LEARN PROGRAMMING

STUDY GROUP- SESSION #7

Weekly: Wednesday 19:15 to 22:15
E037 G29

TODAY?

1. Datatypes
2. Arrays
    a. One dimensional
    b. Two dimensional
    c. Three dimensional
3. Examples
4. Task for today

ARRAY

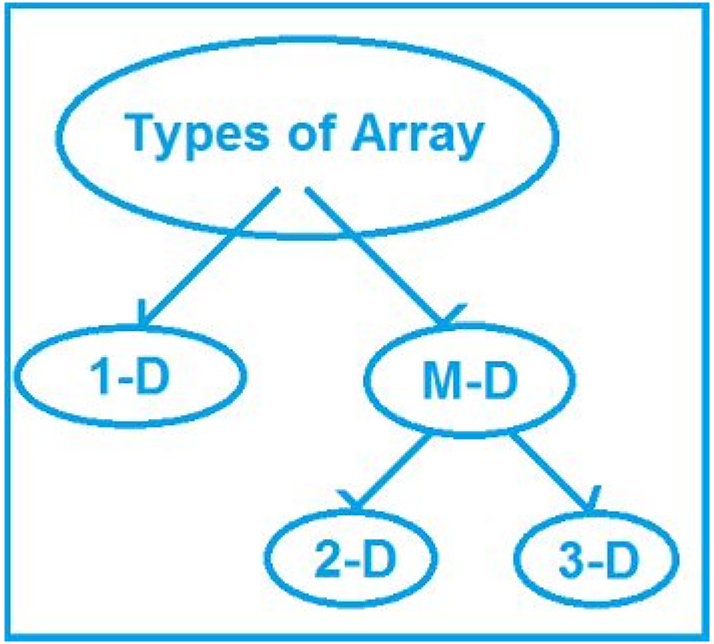
A collection of variable of same data type.

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**.

A specific **element** in an array is accessed by an **index**.

ARRAY Types

- One-dimensional / Single-dimensional
- Two-dimensional
- Three-dimensional...etc

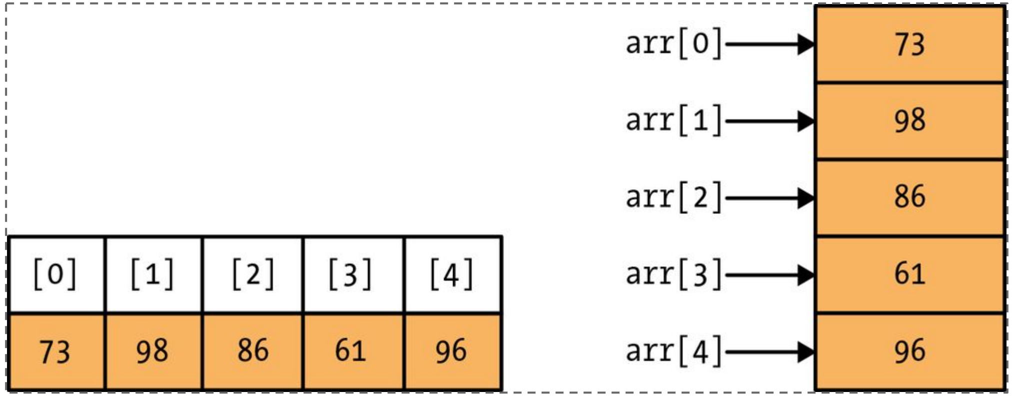


ARRAY TYPES

- One-dimensional

C++ example:

```
<datatype> <ArrayName>[arraySize];
```



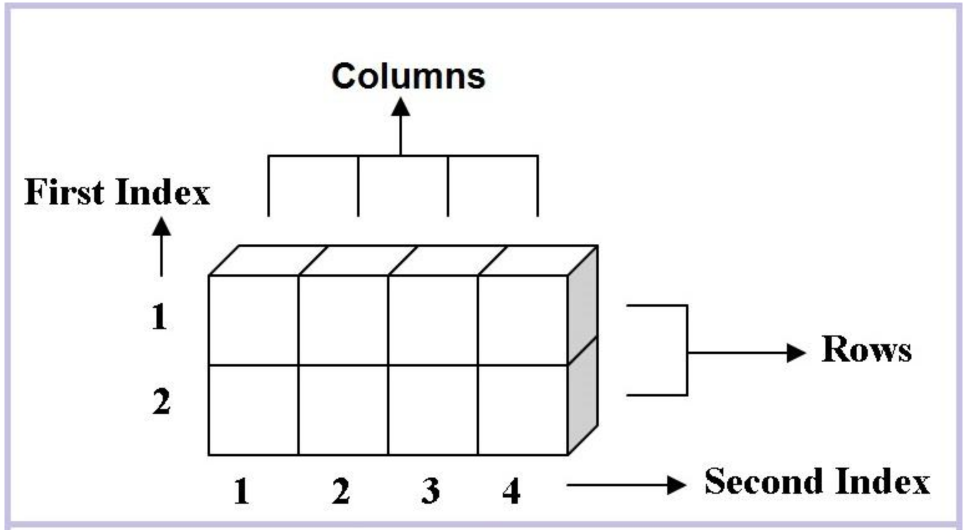
One dimensional Array with 5 elements

ARRAY TYPES

- Two-dimensional

C++ example:

```
<datatype> <ArrayName> [rows][cols];
```



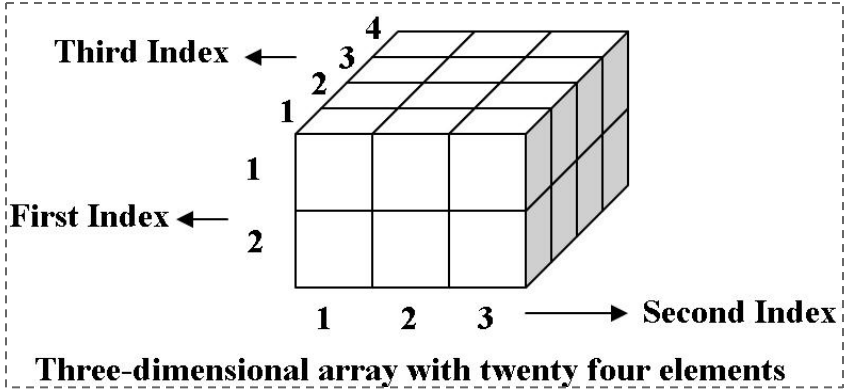
2-D Array with 8 Elements

ARRAY TYPES

- Three-dimensional

C++ example:

```
<datatype> <ArrayName> [index1][index2][index3];
```



Three-dimensional array with twenty four elements

ONE-DIMENSIONAL ARRAY

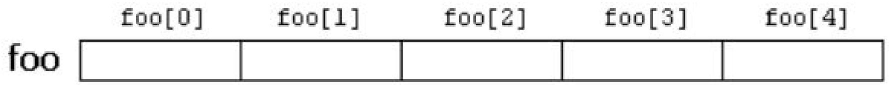
Declaration:

```
int foo [5];
```

A size must be declared given in square brackets, it represents number of elements that this array will hold.

Intiliasing:

```
int foo [5] = { };
```



OR

```
int foo [5] = { 16, 2, 77, 40, 12071}; OR

int foo [] = { 16, 2, 77, 40, 12071 };
```

Accessing Elements:

```
int x = foo[3];
```

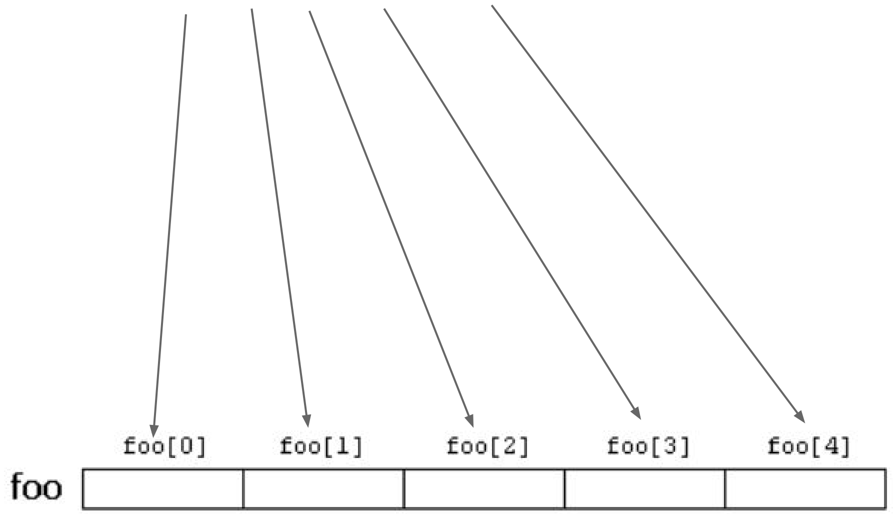
Storing Elements:

```
foo [2] = 75;
```

ONE-DIMENSIONAL ARRAY

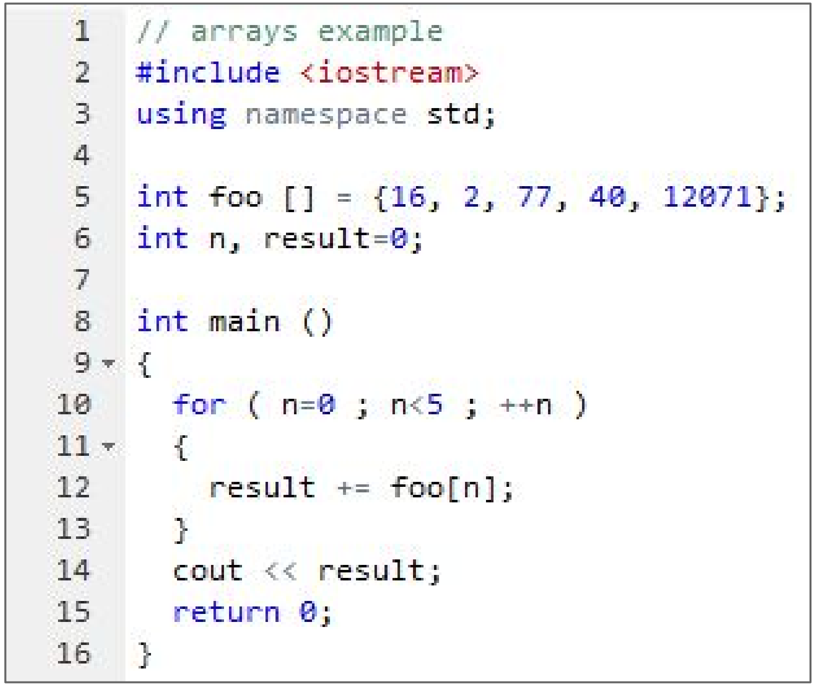
Intiliasing:

```
int foo [5] = { 16, 2, 77, 40, 12071}; OR

int foo [] = { 16, 2, 77, 40, 12071 };
```



ONE-DIMENSIONAL ARRAY

SUM EXAMPLE:



```cpp
// arrays example
#include <iostream>
using namespace std;

int foo [] = {16, 2, 77, 40, 12071};
int n, result=0;

int main ()
{
  for ( n=0 ; n<5 ; ++n )
  {
    result += foo[n];
  }
  cout << result;
  return 0;
}
```

Output:

```
12206
```

TWO-DIMENSIONAL ARRAY

Declaration:

```
int jimmy [3][5];
```

A size must be declared given in square brackets, it represents number of elements that this array will hold. 3 rows and 5 cols

Equivalent:

```
int jimmy [3][5];   // is equivalent to
int jimmy [15];     // (3 * 5 = 15)
```

Intiliasing:

```
int jimmy [3][5] = { };
```

OR

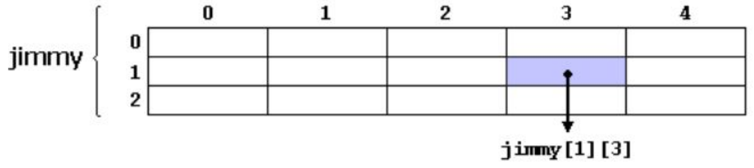
```
int jimmy [3][5] = { 16, 2, 77, 40, 12071,...};  //15 elements OR

int jimmy [][] = { 16, 2, 77, 40, 12071,...}; //15 elements
```

Accessing Elements:

```
int x = jimmy [1][3]; //1st row and 3rd column
```

Storing Elements:

```
jimmy [1][3] = 75;
```



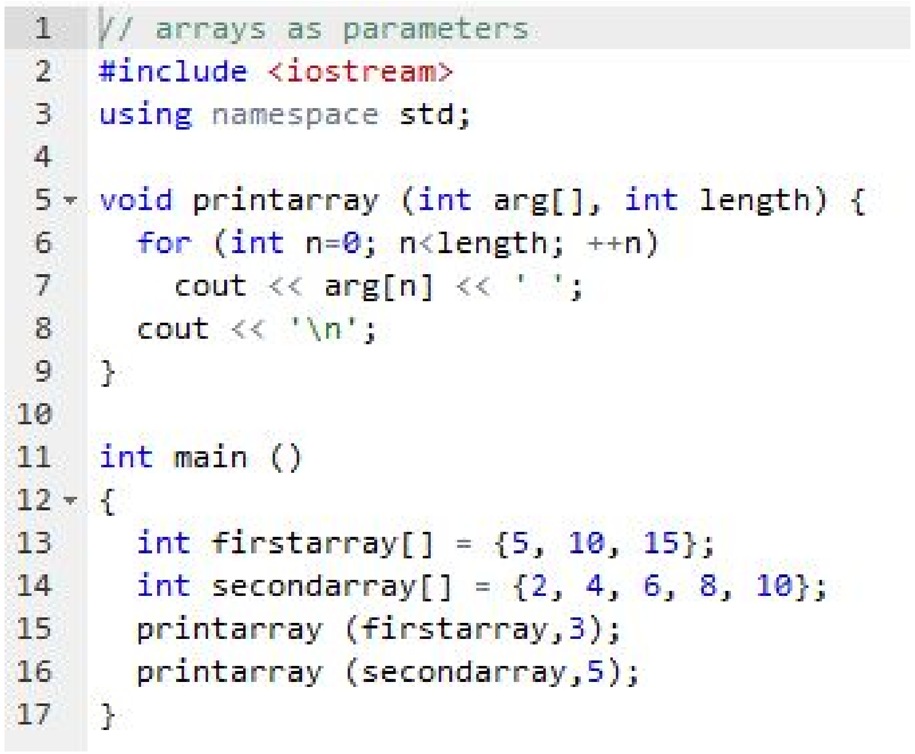
Passing Array

PRINTING ARRAY EXAMPLE:

*Passing array as an argument in function

Output:

```
5,10,15

2,4,6,8,10
```



```cpp
// arrays as parameters
#include <iostream>
using namespace std;

void printarray (int arg[], int length) {
  for (int n=0; n<length; ++n)
    cout << arg[n] << ' ';
  cout << '\n';
}

int main ()
{
  int firstarray[] = {5, 10, 15};
  int secondarray[] = {2, 4, 6, 8, 10};
  printarray (firstarray,3);
  printarray (secondarray,5);
}
```

Task For Today? (USING ARRAYS)

WRITE a C++ program:

1. Read some numbers from the user input and then calculate their sum and average. Print the results.

2. Read student courses grades and calculate its total cgpa and print it.