Lecture 6: C++ Arrays

C++ provides a data structure, the array, which stores a fixed-size sequential collection of elements of the same type. An array is used to store a collection of data, but it is often more useful to think of an array as a collection of variables of the same type.

Instead of declaring individual variables, such as number 0, number1, ..., and number99, you declare one array variable such as numbers and use numbers [0], numbers [1], and ..., numbers[99] to represent individual variables. A specific element in an array is accessed by an index. All arrays consist of contiguous memory locations. The lowest address corresponds to the first element and the highest address to the last element.

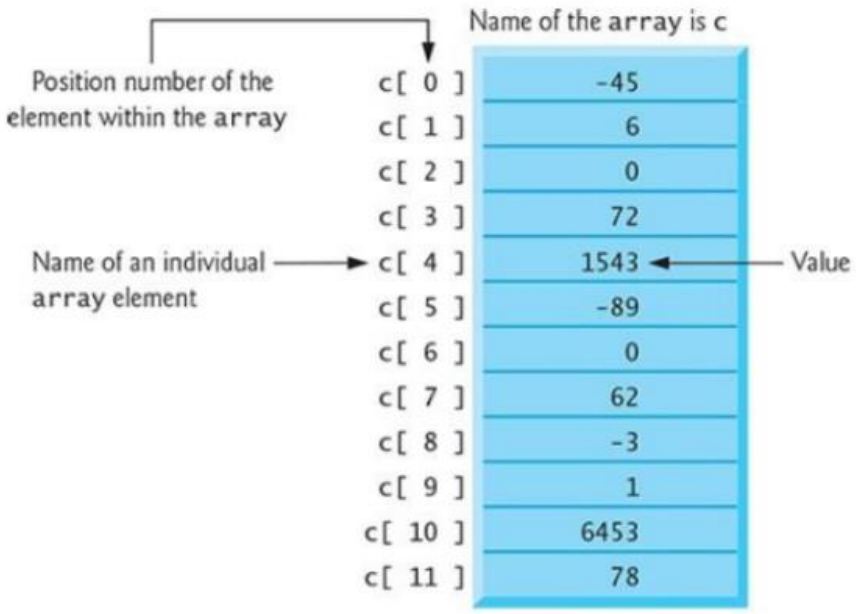


Figure 1: An array of 12 elements

You refer to any one of these elements by giving the array name followed by the particular element's position number in square brackets ([]).

• The position number is more formally called a subscript or index (this number specifies the number of elements from the beginning of the array).

• The first element has subscript 0 (zero) and is sometimes called the zeroth element.

• Thus, the elements of array c are c[0] (pronounced " c of zero "), c[1] , c[2]

and so on.

• Array names follow the same conventions as other variable names.

- **Declaring Arrays:** To declare an array in C++, the programmer specifies the type of the elements and the number of elements required by an array as follows:

Syntax

type arrayName [arraySize];

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

- product part numbers:

int part_numbers[4] = {123, 326, 178, 1209};

- student scores:

int scores[10] = {1, 3, 4, 5, 1, 3, 2, 3, 4, 4};

- characters:

char alphabet[5] = {'A', 'B', 'C', 'D', 'E'};

- names:

char names[5] = {"Peter", "Mary", "Lisa", "John", "George-Simon"};

One-Dimensional Arrays

A one-dimensional array is a 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

Examples:

```
int sample[10];

float float_numbers[100];

char last_name[40];
```

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!

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

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

int main()
{
int sample[10]; // reserves for 10 integers
int t;
// initialize the array
for(t=0; t<10; ++t) sample[t] = t*t;
// display the array
for(t=0; t<10; ++t)
cout << sample[t] << ' ';
return(0);
}
```

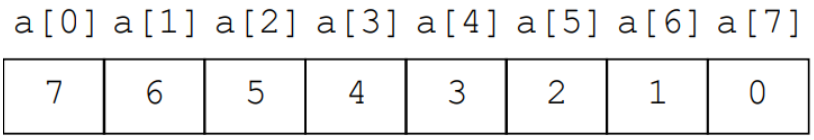
Representation of Arrays in Memory

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.

Example:

```
int a[8];
 int j;
 for(j=0; j<8; j++) a[j] = 7-j;
```

Then the memory representation of the array looks like this:



Change an Array Element

To change the value of a specific element, refer to the index number:

Example

```
string cars[4] = {"Volvo", "BMW", "Ford", "Mazda"};
cars[0] = "Opel";
cout << cars[0];
// Now outputs Opel instead of Volvo
```

```
#include <iostream>

using namespace std;

int main()
{int b [] = {11, 45, 62, 70, 88};
 cout << b[0] << endl;

  //Outputs 11
   cout << b[3] << endl;
}
```