MODULE 7 - C Arrays

C Arrays

- An array is a variable that can store multiple values. For example, if you want to store 100 integers, you can create an array for it.

```
int data[100];
```

How to declare an array?

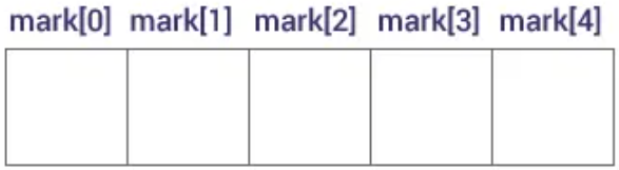
```
dataType arrayName[arraySize];
```

For example,

```
float mark[5];
```

Access Array Elements

- You can access elements of an array by indices.
Suppose you declared an array mark as above. The first element is mark[0], the second element is mark[1] and so on.



Arrays have 0 as the first index, not 1. In this example, mark[0] is the first element.

How to initialize an array?

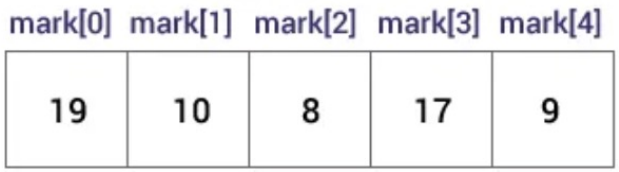
- It is possible to initialize an array during declaration. For example,

```
int mark[5] = {19, 10, 8, 17, 9};
```

You can also initialize an array like this.

```
int mark[] = {19, 10, 8, 17, 9};
```

Here, we haven't specified the size. However, the compiler knows its size is 5 as we are initializing it with 5 elements.



Change Value of Array elements

- int mark[5] = {19, 10, 8, 17, 9}

```
// make the value of the third element to -1
mark[2] = -1;

// make the value of the fifth element to 0
mark[4] = 0;
```

Input and Output Array Elements

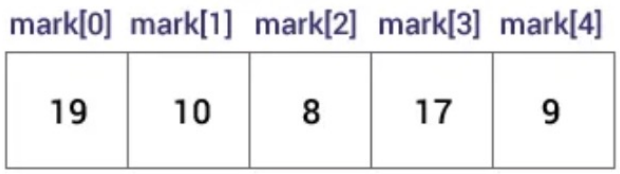
- Here's how you can take input from the user and store it in an array element.

```
// take input and store it in the 3rd element
scanf("%d", &mark[2]);

// take input and store it in the ith(last) element
scanf("%d", &mark[i-1]);
```

Input and Output Array Elements

- Here's how you can print an individual element of an array.




```
// print the first element
printf("%d", mark[0]);

// print the third element
printf("%d", mark[2]);

// print the ith(last) element
printf("%d", mark[i-1]);
```

Access elements out of its bound!

- int testArray[10];

You can access the array elements from testArray[0] to testArray[9].

Now let's say if you try to access testArray[12]. The element is not available. This may cause unexpected output (undefined behavior). Sometimes, you might get an error, and some other times your program may run correctly.
Hence, you should never access elements of an array outside of its bound.

C Multidimensional Arrays

- In C programming, you can create an array of arrays. These arrays are known as multidimensional arrays. For example,

```
float x[3][4];
```

Here, x is a two-dimensional (2d) array. The array can hold 12 elements. You can think the array as a table with 3 rows and each row has 4 columns.

	Column 1	Column 2	Column 3	Column 4
Row 1	x[0][0]	x[0][1]	x[0][2]	x[0][3]
Row 2	x[1][0]	x[1][1]	x[1][2]	x[1][3]
Row 3	x[2][0]	x[2][1]	x[2][2]	x[2][3]

C Multidimensional Arrays

- Similarly, you can declare a three-dimensional (3d) array. For example,

```
float y[2][4][3];
```

Here, the array y can hold 24 elements.

Initializing a multidimensional array

- Here is how you can initialize two-dimensional and three-dimensional arrays:

Initialization of a 2d array

```
// Different ways to initialize two-dimensional array

// Example 1
int c[2][3] = {
        {1, 3, 0},
        {-1, 5, 9}
};

// Example 2
int c[][3] = {{1, 3, 0}, {-1, 5, 9}};

// Example 3
int c[2][3] = {1, 3, 0, -1, 5, 9};
```

Initializing a multidimensional array

- Initialization of a 3d array
  You can initialize a three-dimensional array in a similar way to a two-dimensional array. Here's an example,

```
int test[2][3][4] = {
        {
                {3, 4, 2, 3},
                {0, -3, 9, 11},
                {23, 12, 23, 2}
        },
        {
                {13, 4, 56, 3},
                {5, 9, 3, 5},
                {3, 1, 4, 9}
        }
};
```

PRACTICE

- What is the output?

```c
#include <stdio.h>

int main() {
    int two_dimensional_arr[2][3] =
    {
        {1, 3, 0},
        {-1, 5, 9}
    };

    printf("%d ", two_dimensional_arr[1][1]);

    return 0;

}
```

OUPUT:

PRACTICE

- What is the output?

```c
#include <stdio.h>

int main() {
    int two_dimensional_arr[3][3] =
    {
        {1, 3, 0},
        {-1, 5, 9},
        {2, 4, 8}
    };

    printf("%d ", two_dimensional_arr[2][2]);

    return 0;

}
```

OUPUT:

PRACTICE

- What is the output?

```c
#include <stdio.h>

int main() {
    int three_dimensional_arr[2][3][4] = {
        {
            {3, 4, 2, 3},
            {0, -3, 9, 11},
            {23, 12, 23, 2}
        },
        {
            {13, 4, 56, 3},
            {5, 9, 3, 5},
            {3, 1, 4, 9}
        }
    };


    printf("%d ", three_dimensional_arr[1][2][3]);

    return 0;

}
```

OUPUT:

```
9
```

PROGRAM 1

```c
// Program to take 5 values from the user and store them in an array
// Print the elements stored in the array

#include <stdio.h>

int main() {

  int values[5];

  printf("Enter 5 integers: ");

  // taking input and storing it in an array
  for(int i = 0; i < 5; ++i) {
     scanf("%d", &values[i]);
  }

  printf("Displaying integers: ");

  // printing elements of the array
  for(int i = 0; i < 5; ++i) {
     printf("%d\n", values[i]);
  }
  return 0;
}
```

PROGRAM 2

```c
// Program to find the average of n numbers using arrays

#include <stdio.h>

int main() {

  int marks[10], i, n, sum = 0;
  double average;

  printf("Enter number of elements: ");
  scanf("%d", &n);

  for(i=0; i < n; ++i) {
    printf("Enter number%d: ",i+1);
    scanf("%d", &marks[i]);

    // adding integers entered by the user to the sum variable
    sum += marks[i];
  }

  // explicitly convert the sum to double
  // then calculate average
  average = (double) sum / n;

  printf("Average = %.2lf", average);

  return 0;
}
```

PRACTICE

- What is the output?

```c
#include <stdio.h>

int main() {
    int arrNumber[10] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};

    for(int i = 0; i < 10; i++) {
        printf("%d\n", arrNumber[i]);
    };

    return 0;
}
```

OUPUT:

```
1
2
3
4
5
6
7
8
9
10
```

PRACTICE

- What is the output?

```c
#include <stdio.h>

int main() {
    int arrNumber[] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
    int arrLen = sizeof(arrNumber) / sizeof(arrNumber[0]);


    for(int i = 0; i < arrLen; i++) {
        printf("%d ", arrNumber[i]);
    };

    return 0;
}
```

OUPUT:

```
1 2 3 4 5 6 7 8 9 10
```

PRACTICE

- What is the output?

```c
#include <stdio.h>

int main() {
    int nums[] = {1, 2, 3, 3, 2, 1};

    printf("%d", nums[nums[1]]);
}
```

OUPUT:
3