SNS COLLEGE OF ENGINEERING

Kurumbapalayam (Po), Coimbatore – 641 107

An Autonomous Institution

Accredited by NBA – AICTE and Accredited by NAAC – UGC with 'A' Grade
Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai

DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

COURSE NAME : 23ITT101- PROBLEM SOLVING & C PROGRAMMING

I YEAR /I SEMESTER

Unit 4- Functions

Topic 2: Function prototype, Function Definition

Brain Storming

1. How perform string manipulation operations?

Introduction

- A function is a group of statements that together perform a task.

- The function contains the set of programming statements enclosed by {}.

- The function is also known as *procedure* or *subroutine* in other programming languages.

- Every C program has at least one function, which is **main()**, and all the most trivial programs can define additional functions.

Advantage of functions in C

- By using functions, we can avoid rewriting same logic/code again and again in a program.

- We can call C functions any number of times in a program and from any place in a program.

- We can track a large C program easily when it is divided into multiple functions.

- Reusability is the main achievement of C functions.

- However, Function calling is always a overhead in a C program.

What is the purpose of a function prototype?

- function prototype specifies the input/output interlace to the function

- i.e. what to give to the function and what to expect from the function.

- It tells the return type of the data that the function will return.

- It tells the number of arguments passed to the function.

- It tells the data types of the each of the passed arguments.

- Also it tells the order in which the arguments are passed to the function.

return_type function_name(data_type parameter...)

{

//code to be executed

}

Example:

Void add()

{

//code to be executed

}

Conti...

```
int a = 10, b = 5, c;
```

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

Function Prototype - int is the return type and int x and int y are the function arguments

```
int main(void)
{
    c = product(a,b);

    printf("%i\n",c);

    return 0;
}
```

Main Function - int is always the return type and there are no arguments, hence the (void). Curly braces { } mark the start and end of the main function

Function call - product(a,b); a and b are global variables the function is passed. Here the values returned by the function are assigned to the variable c

```
int product(int x, int y)
{
    return (x * y);
}
```

Function Definition - contains the function statement return(x * y); the function returns x times y to the main function where it was called. Curly braces { } mark the start and end of the function

Conti...

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.

Conti...

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.
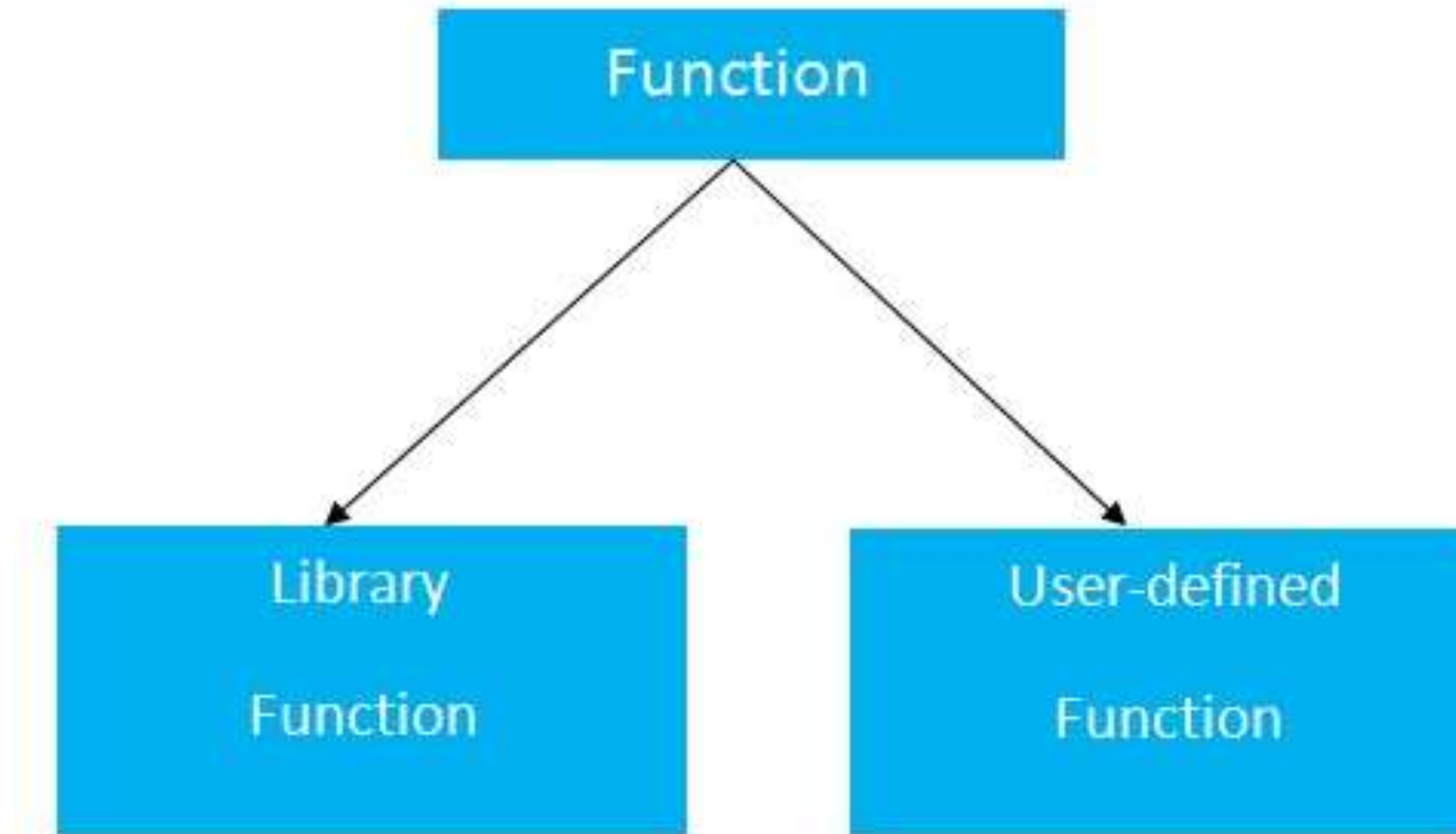
Example:without argument and return value

```c
#include<stdio.h>

void printName();

void main ()

{

    printf("Hello ");

    printName();

}

void printName()

{

    printf("This is function call example");

}
```

Types of Functions

Conti...

- There are two types of functions in C programming:

- **Library Functions**: are the functions which are declared in the C header files such as scanf(), printf(), gets(), puts(), ceil(), floor() etc.

- **User-defined functions**: are the functions which are created by the C programmer, so that he/she can use it many times. It reduces the complexity of a big program and optimizes the code.

Function call

- The function call statement invokes the function. When a function is invoked the compiler jumps to the called function to execute the statements that are part of that function.

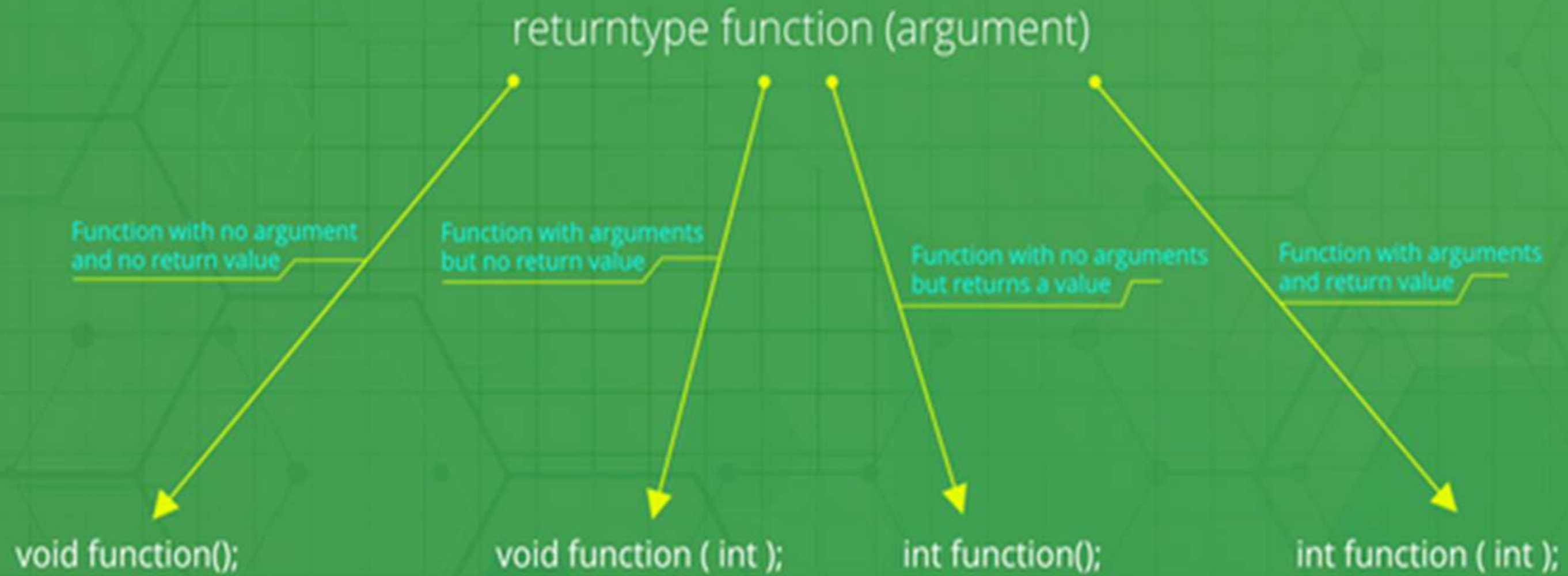- There are four different aspects of function calls.

Types of function call

- function without arguments and without return value

- function without arguments and with return value

- function with arguments and without return value

- function with arguments and with return value

Conti...

```
#include<stdio.h>

void printName();

void main ()

{

 printf("Hello ");

   printName();

}

void printName()

{

   printf("Javatpoint");

}
```

Example for Function without argument and with return value

```c
#include<stdio.h>
int sum();
void main()
{
    int result;
    printf("\nGoing to calculate the sum of two numbe
    rs:");
    result = sum();
    printf("%d",result);
}
int sum()
{
    int a,b;
    printf("\nEnter two numbers");
    scanf("%d %d",&a,&b);
    return a+b;
}
```

Example for Function with argument and without return value

```c
#include<stdio.h>
void sum(int, int);
void main()
{
    int a,b,result;
    printf("\nGoing to calculate the sum of two nu
    mbers:");
    printf("\nEnter two numbers:");
    scanf("%d %d",&a,&b);
    sum(a,b);
}
void sum(int a, int b)
{
    printf("\nThe sum is %d",a+b);
}
```

Example for Function with argument and with return value

```c
#include<stdio.h>
int sum(int, int);
void main()
{
    int a,b,result;
    printf("\nGoing to calculate the sum of two numbe
     rs:");
    printf("\nEnter two numbers:");
    scanf("%d %d",&a,&b);
    result = sum(a,b);
    printf("\nThe sum is : %d",result);
}
int sum(int a, int b)
{
    return a+b;
}
```

Assessment 1

1. What is function?

Ans : _____

2. Write about function prototype?

Ans : _____

References

1. Reema Thareja, "Programming in C", Oxford University Press, Second Edition, 2016

Thank You