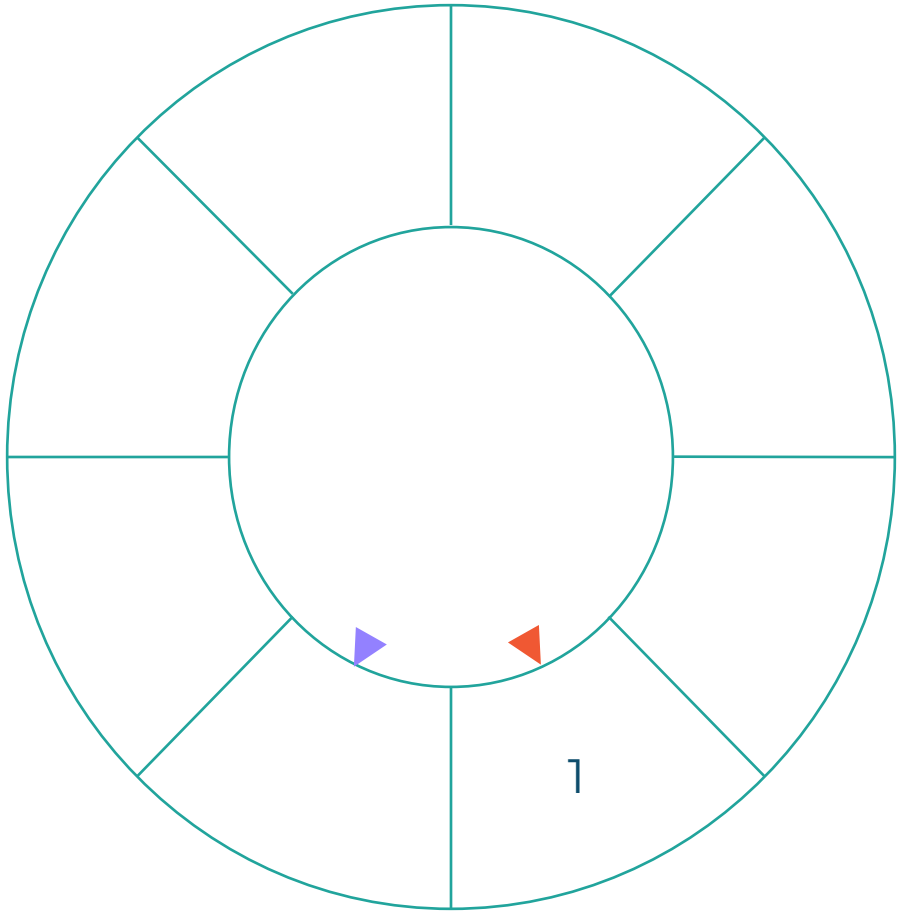


Write index

Read index

Ring buffer

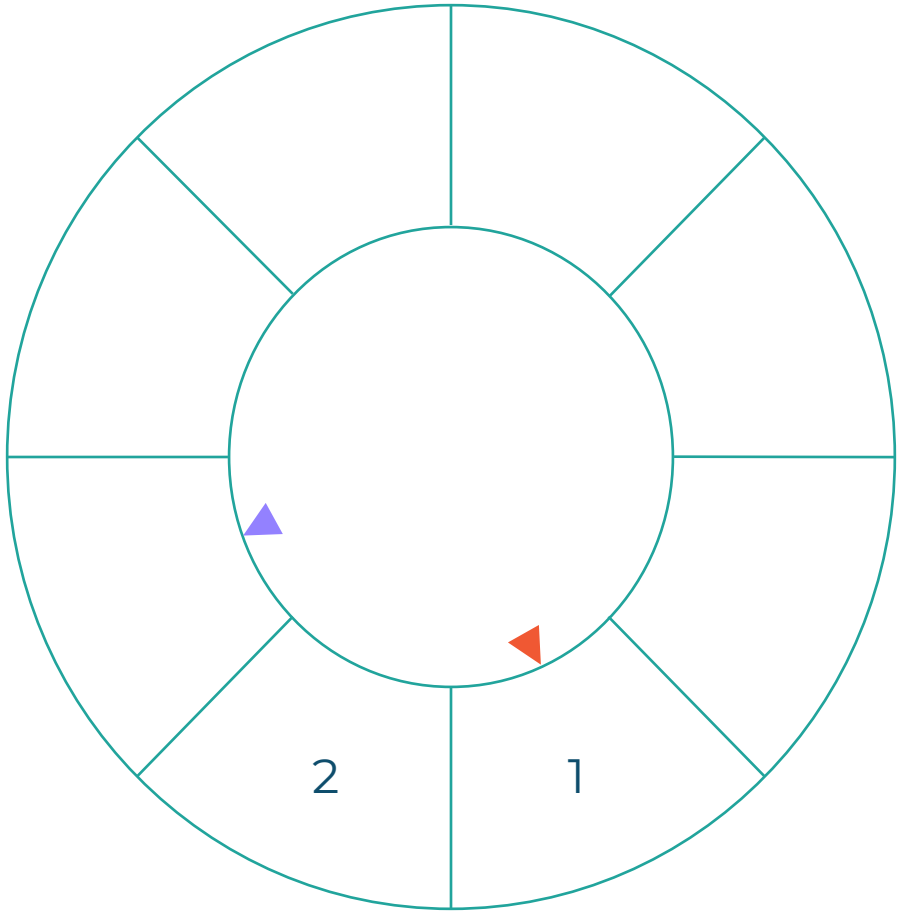
- Write index
- Read index




```
ringBuffer.Write(1);
```

Ring buffer

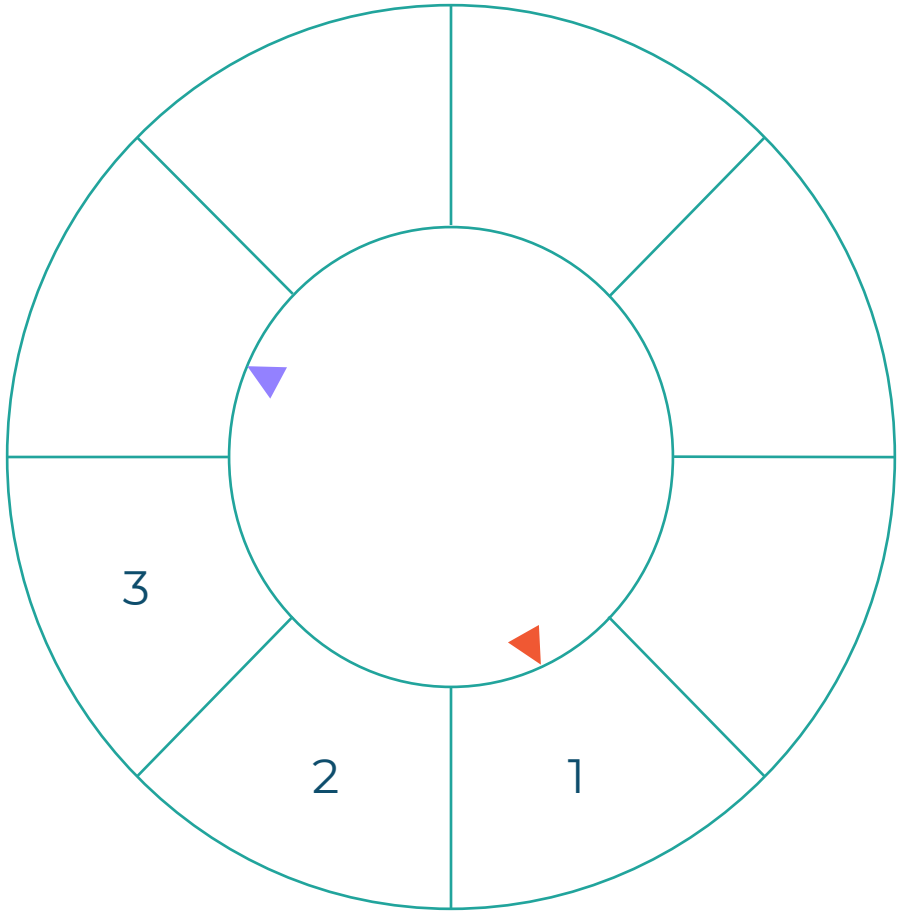
- Write index
- Read index




```
ringBuffer.Write(2);
```

Ring buffer

- Write index
- Read index

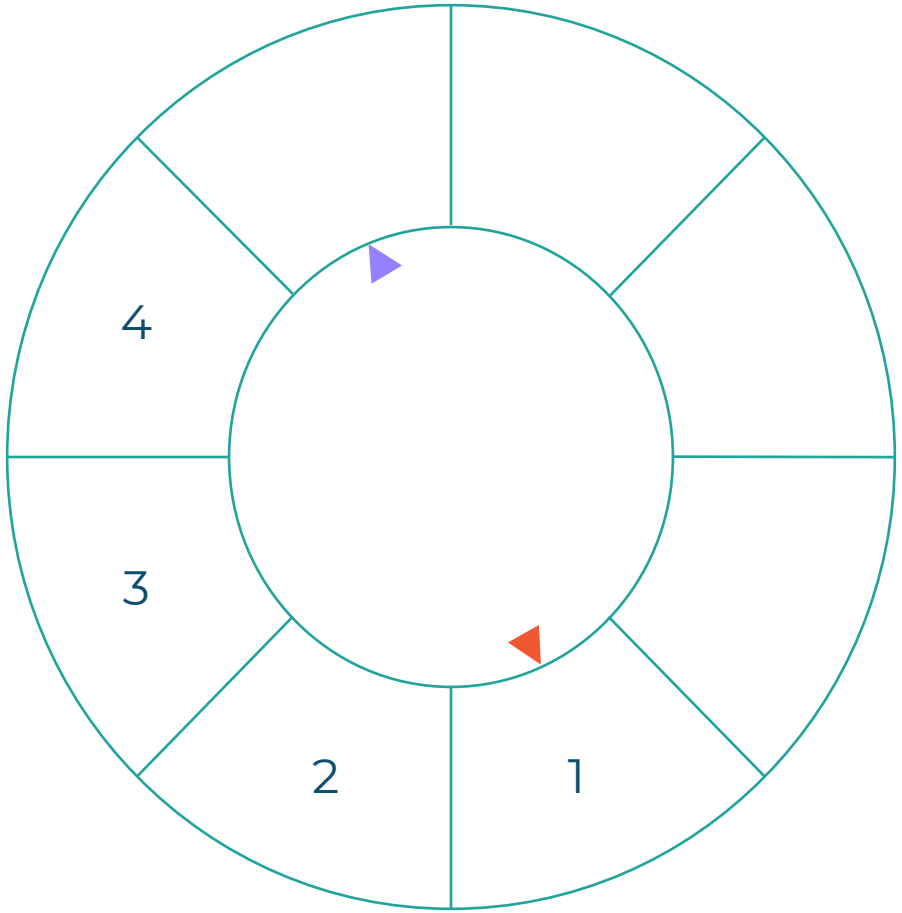



```
ringBuffer.Write(3);
```

Ring buffer

Write index

Read index

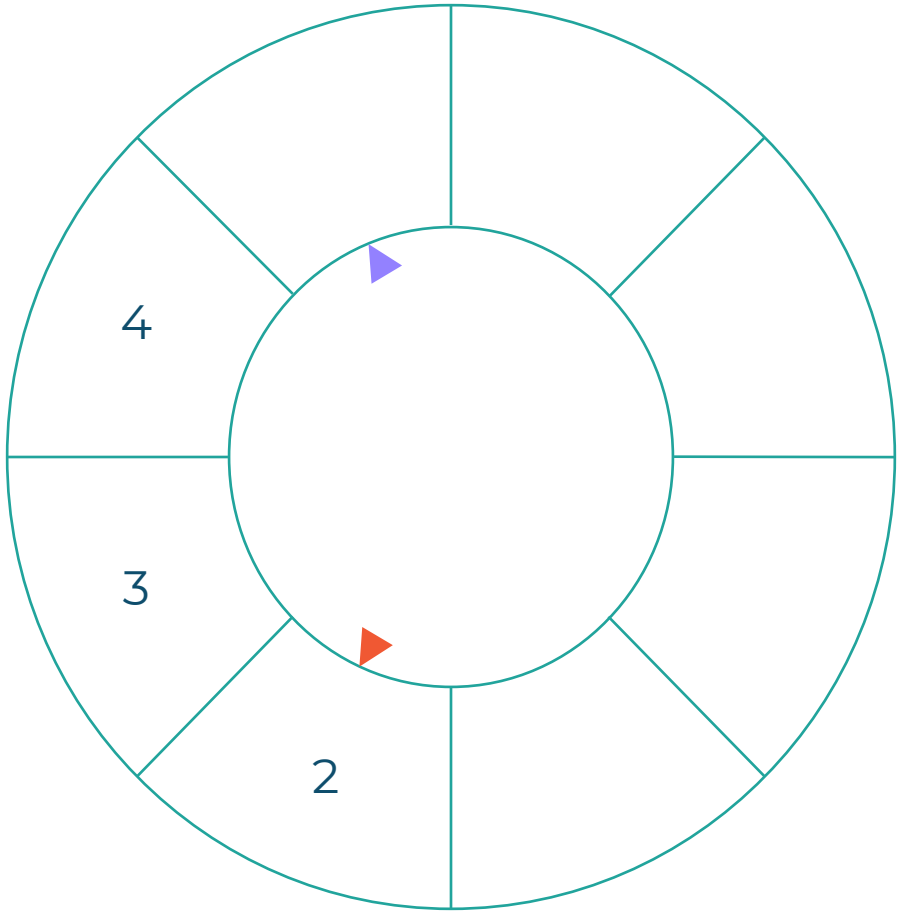



```
ringBuffer.Write(4);
```

Ring buffer

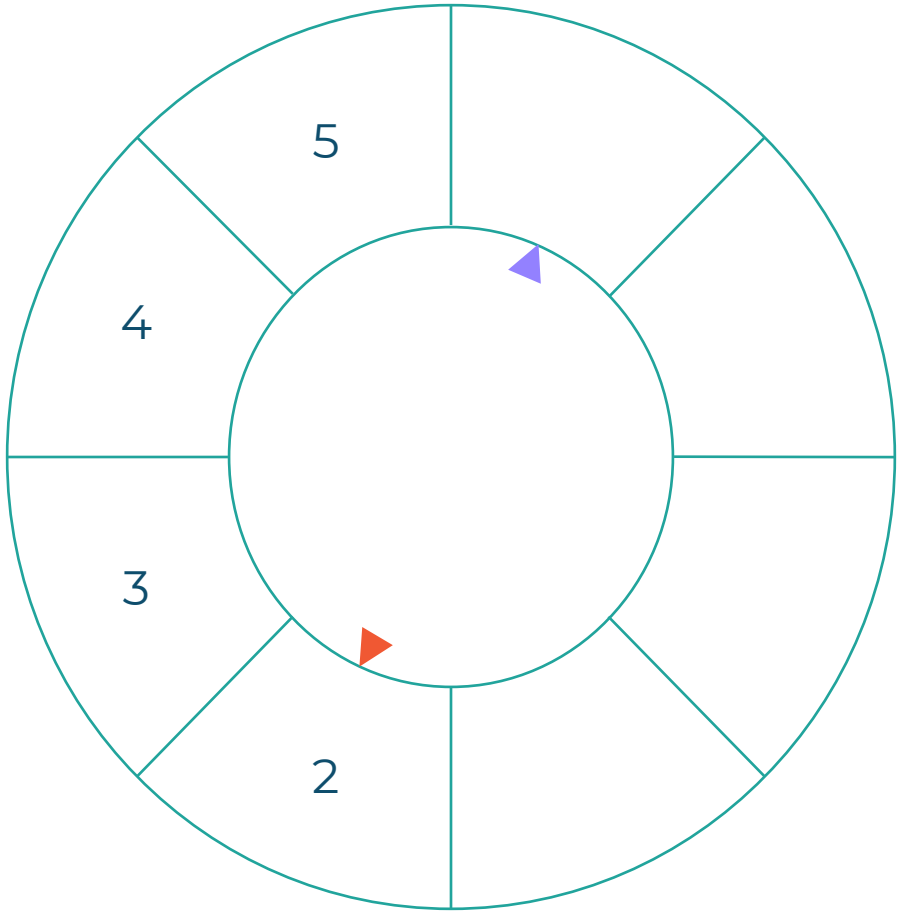
Write index

Read index




```
ringBuffer.Read()
```

Ring buffer



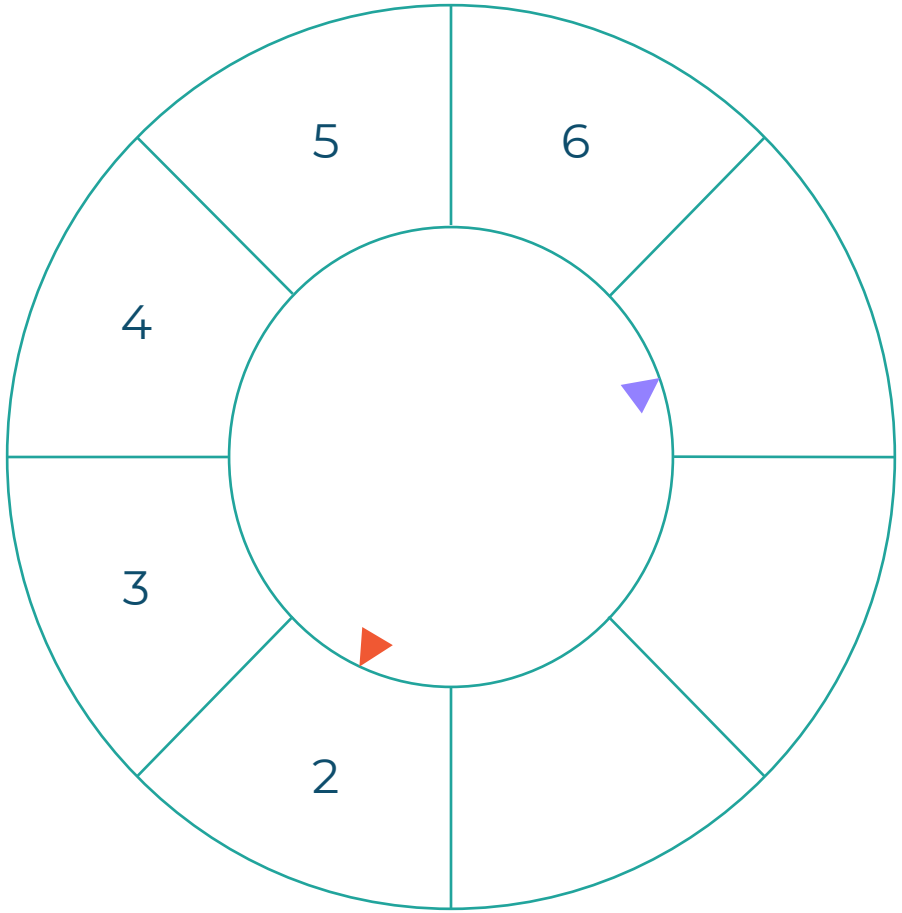
Write index

Read index

```
ringBuffer.Write(5);
```

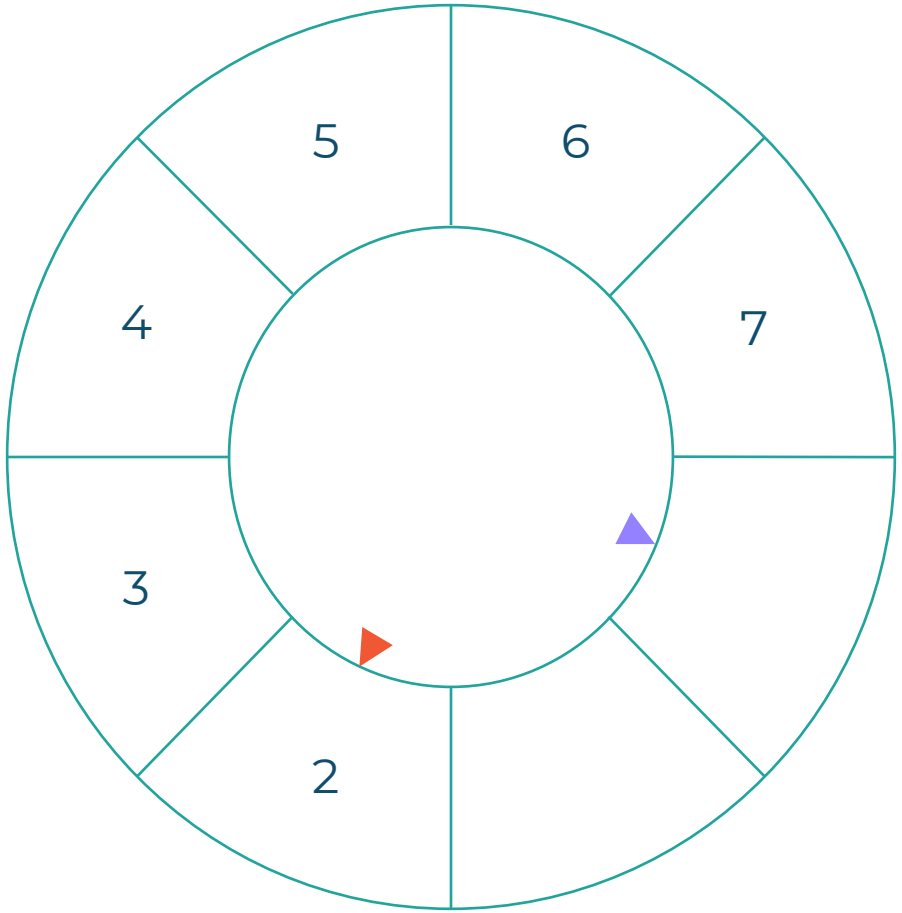
Ring buffer

- Write index
- Read index




```
ringBuffer.Write(6);
```

Ring buffer



Write index

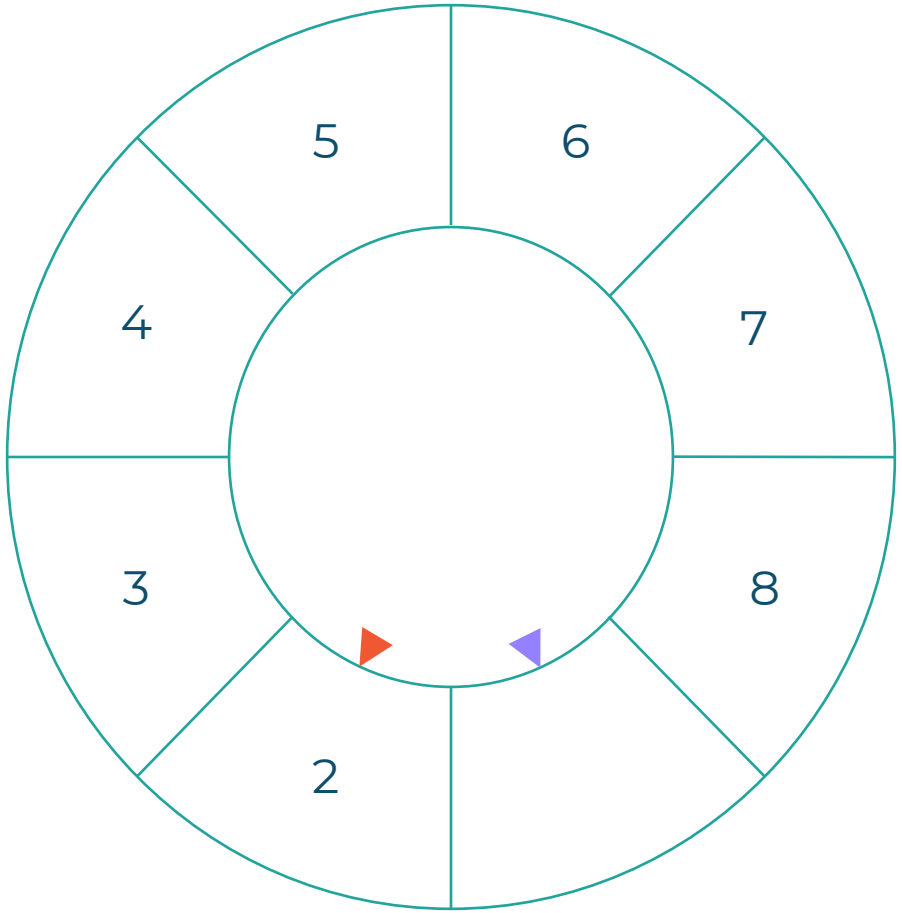
Read index

```
ringBuffer.Write(7);
```

Ring buffer

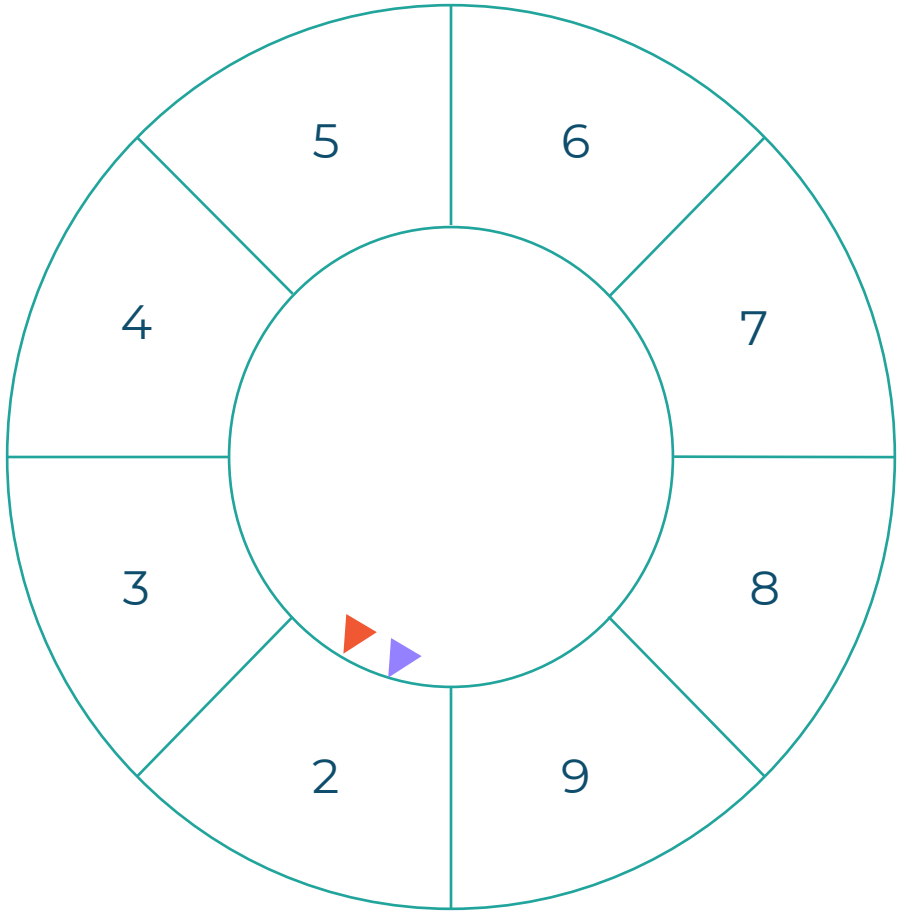
Write index

Read index




```
ringBuffer.Write(8);
```

Ring buffer

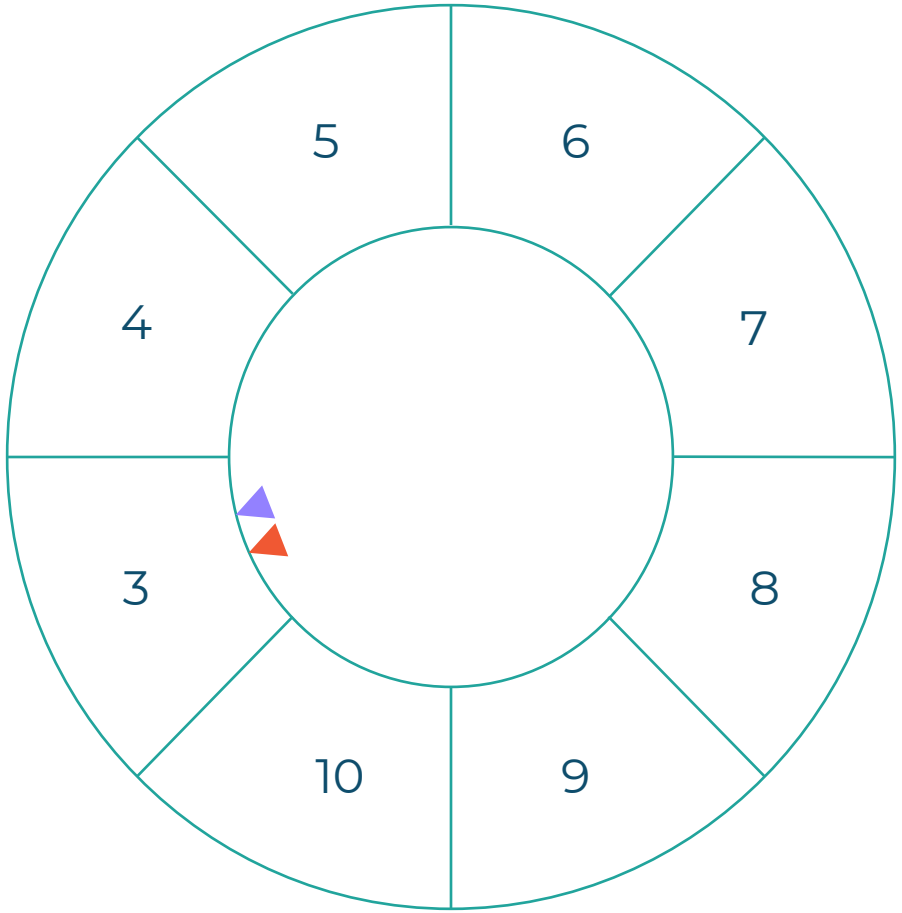


Write index

Read index

```
ringBuffer.Write(9);
```

Ring buffer

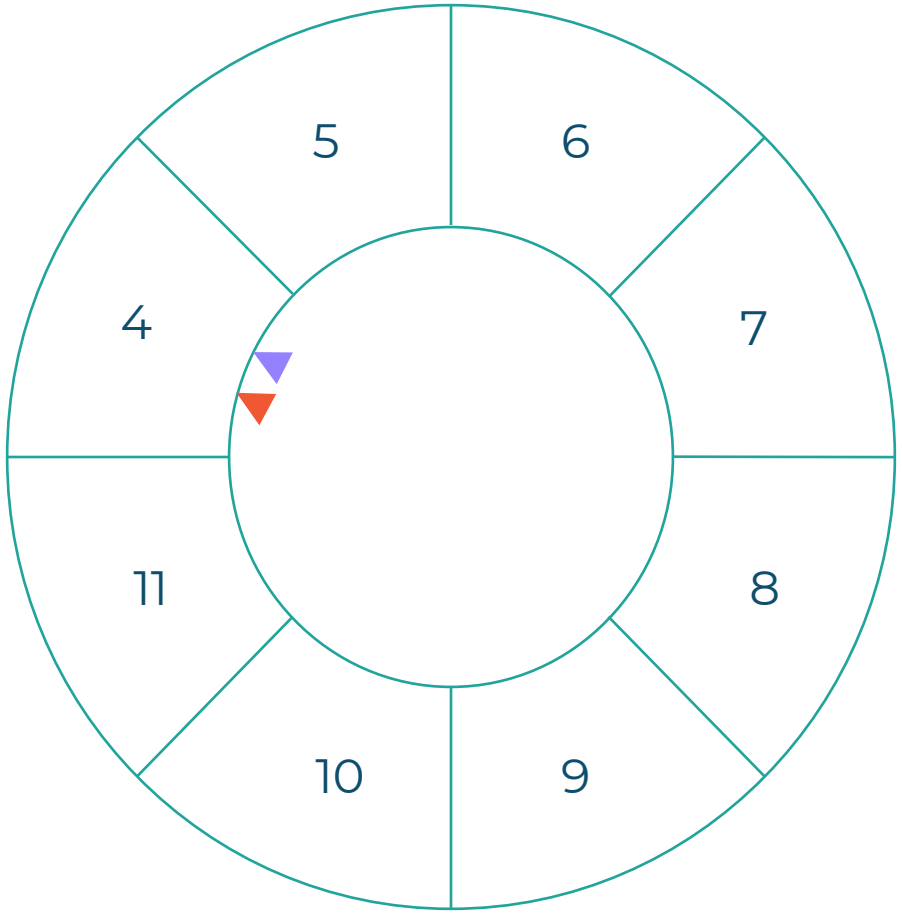


Write index

Read index

```
ringBuffer.Write(10);
```

Ring buffer

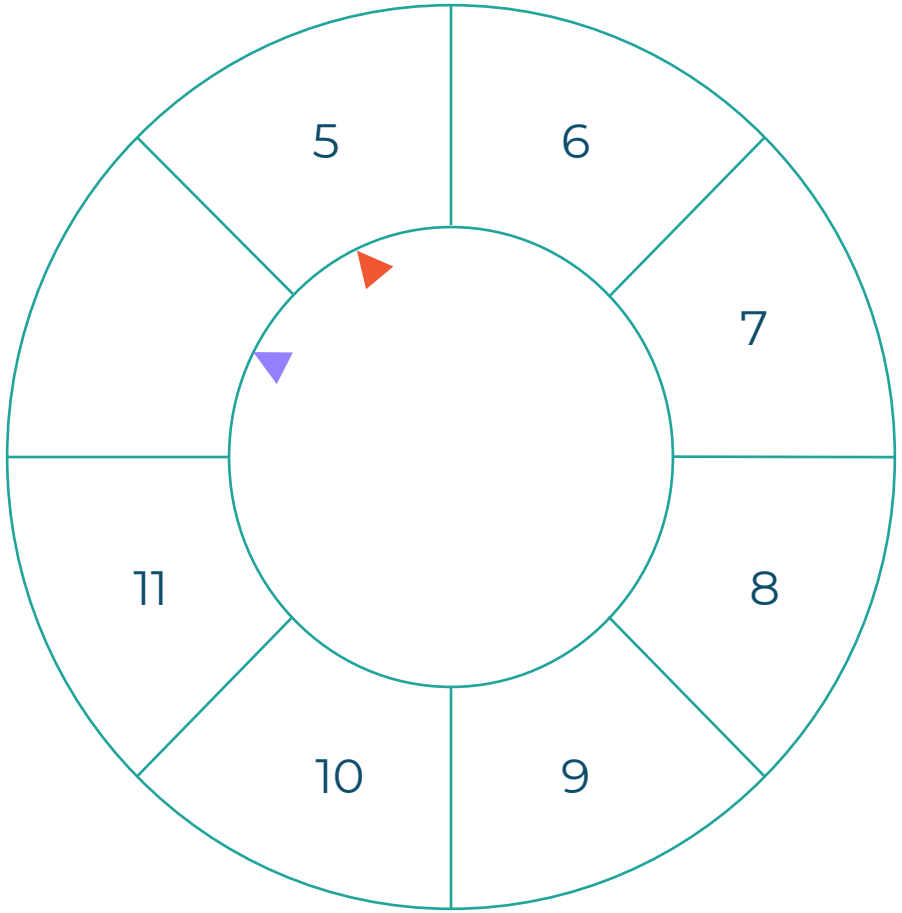


Write index

Read index

```
ringBuffer.Write(11);
```

Ring buffer

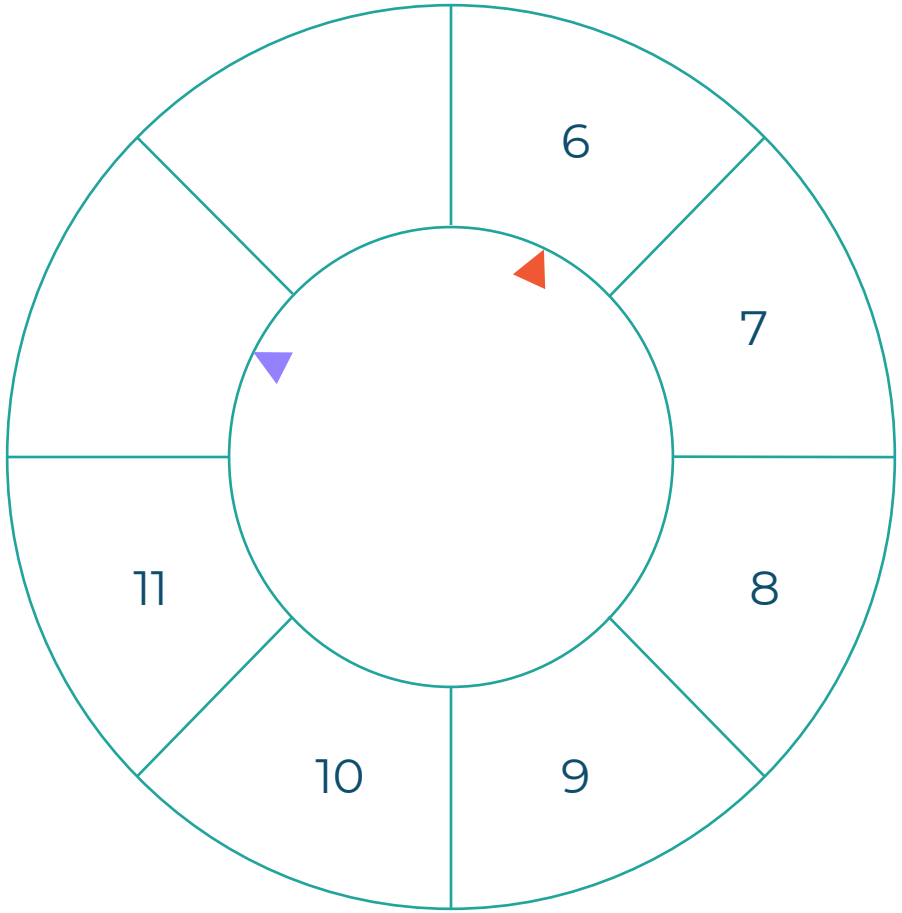


Write index

Read index

```
ringBuffer.Read();
```

Ring buffer



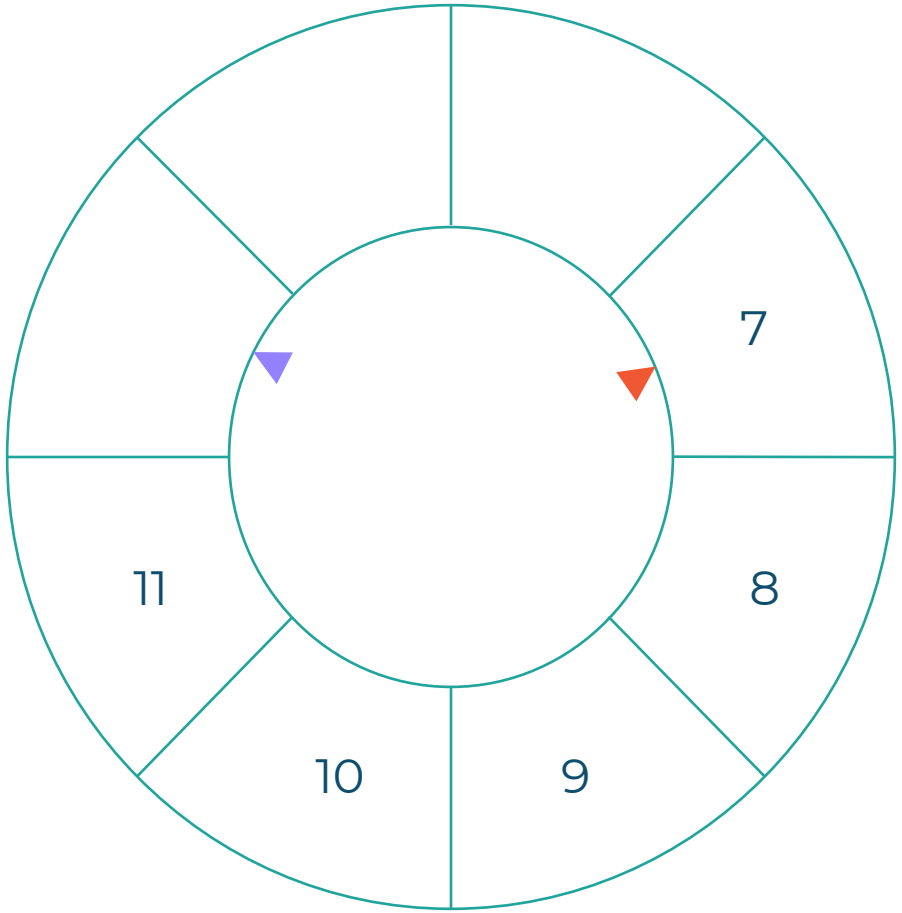
Write index

Read index

```
ringBuffer.Read();
```

Ring buffer

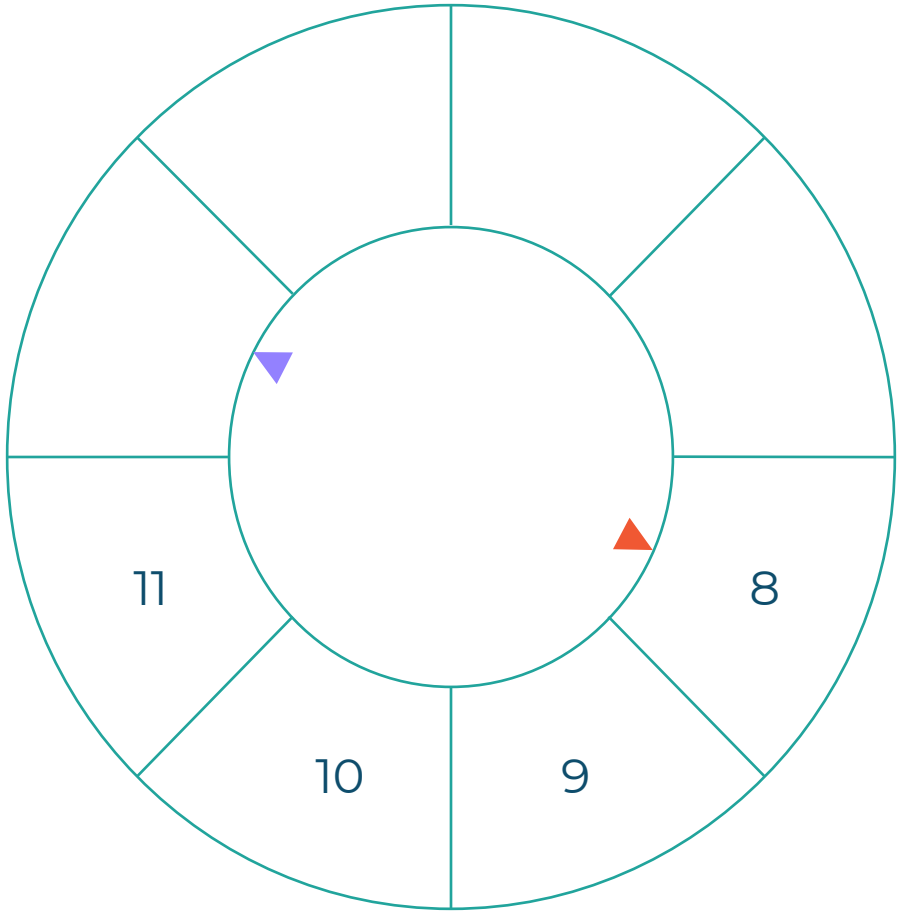
- Write index
- Read index




```
ringBuffer.Read();
```

Ring buffer

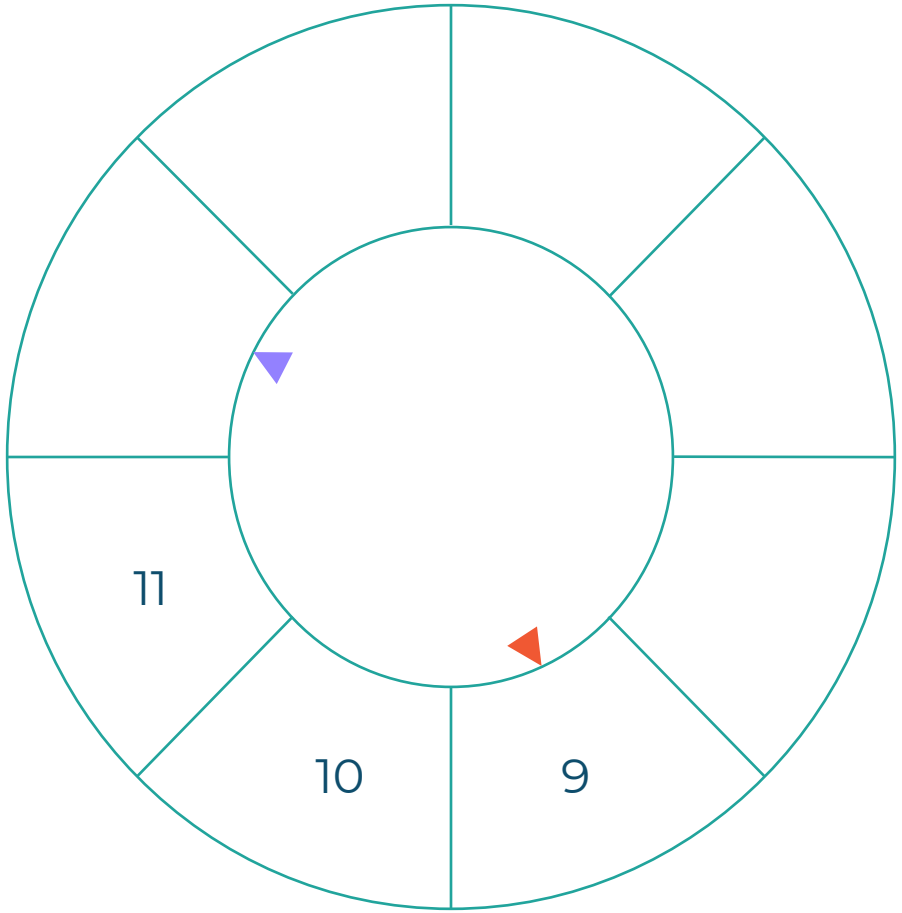
- Write index
- Read index




```
ringBuffer.Read();
```

Ring buffer

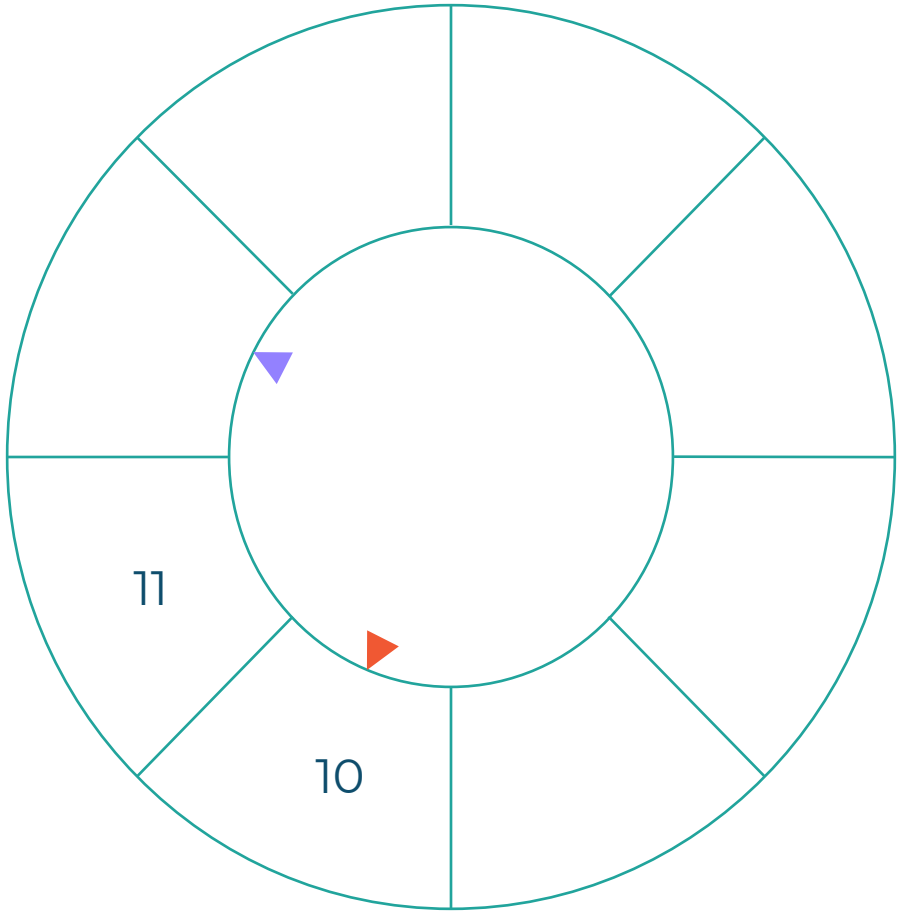
- Write index
- Read index




```
ringBuffer.Read();
```

Ring buffer

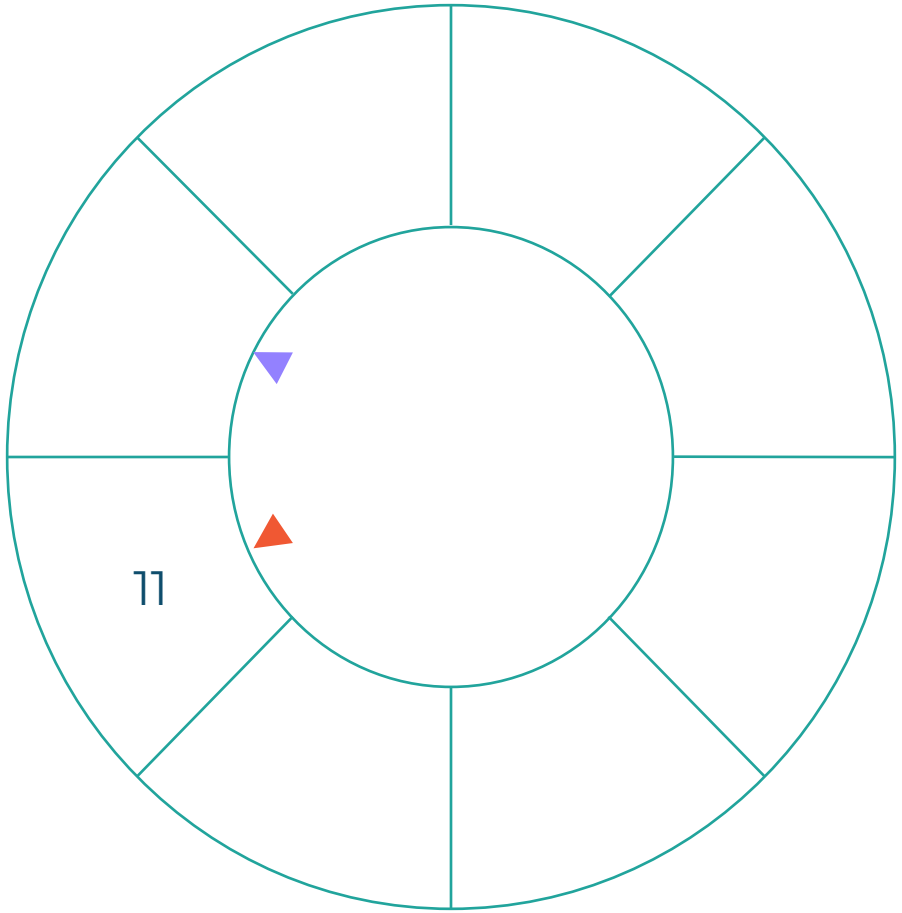
- Write index
- Read index




```
ringBuffer.Read();
```

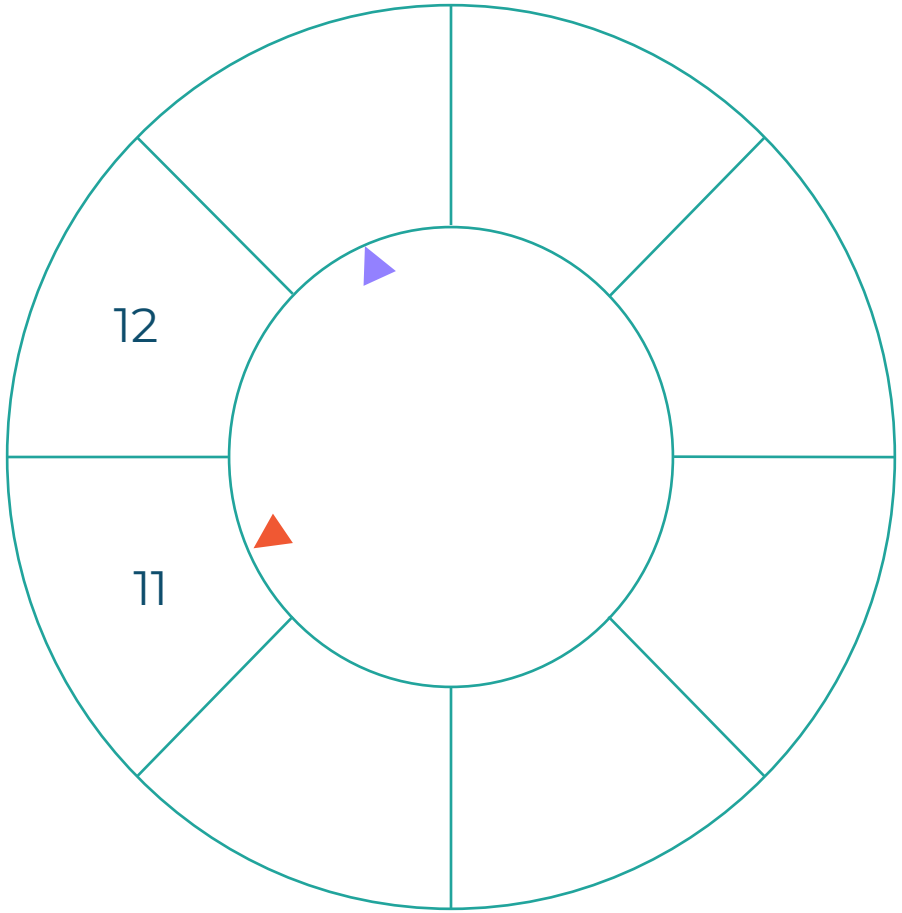
Ring buffer

- Write index
- Read index




```
ringBuffer.Read();
```

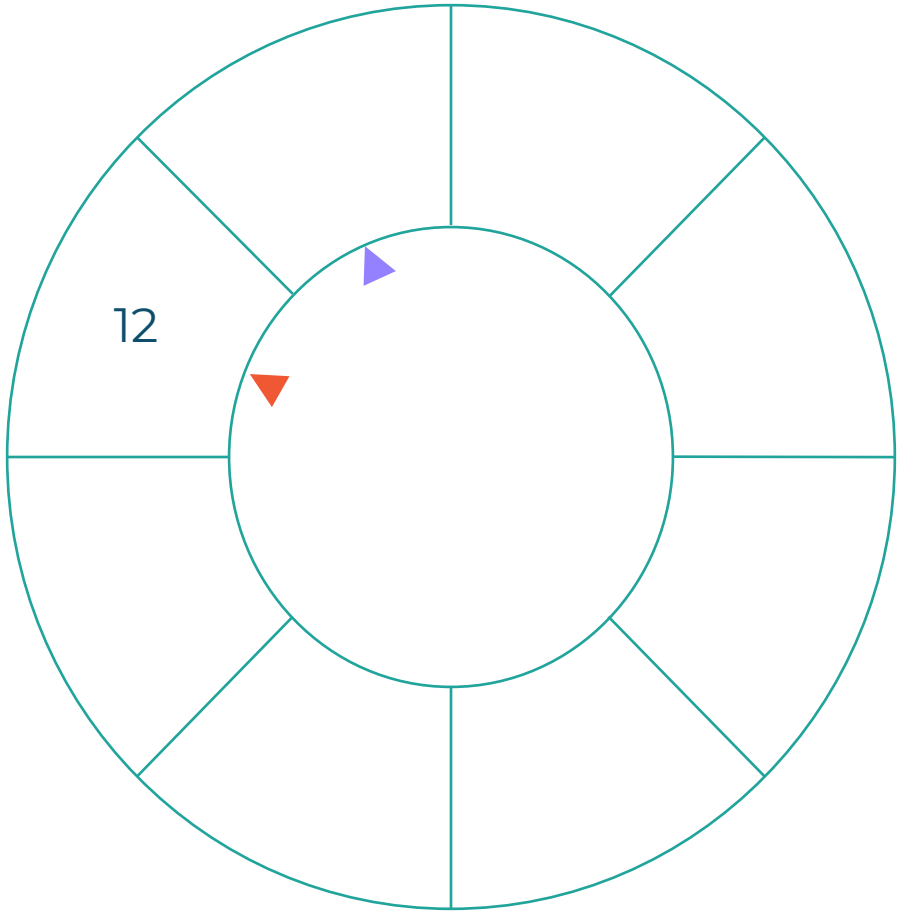
Ring buffer



- Write index
- Read index

```
ringBuffer.Write(12);
```

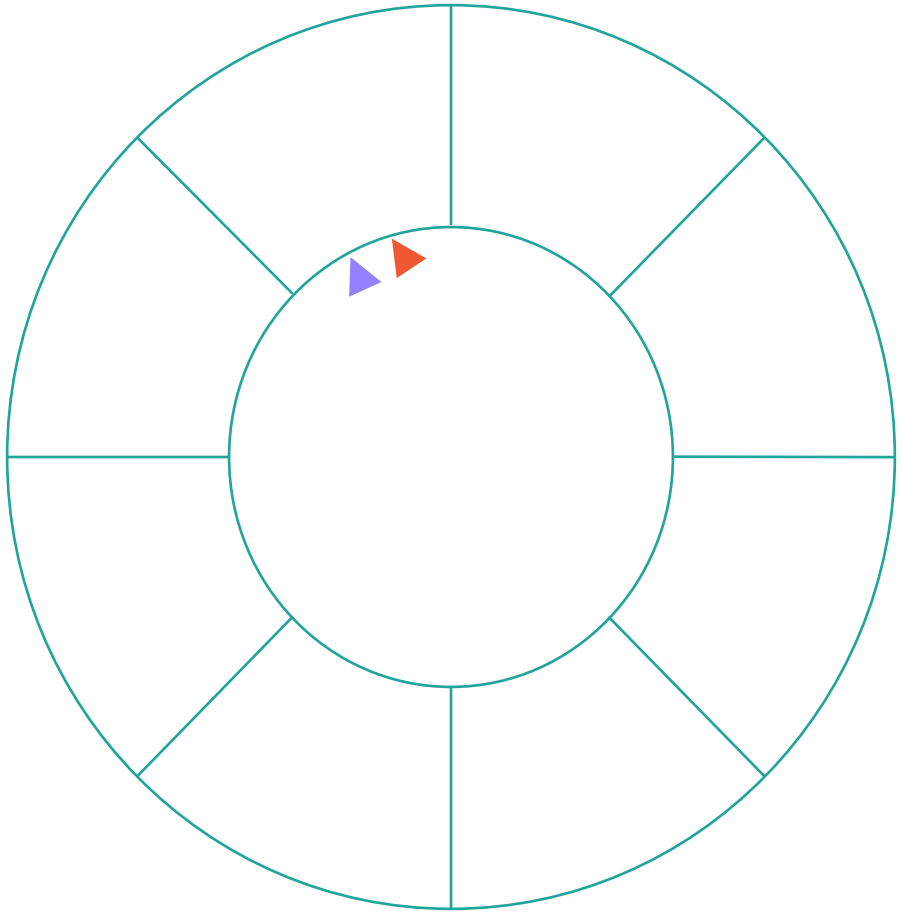
Ring buffer



- Write index
- Read index

```
ringBuffer.Read();
```

Ring buffer



- Write index
- Read index

```
ringBuffer.Read();
```

Requirements

Follow the requirements in the comments of the project

Summary

Collections allow to group and manage objects of the same type (elements)

Arrays are fixed size zero based index collections

Arrays inherit from the array abstract class

Arrays offer good performances for accessing elements