Unit IV – Part 01 (Function)

Lecture Slide

AS2023

Objectives

By the end of this session, students will be able to:

- Define function
- Differentiate between predefined and user defined function
- Explain the need for function
- Identify the characteristics of function
- Identify and explain elements of function
- Solve problem using user defined functions
- Use appropriate return type
- Explain and implement user defined functions

Function

- Block of statements to perform a specific task
- A function is known with various names like a method or a sub-routine or a procedure, modules, subprograms and so on.
- As always, a function is a module of code that takes information in (parameters), does some computation, and (usually) returns a new piece of information based on the parameter information.
- Can be classified either as library functions/inbuilt functions or user defined function
- **add( )** is a example of user-defined functions and **scanf( ) & printf( )** are examples of *library function*

Need for function

- Subprograms are Easier to understand, debug and test
- Facilitates TOP-DOWN modular programming
- The length of the source program can be reduced
- Easier to locate and isolate faulty function for further investigation
- A function may be used by many other programs

Multi-functioned Program

- Once a function has been designed and packed, it can be treated as a '***black box***'
- Consider a set of statements as shown below

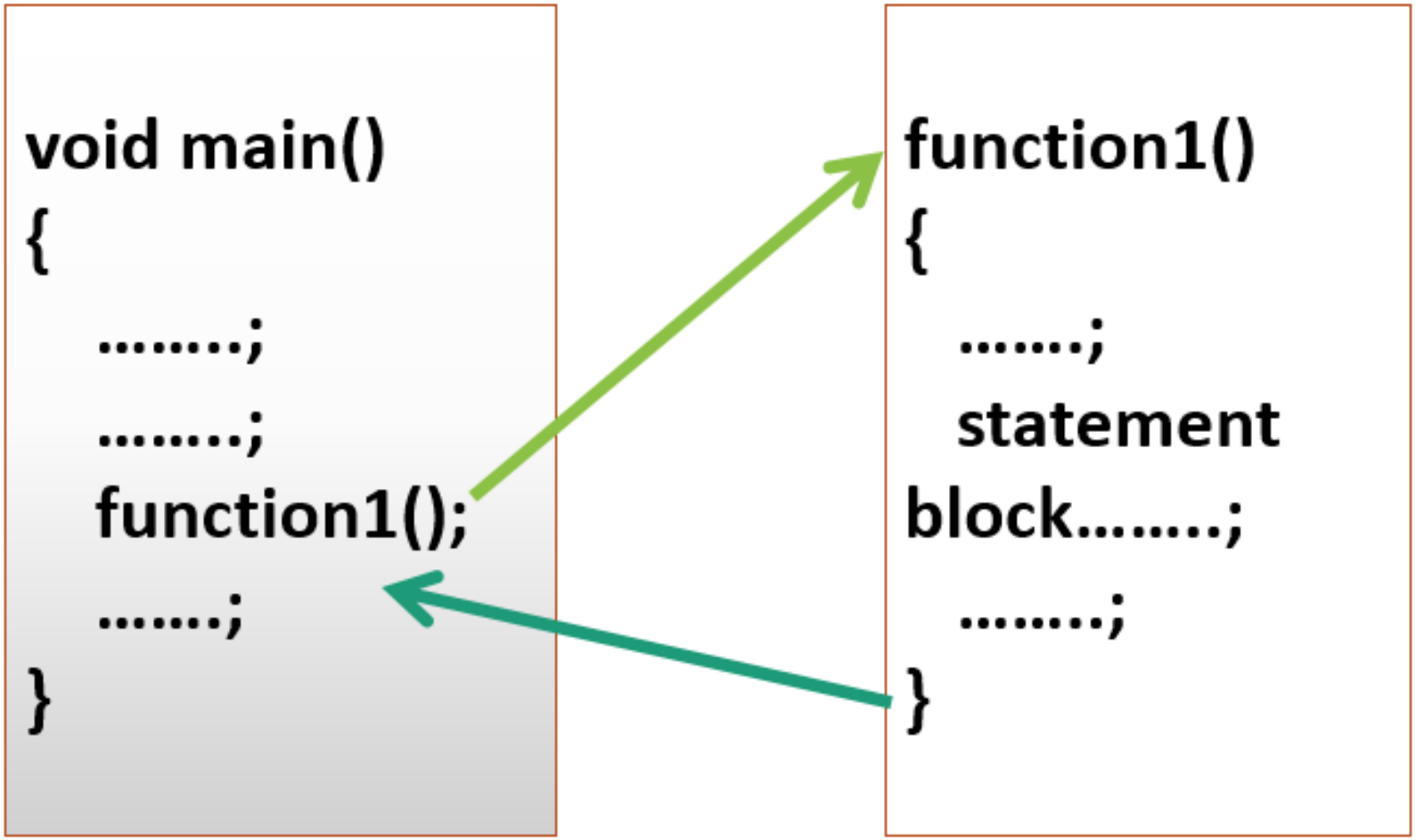
```
void printline(void){
   int i;
   for(i=1; i<40; i++){
      printf("_");
   }
}
```

Multi-functioned Program

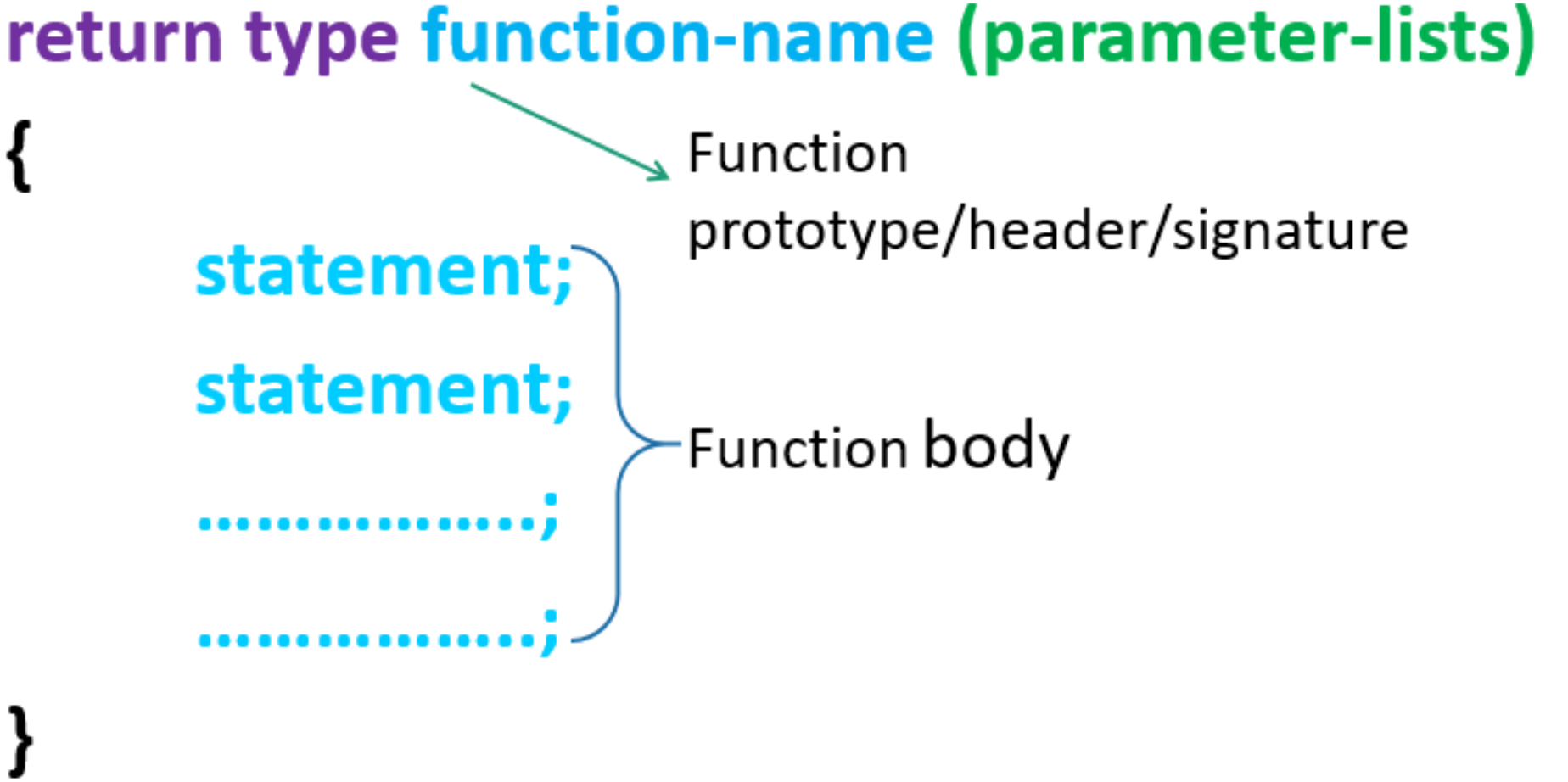
- The function can be used in the program as

```
void printline(void); /*declaration*/
int main( ){
        printline( );
        printf("This illustrates the use of C function\n");
        printline( );
}
void printline(void){
        int i;
        for(i=1; i<40; i++){
                printf("_");
        }
}
```

Function



Basic Structure of Function



Characteristics

- Each function should do only one thing
- Communication between functions is allowed only by a calling function
- A function can be called by one and only one higher module
- No communication can take place directly between modules that do not have *calling-called relationship*
- all modules are designed as *single-entry*, *single-exit* systems using control structures

Elements of Function

- Functions are classified as one of the derived data types in C
- We can define function and use them like a variables in C
- It holds some similarities with variable
  - Both are identifiers and should adhere to the rules for identifiers
  - Both should have type associated with them
  - Both should be declared and defined before their usage in the program
- In order to establish a function, three elements are required that are related to functions
  - Function definition
  - Function call
  - Function declaration

Function Definition

- Is an independent program module that is especially written to implement the requirements of the function
- It is also know as ***function implementation***
- Shall include the following elements

1. Function type
2. Function name
3. List of parameters

Function header

4. Local variable declaration
5. Function statements
6. A return statement

Function body

Function Definition

- General format of function definition

function_type *function_name(parameter list) //Function header*

{

local variable declaration;

executable_statement;

……..

……..

return statement;

}

function body

Function Header

- Consist of three parts as discussed in previous slide
- Type & Name
  - Function types specifies the type of value to be or that function is expected to return to the program calling the function
  - if the return type is not explicitly specified, C will assume that it is an integer type
  - if the function is not returning anything, then we need to specify the return type as **void**
  - it is good programming practice to code return type explicitly even when it is integer type
  - Value return is output produced by the function
  - Function name can be any valid C identifiers
  - Name should be appropriate to the task performed by the function

Function Declarations

- Like variables, all functions in C program must be declared before they are invoke
- Function declaration is known as function prototype and takes the following form

```
function_type function_name (parameter list);
```

- If function doesn't take any values and doesn't return any values, its prototype is written as

```
void display (void);
```

- A prototype declaration can be placed in two places in a program
  - Write the functions above main function
  - Write the functions prototype before main function

Function Header

- 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
  - Since they represent the input values, they are often referred to as formal parameters
- Examples

```
1. float quadratic(int a, int b){
          function body;
   }
2. double power(double a, int b) {
           function body;
   }
3. int sum(int a, int b){
          function body;
   }
```

Remember:

1.there is no semicolon after the closing parenthesis

2. *every parameters is separated with coma*
3. *declaration of parameters cannot be combined. Example* ***int sum(int a,b)*** *is* ***illegal***

Function body

- Contains the statement block (declarations and executable statements) necessary for performing the required task
- Contains three parts
  - Local variable declaration
  - Functions statement
  - Return statements
- If the function doesn't return any values, we can omit return statements but note to declare function type or return type as **void**
- when the function reaches its return statement, the control is transferred back to the calling program. In absence, the closing brace act as a **void return**

Function body cont..

- Return values and their types
  - Function may or may not send back any values to the calling function
  - If it does, it is done through return statement
  - The function can only return one value per call at most
  - The return statement takes the following form

```
return;
Or
return(expression);
```

- A function may have more than one return statement. This arises when the value returned is based on certain conditions

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

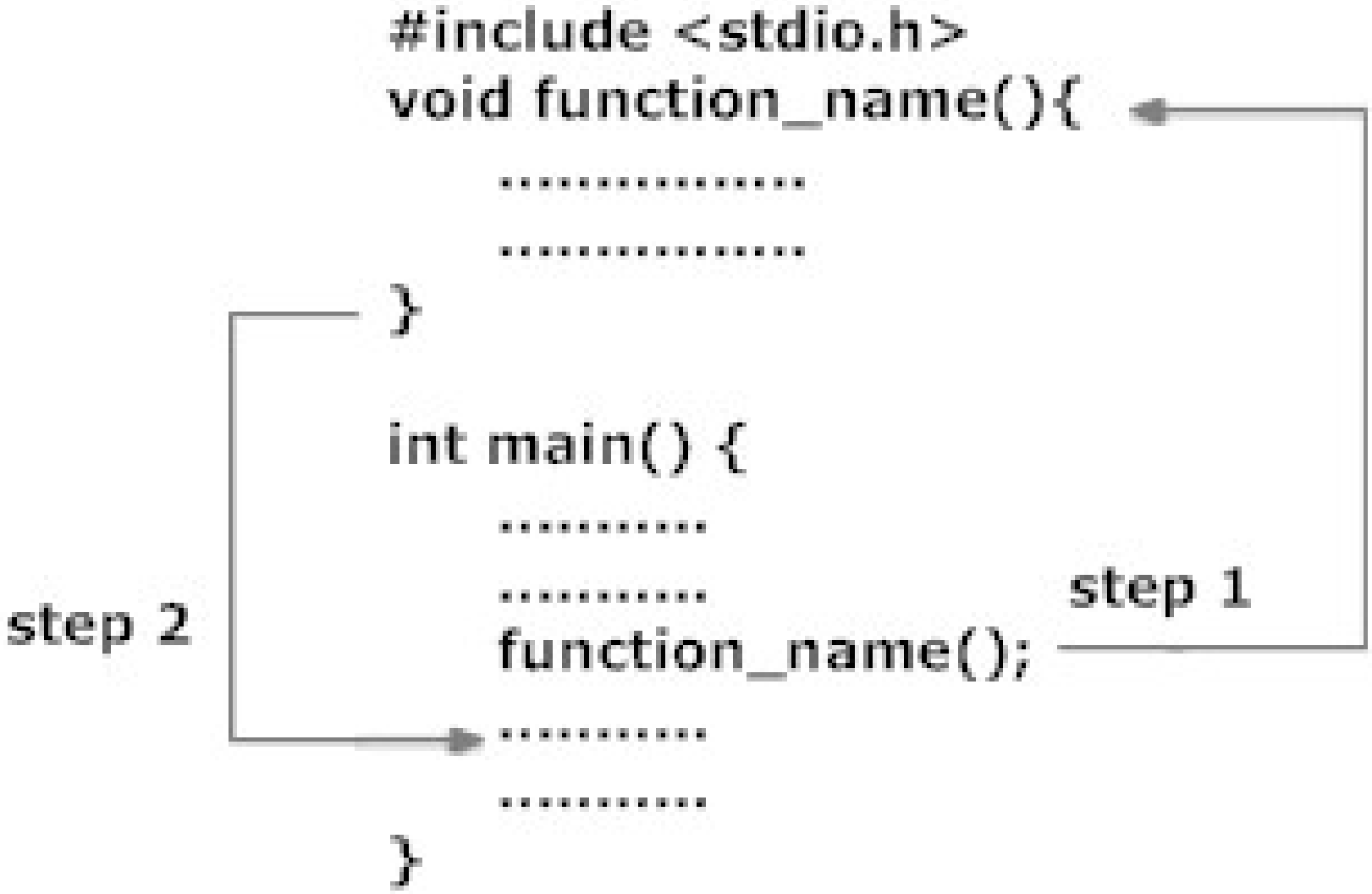
Function call

- A function can be called by simply using the function name followed by list of actual parameters (arguments), if any, in the parenthesis

```
int main( ){
    int y;
    y=mul(10,5); /*function call*/
    printf("%d\n",y);
}
```

- A function which returns a value can be used in expression like any other variable.
- However, a function cannot be used on the right side of an assignment statement
- The actual parameter should match with the formal parameter in type, order and number

How user-defined function work?



Demonstration

- WAP program that uses function to compare two user input number.

Thank you