Advanced Programming

Arrays in C++

Dr. Zeyad Safaa Younus Saffawi

Arrays in C++

- In C++, an **array** is a fundamental **data structure** used to **store a group of elements** that all share the **same data type**.
- An array in C++ is a fixed-size collection of elements of the same type stored sequentially in contiguous memory locations.
- Each element is accessed using an index starting from **zero**.
- Instead of defining many **separate variables**, an **array allows us to store multiple related values** under a **single name**.

For example, instead of declaring multiple variables:

```
int number0, number1, number2, ..., number9
```

It can declare one array:

```
int numbers [10]
```

Each element inside the array behaves like an **individual variable**, but they are organized under **one common identifier.**

Concept of Index

- **Each element** in an array is **identified** by a number called an **index**
- The index always starts from **0**
  - **The first element** → index 0
  - **The second element** → index 1
  - And so on…
- So if an array has **10 elements**, the last index will be **9**.
- This is because arrays use **zero-based indexing**.
- It declare one array variable such as numbers and use **numbers[0], numbers[1], and ..., numbers[9]** to represent individual variables.
- Arrays are stored in **adjacent memory locations**, meaning:
  - All elements are placed next to each other in memory
  - The first element has the lowest memory address
  - The last element has the highest memory address

This arrangement makes arrays efficient for sequential processing and fast access.

One Dimensional Array

- To declare an array in C++, we must specify:
  - The data type of the elements
  - The name of the array
  - The number of elements (size)

- **Syntax:**

type array_Name [array_Size];

Example:

```
int  ar [10];  // This statement creates an array named ar that can store 10 values of type integer.
float  x [15]; // This statement creates an array named x that can store 15 values of type float.
char  ch [25]; // This statement creates an array named ch that can store 25 values of type character.
```

- Important Notes:
  - The **size** must be a **positive integer constant**
  - The **size cannot** be **changed** during program execution
  - This is called a **single-dimensional array**

Initializing an Array

- There are two main ways to **initialize arrays**.

Method 1: Initialize at Declaration
int A [5] = {value list};

int x [5] = {1000.0, 2.0, 3.4, 17.0, 50.0};
Important rule:
The **number** of **values** inside {} must not exceed the **declared size**.

Method 2: Let the Compiler Determine the Size
int x [ ] = {1000.0, 2.0, 3.4, 17.0, 50.0};
In this case, the compiler automatically calculates the size based on the number of values provided.

Assigning Values to Individual Elements

We can also **assign** a **value** to a **specific element** using its **index**.
Example:
balance[4] = 50.0;
This **assigns** the **value 50.0** to the **fifth element**.
Notes:
Index 4 corresponds to the **fifth element** because **indexing starts at 0**.

Accessing Array Elements

- To access an **element**, we write the **array name** followed by **its index** inside **square brackets**.
- Example:
- int salary = x[9];
- This statement **retrieves the tenth element** from the **array** and **stores** it in the **variable salary**.

Reading and writing the array

- Reading the array

```
Int a[10];
For(int i=0 ; i<10 ; i++)
Cin>>a[i];
```

- Writing the array

using for statement

```
For (int i=0 ; i<10 ; i++)
Cout<<a[i];
```

using while statement

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

Practical Example

```
#include <iostream>
using namespace std;
int main()
{
   int n[10];                                  // declare array of 10 integers

// assign values to array elements, initialize elements with 0
      for(int i = 0; i < 10; i++)
   {
      n[i] = i + 20;                           // set element at location i to i + 20
   }
// display array elements
   for(int j = 0; j < 10; j++)
   {
      cout << n[j] << endl;
   }
   return 0;
}
```

Explanation:

1. We declared an array of 10 integers.
2. We used a loop to assign values.
3. We used another loop to print the elements.
4. Each element is accessed using its index.

Practical Example

```
#include <iostream>
using namespace std;
int main()
{
   int a[20];
   int sum = 0;
   float avg;
   // Read array elements
   for(int i = 0; i < 20; i++)
   {
      cin >> a[i];
   }

   // Calculate sum
   for(int i = 0; i < 20; i++)
   {
      sum = sum + a[i];
   }
   // Calculate average
   avg = sum / 20;
   // Display results
   cout << "Sum = " << sum << endl;
   cout << "Average = " << avg << endl;
   return 0;
}
```

Write a C++ program to read an array of 20 numbers, then calculate and display the summation and the average of all elements.

```
// Read array elements and Calculate sum
   for(int i = 0; i < 20; i++)
   {
      cin >> a[i];
      sum = sum + a[i];
   }
avg = sum / 20;
```

Practical Example

Write a C++ program to read an array of 10 lowercase letters (characters), then convert each character to uppercase and display the result.

```
#include <iostream>
using namespace std;
int main()
{
   char ch[10];
   // Read 10 lowercase characters from keyboard
   for(int i = 0; i < 10; i++)
   {
      cin >> ch[i];
   }
   // Convert to uppercase and display
   for(int i = 0; i < 10; i++)
   {
      ch[i] = ch[i] - 32;                    // ASCII conversion
      cout << ch[i] << " ";
   }
   return 0;
}
```

Practical Example

- Write a C++ program to read an array of 15 integers, then print the elements in reverse order.

```
#include <iostream>
using namespace std;
int main()
{
   int a[15];
   // Read array elements
   for(int i = 0; i < 15; i++)
   {
      cin >> a[i];
   }
   // Print elements in reverse order
   for(int i = 14; i >= 0; i--)
   {
      cout << a[i] << " ";
   }
   return 0;
}
```

Practical Example

Write a C++ program to read an array of 10 integers. For each element, calculate its factorial by calling a function. Then compute and display the sum of all factorials in the main program.

```
#include <iostream>
using namespace std;
// Function to calculate factorial
int fact(int n)
{
   int f = 1;
   for(int i = 1; i <= n; i++)
   {
      f = f * i;
   }
   return f;
}
int main()
{
   int a[10];
   int sum = 0;
   int f;
   // Read array elements and calculate factorial
   for(int i = 0; i < 10; i++)
   {
      cin >> a[i];
      f = fact(a[i]);                    // Call function
      sum = sum + f;                     // Add factorial to sum
      cout << f << " ";
   }
   // Print total sum of factorials
   cout << endl;
   cout << "Sum of factorials = " << sum;
   return 0;
}
```

References

- Gaddis, T. (2014). *Starting out with C++: From control structures through objects* (8th ed.). Pearson.
- Soulié, J. (2007, April 24). C++ language tutorial. cplusplus.com.
- Tutorials Point. (n.d.). *Learn C++ programming language*. Tutorials Point.