Freebooks.pk

Free Notes • Past Papers • Guides for Students

2nd Year Computer Science — Punjab Board

CHAPTER 13

Functions in C

Key Concepts · Definitions · Short & Long Questions · MCQs · Diagrams · Exam Tips

This chapter covers Functions in C from the 2nd Year (ICS Part-II) Computer Science syllabus of the Punjab Curriculum and Textbook Board (PTB/PCTB). It explains structured (modular) programming, the importance and types of functions, the function header, body and return statement, function prototypes, calling a function, and the scope of local and global variables. These notes are prepared by freebooks.pk.

Functions are the building blocks of C programs. A function is a self-contained piece of code with a specific purpose, and dividing a program into functions is the basis of structured programming.

Learning Objectives

- Explain structured (modular) programming and the importance of functions.
- Distinguish built-in and user-defined functions.
- Describe the function header, body and return statement.
- Explain the function prototype.
- Explain how a function is called.
- Explain local and global variables and their scope.

Key Concepts

Structured Programming and Functions

Functions are the building blocks of C programs; a function is a self-contained piece of code with a specific purpose. When the whole program logic is placed inside a single main function, the style is called unstructured programming. Structured programming is a modular approach in which the program logic is divided into a number of smaller modules or functions, and the main function calls these functions where they are needed. This makes programs easier to write, read and maintain.

Importance of Functions

Functions provide many benefits. They make programs easier to understand and maintain, because the main program becomes a series of function calls instead of countless lines of code. They increase the reusability of code, since well-written functions can be used in many programs, as the C standard library shows. They allow different programmers to divide a large project by writing different functions, enabling parallel development. A function can be executed as many times as needed from different places, and when an error occurs only the affected function has to be debugged rather than the whole program.

Types of Functions

There are two types of functions in C: built-in functions and user-defined functions. Built-in (library) functions are predefined functions packaged in libraries that provide ready-made functionality, such as printf and scanf in stdio.h, or getch and getche in conio.h; to use one, the header file containing its declaration must be included. User-defined functions are

functions that the programmer writes, according to the nature of the problem, when the built-in functions are not sufficient.

The Function Header

Every function has a function header, which is its first line and identifies the function. Its general form is return_type FunctionName (parameter_list). The header states three things: the type of the return value, the name of the function, and the parameters enclosed in parentheses. The return_type may be any valid data type; if the function returns no value the return type is the keyword void, and a function with no parameters may write void, or leave the parentheses empty, as its parameter list.

The Function Body and return Statement

The function body follows the header and is enclosed in curly braces; it contains the variable declarations and the program logic and makes use of the arguments passed to the function. The return statement, written as return expression;, specifies the value returned by the function. When it is executed the expression is evaluated and returned, and the function stops immediately even if statements remain. If the return type is void, no return statement is needed.

Function Prototype

The compiler must know about a function before it is used, which is why header files are included for built-in functions. A function prototype is a statement that gives the compiler the basic information it needs to check and use a function correctly: the return type, the function name and the parameters. Its general form is the same as the function header but with a semicolon at the end, return_type FunctionName (parameter_list);. Prototypes for functions must appear before the function call, and are usually placed at the top of the source file just before the main function.

Calling a Function

A function call is the mechanism used to invoke a function to perform its task, and it can be made at any point in the program. To call a function, the function name, the required arguments and the statement terminator (a semicolon) are written. When a function call is executed, control is transferred to the called function, memory is allocated to the variables declared in it, and the statements in its body are executed; after the last statement, control returns to the calling function.

Local and Global Variables and Scope

The scope of a variable is the region of a program in which it can be accessed; a variable cannot be used outside its scope, and doing so causes a compiler error. Variables declared inside a block (a pair of curly braces) are local variables with local scope, valid from their declaration to the end of that block; their lifetime is the period during which control stays in that block. A global variable is declared outside all functions and has global scope, so it is accessible to the whole program from its declaration onwards.

Important Definitions

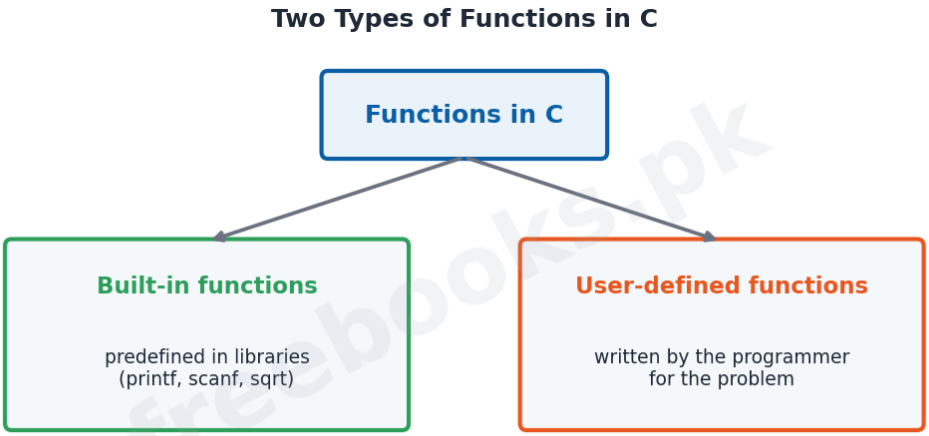
Term	Definition
Function	A self-contained piece of code that performs a specific task.
Structured programming	Dividing a program into smaller modules or functions.
Built-in function	A predefined function packaged in a library (e.g. printf).
User-defined function	A function written by the programmer for a specific problem.
Function header	The first line stating return type, name and parameters.
return statement	A statement that returns a value and ends the function.
Function prototype	A declaration of a function's return type, name and parameters, ending with a semicolon.
Scope	The region of a program in which a variable can be accessed.

Key Facts & Rules

Item	Detail
Function definition	return_type FunctionName(parameter_list) { body }
No value / no parameter	void FunctionName(void) or void FunctionName()
return statement	return expression;
Function prototype	return_type FunctionName(parameter_list); (ends with ;)
Function call	FunctionName(arguments);
Local variable	declared in a block; scope = that block only
Global variable	declared outside all functions; scope = whole program

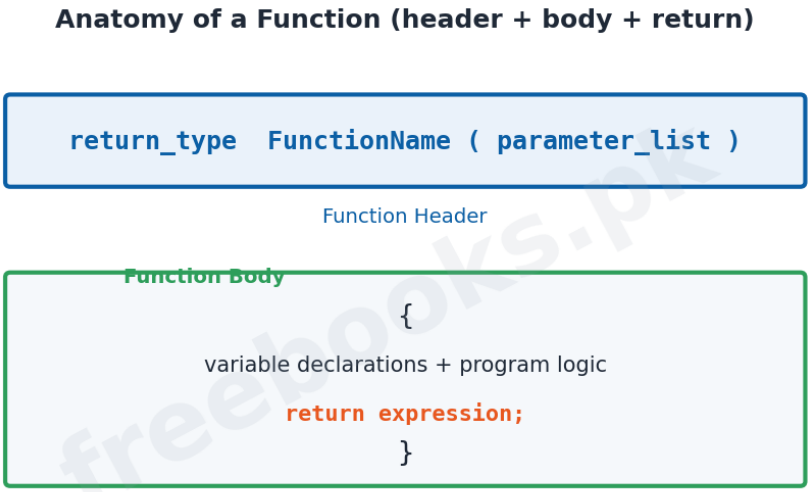
Diagrams & Illustrations

Two types of functions: the two types of functions in C, built-in (library) functions and user-defined functions.



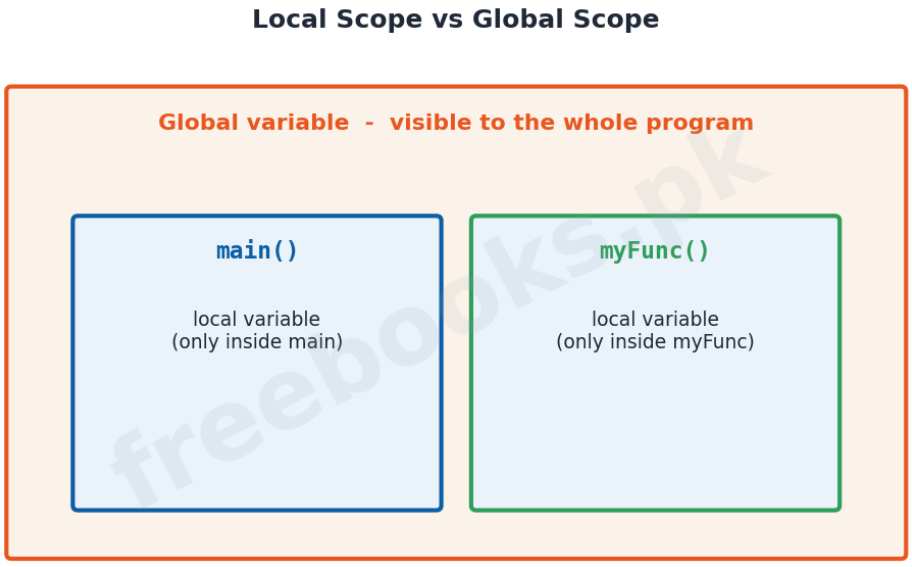
Two types of functions

Anatomy of a function: the parts of a function, the header and the body with its return statement.



Anatomy of a function

Local vs global scope: how a local variable is visible only in its block while a global variable is visible to the whole program.



Local vs global scope

Solved Examples & Practice

A function header

int add(int a, int b) is a header for a function named add that takes two integers and returns an integer.

A void function

void greet(void) is a function that takes no parameters and returns no value.

A prototype

int add(int, int); written before main is the prototype of the add function.

A function call

sum = add(5, 3); calls add with arguments 5 and 3 and stores the returned value in sum.

Short Questions & Answers

Q1. What is a function?
Ans. A self-contained piece of code that performs a specific task.

Q2. Name the two types of functions in C.
Ans. Built-in (library) functions and user-defined functions.

Q3. What is a function header?
Ans. The first line of a function stating its return type, name and parameters.

Q4. What is the use of the return statement?

Ans. It returns a value from the function and stops the function's execution.

Q5. What is a function prototype?

Ans. A statement giving the compiler the return type, name and parameters of a function, ending with a semicolon.

Q6. What is the scope of a local variable?

Ans. From its declaration to the end of the block in which it is declared.

Long Questions & Answers

Q1: Explain structured programming, the importance of functions and the two types of functions.

A function is a self-contained piece of code with a specific purpose, and functions are the building blocks of C programs. In the simplest programs the whole logic is placed inside a single main function, a style known as unstructured programming; structured programming, by contrast, is a modular approach in which the program logic is divided into a number of smaller modules called functions, and the main function calls these functions wherever they are needed. This idea is inspired by hardware manufacturing, where a device is built from replaceable components that can be combined and, if faulty, replaced individually. Dividing a program into functions brings several important benefits. First, it makes programs much easier to understand and maintain, because the main program becomes a short series of function calls rather than countless lines of code. Second, it increases the reusability of code, since a well-written function can be used again in many programs, and the C standard library is itself a large collection of reusable functions. Third, it supports parallel development, because different programmers working on one large project can divide the work by each writing different functions. Fourth, a function can be executed as many times as necessary from different places in the program, avoiding repeated code. Finally, it simplifies debugging, because when an error arises only the affected function needs to be examined rather than the whole program. There are two types of functions in C. Built-in or library functions are predefined functions packaged in libraries, such as printf and scanf in the standard input/output library and getch and getche in the console input/output library; to use one, the header file that declares it must be included in the program. User-defined functions are functions that the programmer writes when the built-in functions are not sufficient for the problem at hand.

Q2: Describe the structure of a function in C: the header, the body, the return statement and the function prototype.

Every function in C has almost the same basic structure, consisting of a function header followed by the function body enclosed in curly braces. The function header is the first line of the definition and identifies the function; its general form is return_type FunctionName (parameter_list), and it states three things: the type of the value the function returns, the name of the function, and the parameters of the function enclosed in parentheses. The return type may be any valid data type, but if the function does not return a value the return type is written as the keyword void, and a function that takes no parameters may write void

as its parameter list or simply leave the parentheses empty, so that a function returning nothing and taking nothing may be written void FunctionName(). The function body follows the header and is enclosed in curly braces; it contains the declarations of the variables used by the function and the program logic, and it makes use of the arguments passed to the function. Inside the body the return statement, whose general form is return expression;, is used to specify the value the function sends back to the caller; when it is executed the expression is evaluated and returned as the value of the function, and the function stops immediately even if further statements remain, while a void function needs no return statement. Because the compiler must know about a function before it is used, C also requires a function prototype, which is a statement that provides the compiler with the return type, the name and the parameter list of the function. The prototype looks exactly like the function header but ends with a semicolon, as in return_type FunctionName (parameter_list);, and it must appear before the function is called, usually at the top of the source file just before the main function.

Q3: Explain how a function is called, and describe local and global variables and their scope.

A function call is the mechanism used to invoke a function so that it performs its task, and a call can be made at any point in the program. To call a function, the programmer writes the function name, the arguments the function requires, and the statement terminator, which is a semicolon. When a function call statement is executed, control is transferred from the calling function to the called function; memory is then allocated to the variables declared inside the called function, and the statements in its body are executed one by one. After the last statement of the function, or when a return statement is reached, control returns to the calling function, which continues from where it left off. Closely related to functions is the idea of the scope of a variable, which is the region of a program in which the variable can be accessed; the name of a variable is only valid within its scope, and any attempt to refer to it outside its scope causes a compiler error. Variables declared inside a block, that is within a pair of curly braces such as the body of a function or of an if statement, are called local variables and have local scope, which extends from the point where the variable is declared to the end of the block containing the declaration. The lifetime of a local variable is the period during which control remains inside that block; as soon as control leaves the block the variable is destroyed, meaning the memory allocated to it is returned to the operating system. A global variable, in contrast, is declared outside all functions and has global scope, so it is accessible to the whole program from the point of its declaration onwards. Local variables keep functions independent of one another, while global variables allow information to be shared across the whole program.

MCQs with Answers

1. The building blocks of C programs are:
(a) loops (b) functions (c) arrays (d) headers
Correct Answer: (b) functions. Functions.

2. Dividing a program into smaller modules is called:

(a) unstructured programming (b) structured programming (c) looping (d) casting
Correct Answer: (b) structured programming. Structured programming.

3. printf and scanf are examples of:
(a) user-defined functions (b) built-in functions (c) global variables (d) prototypes
Correct Answer: (b) built-in functions. Built-in functions.

4. A function written by the programmer is a:
(a) built-in function (b) user-defined function (c) library function (d) header
Correct Answer: (b) user-defined function. User-defined function.

5. If a function returns no value its return type is:
(a) int (b) void (c) float (d) char
Correct Answer: (b) void. void.

6. The statement that returns a value from a function is:
(a) break (b) return (c) exit (d) stop
Correct Answer: (b) return. return.

7. A function prototype ends with a:
(a) brace (b) semicolon (c) comma (d) colon
Correct Answer: (b) semicolon. Semicolon.

8. A variable declared inside a block is a:
(a) global variable (b) local variable (c) constant (d) parameter
Correct Answer: (b) local variable. Local variable.

9. A variable accessible to the whole program is a:
(a) local variable (b) global variable (c) temporary variable (d) register
Correct Answer: (b) global variable. Global variable.

10. Using a variable outside its scope causes a:
(a) warning only (b) compiler error (c) new variable (d) loop
Correct Answer: (b) compiler error. Compiler error.

Quick Revision Summary

- Function = self-contained code with a specific purpose; building block of C programs.
- Structured programming divides logic into functions called by main.
- Two types: built-in (library, e.g. printf) and user-defined functions.
- Function = header (return_type name(params)) + body { ... return expr; }.
- Prototype = header ending with a semicolon, placed before the call.
- Local variable: scope = its block; global variable: scope = whole program. Notes by freebooks.pk.

Exam Tips

- Define a function and explain structured programming.
- List the benefits (importance) of functions.

- Differentiate built-in and user-defined functions with examples.
- Describe the header, body and return statement.
- Explain the function prototype and where it is placed.
- Differentiate local and global variables by their scope.