WESTERN SYDNEY
UNIVERSITY

Accessing Array Contents

ELEC1006: ENGINEERING COMPUTING

Accessing Array Contents

WESTERN SYDNEY UNIVERSITY

- Can access element with a constant or literal subscript:

```
cout << tests[3] << endl;
```

- Can use integer expression as subscript:

```
int i = 5;
cout << tests[i] << endl;
```

Using a Loop to Step Through an Array

WESTERN SYDNEY UNIVERSITY

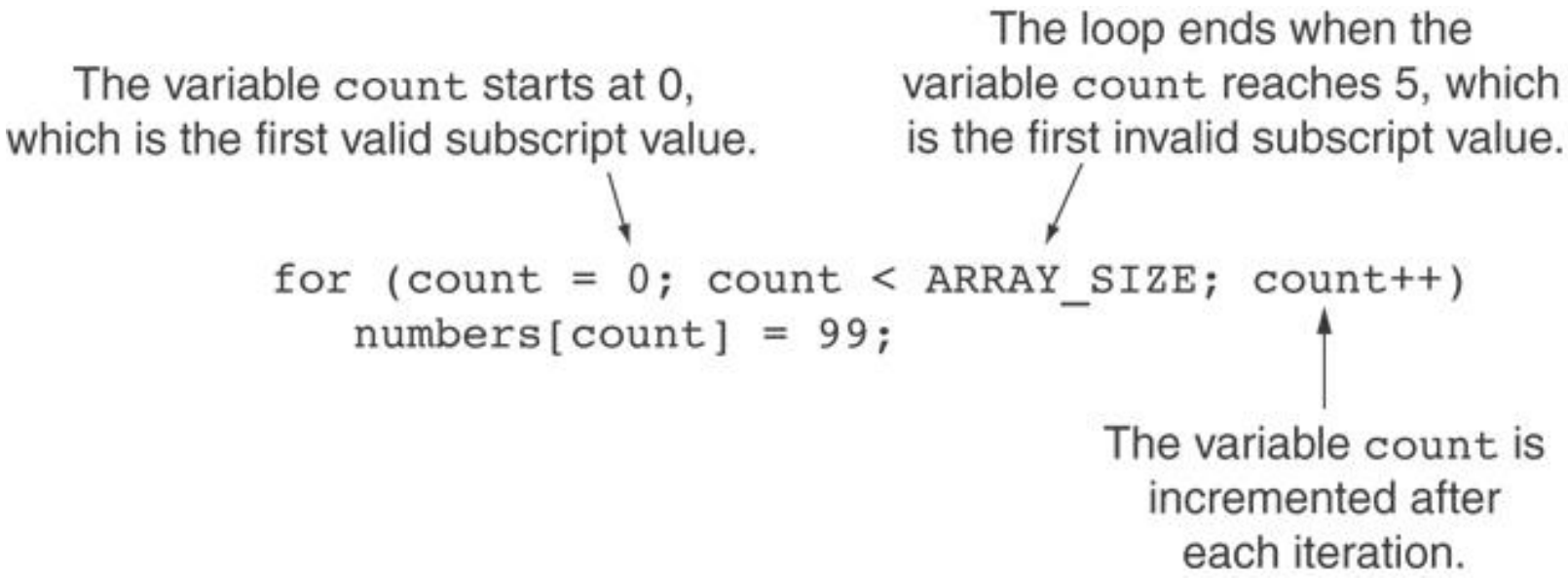
- Example – The following code defines an array, `numbers`, and assigns 99 to each element:

```
const int ARRAY_SIZE = 5;
int numbers[ARRAY_SIZE];

for (int count = 0; count < ARRAY_SIZE; count++)
     numbers[count] = 99;
```

A Closer Look at the Loop

WESTERN SYDNEY UNIVERSITY



Default Initialisation

WESTERN SYDNEY UNIVERSITY

- Global array → all elements initialized to `0` by default

- Local array → all elements *uninitialized* by default

No Bounds Checking in C++

WESTERN SYDNEY UNIVERSITY

- When you use a value as an array subscript, C++ does not check it to make sure it is a *valid* subscript.

- In other words, you can use subscripts that are beyond the bounds of the array.

Example 2

WESTERN SYDNEY UNIVERSITY

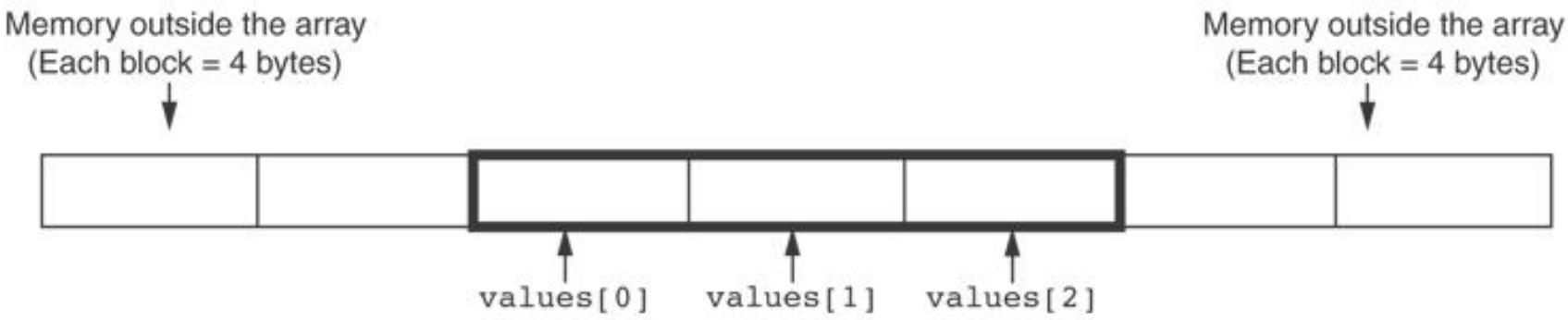
- The following code defines a three-element array, and then writes five values to it!

```
 9     const int SIZE = 3;   // Constant for the array size
10     int values[SIZE];     // An array of 3 integers
11     int count;            // Loop counter variable
12
13     // Attempt to store five numbers in the three-element array.
14     cout << "I will store 5 numbers in a 3 element array!\n";
15     for (count = 0; count < 5; count++)
16        values[count] = 100;
```

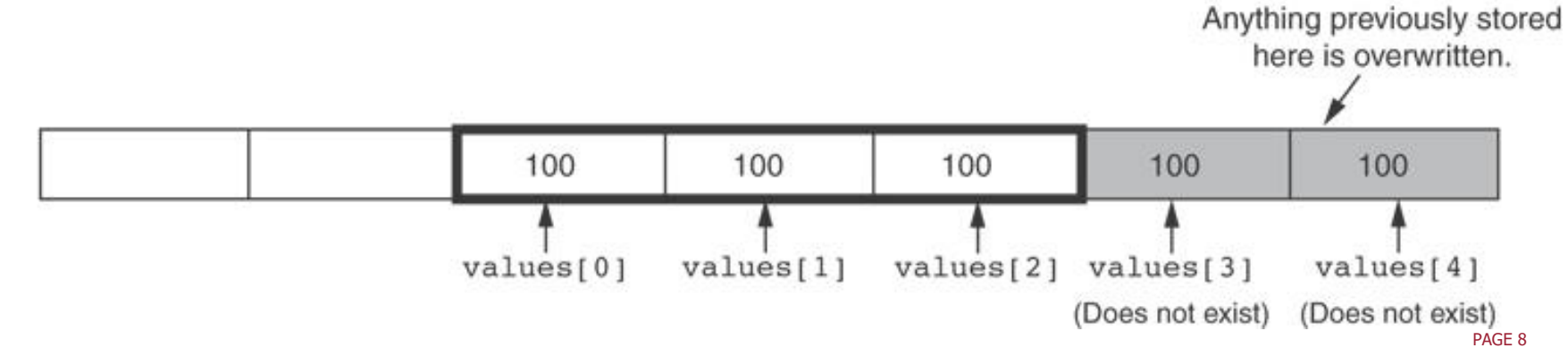
Example 2 (continued)

WESTERN SYDNEY UNIVERSITY

The way the `values` array is set up in memory.
The outlined area represents the array.



How the numbers assigned to the array overflow the array's boundaries.
The shaded area is the section of memory illegally written to.



No Bounds Checking in C++

WESTERN SYDNEY UNIVERSITY

- **Be careful not to use invalid subscripts**.
- **Doing so can corrupt other memory locations, crash program, or lock up computer, and cause elusive bugs**.

Off-by-One Errors

WESTERN SYDNEY UNIVERSITY

- **An off-by-one error happens when you use array subscripts that are off by one**.
- This can happen when you start subscripts at 1 rather than 0:

```
// This code has an off-by-one error.
const int SIZE = 100;
int numbers[SIZE];
for (int count = 1; count <= SIZE; count++)
   numbers[count] = 0;
```

More info

WESTERN SYDNEY UNIVERSITY

[1] cplusplus.com: Arrays
https://www.cplusplus.com/doc/tutorial/arrays/

[2] learncpp.com: 6.3 – Arrays and loops
https://www.learncpp.com/cpp-tutorial/63-arrays-and-loops/