

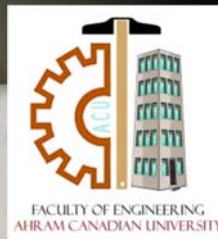


# Lecture (07) Arrays

By:

Dr. Ahmed ElShafee

Dr. Ahmed ElShafee, ACU : Fall 2015, Programming I



## introduction

- 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 number0, 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.

٢

Dr. Ahmed ElShafee, ACU : Fall 2015, Programming I

## Declaring Arrays

```
type arrayName [ arraySize ];
```

This is called a single-dimension array. The **arraySize** must be an integer constant greater than zero and **type**

```
double balance[10];
```

## Initializing Arrays

```
double balance[5] = {1000.0, 2.0, 3.4, 17.0, 50.0};
```

The number of values between braces { } can not be larger than the number of elements that we declare

```
double balance[] = {1000.0, 2.0, 3.4, 17.0, 50.0};
```

You will create exactly the same array as you did

```
balance[4] = 50.0;
```

statement assigns element number 5th in the array a value of 50.0

	0	1	2	3	4
balance	1000.0	2.0	3.4	7.0	50.0

٣

Dr. Ahmed ElShafee, ACU : Fall 2015, Programming I

٤

Dr. Ahmed ElShafee, ACU : Fall 2015, Programming I

# Accessing Array Elements

```
double salary = balance[9];
```

```
Cout<< "balance[10];
```

# Example 01

```
#include "stdafx.h"
#include <iostream>
using namespace std;

int _tmain(int argc, _TCHAR* argv[])
{
    int n[ 10 ];
    for ( int i = 0; i < 10; i++ )
    {
        n[ i ] = i + 100;
    }
    cout << "Element" << "\t\t" << "Value" << endl;
    for ( int j = 0; j < 10; j++ )
    {
        cout << j << "\t\t" << n[ j ] << endl;
    }
    cin.get();
    return 0;
}
```

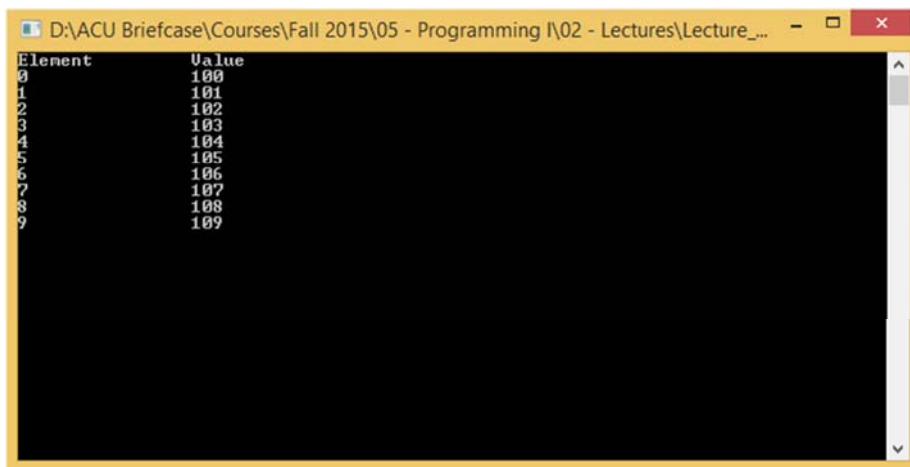
# Multi-dimensional arrays

the general form

```
type name[size1][size2]...[sizeN];
```

three dimensional

```
int threedim[5][10][4];
```



# Two-Dimensional Arrays

The simplest form of the multidimensional array

```
type arrayName [ x ][ y ];
```

A two-dimensional array can be think as a table

	Column 0	Column 1	Column 2	Column 3
Row 0	a[ 0 ][ 0 ]	a[ 0 ][ 1 ]	a[ 0 ][ 2 ]	a[ 0 ][ 3 ]
Row 1	a[ 1 ][ 0 ]	a[ 1 ][ 1 ]	a[ 1 ][ 2 ]	a[ 1 ][ 3 ]
Row 2	a[ 2 ][ 0 ]	a[ 2 ][ 1 ]	a[ 2 ][ 2 ]	a[ 2 ][ 3 ]

9

Dr. Ahmed ElShafee, ACU : Fall 2015, Programming I

# Initializing Two-Dimensional Arrays

initialized by specifying bracketed values for each row

```
int a[3][4] = {  
{0, 1, 2, 3}, /* initializers for row indexed by 0 */  
{4, 5, 6, 7}, /* initializers for row indexed by 1 */  
{8, 9, 10, 11} /* initializers for row indexed by 2 */  
};
```

infested braces, which indicate the intended row, are optional.

The following initialization is equivalent to previous example

```
int a[3][4] = {0,1,2,3,4,5,6,7,8,9,10,11};
```

10

Dr. Ahmed ElShafee, ACU : Fall 2015, Programming I

# Accessing Two-Dimensional Array Elements

Using row index and column index of the array

```
int val = a[2][3];
```

11

Dr. Ahmed ElShafee, ACU : Fall 2015, Programming I

# Example 02

```
#include "stdafx.h"  
#include <iostream>  
using namespace std;  
int _tmain(int argc, _TCHAR* argv[])  
{  
    int a[5][2] = { {0,0}, {1,2}, {2,4}, {3,6},{4,8}};  
    cout << "[*]\t[0]\t[1]"<<endl;  
    for ( int i = 0; i < 5; i++ )  
    {  
        cout<< "["<<i<< "]:\t";  
        for ( int j = 0; j < 2; j++ )  
        {  
            cout << " "<<a[i][j]<< "\t";  
        }  
        cout<<endl;  
    }  
    cin.get();  
    return 0;  
}
```

# Pointer to an array

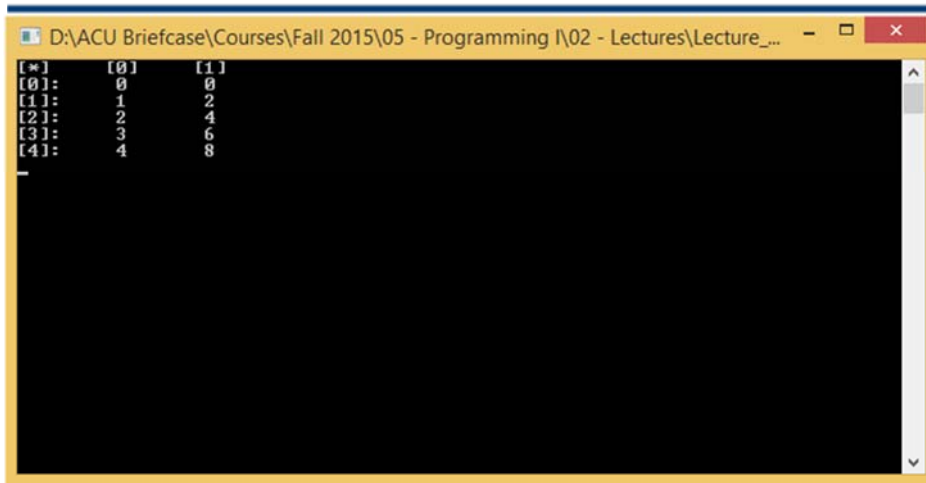
- An array name is a constant pointer to the first element of the array.

```
double balance[50];
```

- balance** is a pointer to &balance[0], which is the address of the first element of the array balance

```
double *p;  
double balance[10];  
p = balance;
```

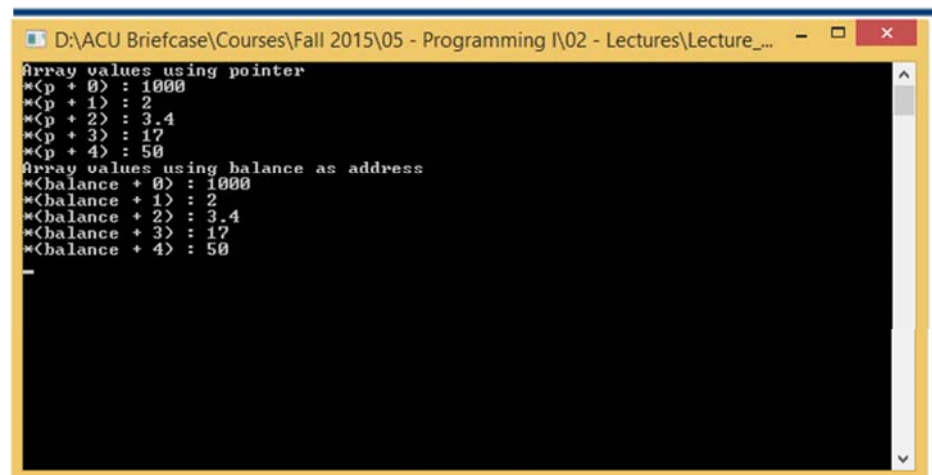
- Therefore, \*(balance + 4) is a legitimate way of accessing the data at balance[4].



```
D:\ACU Briefcase\Courses\Fall 2015\05 - Programming I\02 - Lectures\Lecture_...  
[*] [0] [1]  
[0]: 0 0  
[1]: 1 2  
[2]: 2 4  
[3]: 3 6  
[4]: 4 8
```

## Example 03

```
#include "stdafx.h"  
#include <iostream>  
using namespace std;  
  
int _tmain(int argc, _TCHAR* argv[])  
{  
    double balance[5] = {1000.0, 2.0, 3.4, 17.0, 50.0};  
    double *p;  
    p = balance;  
    cout << "Array values using pointer " << endl;  
    for ( int i = 0; i < 5; i++ )  
    {  
        cout << "*(p + " << i << ") : ";  
        cout << *(p + i) << endl;  
    }  
    cout << "Array values using balance as address " << endl;  
    for ( int i = 0; i < 5; i++ )  
    {  
        cout << "*(balance + " << i << ") : ";  
        cout << *(balance + i) << endl;  
    }  
    cin.get();  
    return 0;  
}
```



```
D:\ACU Briefcase\Courses\Fall 2015\05 - Programming I\02 - Lectures\Lecture_...  
Array values using pointer  
*(p + 0) : 1000  
*(p + 1) : 2  
*(p + 2) : 3.4  
*(p + 3) : 17  
*(p + 4) : 50  
Array values using balance as address  
*(balance + 0) : 1000  
*(balance + 1) : 2  
*(balance + 2) : 3.4  
*(balance + 3) : 17  
*(balance + 4) : 50
```

# Passing arrays to functions

- C++ does not allow to pass an entire array as an argument to a function. However, You can pass a pointer to an array by specifying the array's name without an index.

## Way-1

```
void myFunction(int *param)
{
    .
    .
    .
}
```

## Way-2

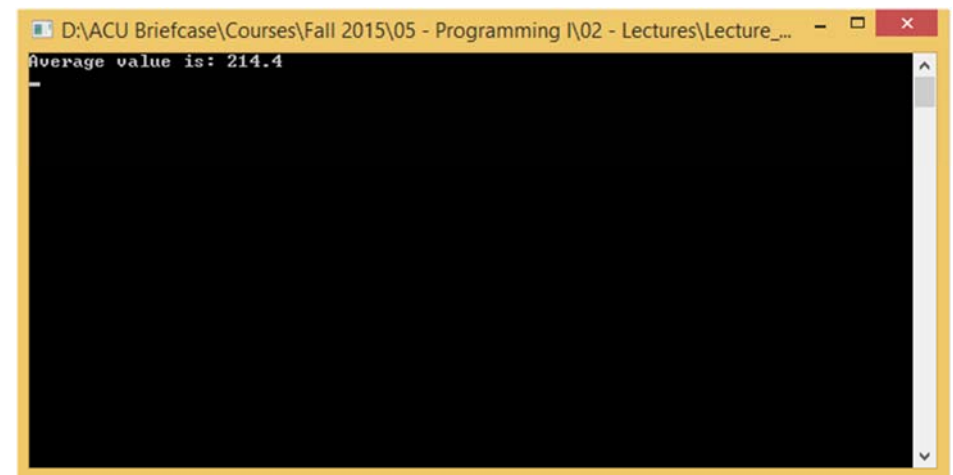
```
void myFunction(int param[10])
{
    .
    .
    .
}
```

## Way-3

```
void myFunction(int param[])
{
    .
    .
    .
}
```

# Example 04

```
#include "stdafx.h"
#include <iostream>
using namespace std;
double getAverage(int arr[], int size);
int _tmain(int argc, _TCHAR* argv[])
{
    int balance[5] = {1000, 2, 3, 17, 50};
    double avg;
    avg = getAverage( balance, 5 );
    cout << "Average value is: " << avg << endl;
    cin.get();
    return 0;
}
double getAverage(int arr[], int size)
{
    int i, sum = 0;
    double avg;
    for (i = 0; i < size; ++i)
    {
        sum += arr[i];
    }
    avg = double(sum) / size;
    return avg;
}
```



## Return array from functions

- If you want to return a single-dimension array from a function, you would have to declare a function returning a pointer

```
int * myFunction()
{
.
.
.
}
```

- C++ does not advocate to return the address of a local variable to outside of the function so you would have to define the local variable as **static** variable

Cancelled

## Example 05

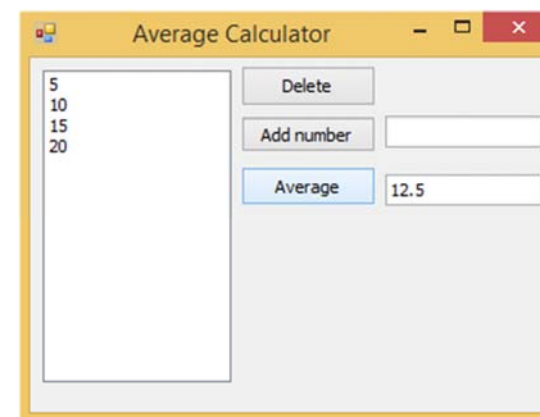
```
#include "stdafx.h"
#include <iostream>
#include <ctime>
using namespace std;
int * getRandom()
{
    static int r[10];
    /* initialize random seed: */
    srand(time( NULL ) );
    for (int i = 0; i < 10; ++i)
    {
        r[i] = rand();
        cout << r[i] << endl;
    }
    cin.get();
    return r;
}

int _tmain(int argc, _TCHAR* argv[])
{
    int *p;
    p = getRandom();
    for ( int i = 0; i < 10; i++ )
    {
        cout << *(p + " << i << " : ";
        cout << *(p + i) << endl;
    }
    return 0;
}
```

Cancelled

## Example 06

Build a GUI application that calculates average a number entered by the user as follow



Cancelled

```
double getAverage(double arr[], int size)
```

```
{  
    double avg,sum=0;  
    for (int i = 0; i < size; ++i)  
    {  
        sum += arr[i];  
    }  
    avg = double(sum) / size;  
    return avg;  
}
```

```
private: System::Void button3_Click(System::Object^ sender, System::EventArgs^ e) {  
    int index=listBox1->SelectedIndex ;  
    listBox1->Items->RemoveAt( index );  
}
```


٢٥

Dr. Ahmed ElShafee, ACU : Fall 2015, Programming I

```
} private: System::Void button1_Click(System::Object^ sender, System::EventArgs^ e) {  
    listBox1->Items->Add(textBox1->Text);  
    textBox1->Text="";  
}  
} private: System::Void button2_Click(System::Object^ sender, System::EventArgs^ e) {  
    double nums[100];  
    double result;  
    int length=listBox1->Items->Count;  
    for(int n=0;n<length;n++)  
    {  
        nums[n]=double::Parse(listBox1->Items[n]->ToString());  
        result=getAverage(nums,length);  
    }  
    textBox2->Text=result.ToString();  
}
```

٢٦

Dr. Ahmed ElShafee, ACU : Fall 2015, Programming I



**Thanks,..  
Wish you all the best 😊**

Dr. Ahmed ElShafee, ACU : Fall 2015, Programming I