Introduction to Computing

Lecture 10

Dr. Naveed Anwar Bhatti

Webpage: naveedanwarbhatti.github.io

Arrays

Arrays

Definition:

An **array** is a collection of variables of the **same type** that are referred to by a common name.

Arrays offer a convenient means of grouping together several related variables, in one dimension (or more dimensions).

Arrays

Types of arrays:

- One-dimensional array
- Two-dimensional array
- Multi-dimensional array

One-Dimensional Arrays

Definition:

- A one-dimensional array is a single list of related variables.
- The general form of a one-dimensional array declaration is:

```
type variable_name[size]
```

  - **type**: base type of the array, determines the data type of each element in the array
  - **size**: how many elements the array will hold
  - **variable_name**: the name of the array

One-Dimensional Arrays

Examples:

```
int sample[10];
float float_numbers[100];
char last_name[40];
```

One-Dimensional Arrays – Initialization

The general form of array initialization is similar to that of other variables:

```
type variable_name[size] = { list-of-values };
```

The list-of-values has to be a comma-separated list of constants that are type-compatible with the base type of the array.

In the following example, a 10-element int array is initialized with the numbers 1 through 10:

```
int i[10] = {1,2,3,4,5,6,7,8,9,10};
```

One-Dimensional Arrays – Initialization

Some more examples:

```
float i[10] = {1.1, 2, 3.5, 4, 5, 6, 7, 8, 9.1, 10};

int quiz_marks[5] = {10, 5, 2, 0, 7};
```

One-Dimensional Arrays – Accessing Array Elements

- An individual element within an array is accessed by use of an **index**.
- An **index** describes the position of an element within an array.

Note: In C++ the first element has the index zero!

array-name[index] ;

- In C++, any array is mapped to a contiguous memory location.
- All memory elements reside next to each other.
- The lowest address corresponds to the first element, and the highest address to the last element.

One-Dimensional Arrays – Memory Mapping

For Example:

```
int a[10] = { 1,2,3,4,5,6,7,8,9,10 };
```

a[0]	1
a[1]	2
a[2]	3
a[3]	4
a[4]	5
a[5]	6
a[6]	7
a[7]	8
a[8]	9
a[9]	10

One-Dimensional Arrays – Accessing Array Elements

Accessing Array Elements:

```
#include <iostream>
using namespace std;

int main()
{
    int a[10] = { 1,2,3,4,5,6,7,8,9,10 };

    cout << a[0] << endl;
    cout << a[1] << endl;
    cout << a[2] << endl;
    cout << a[3] << endl;
    cout << a[4] << endl;
    cout << a[5] << endl;
    cout << a[6] << endl;
    cout << a[7] << endl;
    cout << a[8] << endl;
    cout << a[9] << endl;

    return 0;
}
```

One-Dimensional Arrays – Accessing Array Elements

Accessing Array Elements:

```
#include <iostream>
using namespace std;

int main()
{
    int a[10] = { 1,2,3,4,5,6,7,8,9,10 };

    for (int i = 0; i < 10; i++)
    {
        cout << a[i] << endl;
    }

    return 0;
}
```

One-Dimensional Arrays – Accessing Array Elements

Exercise: Find the Maximum

```
#include <iostream>
using namespace std;

int main()
{
    int a[10] = { 1,10,3,7,5,6,2,11,4,0 };









}
```

One-Dimensional Arrays – Accessing Array Elements

Exercise: Find the Maximum

```
#include <iostream>
using namespace std;

int main()
{
    int a[10] = { 1,10,3,7,5,6,2,11,4,0 };

    int max=0;

    for (int i = 0; i < 10; i++)
    {
        if( max < a[i])
            max= a[i] ;
    }

    cout << max << endl;

    return 0;
}
```

One-Dimensional Arrays – Accessing Array Elements

Example: Initialize the array **sample** with the numbers 0^2 through 9^2

0	1	4	9	16	25	36	49	64	81

One-Dimensional Arrays – Accessing Array Elements

Example: Initialize the array **sample** with the numbers 0^2 through 9^2

```
int main()
{
    int sample[10];
    int t;

    for (int t = 0; t < 10; t++)
    {
        sample[t] = t * t;
    }

    for (int t = 0; t < 10; t++)
    {
        cout << sample[t] << endl;
    }

    return(0);
}
```

One-Dimensional Arrays

No Array-to-Array Assignments

- You cannot assign one array to another in C++.
- The following is illegal:

```
int a[10], b[10];
// do something
// assign all elements of array b to array a
a = b; // error -- illegal
```

- Instead, you have to do the assignments for each element:

```
int i;
// assign all elements of array b to array a
for (i = 0; i < 10; i++)
{
    a[i] = b[i];
}
```

One-Dimensional Arrays

No Bounds Checking

- C++ performs no bounds checking on arrays.

- Nothing will stop you from overrunning the end of an array:

  ➞ You will assign values to some other variables´ data!!!
  ➞ You might even write into a piece of the program code!!!