



# Array

UTM RESEARCH UNIVERSITY



# Array

- Array: variable that can store a collection of data of the same type
  - Examples: A list of names, A list of temperatures
- Why do we need arrays?
  - Imagine keeping track of 5 test scores, or 100, or 1000 in memory
    - How would you name all the variables?
    - How would you process each of the variables?

UTM RESEARCH UNIVERSITY



## Declaring an Array

- An array, named test, containing five variables of type int can be declared as

```
int tests[5];
```

- The value in brackets is called
  - A subscript
  - An index

UTM UNIVERSITY

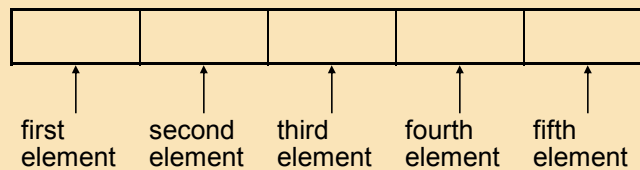


## Array - Memory Layout

- The definition:

```
int tests[5];
```

allocates the following memory:



UTM UNIVERSITY



## Array Terminology

In the definition `int tests[5];`

- `int` is the data type of the array elements
- `tests` is the name of the array
- `5`, in `[5]`, is the size declarator. It shows the number of elements in the array.
- The size of an array is (number of elements) \* (size of each element)

UTM: MORE CREATIVE & INNOVATIVE CAMPUS



## Array Terminology

- The size of an array is:
  - the total number of bytes allocated for it
  - (number of elements) \* (number of bytes for each element)
- Examples:
  - `int tests[5]` is an array of 20 bytes, assuming 4 bytes for an `int`
  - `long double measures[10]` is an array of 80 bytes, assuming 8 bytes for a `long double`

UTM: MORE CREATIVE & INNOVATIVE CAMPUS



## Size Declarators

- Named constants are commonly used as size declarators.

```
const int SIZE = 5;  
int tests[SIZE];
```

- This eases program maintenance when the size of the array needs to be changed.

UTM: MORE CREATIVE & INNOVATIVE CAMPUS



## Accessing Array Elements

- Each element in an array is assigned a unique *subscript*.
- Subscripts start at 0

subscripts:

0	1	2	3	4

UTM: MORE CREATIVE & INNOVATIVE CAMPUS



## Accessing Array Elements

- The last element's subscript is  $n-1$  where  $n$  is the number of elements in the array.

subscripts:

0	1	2	3	4

UTM: BE CREATIVE & INNOVATIVE



## Accessing Array Elements

- Array elements can be used as regular variables:

```
tests[0] = 79;
cout << tests[0];
cin >> tests[1];
tests[4] = tests[0] + tests[1];
```
- Arrays must be accessed via individual elements:

```
cout << tests; // not legal
```

UTM: BE CREATIVE & INNOVATIVE



## Accessing Array Elements - example

### Program 7-1

```

1  // This program asks for the number of hours worked
2  // by six employees. It stores the values in an array.
3  #include <iostream>
4  using namespace std;
5
6  int main()
7  {
8      const int NUM_EMPLOYEES = 6;
9      int hours[NUM_EMPLOYEES];
10
11     // Get the hours worked by six employees.
12     cout << "Enter the hours worked by six employees: ";
13     cin >> hours[0];
14     cin >> hours[1];
15     cin >> hours[2];
16     cin >> hours[3];
17     cin >> hours[4];
18     cin >> hours[5];
19

```

*(Program Continues)*



## Accessing Array Elements - example

```

20     // Display the values in the array.
21     cout << "The hours you entered are:";
22     cout << " " << hours[0];
23     cout << " " << hours[1];
24     cout << " " << hours[2];
25     cout << " " << hours[3];
26     cout << " " << hours[4];
27     cout << " " << hours[5] << endl;
28     return 0;
29 }

```

### Program Output with Example Input

Enter the hours worked by six employees: **20 12 40 30 30 15** [Enter]  
 The hours you entered are: 20 12 40 30 30 15

Here are the contents of the `hours` array, with the values entered by the user in the example output:

hours[0]	hours[1]	hours[2]	hours[3]	hours[4]	hours[5]
↓	↓	↓	↓	↓	↓
20	12	40	30	30	15



## Accessing Array Contents

- 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;
```

UTM | BE CREATIVE & INNOVATIVE



## Using a Loop to Step Through an Array

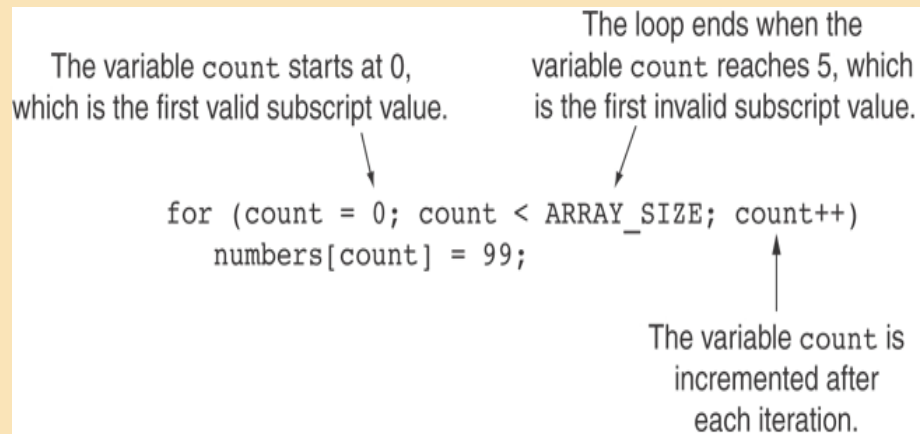
- Example – The following code defines an array, `numbers`, and assigns 99 to each element:

```
const int ARRAY_SIZE = 5;  
int numbers[ARRAY_SIZE];  
  
for (int count = 0; count < ARRAY_SIZE; count++)  
    numbers[count] = 99;
```

UTM | BE CREATIVE & INNOVATIVE



## A Closer Look At the Loop



UTM: BE CREATIVE & INNOVATIVE



## Default Initialization

- Global array → all elements initialized to 0 by default
- Local array → all elements *uninitialized* by default

UTM: BE CREATIVE & INNOVATIVE





## In-class Exercise

- Do Lab 12, Exercise 1, no. 1 (pg. 172)
- Do Lab 12, Exercise 1, No. 2 (pg. 172)

UTM: MORE CREATIVE & INNOVATIVE CAMPUS



## No Bounds Checking in C++

- 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.

UTM: MORE CREATIVE & INNOVATIVE CAMPUS



## Example

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

```

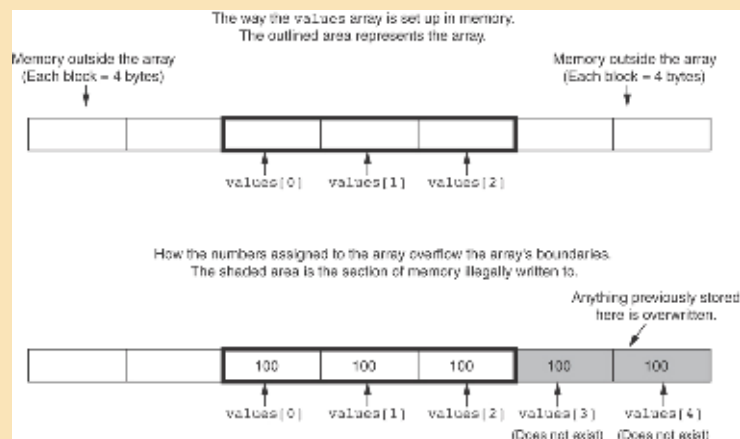
9   const int SIZE = 3; // Constant for the array size
10  int values[SIZE];   // An array of 3 integers
11  int count;          // Loop counter variable
12
13  // Attempt to store five numbers in the three-element array.
14  cout << "I will store 5 numbers in a 3 element array!\n";
15  for (count =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

```

UTM: MULTICREATIVE & INNOVATIVE CAMPUS



## What the Code Does



UTM: MULTICREATIVE & INNOVATIVE CAMPUS



## No Bounds Checking in C++

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

UTM: MORE CREATIVE & INNOVATIVE CAMPUS



## Array Initialization

- Arrays can be initialized with an initialization list:

```
const int SIZE = 5;  
int tests[SIZE] = {79, 82, 91, 77, 84};
```

- The values are stored in the array in the order in which they appear in the list.
- The initialization list cannot exceed the array size.

UTM: MORE CREATIVE & INNOVATIVE CAMPUS



## Example

```

7   const int MONTHS = 12;
8   int days[MONTHS] = { 31, 28, 31, 30,
9                        31, 30, 31, 31,
10                       30, 31, 30, 31};
11
12   for (int count = 0; count < MONTHS; count++)
13   {
14       cout << "Month " << (count + 1) << " has ";
15       cout << days[count] << " days.\n";
16   }

```

### Program Output

```

Month 1 has 31 days.
Month 2 has 28 days.
Month 3 has 31 days.
Month 4 has 30 days.
Month 5 has 31 days.
Month 6 has 30 days.
Month 7 has 31 days.
Month 8 has 31 days.
Month 9 has 30 days.
Month 10 has 31 days.
Month 11 has 30 days.
Month 12 has 31 days.

```

UTM: MULTICREATIVE & INNOVATIVE CAMPUS



## Array Initialization

- Valid

```
int tests[3] = { 3, 5, 11 };
```

- Invalid

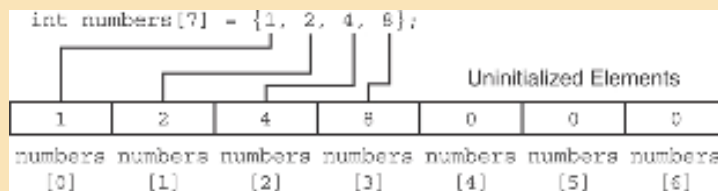
```
int tests[3];
tests= { 3, 5, 11 };
```

UTM: MULTICREATIVE & INNOVATIVE CAMPUS



## Partial Array Initialization

- If array is initialized with fewer initial values than the size declarator, the remaining elements will be set to 0 :



UTM: BE CREATIVE & INNOVATIVE



## Implicit Array Sizing

- Can determine array size by the size of the initialization list:

```
int quizzes[]={12,17,15,11};
```

12	17	15	11
----	----	----	----

- Must use either array size declarator or initialization list at array definition

UTM: BE CREATIVE & INNOVATIVE



## Initializing With a String

- Character array can be initialized by enclosing string in " ":

```
const int SIZE = 6;
char fName[SIZE] = "Henry";
```

- Must leave room for \0 at end of array
- If initializing character-by-character, must add in \0 explicitly:

```
char fName[SIZE] =
{ 'H', 'e', 'n', 'r', 'y', '\0' };
```

UTM: MULTICREATIVE & INNOVATIVE CAMPUS



## In-Class Exercise

- Are each of the following valid or invalid array definitions? (If a definition is invalid, explain why)

```
int numbers[10] = {0, 0, 1, 0, 0, 1, 0, 0, 1, 1};
int matrix[5] = {1, 2, 3, 4, 5, 6, 7};
double radix[10] = {3.2, 4.7};
int table[7] = {2, , , 27, , 45, 39};
char codes [] = {'A', 'X', '1', '2', 's'};
int blanks[];
char name[6] = "Joanne";
```

UTM: MULTICREATIVE & INNOVATIVE CAMPUS



## Processing Array Contents

- Array elements can be treated as ordinary variables of the same type as the array
- When using ++, -- operators, don't confuse the element with the subscript:

```
tests[i]++; // add 1 to tests[i]
tests[i++]; // increment i, no
            // effect on tests
```

UTM MALAYSIA: CREATIVE & INNOVATIVE CAMPUS



## Array Assignment

To copy one array to another,

- Don't try to assign one array to the other:

```
newTests = tests; // Won't work
```

- Instead, assign element-by-element:

```
for (i = 0; i < ARRAY_SIZE; i++)
    newTests[i] = tests[i];
```

UTM MALAYSIA: CREATIVE & INNOVATIVE CAMPUS



## Printing the Contents of an Array

- You can display the contents of a *character* array by sending its name to cout:

```
char fName[] = "Henry";  
cout << fName << endl;
```

But, this **ONLY** works with character arrays!

UTM MALAYSIA CREATIVE & INNOVATIVE CAMPUS



## Printing the Contents of an Array

- For other types of arrays, you must print element-by-element:

```
for (i = 0; i < ARRAY_SIZE; i++)  
    cout << tests[i] << endl;
```

UTM MALAYSIA CREATIVE & INNOVATIVE CAMPUS





## Summing and Averaging Array Elements

- Use a simple loop to add together array elements:

```
int tnum;
double average, sum = 0;
for(tnum = 0; tnum < SIZE; tnum++)
    sum += tests[tnum];
```

- Once summed, can compute average:

```
average = sum / SIZE;
```

UTM: BE CREATIVE & INNOVATIVE



## Finding the Highest Value in an Array

```
int count;
int highest;
highest = numbers[0];
for (count = 1; count < SIZE; count++)
{
    if (numbers[count] > highest)
        highest = numbers[count];
}
```

When this code is finished, the `highest` variable will contain the highest value in the `numbers` array.

UTM: BE CREATIVE & INNOVATIVE



## Finding the Lowest Value in an Array

```
int count;  
int lowest;  
lowest = numbers[0];  
for (count = 1; count < SIZE; count++)  
{  
    if (numbers[count] < lowest)  
        lowest = numbers[count];  
}
```

When this code is finished, the `lowest` variable will contain the lowest value in the `numbers` array.

UTM MALAYSIA: CREATIVE & INNOVATIVE CAMPUS



## Partially-Filled Arrays

- If it is unknown how much data an array will be holding:
  - Make the array large enough to hold the largest expected number of elements.
  - Use a counter variable to keep track of the number of items stored in the array.

UTM MALAYSIA: CREATIVE & INNOVATIVE CAMPUS



## Comparing Arrays

- To compare two arrays, you must compare element-by-element:

```
const int SIZE = 5;
int firstArray[SIZE] = { 5, 10, 15, 20, 25 };
int secondArray[SIZE] = { 5, 10, 15, 20, 25 };
bool arraysEqual = true; // Flag variable
int count = 0;           // Loop counter variable
// Compare the two arrays.
while (arraysEqual && count < SIZE)
{
    if (firstArray[count] != secondArray[count])
        arraysEqual = false;
    count++;
}
if (arraysEqual)
    cout << "The arrays are equal.\n";
else
    cout << "The arrays are not equal.\n";
```

UTM UNIVERSITY



## In-Class Exercise

- Given the following array definition:

```
int values[] = {2,6,10,14};
```

What does each of the following display?

- `cout<<values[2];`
- `cout<<++values[0];`
- `cout<< values[1]++;`
- `x = 2;`  
`cout<<values[++x];`

UTM UNIVERSITY



### In-Class Exercise

- Do Lab 12, Exercise 2, No. 1 (pg. 176)
- Do Lab 12, Exercise 2, No. 2 (pg. 176)
- Do Lab 12, Exercise 2, No. 3 (pg. 177)
- Do Lab 12, Exercise 2, No. 4 (pg. 178)
- Declare an integer array named `names` with 20 elements. Write a loop that prints each element of the array.

UTM MALAYSIA: CREATIVE & INNOVATIVE CAMPUS



### In-Class Exercise

- Write a program that lets the user enter 10 values into an array. The program should then display the largest and smallest values stored in the array.
- Write a program that lets the user enter the total rainfall for each of 12 months into an array of doubles. The program should then calculate and display the total rainfall for the year, the average monthly rainfall, and the months with the highest and lowest amounts.

*Input Validation: Do not accept negative numbers for monthly rainfall figures.*

UTM MALAYSIA: CREATIVE & INNOVATIVE CAMPUS