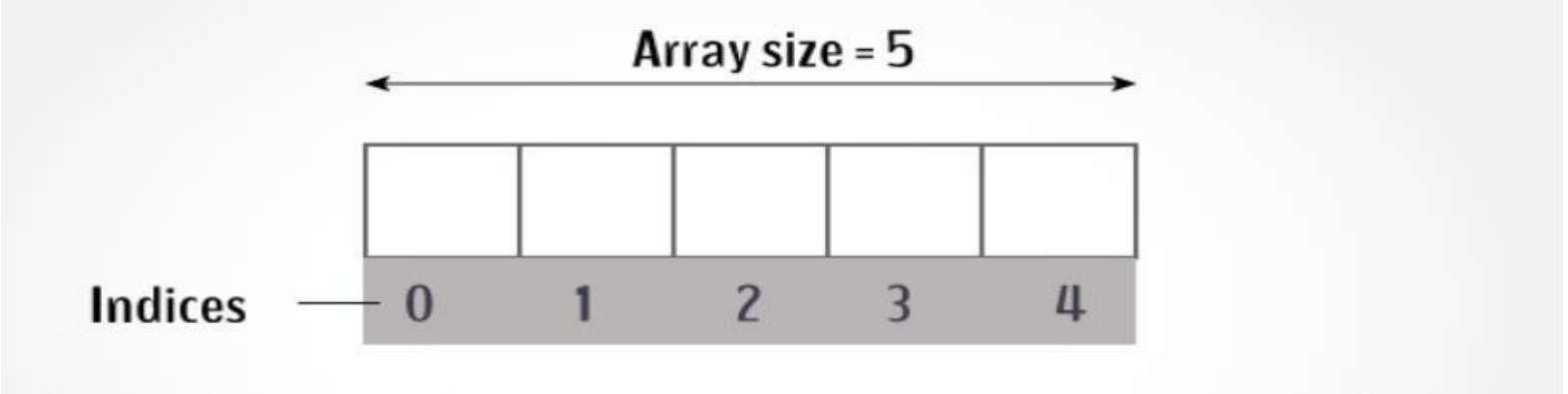
MNNIT
COMPUTER CODING
CLUB

C L A S S - 3

Basics Of C++

1D ARRAY

- An array in C++ is a collection of homogenous items stored at contiguous memory locations.
- The elements of the array share the same variable name but each element has its own unique index number.
- An array can be of any type. For example: `int`, `float`, `char` etc. If an array is of type `int` then it's elements must be of type `int` only.
- We can use normal variables (v1, v2, v3, ..) when we have a small number of objects, but if we want to store a large number of instances, it becomes difficult to manage them with normal variables. The idea of an array is to represent many instances in one variable.



DECLARING 1D ARRAY

dataType arrayName[arraySize];

For example, `float mark[5];`

- Here, we declared an array, mark, of floating-point type. And its size is 5. Meaning, it can hold 5 floating-point values.
- It's important to note that the size and type of an array cannot be changed once it is declared.

I NI TI ALI ZI NG 1D ARRAY

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

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

An array can also be initialized like

```
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.

ACCESSING 1D ARRAY ELEMENTS

- Elements of array can be accessed by indices.
- Array index starts from 0 and goes till size of array minus 1.

  Considering the array mark declared in previous slide. The first element is mark[0], the second element is mark[1] and so on.


```
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
```

2D ARRAY

- An array of arrays is known as 2D array.
- Just like 1D array, 2D array also store homogenous values (i.e. same data type values)

Here, **x** is a two-dimensional (2d) array. The array can hold 12 elements. This 2D array can be visualized 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]

DECLARING 2D ARRAY

The syntax to declare the 2D array is given below.

```
dataType arrayName[rows][columns];
```

Consider the following example.

```
int arr[4][3];
```

Here, 4 is the number of rows, and 3 is the number of columns.

INITIALIZING 2D ARRAY

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

2D array can also be initialized like

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

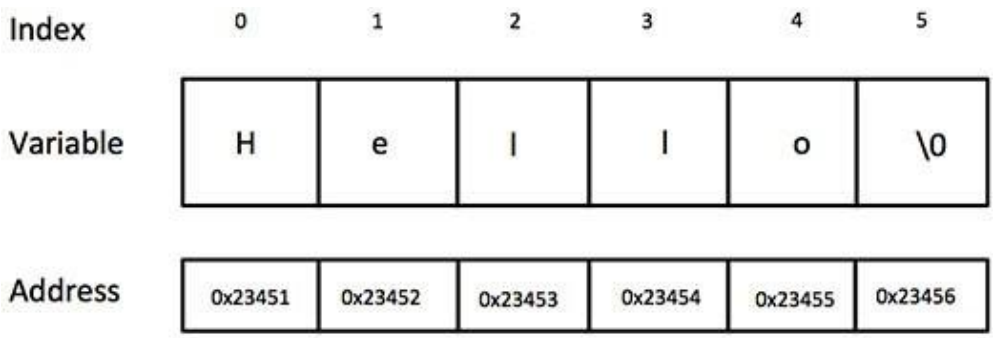
Note: Although both the above declarations are valid, we recommend you to use the first method as it is more readable.

We already know, when we initialize a 1D array during declaration, we need not to specify the size of it. However that's not the case with 2D array, you must always specify the second dimension even if you are specifying elements during the declaration.

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

CHAR ARRAY (STRINGS)

- Strings are actually one-dimensional array of characters terminated by a **null** character '\0'.
- `char greeting[6] = {'H', 'e', 'l', 'l', 'o', '\0'};`
- **`char`** greeting`[] = "Hello";`
- Functions of Strings are found in string.h header file;



Question:

Write a program to initialise an array of size 10 by taking input from user and print the sum of values at even indices.

What is the output of the following code?

```
#include <iostream>

using namespace std;

int main() {

   int i;

   int arr[6] = { 10, 20, 30, 40 };

  // cout<<arr[5];

  for(i=0;i<6;i++)

   cout<<arr[i]<<” “;

   return 0;

}
```

Options

1. 10 20 30 40 0 0
2. 10 20 30 40 46854 87867
3. 10 20 30 40 454 343
4. Error

STRING FUNCTIONS

Sr.No.	Function & Purpose
1	**strcpy(s1, s2);** Copies string s2 into string s1.
2	**strcat(s1, s2);** Concatenates string s2 onto the end of string s1.
3	**strlen(s1);** Returns the length of string s1.
4	**strcmp(s1, s2);** Returns 0 if s1 and s2 are the same; less than 0 if s1<s2; greater than 0 if s1>s2.
5	**strchr(s1, ch);** Returns a pointer to the first occurrence of character ch in string s1.
6	**strstr(s1, s2);** Returns a pointer to the first occurrence of string s2 in string s1.

F U N C T I O N S I N C++

A function is a group of statements that together perform a task. Every C++ program has at least one function, which is main(), and all the most trivial programs can define additional functions.

You can divide up your code into separate functions. How you divide up your code among different functions is up to you, but logically the division is such that each function performs a specific task.

Advantages of using functions -

- Reusability of code, avoids repetition of code
- Makes program easier to modify and debug in case of errors
- Enhances readability of code, a big code is difficult to read

DEFINITION OF A FUNCTION

The general form of a function definition in C++ programming language is as follows −

```
return_type function_name
        (parameters) { body of the
        function
}
```

Return Type − A function may return a value. The return_type is the data type of the value the function returns. Some functions perform the desired operations without returning a value. In this case, the return_type is the keyword void.

Function Name − This is the actual name of the function. The function name and the parameter list together constitute the function signature.

Parameters − A parameter is like a placeholder. When a function is invoked, you pass a value to the parameter. This value is referred to as actual parameter or argument. The parameter list refers to the type, order, and number of the parameters of a function. Parameters are optional; that is, a function may contain no parameters.

Function Body − The function body contains a collection of statements that define what the function does.

DECLARATION OF FUNCTION

Here the function sumOfDigits() is written before main(), so main() knows everything about the function sumOfDigits(). But generally, the function main() is placed at the top and all other functions are placed after it. In this case, function declaration is needed. The function declaration is also known as the function prototype, and it informs the compiler about the following three things -

1. Name of the function.
2. Number and type of arguments received by the function.
3. Type of value returned by the function.

A function declaration has the following parts −

```
return_type function_name ( parameters );
```

For the above defined function sumOfDigits(), the function declaration is as follows −

```
int sumOfDigits(int num);
```

Parameter names are not important in function declaration only their type is required, so the following is also a valid declaration −

```
int sumOfDigits(int);
```

C A L L I N G

- Function definition tells what it does and to use the function you have to call it.
- The control and current line shift to the function
- Returned value can either be used or stored in different variable
- Calling of functions is done on stack (Ignore for now)
- If a function is not called, it won't execute on its own.
- In previous example sumOfDigits() is referred as called function and main() is the calling function

A R G U M E N T S

Actual arguments: The arguments which are mentioned in the function call.

Formal arguments: The name of the arguments, which are mentioned in the function definition are called formal or dummy arguments

- The order, number and type of actual arguments in the function call should match with the order, number and type of formal arguments in the function definition

TYPES OF USER DEFINED FUNCTIONS

The functions can be classified into four categories on the basis of the arguments and return value

1. Functions with arguments and a return value
2. Functions with arguments and no return value
3. Functions with no arguments and a return value
4. Functions with no arguments and no return value

M A I N

- Execution of every C++ program always begins with the function main().
- Every function is called in main() and after execution control is given back to main()
- The name, number and type of arguments are predefined in the language.
- main() returns 0 on successful execution and nonzero (garbage value) when error.
- The definition, declaration and call of main() function
- Function Declaration - By the C++ compiler
- Function Definition - By the programmer
- Function Call - By the operating system

LIBRARY FUNCTIONS

- The definition, declaration and call of library functions-
    1. Function Definition - Predefined, precompiled and present in the library.
    2. Function Declaration. - In header files (files with a .h extension)
    3. Function Call - By the programmer

- To use a library function in our program we should know-
    1. Name of the function and its purpose
    2. Type and number of arguments it accepts
    3. Type of the value it returns
    4. Name ,of the header file to be included.

- We can deflne any function of our own with the same name as that of any function in the C++ library. If we do so then the function that we have defined will take precedence over the library function with the same name.

L O C A L , G L O B A L & S T A T I C

- **Local Variable:** Variables that are defined within the body of the function or a block are local to that function or block and are called Local Variables.

- **Global Variable:** Any variable that is defined outside any function are called global variables. All the functions in the program can access and modify these variables. They are automatically initialized to 0 at the time of declaration.

- **Static Variable:** Any variable defined using keyword static before its type is called a static variable. These variables are initialized only once and retain their value between function calls.

Question:

Write a program to find the factorial of a given number by using a function in C++ programming language.