Chapter 3
Functions

The process of breaking large programs into smaller sub programs is called **modularisation**. The sub programs are generally called functions.

Merits of modular programming

- Reduces the size of the program.
- Less chance of error occurrence.
- Reduces programming complexity.
- Improves reusability.

Function is a named unit of statements in a program to perform a specific task. Classified into predefined (built-in) functions and user-defined functions.

Parameters or Arguments of functions

These are the data required for performing the task given to the function. These are provided within the pair of parentheses of the function name. There are certain functions which give results after performing the task. This result is known as *value returned* by the function. A function can return only one value, but some functions do not return any value.

Predefined functions

These are the ready-to-use functions available in header files. So, the respective header file is to be included to use a built-in function.

Type	Functions	Syntax / Example	Operation
Console functions (cstdio / stdio.h)	**getchar()**	`ch=getchar()`	To input a character
	putchar()	`putchar('a')`	To display a character
Stream functions (iostream)	**get()**	`cin.get(ch)` `cin.get(str,10)`	To input a character To input a string of maximum 10 characters
	getline()	`cin.getline(str,10)`	To input a string of maximum 10 characters
	put()	`cout.put('a')`	To display a character
	write()	`Cout.write(str,10)`	To display a string of maximum 10 characters

String Functions (cstring / string.h)	**strlen()**	strlen(string)	To find the length of a string.
	strcpy()	strcpy(string1, string2)	To copy one string into another
	strcat()	strcat(string1, string2)	To append (concatenate) string2 to string1
	strcmp()	strcmp(string1, string2)	To compare two strings. • Returns 0 if string1 and string2 are same. • Returns a –ve value if string1 is alphabetically lower than string2. • Returns a +ve value if string1 is alphabetically higher than string2.
	strcmpi()	strcmp(string1, string2)	Same as strcmp(), except that the case of the two strings are ignored.
Mathematical Functions (cmath / math.h)	**abs()**	abs(int)	To find the absolute value of an integer.
	fabs()	fabs(float)	To find the absolute value of a floating point number.
	sqrt()	sqrt(double)	To find the square root of a number.
	pow()	pow(double, int)	To find the power of a number. It takes two arguments **x** and **y**. Returns the value of $\mathbf{x}^{\mathbf{y}}$.
Character Functions (cctype / ctype.h)	**isupper()**	isupper(char)	To check whether a character is in upper case or not. The function returns non-zero if the given character is in uppercase, and 0 otherwise.
	islower()	islower(char)	To check whether a character is in lower case or not. The function returns non-zero if the given character is in lowercase, and 0 otherwise.
	isalpha()	isalpha(char)	To check whether a character is alphabet or not. The function returns non-zero if the given character is an alphabet, and 0 otherwise.

	isdigit()	isdigit(char)	To check whether a character is digit or not. The function returns non-zero if the given character is a digit, and 0 otherwise.
	isalnum()	isalnum(char)	To check whether a character is alphanumeric or not. The function returns non-zero if the given character is alphanumeric, and 0 otherwise.
	toupper()	toupper(char c)	This function is used to convert the given character into its uppercase.
	tolower()	tolower(char c)	This function is used to convert the given character into its lowercase.

User defined functions

The syntax of a function definition is given below:

```
data_type function_name(argument_list)
{
    statements in the body;
}
```

The `data_type` is any valid data type of C++. The `function_name` is a user-defined word (identifier). The `argument_list`, which is optional, is a list of parameters, i.e. a list of variables preceded by data types and separated by commas. The `body` comprises of C++ statements required to perform the task assigned to the function.

A **function prototype** is the declaration of a function by which compiler is provided with the information about the function such as the name of the function, its return type, the number and type of arguments, and its accessibility. The following is the format:

```
data_type function_name(argument_list);
```

Arguments or parameters are the means to pass values from the calling function to the called function. The variables used in the function definition as arguments are known as *formal arguments*. The constants, variables or expressions used in the function call are known as *actual (original) arguments*. If formal arguments are initialized with values, they are called *default arguments*.

Two type of Function Calling

Call by Value Method	Call by Reference Method
• Ordinary variables are used as formal parameters. • Actual parameters may be constants, variables or expressions. • The changes made in the formal arguments do not reflect in actual arguments. • Exclusive memory allocation is required for the formal arguments.	• Reference variables are used as formal parameters. • Actual parameters will be variables only. • The changes made in the formal arguments do reflect in actual arguments. • Memory of actual arguments is shared by formal arguments.

Scope and Life of Variables and Functions

Scope & life	Local	Global
Variables	• Declared within a function or a block of statements. • Available only within that function or block. • Memory is allocated when the function or block is active and freed when the execution of the function or block is completed.	• Declared outside all the functions. • Available to all functions in the program. • Memory is allocated just before the execution of the program and freed when the program stops execution.
Functions	• Declared within a function or a block of statements and defined after the calling function. • Accessible only within that function or the block.	• Declared or defined outside all other functions. • Accessible by all functions in the program

Questions from Previous Years' Question Papers

1. Consider the following code:

```
char S1[] = "program";
char S2[] = "PROGRAM";
int n;
n = strcmpi(S1, S2);
```

What is the value of n?

(a) n = 0 (b) n = 1 (c) n > 1 (d) n < 0 (1) (March 2016)

2. Explain any three stream functions for I/O operations. (3) (March 2016)

3. Write a code to do the following:
   (a) A function named largest accepts two integer numbers and return the largest number.
   (b) Use the function to find the largest of two numbers. (3) (March 2016)
4. _____function is used to check whether a character is alphanumeric.
   (a) isdigit() (b) isalnum() (c) isupper() (d) islower() (1) (SAY 2016)
5. Explain any three string functions with examples. (3) (SAY 2016)
6. Write a function that accepts 3 numbers of type float as arguments and return the average of three numbers. Write program which use this function to find the average of three numbers using C++. (3) (SAY 2016)
7. Write the output of the following C++ code segment:

```
char S1[10] = "Computer";
char S2[15] = "Applications";
strcpy(S1,S2);
cout<<S2;
```

   (1) (March 2017)
8. Explain two stream functions for input operation with examples. (2) (March 2017)
9. "Initialized arguments are called default arguments". Using this concept, write the function prototype and definition of a user-defined function Sum() which accepts two or three integer numbers and return their sum. (3) (March 2017)
10. Explain three string functions in C++. (3) (SAY 2017)
11. Write a program using a function to interchange the values of two variables. (Use call-by-reference method for passing arguments) (3) (SAY 2017)
12.
```
char s1[10] = "hello", s2[10];
strcpy(s2,s1);
cout<<s2;
```
   What will be the output?
   (a) hello (b) hel (c) hell (d) No output (1) (SAY 2017)