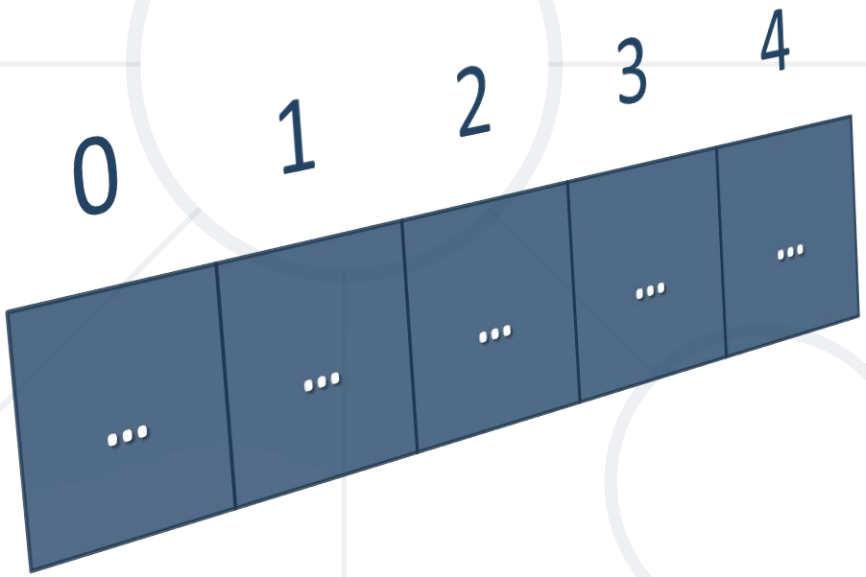
Arrays

Fixed-Size Sequences of Elements



SoftUni Team

Technical Trainers

Software University

SoftUni

Software University

https://softuni.bg/

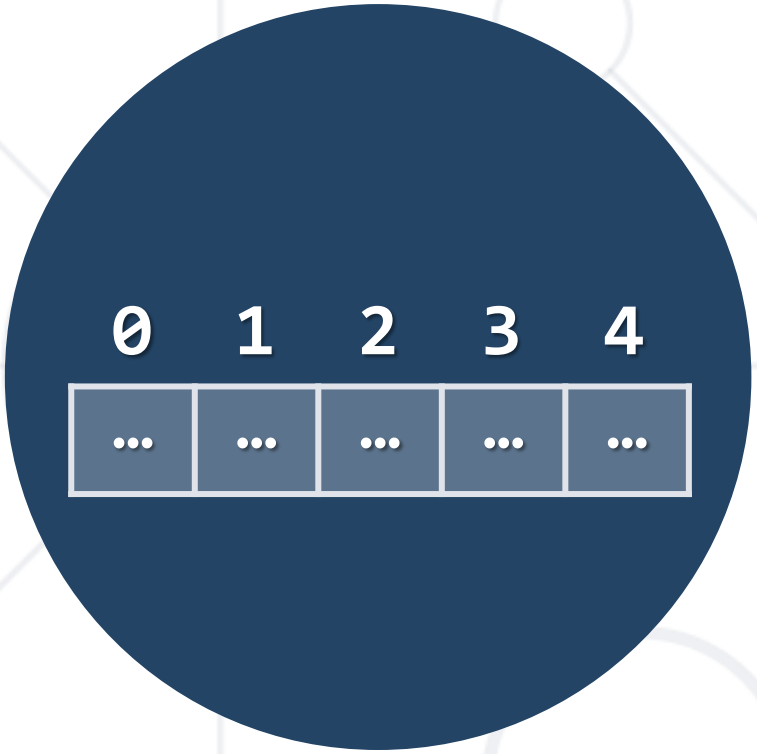
Table of Contents

Have a Question?

Software University

sli.do

#cpp-fundamentals

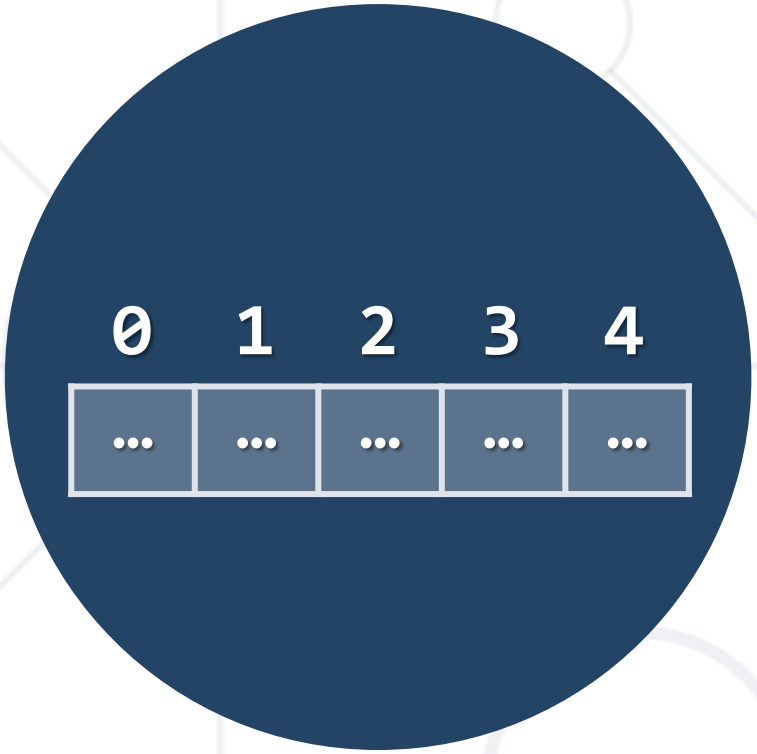


Arrays

What Are Arrays?

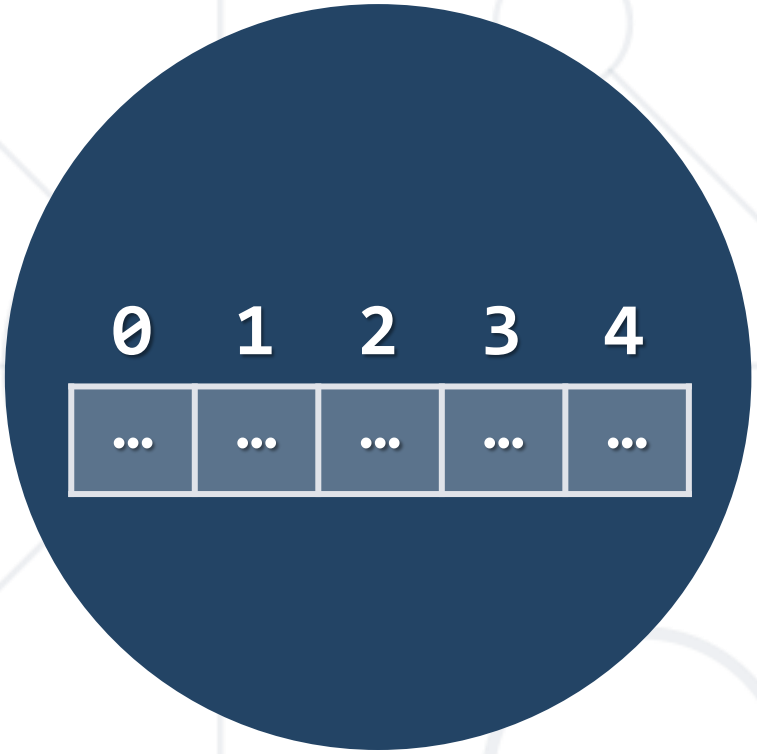
- In programming, an **array** is a **sequence of elements**

- Elements are numbered from **0** to **Length-1**
- Elements are of the **same type** (e.g. integers)
- Arrays have **fixed size** and cannot be resized



Arrays

LIVE DEMO



Creating C++ Arrays

Creating C++ Arrays

- Declaring

```
DataType identifier[arraySize];
```

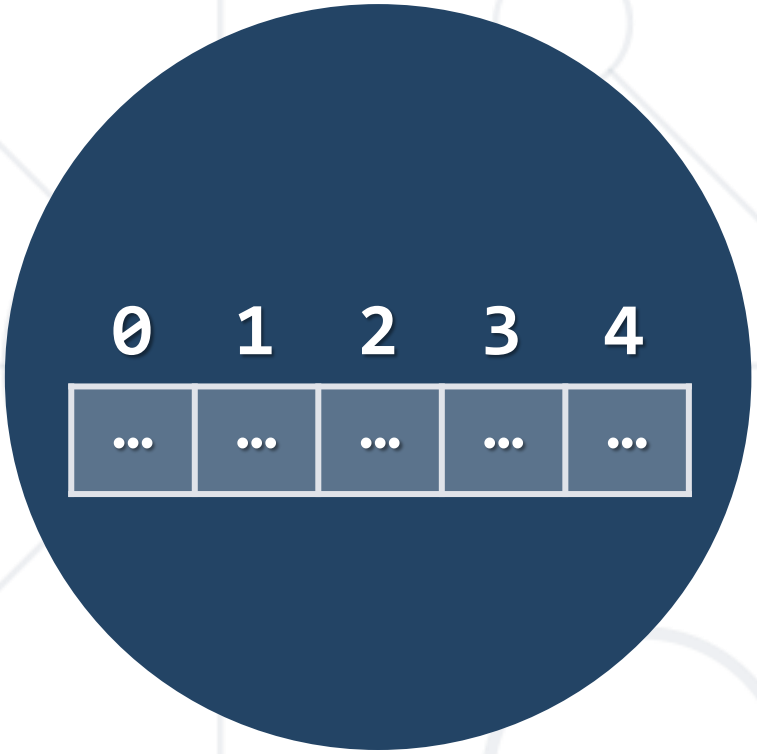
- C++ arrays have some special initialization syntax

```
DataType identifier[N] = {elem0, elem1, ..., elemN-1};
```

  - There can be less than N elements, **but not more**

```
int numbers[5] = { 10, 9, 12, 31, 15 };
```

Index	0	1	2	3	4
Value	10	9	12	31	15



Declaring and Initializing

Array Declaration

- C++ Array size must be an integer, known compile-time
  - i.e. size can be a literal
  - or a **const** value/expression

```
double numbers[7];
const int NumLetters = 26;
char alphabet[NumLetters];
```

- You can also declare an array **without a specified size**
  - The compiler is smart enough to get the elements' count we put inside the braces

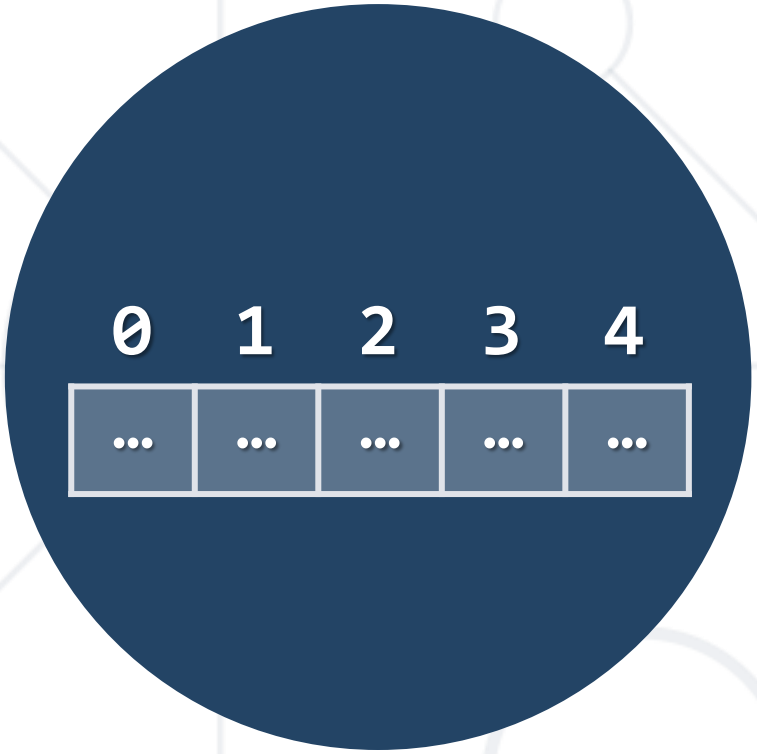
```
int numbersToFive[] = { 1, 2, 3, 4, 5 };
```

Array Initialization

- **{ }** initializes elements (comma-separated values)
  - if less values than array size -> remaining get default values

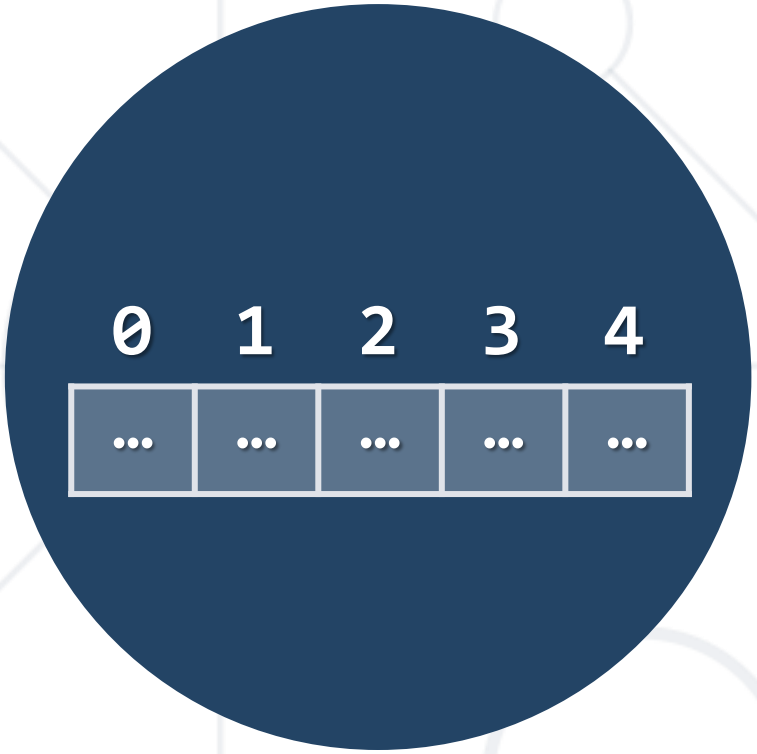
```
double values[3] = {3.14};
double sameValues[3] = {3.14, 0, 0};
```

  - if more values than array size -> compilation error
- Other rules are the same as for primitives
  - Can only be initialized once
  - Can be made **const**



Declaring and Initializing

LIVE DEMO



Reading and Printing Arrays

Accessing Array Elements

- The indexing operator [ ] gives access to any array element

```
int array[index] = value;
DataType value = arrayName[index]
```

- Once you access the element, treat it as a normal variable

```
int array[3] = { 20, 30, 40 };
cout << "First->" << array[0]
     << "Second->"<< array[1]
     << "Third->" << array[2] << endl;
```

Reading-in an Array

- Arrays are often read-in from some input, instead of initialized
- That's the point of arrays – to store arbitrary amounts of data
- Common approach: run a loop to read-in a number of elements
  - Example: read-in a specified number of elements from console

```
for (int i = 0; i < n; i++) {
  std::cin << arr[i];
}
```

Writing-out (printing) an Array

- You will commonly need to display all elements of an array
- Common approach: loop over the elements, print each
- Note: need to know how long the array is – keep a variable

```
for (int i = 0; i < n; i++) {
  std::cout << arr[i] << " ";
}
```

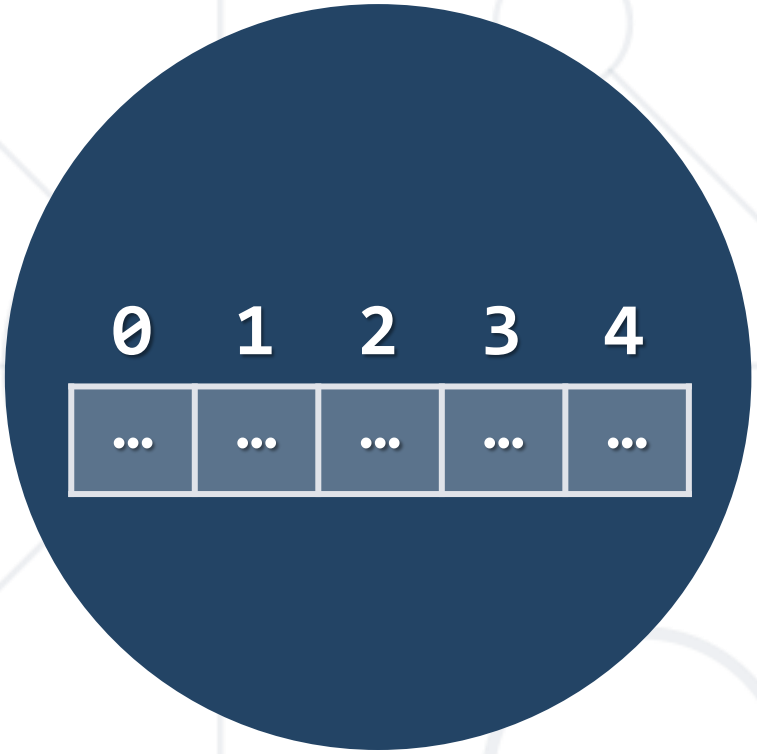
Read and Print an Array

```
const int lenght = 20;

int array[lenght];
cout << "Enter elements in array: " << endl;
for(int i = 0; i < lenght; i++) {
    cin >> array[i];
}
cout << "Elements in array: " << endl;
for(int i = 0; i < lenght; i++) {
    cout << array[i] << " ";
}
cout << endl;
cout << "End of elements" << endl;
```

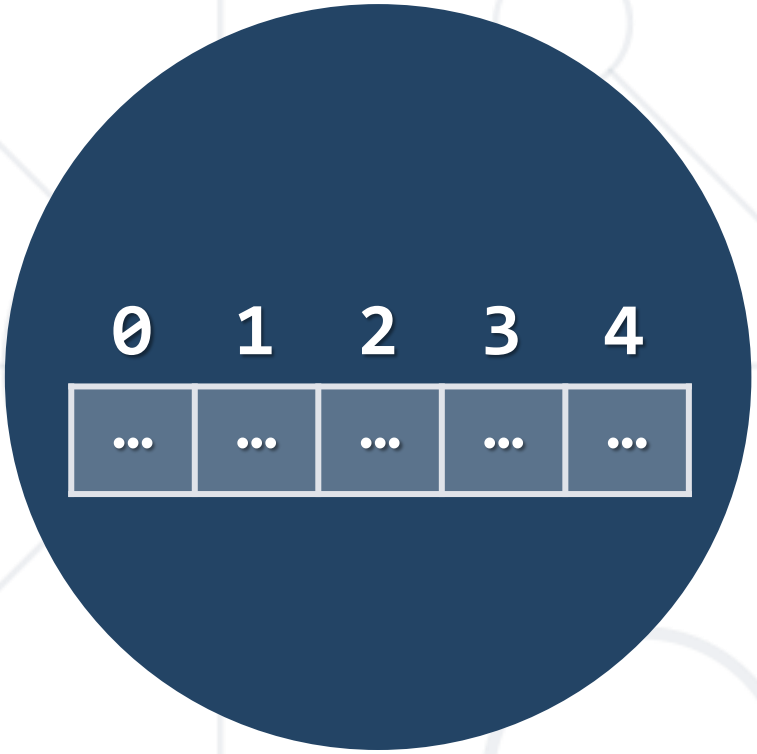
How NOT to Read and Print an Array

```
int lenght = 0;
cout << "Enter a lenght of array: " << endl;
cin >> lenght;
int array[lenght];
cout << "Enter elements in array: " << endl;
for(int i = 0; i < lenght; i++) {
    cin >> array[i];
}
cout << "Elements in array: " << endl;
for(int i = 0; i < lenght; i++) {
    cout << array[i] << " ";
}
cout << endl;
cout << "End of elements" << endl;
```



Reading and Printing Arrays

LIVE DEMO



Arrays as Function Parameters

Arrays as Function Parameters

- Array parameters are declared the same way arrays are declared
  - First dimension size can be omitted in array parameter
  - Usually necessary to add an **int** with the size

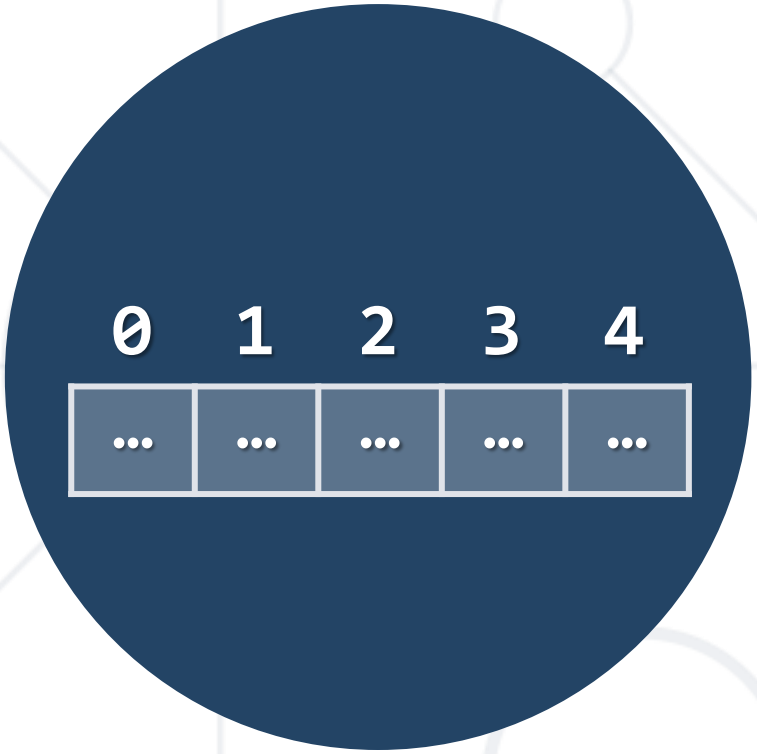
```
void print(int a[], int size) {
  for (int i = 0; i < size; i++) {
    cout << a[i] << " ";
  }
  cout << endl;
}
```

```
int main() {
  int numbers[] = {1,2,3};
  print(numbers, 3);

  return 0;
}
```

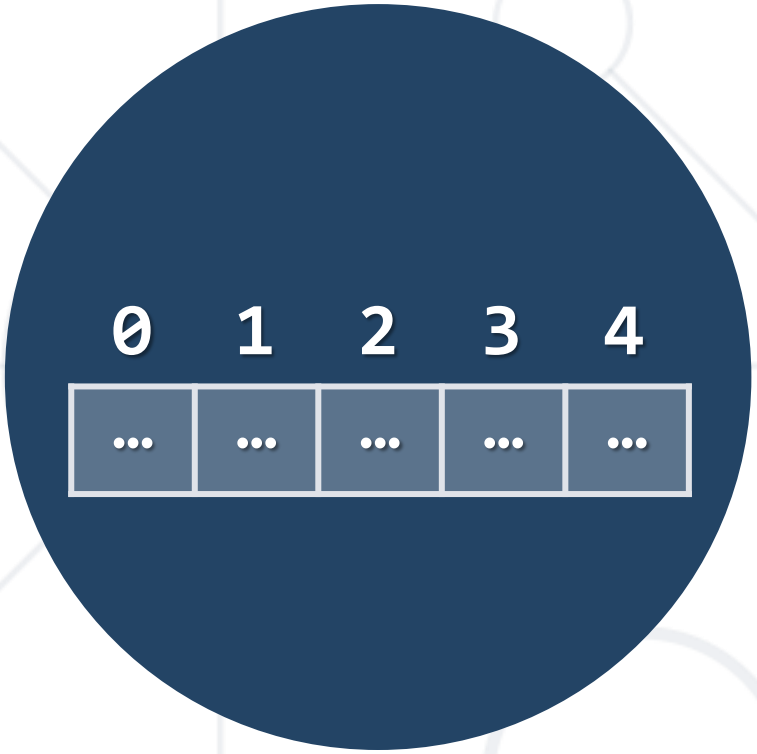
Arrays and Functions - Specifics

- Functions work with the original array the caller uses
  - If the function **changes an element**, the caller's array is **modified**
  - Array elements are passed "by reference"
- Functions can't return C++ "static" arrays created in them
  - Arrays are essentially memory addresses
  - The memory they point to is freed when the function exits
  - We will later discuss other ways to return sequences of elements



Arrays as Function Parameters

LIVE DEMO



Arrays as Pointers

Arrays as Pointers

- **Pointer** - a variable that holds the **address** of another variable (in the memory)
  - The address stores its **value**
  - Pointers have **data type**
- **Arrays** in C++ can be represented as pointers
  - The array is a **sequence of variables stored in memory**
  - The array name points to the first item
- *We will learn more about pointers in the next courses*

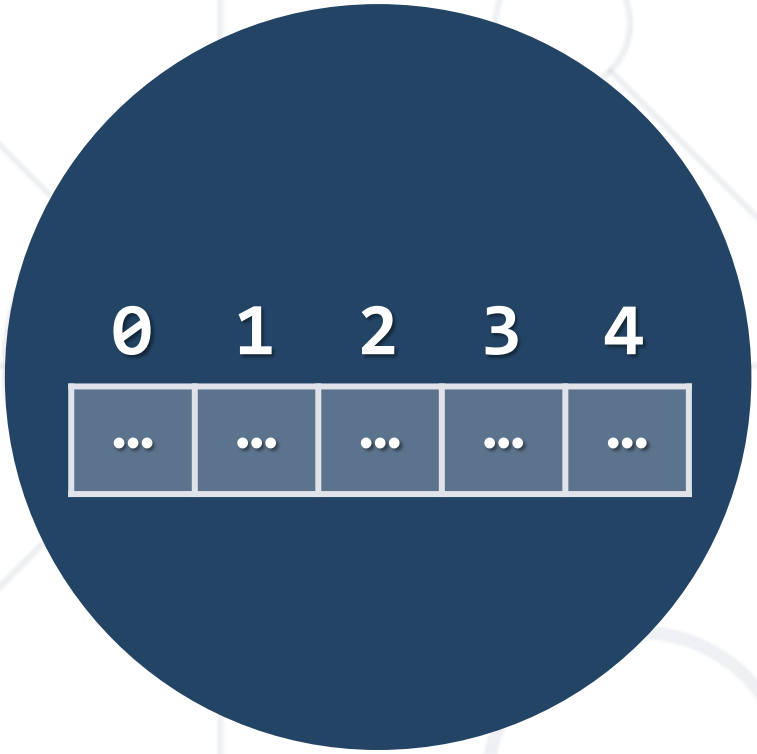
Arrays as Pointers

- Examples:

```
int *ptr;
int arr[4];
ptr = arr;

ptr + 0 is equivalent to &arr[0];
ptr + 1 is equivalent to &arr[1];
ptr + 2 is equivalent to &arr[2];
ptr + 3 is equivalent to &arr[3];
```

```
*ptr == arr[0];
*(ptr + 1) is equivalent to arr[1];
*(ptr + 2) is equivalent to arr[2];
```



Arrays as Pointers

LIVE DEMO

C++11 Range-Based for Loop

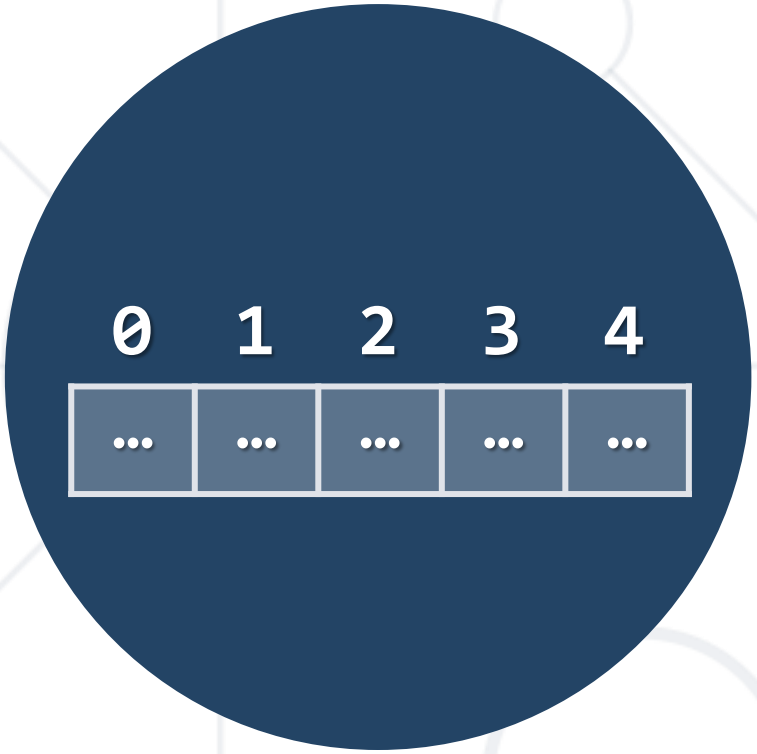
C++11 Range-Based for Loop

- Tired of writing for loops with indices to iterate over an array?
- C++11 added a loop for that use-case
- Syntax (for arrays): `for (DataType element : array)`
  - Body will execute once for each element in the array
  - On each iteration, `element` will be the next item in the array

```
int numbers[] = { 13, 42, 69 };
for (int i : numbers) {
    cout << i << endl;
}
```

Range-Based for Loop

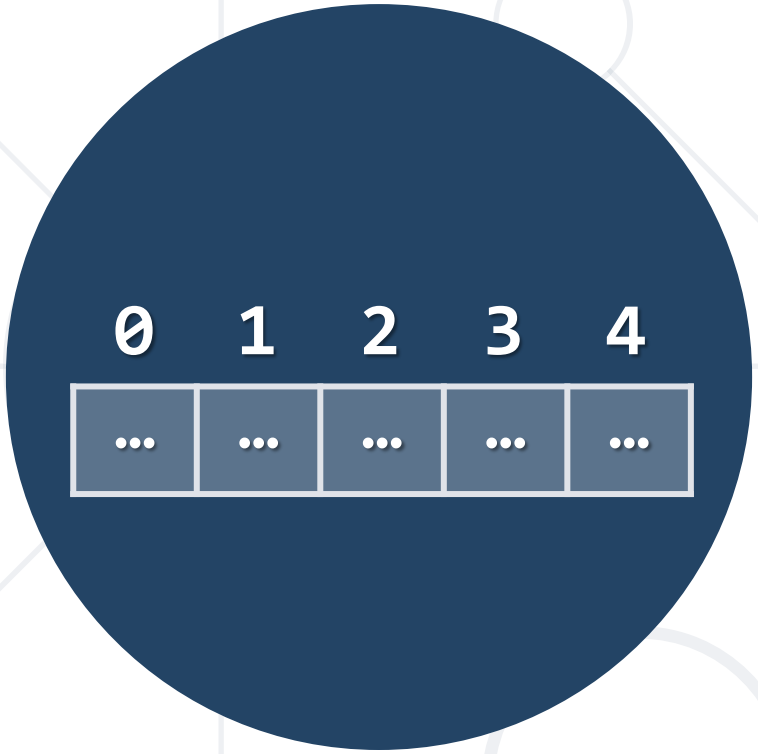
LIVE DEMO



C++11 <array> Header

C++11 <array> Header

- C++11 provides an alternative array, which is a bit smarter
- The `array` class knows its size, can be returned from functions
  - `#include<array>`
  - Declaring: `array<int, 5> arr;` is the same as `int arr[5];`
  - Declaring & Initializing:
  - `array<int, 5> arr = { 1, 2, 3, 4, 5 };`
  - `arr.size()` gives you the size of the array
  - Accessing elements: use the `[ ]` operator like with normal arrays



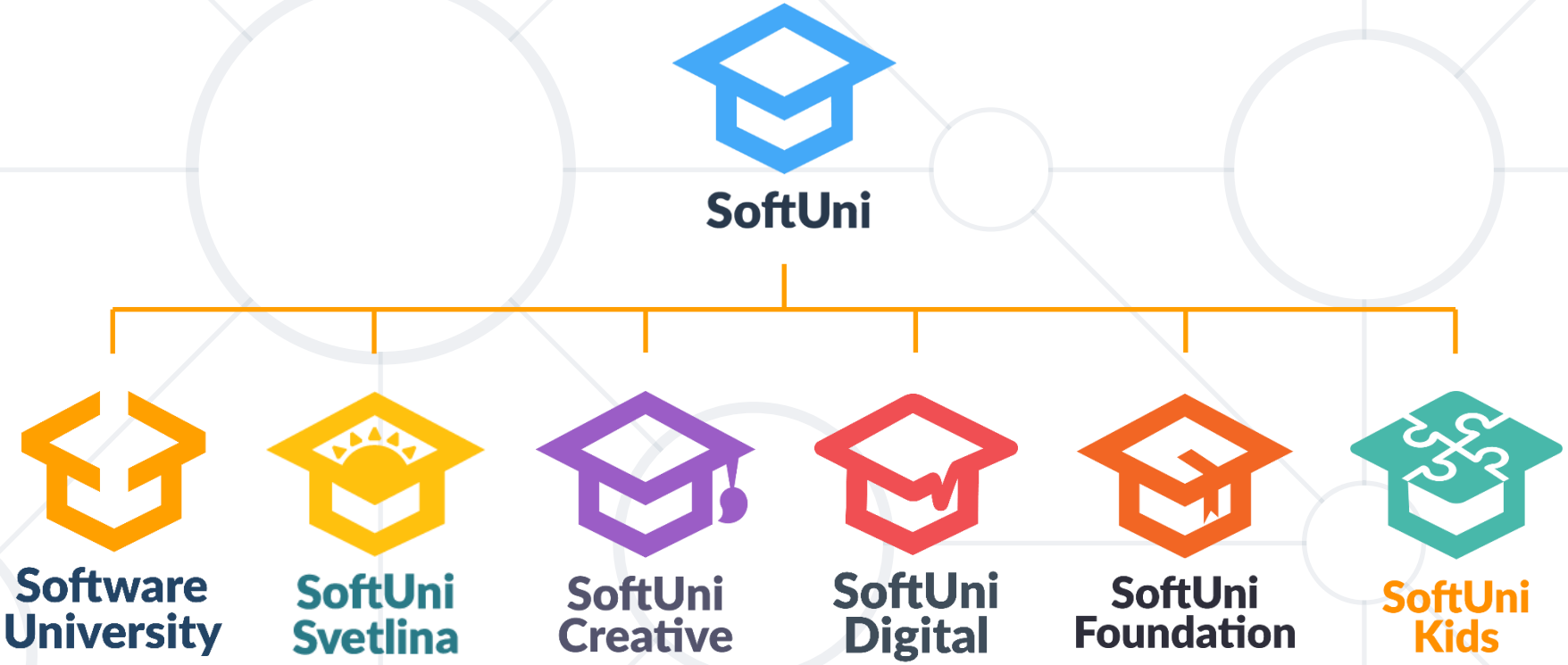
C++11 <array> Header

LIVE DEMO

Summary

- Arrays hold a **sequence** of elements
  - Elements are numbered from **`0`** to **`length-1`**
- Creating (allocating) an array
- Accessing array elements by **index**
- Printing array elements
- Range-Based for Loop

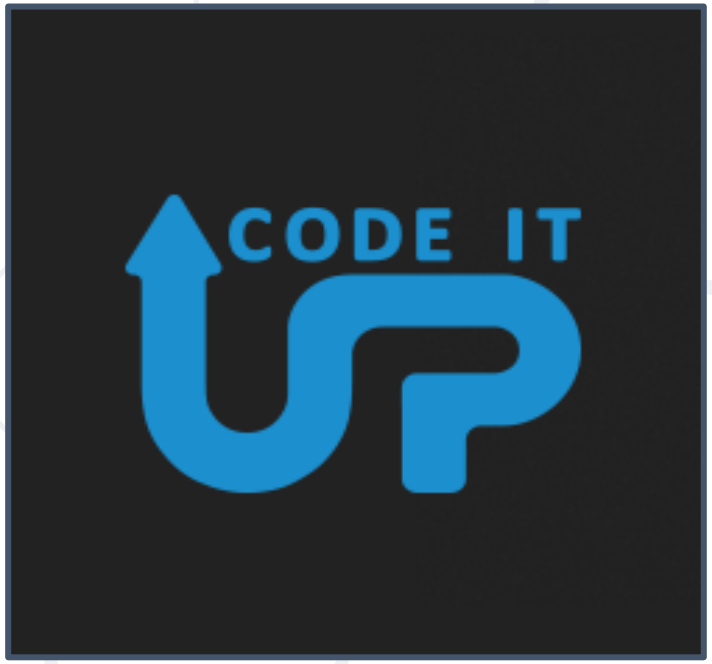
Questions?



SoftUni Diamond Partners

Software University

Educational Partners



VIRTUAL RACING SCHOOL

License

- This course (slides, examples, demos, exercises, homework, documents, videos and other assets) is **copyrighted content**
- Unauthorized copy, reproduction or use is illegal
- © SoftUni – https://about.softuni.bg/
- © Software University – https://softuni.bg

Trainings @ Software University (SoftUni)

- Software University – High-Quality Education, Profession and Job for Software Developers
  - softuni.bg, about.softuni.bg
- Software University Foundation
  - softuni.foundation
- Software University @ Facebook
  - facebook.com/SoftwareUniversity
- Software University Forums
  - forum.softuni.bg