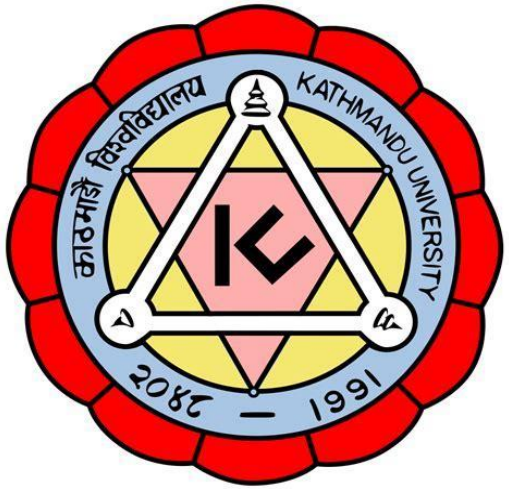
# KATHMANDU UNIVERSITY SCHOOL OF MANAGEMENT

BBIS

COM 102 : 3 Credit Hours




# 7. Functions

# Outlines

## Functions (6 hrs)

- 8.1 Defining Function
- 8.2 Use of function
- 8.3 Function Prototypes
- 8.4 Passing Argument to a Function
- 8.5 Recursive function
- 8.6 Storage Class

# Introduction

- A function is a collection of statements that work together to complete a task.
- Every C program contains at least one function, main(), and even the simplest programs can specify more functions.
- divide up our code into separate functions.
- The division is such that each function performs a specific task.
- A function declaration tells the compiler about a function's name, return type, and parameters.
- A function definition provides the actual body of the function.

…

- A function is a set of statements enclosed within curly brackets ({}) that take inputs, do the computation, and provide the resultant output.
- You can call a function multiple times, thereby allowing reusability and modularity in C programming.

The function main() in the program is executed first.

```
main()
{
//statements
}
```

- All other functions are executed from main which calls them directly or indirectly

# Need of Functions???

- Enables reusability and reduces redundancy.
- Makes a code modular.
- Provides abstraction functionality.
- The program becomes easy to understand and manage.
- Breaks an extensive program into smaller and simpler pieces.

# Types of Functions

1. Built-in Functions or Library functions
2. User defined functions

## 1. Built-in Functions or Library functions

- Also referred to as predefined functions, library functions are already defined in the C libraries.
- do not have to write a definition or the function's body to call them.
  - simply call them
- need to include the library at the beginning of the code for calling a library function.
- Printf(), scanf(), ceil(), and floor() are examples of library functions.

# 2. User-defined functions

- These are the functions that a developer or the user declares, defines, and calls in a program.
- Increases the scope and functionality, and reusability of C programming as we can define and use any function we want.
- can add a user-defined to any library to use it in other programs.

# Local and Global Variable

- Local variable
  - A local variable is a variable that is declared inside a function.
  - local variable can only be used in the function where it is declared.

- Global variable
  - A global variable is a variable that is declared outside all function.
  - A global variable can be used in all function

# Defining a Function

- General form:

```
return_type function_name ( parameter list ) {
   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.

# Example

```
/* function returning the max between two numbers */

int max(int num1, int num2) {
   /* local variable declaration */
   int result;

   if (num1 > num2)
      result = num1;
   else
      result = num2;

   return result;
}
```

# More…

```
int add(int x, int y);

void main()
{
   int c, a=10, b=20;

   c= add(a, b);

   printf("Sum=%d", c);

}
int  add (int x , int y)
{
   int sum;
   sum =x+y;
   return sum;
}
```

Function declaration / function prototype

Function call

Function declarator

Function body

Function definition

# Functions

| SN | C function aspects | Syntax |
|---|---|---|
| 1 | Function declaration | return_type function_name (argument list); |
| 2 | Function call | function_name (argument_list) |
| 3 | Function definition | return_type function_name (argument list) {function body;} |

# Function declaration / function prototype

- A function declaration tells the compiler about the number of parameters function takes, data-types of parameters, and return type of function.
- Putting parameter names in function declaration is optional in the function declaration, but it is necessary to put them in the definition.

```
// A function that takes two integers as parameters
// and returns an integer
int max(int , int );

// A function that takes an int pointer and an int variable as parameters
// and returns a pointer of type int
int *swap(int*,int);
```

```
// A function that takes a charas parameters
// and returns a pointer of type char
char *call(char b);

// A function that takes a char and an int as parameters
// and returns an integer
int fun(char, int);
```

# Example

```
int sum(int, int);  // function prototype

int sum(int x, int y) {  // function definition
        z = x + y;
        return z;
}

Int main () {
        int a=5,b=5,c;
        c = sum(a,b);  // calling function
}
```

# Classwork

- Program to calculate maximum of three numbers using function.
  - You will learn:
    - Functions
    - Decision making statements (if else)

# The main function

- The main function is a special function. Every C++ program must contain a function named main.
- It serves as the entry point for the program. The computer will start running  the code from the beginning of the main function.

- **Types of main Function:**

1) The first type is – main function without parameters :

```
// Without Parameters
    int main()
    {
      ...
      return 0;
    }
```

…

2) Second type is main function with parameters :

```
// With Parameters
int main(int argc, char * const argv[])
{
   ...
   return 0;
}
```

- The reason for having the parameter option for the main function is to allow input from the command line.
- When you use the main function with parameters, it saves every group of characters (separated by a space) after the program name as elements in an array named argv.
- Since the main function has the return type of int, the programmer must always have a return statement in the code. The number that is returned is used to inform the calling program what the result of the program's execution was.
- Returning 0 signals that there were no problems.

# Functions

- Depending on argument and the return value function can be classified as:
  - Function with no argument and no return value.
    - Eg ;void Print()
  - Function with argument but no return value.
    - Eg: void add(int x, int y);
  - Function with no argument but return value.
    - Eg: int add();
  - Function with both arguments and return value.
    - Eg: int add(int , int);

# Function with no argument and no return value.

```
void display() {
        printf(“Hello world”);
}

Void main() {
        display();
}
```

# Function with argument but no return value.

```
void addition(int x, int y) {
        printf(“The sum is: %d”, x+y);
}

Void main() {
        addition(2,3);
         addition(10,20);
}
```

# Function with no argument but return value.

```
int addition() {
        int x=5,y=10;
        return x+y;
}

Void main() {
        int sum;
        sum = addition();
        printf(“The addition is: %d”, sum);

}
```

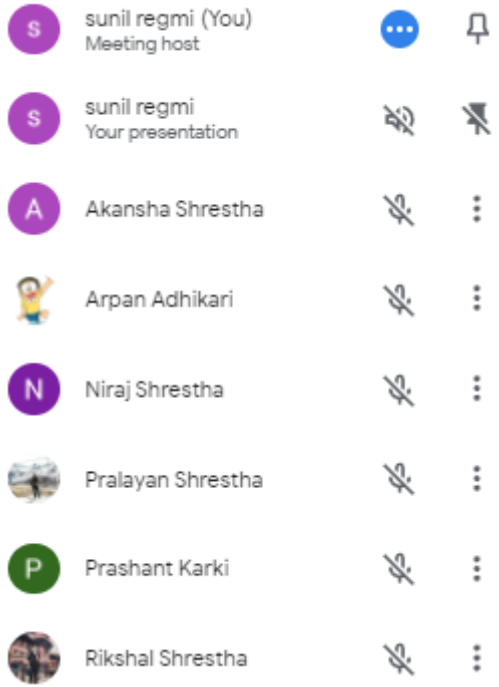
# Function with both argument and return value.

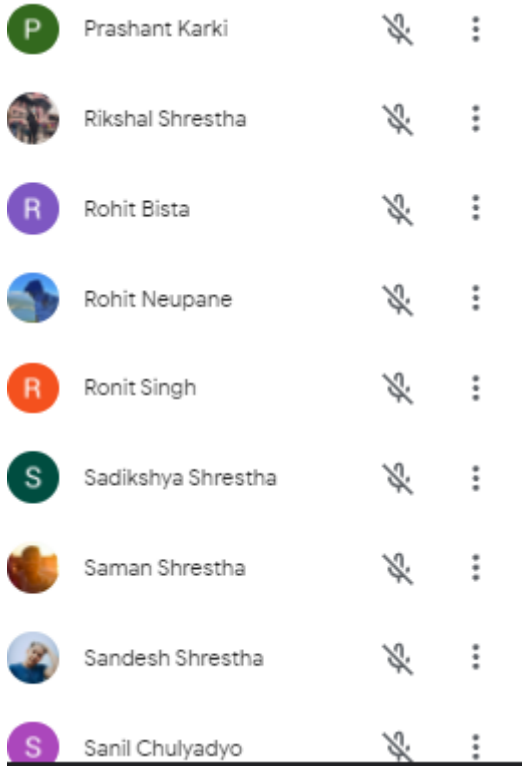
```
int addition(int x, int y) {
        int add;
        add = x+y;
        return add;
}

Void main() {
        int sum;
        sum = addition(10,20);
        printf("The addition is: %d", sum);

}
```

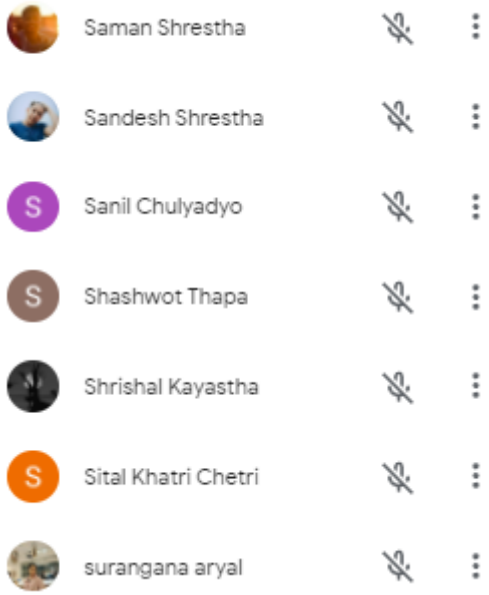
# Nesting functions

```
void first_func();
void second_func();
void third_func();
void main()
{
    first_func();
}
void first_func()
{
   printf("I am in first");
   second_func();
}
```

```
void second_func()
{
   printf("I am in second ");
    third_func();
}
void third_func()
{
   printf("I am in third");
}
```


sunil regmi (You)
Meeting host
sunil regmi
Your presentation
Akansha Shrestha
Arpan Adhikari
Niraj Shrestha
Pralayan Shrestha
Prashant Karki
Rikshal Shrestha


Prashant Karki
Rikshal Shrestha
Rohit Bista
Rohit Neupane
Ronit Singh
Sadikshya Shrestha
Saman Shrestha
Sandesh Shrestha
Sanil Chulyadyo


Saman Shrestha
Sandesh Shrestha
Sanil Chulyadyo
Shashwot Thapa
Shrishal Kayastha
Sital Khatri Chetri
surangana aryal