# Programming In C++

Course 2: Lecture 6, Arrays and functions

Prepared By Dr. Ali Mohsin

Nineveh University- Faculty of IT- department of software

# Arrays as Function Arguments

To pass an array as an argument to a function, pass the name of the array.

The function can accept an array in two ways:

1- By sending the name of the function from the main

Int a[5];
showValues(a);

In the function we can write,
showValues(int a[5]);

```
1   #include <iostream>
2
3   using namespace std;
4
5   float sum(int a[5]){
6   float s;
7   for(int i=0;i<5;i++)
8      s+=a[i];
9
10  return s;
11  }
12
13  int main()
14  {
15     int a[5]={10,20,30,40,50};
16     cout<<sum(a);
17
18     return 0;
19  }
```

# Arrays as Function Arguments

When an entire array is passed to a function, it is not passed by value, but passed by reference.

It means any changing in the array inside the array, it can be seen by the main function.

The name of the array considers the address of that array.

# Arrays as Function Arguments

2- The second way of sending an array to function

The first parameter is followed by an empty set of brackets. The second argument indicates the size of the array.

```
void showValues(int nums[], int size)
{
    for (int index = 0; index < size; index++)
        cout << nums[index] << " ";
    cout << endl;
}
```

```
int main()
{
    const int ARRAY_SIZE = 8;
    int numbers[ARRAY_SIZE] = {5, 10, 15, 20, 25, 30, 35, 40};

    showValues(numbers, ARRAY_SIZE);
    return 0;
}
```

# Arrays as Function Arguments

The second way is more efficient because the same function can work with many arrays in different sizes.

So, we can use the showValues function to display the contents of any integer array by passing the name of the array and its size as Arguments to display the contents of two different arrays.

# Arrays as Function Arguments

```
void showValues(int nums[], int size)
{
     for (int index = 0; index < size; index++)
          cout << nums[index] << " ";
     cout << endl;
}


int main()
{
     const int SIZE1 = 8; // Size of set1 array
     const int SIZE2 = 5; // Size of set2 array
     int set1[SIZE1] = {5, 10, 15, 20, 25, 30, 35, 40};
     int set2[SIZE2] = {2, 4, 6, 8, 10};

     // Pass set1 to showValues.
     showValues(set1, SIZE1);

     // Pass set2 to showValues.
     showValues(set2, SIZE2);
     return 0;
}
```

# Passing 2-D array to functions

Similarly with the 1-D array, there are two ways of passing 2-D array to function.

1- sending the name of the array (similar way that is used in 1-D array)

2- sending the name of the array followed by two brackets (the first bracket is empty and the second one has the number of columns).

When a two-dimensional array is passed to a function, the parameter type must contain a size declarator for the number of columns.

```
void showArray(const int numbers[][COLS], int rows)
```

COLS is a global named constant

# Passing 2-D array to functions

C++ requires the columns to be specified in the function prototype and header because of the way two-dimensional arrays are stored in memory. One row follows another, as shown in the following Figure

When the compiler generates code for accessing the elements of a two-dimensional array, it needs to know how many bytes separate the rows in memory.

# Passing 2-D array to functions

Example of using string with function.

A function to count the occurrence of given character in a string.

# The End