FUNCTIONS

Presented by
T.Sirija

WHAT IS A FUNCTION ?

A subprogram (or) subroutine that is used to perform a specified task is called a function. It is a block of code which only runs when it is called. You can pass data, known as parameters, into a function. Functions are used to perform certain actions, and they are important for reusing code: Define the code once, and use it many times. A C program is made of one or more functions, and one and only one of which must be main( ) function. The execution of the program always starts with main( )

WHAT IS DIFFERENCE BETWEEN A PROGRAM AND FUNCTION ?

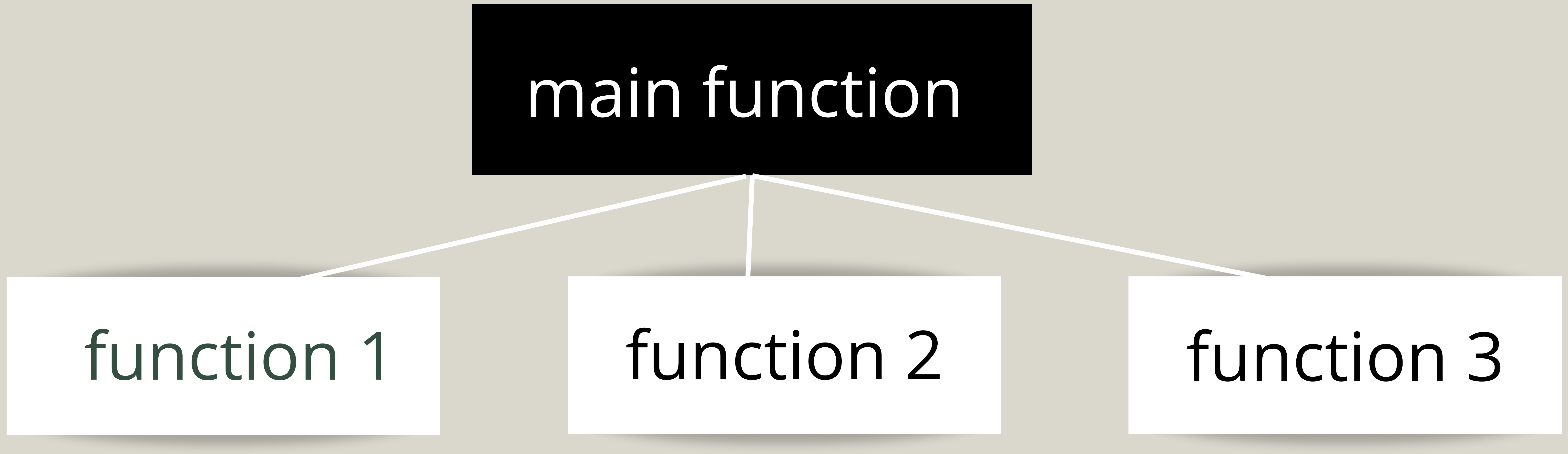
Program is set or collection of instructions used by computer to execute specific task and these are created using particular programming languages such as C,C++ , Python, Ruby, etc,

Function, as name suggests, is basic concept in computer programming that one calls to control flow of program as well as perform a specific task, returns a value and resumes program where it was called.

A function is a part of program.

ADVANTAGES OF FUNCTIONS

A function helps in dividing a complex program into simpler segments .It facilitates top-down modular programming approach as shown in figure below.



TYPES OF FUNCTIONS

in C language, functions are classified into two types. They are

a) Library functions (or) Built-in functions

b) User defined functions

a) Library functions : Library functions are predefined functions and supplied along with the compiler and these can be used in any C program. They are also known as Built-in functions and all these functions are available in C library. Ex: scanf( ), printf( ), gets( ) etc.

b) User defined Functions: The functions that are defined by the users are called as user-defined functions. Ex: main( ) is an example of user defined function.

ELEMENTS OF USER DEFINED FUNCTIONS (OR) FUNCTION COMPONENTS:

Every function has the following elements associated with it.

i. Function declaration (or) Function prototype

ii. Function call

iii. Function definition (or) Function implementation

FUNCTION DECLARATION

All functions in a C program must be declared, before they are invoked. A function declaration consists of four parts.

a) return_type
b) function_name
c) parameter list
d) terminating semicolon

The general form of declaring the function is return_type function_name(parameter list);
Ex: int maximum(int m, int n);

FUNCTION CALL

A function can be called by writing the function name followed by a list of actual parameters, if any, enclosed in the parenthesis and terminated by a semicolon.

Ex:
```
main( )
{
int z, x, y;
scanf("%d%d",&x,&y)
;
z=maximum(x,y); /*function call*/
printf("z=%d",z);
```

FUNCTION DEFINITION

The function definition is an independent program module that iswritten to implement t

specific task.

A function definition has two parts namely

1. Function body
2. Function header

The general form of a function definition to implement two parts is given below.

return_type function_name (parameter list)

{

Local variable

declaration;Executable

statement1; Executable

statement2;

----

return statement;

FUNCTION HEADER

The function header consists of three parts

a) return_type
b) function_name
c) parameter(formal) list

Return type: A function may or may not send back any value to the calling function. A function
type specifies the data type of value that the function is expected to return to the program calling
the function. If a function is not returning anything, then we need to specify the function type as
void.

FUNCTION NAME

The function name is any valid C identifier and the name should be appropriate to the task performed by the function.

Parameter list: The parameter list contains declaration of variables separated by commas and surrounded by parenthesis. The parameter list declares the variables that will receive the data sent by the calling program. They serve as input data to the function to carry out the specified
task.

The function body contains the declarations and statements necessary for performing the required task. The body enclosed in braces, contains three parts in the order given below.

a) Local declarations that specify the variables needed by the function

b) Function statements that perform the task of the function

FUNCTION BODY

c) A return statement that returns the value evaluated by the function.

If a function does not return any value, we can omit the return statement. But the functiontype should be specified as void.

```
Ex: int sum(int a, int b) /*function header*/
{  int rst;
   rst=a+b;                    /*function
   body*                      /return(rst);
}
```

RETURN STATEMENT

A function may or may not send back any value to the calling function. It
can be done through the return statement. When a return statement is executed, the contr
transferred to the calling function.

The return statement can take any one of the following forms

return;
(or)
return (expression);

The statement return; does not return any value and it acts as the closing brace of
thefunction.

The statement return (expression); return the value of the expression and control is transferred to the calling function.

A function may have more than one return statements. This situation arises when the value returned is based on the certain condition. for example

```
if(x<=0)
return (0);
else
return (1);
```

PARAMETERS

Parameters (also known as arguments) are used in three places

o in function declaration
o in function call
o in function definition

The parameters used in function declaration and function definition are called formal parameters and the parameters used in function call are called actual parameters. Actual parameters used in a calling statement may be simple constants, variables or expressions. The formal parameters and actual parameters must match exactly in type, number and order. But their names need not match.

CATEGORY OF FUNCTIONS

Depending on whether arguments are present (or) not and whether a
value is returned (or) not, the functions are classified into following four categories.

Category 1: functions with no arguments and no return value
Category 2: functions with no arguments and return value
Category 3: functions with arguments and no return value
Category 4: functions with arguments and

CATEGORY 1: FUNCTIONS WITH NO ARGUMENTS AND NO RETURN VALUE

When a function has no arguments, it does not receive any data from the calling function. Similarly, when it does not return a value, the calling function does not receive any data from the called function. So there is no data transfer between the calling function and the called function.

CATEGORY 2: FUNCTIONS WITH NO ARGUMENTS AND RETURN VALUE

: When a function has no arguments, it does not receive any data from the calling function. When a function returns a value, the calling function receives data from the called function. So there is one way data communication from the called function to calling function.

CATEGORY 3: FUNCTIONS WITH ARGUMENTS AND NO RETURN VALUE

When a function has arguments, it receives data from the calling function. Similarly, when it does not return a value,

he calling function does not receive any data from the called function. So there is one way data communication from the calling function to the called function.

CATEGORY 4: FUNCTIONS WITH ARGUMENTS AND RETURN VALUE

When a function has arguments, it receives data from the calling function. Also, when it returns a value, the calling functions receive data from the called function. So there is two way data communication between the calling function and the called function.

THANK YOU

THANK U

FOR LISTENING