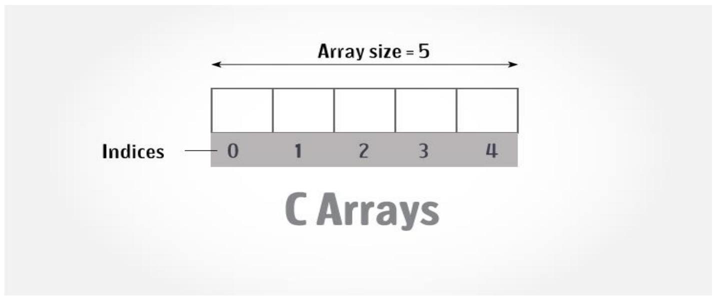


- An array is a collection of fixed number of values of a single type. For example: if you want to store 100 integers in sequence, you can create an array for it.
  ```
  int data[100];
  ```
- The size and type of arrays cannot be changed after its declaration.
- Arrays are of two types:
  - One-dimensional arrays
  - Multidimensional arrays

How to declare arrays?

```
data_type array_name[array_size];
```

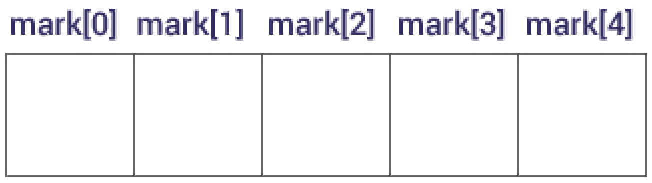
For example, `float mark[5];`

Here, we declared an array, `mark`, of floating-point type and size 5. Meaning, it can hold 5 floating-point values.

Elements of an Array and How to access them?

You can access elements of an array by indices.

Suppose you declared an array `mark` as above. The first element is `mark[0]`, second element is `mark[1]` and so on.



Few key notes:

- Arrays have 0 as the first index not 1. In this example, `mark[0]`
- If the size of an array is `n`, to access the last element, `(n-1)` index is used. In this example, `mark[4]`
- Suppose the starting address of `mark[0]` is 2120d. Then, the next address, `a[1]`, will be 2124d, address of `a[2]` will be 2128d and so on. It's because the size of a float is 4 bytes.

How to initialize an array?

It's possible to initialize an array during declaration. For example,

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

Another method to initialize array during declaration:

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

mark[0]	mark[1]	mark[2]	mark[3]	mark[4]
19	10	8	17	9

Here,

```
mark[0] is equal to 19
mark[1] is equal to 10
mark[2] is equal to 8
mark[3] is equal to 17
mark[4] is equal to 9
```

How to insert and print array elements?

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

      // insert different value to third element
      mark[3] = 9;

      // take input from the user and insert in third element
      scanf("%d", &mark[2]);

      // take input from the user and insert in (i+1)th element
      scanf("%d", &mark[i]);

      // print first element of an array
      printf("%d", mark[0]);

      // print ith element of an array
      printf("%d", mark[i-1]);
```

Example: C Arrays

```
// Program to find the average of n (n < 10) numbers using arrays
```

```
      #include <stdio.h>
      int main()
      {
          int marks[10], i, n, sum = 0, average;
          printf("Enter n: ");
          scanf("%d", &n);
          for(i=0; i<n; ++i)
          {
              printf("Enter number%d: ",i+1);
              scanf("%d", &marks[i]);
              sum += marks[i];
          }
          average = sum/n;
```

```
    printf("Average = %d", average);

    return 0;
}
```

Output

```
Enter n: 5
Enter number1: 45
Enter number2: 35
Enter number3: 38
Enter number4: 31
Enter number5: 49
Average = 39
```

Important thing to remember when working with C arrays

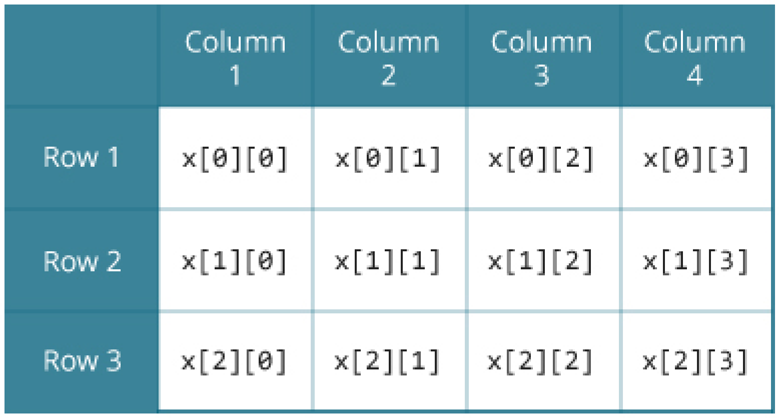
- Suppose you declared an array of 10 elements. Let's say,
  `int testArray[10];`
- You can use the array members from `testArray[0]` to `testArray[9]`.
- If you try to access array elements outside of its bound, let's say `testArray[12]`, the compiler may not show any error. However, this may cause unexpected output (undefined behavior).

Multidimensional Arrays

In C programming, you can create array of an array known as multidimensional array. 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 table with 3 row and each row has 4 column.



	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]

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

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

Here,The array `y` can hold 24 elements.
You can think this example as: Each 2 elements have 4 elements, which makes 8 elements and each 8 elements can have 3 elements. Hence, the total number of elements is 24.

How to initialize a multidimensional array?

There is more than one way to initialize a multidimensional array.

Initialization of a two dimensional array

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

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

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

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

Above code are three different ways to initialize a two dimensional arrays.

Initialization of a three dimensional array.

You can initialize a three dimensional array in a similar way like 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} }
                 };
```

Example 1: Two Dimensional Array to store and print values

```
// C program to store temperature of two cities for a week and display it.

#include <stdio.h>

const int CITY = 2;
const int WEEK = 7;

int main()
{
    int temperature[CITY][WEEK];
    for (int i = 0; i < CITY; ++i) {
        for(int j = 0; j < WEEK; ++j) {
            printf("City %d, Day %d: ", i+1, j+1);
            scanf("%d", &temperature[i][j]);
        }
    }

    printf("\nDisplaying values: \n\n");
    for (int i = 0; i < CITY; ++i) {
        for(int j = 0; j < WEEK; ++j)
        {
            printf("City %d, Day %d = %d\n", i+1, j+1, temperature[i][j]);
        }
    }
    return 0;
}
```

Output

```
City 1, Day 1: 33
City 1, Day 2: 34
City 1, Day 3: 35
City 1, Day 4: 33
City 1, Day 5: 32
City 1, Day 6: 31
City 1, Day 7: 30
City 2, Day 1: 23
City 2, Day 2: 22
City 2, Day 3: 21
City 2, Day 4: 24
City 2, Day 5: 22
City 2, Day 6: 25
City 2, Day 7: 26

Displaying values:

City 1, Day 1 = 33
City 1, Day 2 = 34
City 1, Day 3 = 35
City 1, Day 4 = 33
City 1, Day 5 = 32
City 1, Day 6 = 31
City 1, Day 7 = 30
City 2, Day 1 = 23
City 2, Day 2 = 22
City 2, Day 3 = 21
City 2, Day 4 = 24
City 2, Day 5 = 22
City 2, Day 6 = 25
City 2, Day 7 = 26
```

Example 2: Sum of two matrices using Two dimensional arrays

```c
// C program to find the sum of two matrices of order 2*2

#include <stdio.h>
int main()
{
   float a[2][2], b[2][2], c[2][2];
   int i, j;

   // Taking input using nested for loop
   printf("Enter elements of 1st matrix\n");
   for(i=0; i<2; ++i)
    for(j=0; j<2; ++j)
    {
       printf("Enter a%d%d: ", i+1, j+1);
       scanf("%f", &a[i][j]);
    }

  // Taking input using nested for loop
   printf("Enter elements of 2nd matrix\n");
   for(i=0; i<2; ++i)
    for(j=0; j<2; ++j)
    {
       printf("Enter b%d%d: ", i+1, j+1);
       scanf("%f", &b[i][j]);
    }

   // adding corresponding elements of two arrays
   for(i=0; i<2; ++i)
    for(j=0; j<2; ++j)
    {
       c[i][j] = a[i][j] + b[i][j];
    }

   // Displaying the sum
   printf("\nSum Of Matrix:");

   for(i=0; i<2; ++i)
    for(j=0; j<2; ++j)
    {
       printf("%.1f\t", c[i][j]);

       if(j==1)
          printf("\n");
    }
return 0;
}
```

Ouput

```
Enter elements of 1st matrix
Enter a11: 2;
Enter a12: 0.5;
Enter a21: -1.1;
Enter a22: 2;
Enter elements of 2nd matrix
Enter b11: 0.2;
Enter b12: 0;
Enter b21: 0.23;
Enter b22: 23;

Sum Of Matrix:
2.2     0.5
-0.9    25.0
```

STRINGS IN C

Read String from the user

- You can use the scanf() function to read a string.
- The scanf() function reads the sequence of characters until it encounters a whitespace (space, newline, tab etc.).

Example 1: scanf() to read a string

```
#include <stdio.h>
int main()
{
    char name[20];
    printf("Enter name: ");
    scanf("%s", name);
    printf("Your name is %s.", name);
    return 0;
}
```

Output

```
Enter name: Dennis Ritchie
Your name is Dennis.
```

Even though Dennis Ritchie was entered in the above program, only "Ritchie" was stored in the name string. It's because there was a space after Ritche.

How to read a line of text?

You can use gets() function to read a line of string. And, you can use puts() to display the string.

Example 2: gets() and puts()

```
#include <stdio.h>
int main()
{
    char name[30];
    printf("Enter name: ");
    gets(name);     // read string
    printf("Name: ");
    puts(name);    // display string
    return 0;
}
```

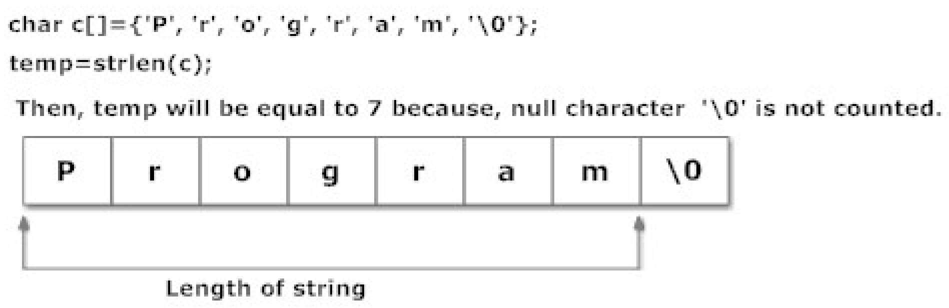
When you run the program, the output will be:

```
Enter name: Tom Hanks
Name: Tom Hanks
```

Main String Functions:

Discussed are following four functions included in string.h header file.

1. The strlen() function calculates the length of a given string. The `strlen()` function is defined in <string.h> header file.



```
char c[]={'P', 'r', 'o', 'g', 'r', 'a', 'm', '\0'};
temp=strlen(c);
```

Then, temp will be equal to 7 because, null character '\0' is not counted.

Example: C strlen() function

```
#include <stdio.h>
#include <string.h>
int main()
{
   char a[20]="Program";
   char b[20]={'P','r','o','g','r','a','m','\0'};
   char c[20];

   printf("Enter string: ");
   gets(c);

   printf("Length of string a = %d \n",strlen(a));

   //calculates the length of string before null charcter.
   printf("Length of string b = %d \n",strlen(b));
   printf("Length of string c = %d \n",strlen(c));

   return 0;
}
```

Output

```
Enter string: String
Length of string a = 7
Length of string b = 7
Length of string c = 6
```

2. strcat() concatenates (joins) two strings.

It takes two arguments, i.e, two strings or character arrays, and stores the resultant concatenated string in the first string specified in the argument.

```
#include <stdio.h>
#include <string.h>
int main()
{
    char str1[] = "This is ", str2[] = "programiz.com";

    //concatenates str1 and str2 and resultant string is
stored in str1.
    strcat(str1,str2);

    puts(str1);
    puts(str2);

    return 0;
}
```

Output

```
This is programiz.com
programiz.com
```

3. The strcmp() function compares two strings and returns 0 if both strings are identical.

The strcmp() function takes two strings and return an integer.

The strcmp() compares two strings character by character. If the first character of two strings are equal, next character of two strings are compared. This continues until the corresponding characters of two strings are different or a null character '\0' is reached.

Return Value from strcmp()

Return Value	Remarks
0	if both strings are identical (equal)
negative	if the ASCII value of first unmatched character is less than second.
positive integer	if the ASCII value of first unmatched character is greater than second.

```
#include <stdio.h>
#include <string.h>

int main()
{
    char str1[] = "abcd", str2[] = "abCd", str3[] = "abcd";
    int result;
```

```c
    // comparing strings str1 and str2
    result = strcmp(str1, str2);
    printf("strcmp(str1, str2) = %d\n", result);

    // comparing strings str1 and str3
    result = strcmp(str1, str3);
    printf("strcmp(str1, str3) = %d\n", result);

    return 0;
}
```

Output

```
strcmp(str1, str2) = 32
strcmp(str1, str3) = 0
```

The first unmatched character between string `str1` and `str2` is third character. The ASCII value of 'c' is 99 and the ASCII value of 'C' is 67. Hence, when strings `str1` and `str2` are compared, the return value is 32.
When strings `str1` and `str3` are compared, the result is 0 because both strings are identical.

4. The strcpy() function copies the string pointed by `source` (including the null character) to the character array `destination`.

This function returns character array `destination`.

The strcpy() function is defined in `string.h` header file.

```c
#include <stdio.h>
#include <string.h>

int main()
{
    char str1[10]= "awesome";
    char str2[10];
    char str3[10];

    strcpy(str2, str1);
    strcpy(str3, "well");
    puts(str2);
    puts(str3);

    return 0;
}
```

Output

```
awesome
well
```

It is important to note that, the destination array should be large enough otherwise it may result in undefined behavior.