Intermediate programming(C++)- *Lab 5 – Array*

CSCI112
Nile University

Content

- Recursion
- 1D arrays
- 2D arrays
- Strings

Recursion - Ex1

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

void countDown(int count)
{
    cout << "Push " << count << '\n';

    if (count > 1) // base case
        countDown(count-1);

    cout << "Pop " << count << '\n';
}

int main()
{
    countDown(5);
    return 0;
}
```

Output:

```
Push 5
Push 4
Push 3
Push 2
Push 1
Pop 1
Pop 2
Pop 3
Pop 4
Pop 5
```

Recursion - Ex2

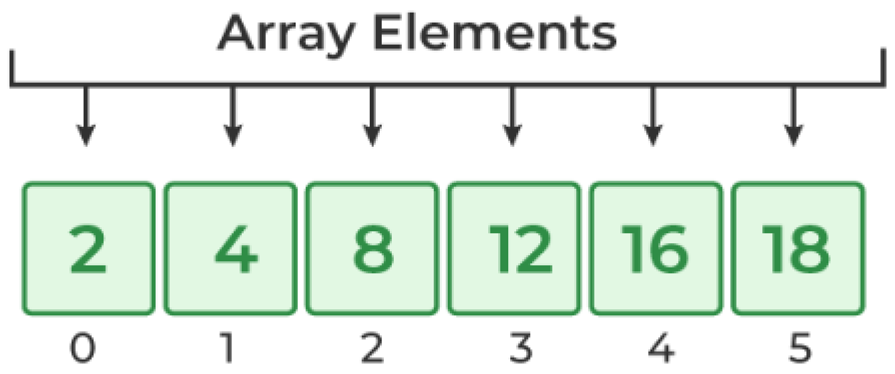
```
// return the sum of all the integers between 1 (inclusive) and sumto
(inclusive)
// returns 0 for negative numbers
int sumTo(int sumto)
{
    if (sumto <= 0)
        return 0; // base case (termination condition) when user passed
in an unexpected argument (0 or negative)
    if (sumto == 1)
        return 1; // normal base case (termination condition)

    return sumTo(sumto - 1) + sumto; // recursive function call
}

int main() {
    sumTo(5);
}
```

Output:

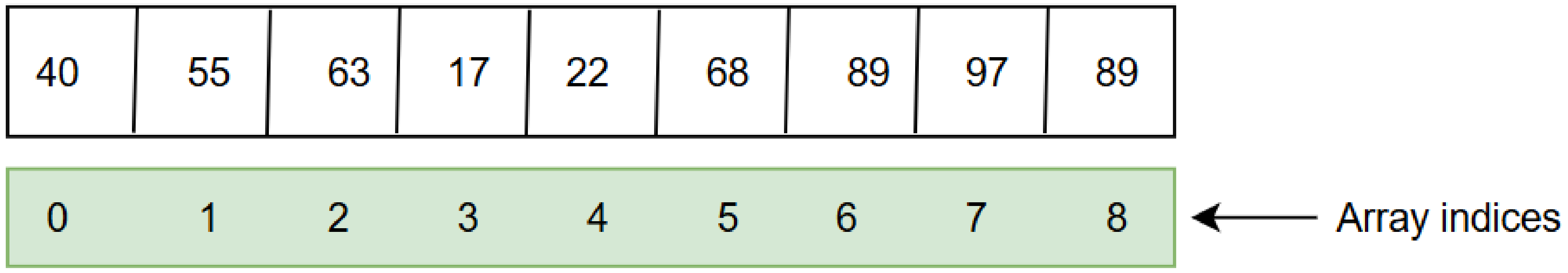
```
15
```

Array

- An array in C is a collection of items stored at contiguous memory locations.
- Elements can be accessed randomly using indices of an array.
- They are used to store similar type of elements as in the data type must be the same for all elements.
- They can be used to store collection of primitive data types such as: int, float, double, char, etc.. of any particular type.



Array - 1D



Array length = 9
First index = 0
Last index = 8

Array

When to use Arrays?

1. We can use normal variables (v1, v2, v3, ..) when we have a small number of objects.

1. But if we want to store a large number of instances, it becomes difficult to manage them with normal variables.

1. The idea of an array is to represent many instances in one variable.

Array initialization:

1. Initialize Array with Values in C++

```
int arr[5] = {1, 2, 3, 4, 5};
```

2. Initialize Array with Values and without Size in C++

```
int arr[] = {1, 2, 3, 4, 5};
```

Array

Array initialization:

3. Initialize Array after declaration (using loops)

```
int arr[5];
for (int i = 0; i < N; i++) {
    arr[i] = value;
}
```

4. Initialize Array partially

```
int partialArray[5] = {1, 2};
```

5. Initialize array with Zeros

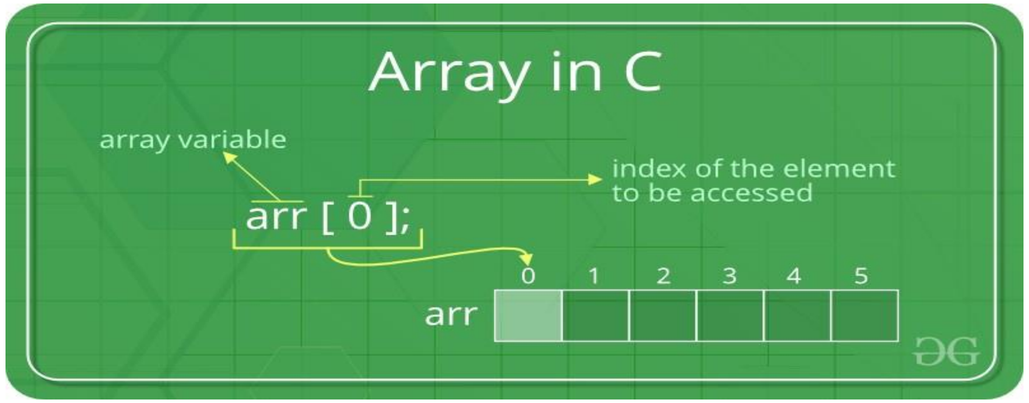
```
int zero_array[5] = {0};
```

Access elements of array:

```
int arr[5] = {1, 2, 3, 4, 5};
cout << arr[0];  // 1
cout << arr[3];  // 4
cout << arr[7];   // garbage value
```

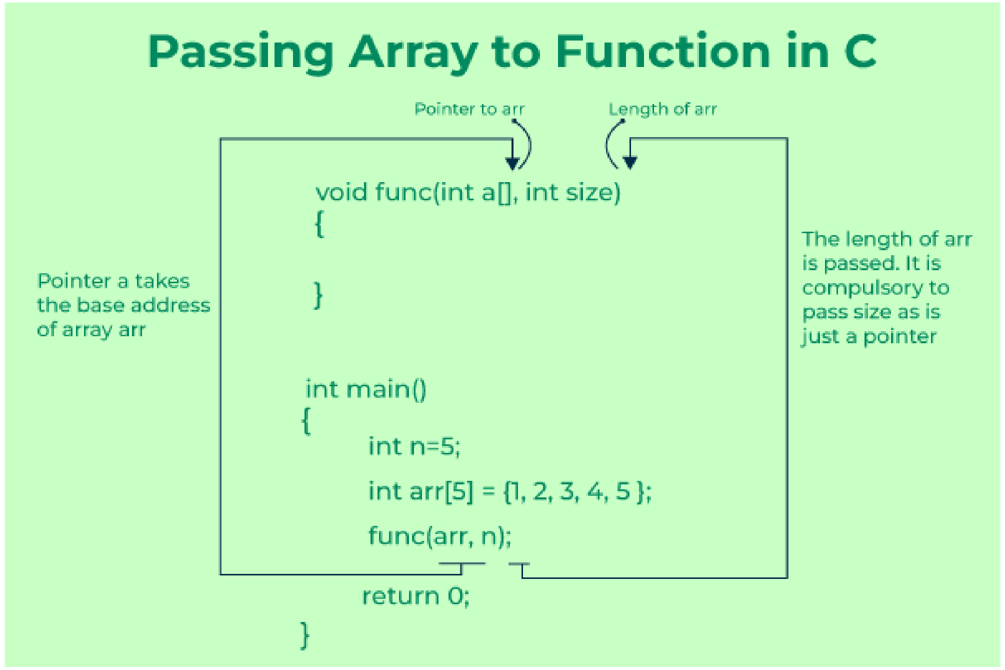
Update elements of array:

```
arr[0] = 10;
```



Array – Passing arrays to functions

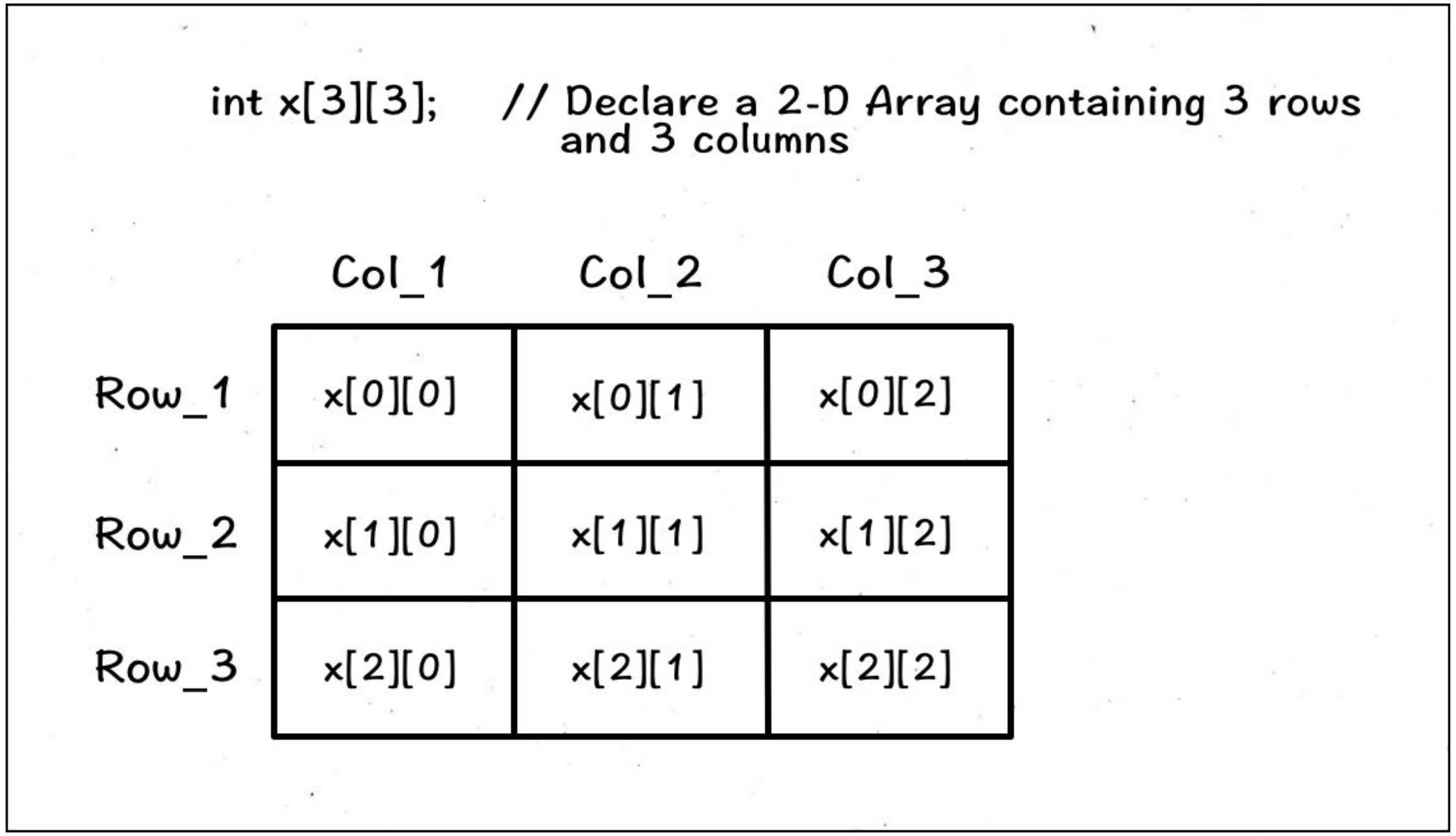
```cpp
#include <iostream>
using namespace std;

void printarray(int a[],int size)
{
    for (int i = 0; i < size; i++)
        a[i] = a[i] + 5;
}

int main()
{
    int a[5] = { 1, 2, 3, 4, 5 };
    int n=5;
    printarray(a,n); // Passing array to function

    for (int i = 0; i < n; i++)
        cout << a[i] << " ";
    return 0;
}
```



Passing Array to Function in C
Pointer to arr
Length of arr
Pointer a takes the base address of array arr
The length of arr is passed. It is compulsory to pass size as is just a pointer

```c
void func(int a[], int size)
{

}

int main()
{
    int n=5;
    int arr[5] = {1, 2, 3, 4, 5 };
    func(arr, n);
    return 0;
}
```

Array - 2D



```
int x[3][3];    // Declare a 2-D Array containing 3 rows
                   and 3 columns
```

	Col_1	Col_2	Col_3
Row_1	x[0][0]	x[0][1]	x[0][2]
Row_2	x[1][0]	x[1][1]	x[1][2]
Row_3	x[2][0]	x[2][1]	x[2][2]

Array – 2D – Looping on each element

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

int main()
{
    // Declaring 2D array
    int arr[4][4];
    // Initialize 2D array using loop
    for (int i = 0; i < 4; i++)
        for (int j = 0; j < 4; j++)
            arr[i][j] = i + j;

    // Printing the element of 2D array
    for (int i = 0; i < 4; i++) {
        for (int j = 0; j < 4; j++) {
            cout << arr[i][j] << " ";
        }
        cout << endl;
    }

    return 0;
}
```

Output:

```
0 1 2 3
1 2 3 4
2 3 4 5
3 4 5 6
```