# fundamentals of programming I

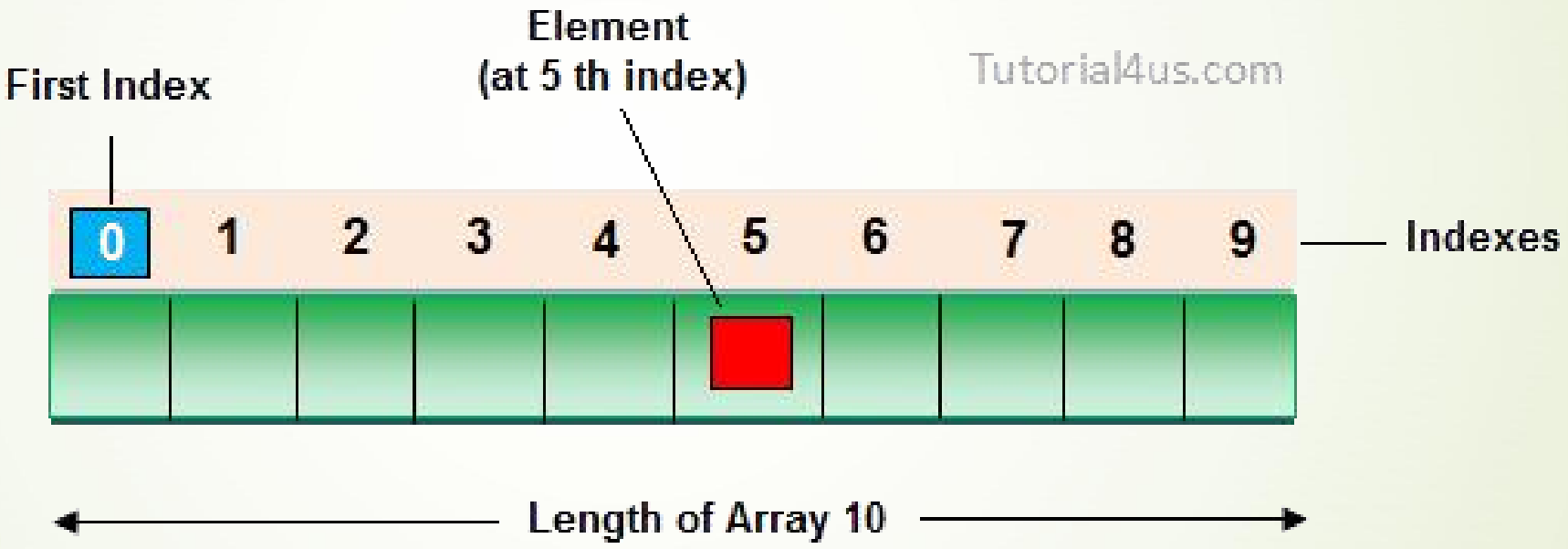
Lab 4

# Arrays

- Arrays are used to store multiple values in a single variable, instead of declaring separate variables for each value and a collection of fixed number of elements, wherein all of elements have same data type.
- Consecutive group of memory locations that all have the same type.
- The collection of data is indexed, or numbered, and at starts at 0 and The highest element index is one less than the total number of elements in the array

- int arr[4]={1,2,3,4,5};
- `string cars[4] = {"Volvo", "BMW", "Ford", "Mazda"};`

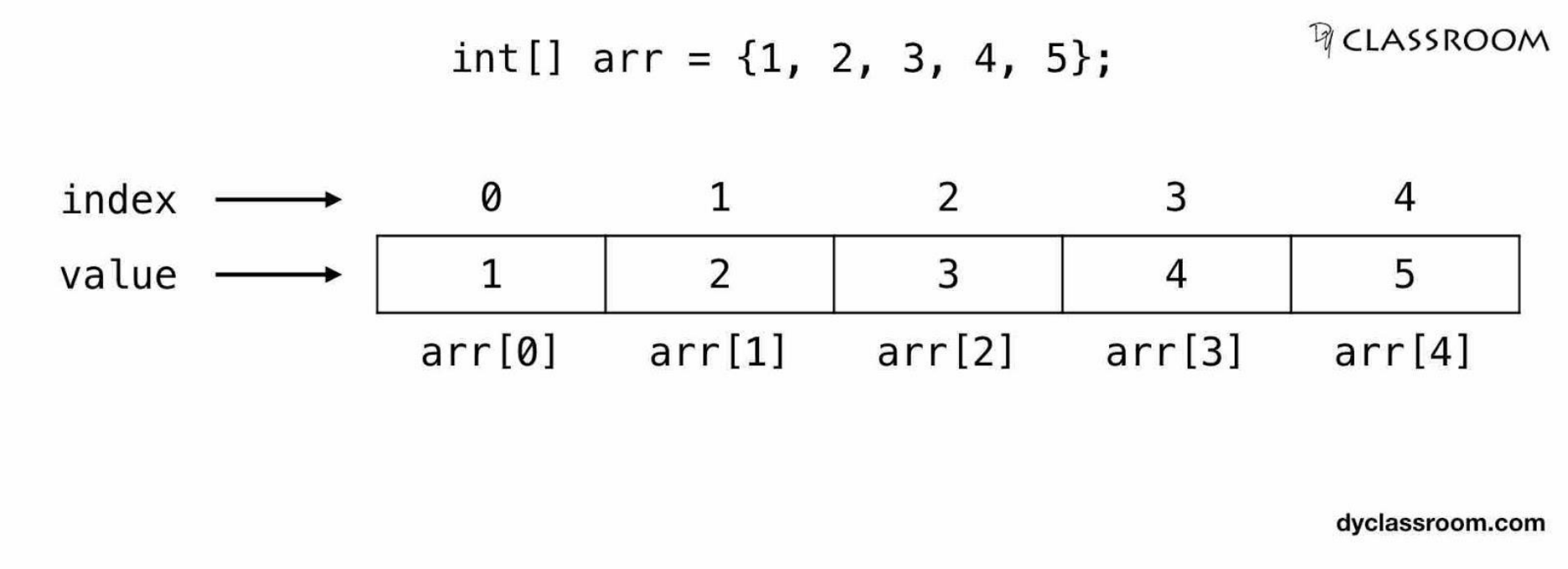
# Arrays


Element
(at 5 th index)
First Index
Tutorial4us.com
0
1
2
3
4
5
6
7
8
9
Indexes
Length of Array 10

# Syntax for declaring a one-dimensional array :

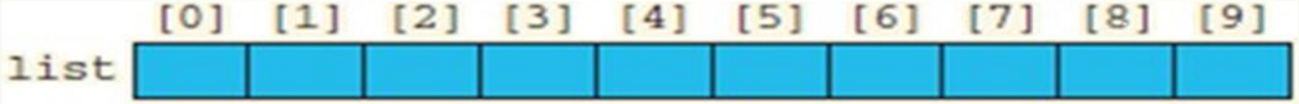
- Datatype ArrayName [ArraySize] ;
- ArraySize: any positive integer or constant variable.
- Example: int num[5];
- Example: const int size = 5 ;
- int num[size];

# Arrays

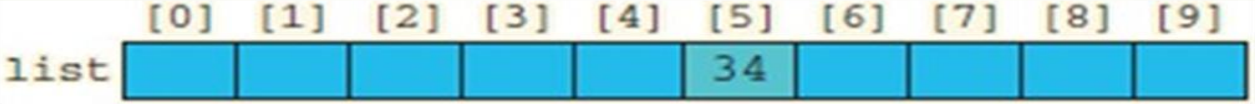

int[] arr = {1, 2, 3, 4, 5};
CLASSROOM
index
value
0
1
2
3
4
1
2
3
4
5
arr[0]
arr[1]
arr[2]
arr[3]
arr[4]
dyclassroom.com

# Accessing array Elements

***Arrayname***[element index].




```
list[5] = 34;
```




```
Cout << list [5];
```

# Array Initialization

- Example 1: int Items[] = {12, 32, 16, 23, 45};
- Example 2: int items [10] = { 0 };
- Example 3: int items [10] = { 5, 7, 10 }
- After declaring the array you can use the For .. Loop to initialize it with values submmited by the user.
- • Using for loops to access array elements:

Example: for (int i = 0; i < 10; i++) cin >> list[i];

# Get the Size of an Array

- To get the size of an array, you can use the sizeof(arrayName) :

Example

```
int myNumbers[5] = {10, 20, 30, 40, 50};
cout << sizeof(myNumbers);
```

Result:

```
20
```

# Get the Size of an Array

- Why did the result show 20  instead of 5 ?
- It is because the sizeof() operator returns the size of a type in bytes.
- You learned from the Data Types chapter that an int type is usually 4 bytes, so from the example above, *4 x 5 (4 bytes x 5 elements) = 20 bytes*.
- To find out how many elements an array has, you have to divide the size of the array by the size of the data type it contains

Example

```
int myNumbers[5] = {10, 20, 30, 40, 50};
int getArrayLength = sizeof(myNumbers) / sizeof(int);
cout << getArrayLength;
```

Result:

```
5
```

# Multi-Dimensional Arrays

- Used when data is provided in a table form.
- For Example , to store 4 Marks for 6 students.

| | M 1 | M2 | M3 | M4 |
|---|---|---|---|---|
| Student 1 | | | | |
| Student 2 | | | | |
| Student 3 | | | | |
| Student 4 | | | | |
| Student 5 | | | | |
| Student 6 | | | | |

# Multi-Dimensional Arrays

- Two dimensional Array declaration
- Datatype ArrayName [ Rows] [Columns] ;
- Example : Float marks [6] [4] ;
- Two dimensional Array intialization
- Marks[4][2]=20;

- Using 2 nested for loops to access array elements:
- for (int row = 0; row < 6; row++)

```
        for (int col = 0; col < 4; col++)
                cin >> marks[ row ][col];
```

# Multi-Dimensional Arrays

```
string letters[2][4] = {
  { "A", "B", "C", "D" },
  { "E", "F", "G", "H" }
};
```

# Example

- Take Inputs from User and Store Them in an Array

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

int main() {

  int numbers[5];

  cout << "Enter 5 numbers: " << endl;

  //  store input from user to array
  for (int i = 0; i < 5; ++i) {
    cin >> numbers[i];
  }

  cout << "The numbers are: ";

  //  print array elements
  for (int n = 0; n < 5; ++n) {
    cout << numbers[n] << "  ";
  }

  return 0;
}
```

# References

- A reference variable is a "reference" to an existing variable, and it is created with the & operator

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

int main() {

 string food ="pizza";
 string &meal=food;
 cout<<food<<endl;
 cout<<&food<<endl;
 cout<<meal<<endl;
 cout<<&meal;

  return 0;
}
```

```
C:\Users\monic\OneDrive\De:

pizza
0x61fde0
pizza
0x61fde0
Process returned 0 (0x0)   execution time : 0.762 s
Press any key to continue.
```

# Poniter

- we can get the memory address of a variable by using the & operator and the poniter is a variable that stores the memory address as its value.

- A pointer variable points to a data type (like int or string) of the same type, and is created with the * operator. The address of the variable you're working with is assigned to the pointer.
- Declare  : DataType* poniterName ;

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

int main() {

 string food = "Pizza";  // A food variable of type string
string* ptr = &food;    // A pointer variable, with the name ptr, that stores the address of food

// Output the value of food (Pizza)
cout << food << "\n";

// Output the memory address of food (0x6dfed4)
cout << &food << "\n";

// Output the memory address of food with the pointer (0x6dfed4)
cout << ptr << "\n";

// Output the memory address of food with the pointer (Pizza)
cout<<*ptr;

  return 0;
}
```

# Modify the Pointer Value

Example

```
string food = "Pizza";
string* ptr = &food;

// Output the value of food (Pizza)
cout << food << "\n";

// Output the memory address of food (0x6dfed4)
cout << &food << "\n";

// Access the memory address of food and output its value (Pizza)
cout << *ptr << "\n";

// Change the value of the pointer
*ptr = "Hamburger";

// Output the new value of the pointer (Hamburger)
cout << *ptr << "\n";

// Output the new value of the food variable (Hamburger)
cout << food << "\n";
```

# Pointers and arrays

```
1   #include <iostream>
2   using namespace std;
3
4   int main() {
5
6   char word[]="hello";
7   char* ptr ;
8   ptr=word;
9   *ptr='a';
10  ptr++;
11  *ptr='b';
12  ptr+=2;
13  *ptr='c';
14  cout<<word;
15    return 0;
16  }
17
```

Thank you !