Freebooks.pk

Free Notes • Past Papers • Guides for Students

2nd Year Computer Science — Punjab Board

CHAPTER 8

Getting Started with C

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

This chapter covers Getting Started with C from the 2nd Year (ICS Part-II) Computer Science syllabus of the Punjab Curriculum and Textbook Board (PTB/PCTB). It introduces the C language, the structure of a C program, comments and header files, the compilation process and a first simple program. These notes are prepared by freebooks.pk.

C is a powerful and widely used programming language. This chapter introduces C and shows how a C program is written, structured and turned into a running program.

Learning Objectives

- State what C is and its features.
- Describe the structure of a C program.
- Explain header files and comments.
- Describe the compilation process.
- Write and understand a simple C program.
- State the C character set and keywords.

Key Concepts

The C Language

C is a general-purpose, high-level programming language developed by Dennis Ritchie in the early 1970s. It is powerful, efficient and portable, meaning a C program can run on many different computers with little change, and it is used to write operating systems, application software and much other software. Because it teaches the fundamentals of programming clearly, C is widely used as a first serious programming language.

Structure of a C Program

A C program has a definite structure. It usually begins with preprocessor directives such as #include, which bring in header files. It then contains one or more functions, and every C program must have a function called main(), where execution begins. Inside a function, statements are written between braces { and }, each statement normally ending with a semicolon. A simple program consists of the header, the main() function, some declarations and statements, and a return statement.

Header Files and Comments

A header file is a file, included with a #include directive, that contains ready-made declarations the program needs; for example #include <stdio.h> brings in the standard input/output functions such as printf and scanf. Comments are explanatory notes added to the source code to make it easier to understand; they are ignored by the compiler and are written between /* and */ (or after // for a single line). Good comments make a program easier to read and maintain.

The Compilation Process

A C program is written as source code in a file (with the extension .c), but the computer cannot run this directly. A special program called a compiler translates the source code into machine code, first producing object code and then, after linking, an executable file (with the extension .exe on Windows) that the computer can run. If the source contains errors, the compiler reports them so that the programmer can correct them before the program will compile.

A First C Program

A simple first C program prints a message on the screen. It includes the header stdio.h, defines the main() function, and uses the printf() function to display text; for example printf("Hello, World!"); prints the words Hello, World! on the screen. The program ends with return 0; inside main(). Running this program shows how source code produces visible output.

The C Character Set

The C character set is the set of characters that may be used to write a C program. It includes the letters A to Z and a to z, the digits 0 to 9, special symbols such as + - * / = ( ) { } ; and white-space characters such as spaces and new lines. These characters are combined to form the words and symbols of the language.

Keywords and Identifiers

Keywords are reserved words that have a fixed, special meaning in C, such as int, float, if, else, while and return; they cannot be used for any other purpose. Identifiers are the names that the programmer gives to things such as variables and functions; an identifier must begin with a letter or underscore, may contain letters, digits and underscores, and must not be a keyword. Choosing clear identifiers makes a program easier to understand.

Important Definitions

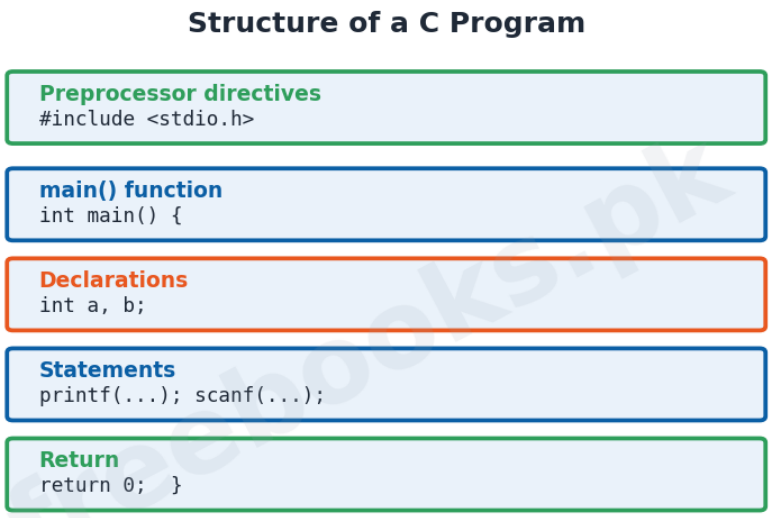
Term	Definition
C	A general-purpose high-level programming language by Dennis Ritchie.
main()	The function where a C program begins execution.
Preprocessor directive	A line beginning with # (e.g. #include) processed before compilation.
Header file	A file included to provide ready-made declarations (e.g. stdio.h).
Comment	An explanatory note in the code, ignored by the compiler.
Compiler	A program that translates source code into machine code.
Keyword	A reserved word with a fixed meaning (e.g. int, if, while).
Identifier	A name given by the programmer to a variable or function.

Key Facts & Rules

Item	Detail
C	High-level language by Dennis Ritchie (~1972)
Program start	main() function
Include a header	#include <stdio.h>
Comment	/* ... */ or // ... (ignored by compiler)
Compile	source (.c) -> object -> executable (.exe)
Print text	printf("...");

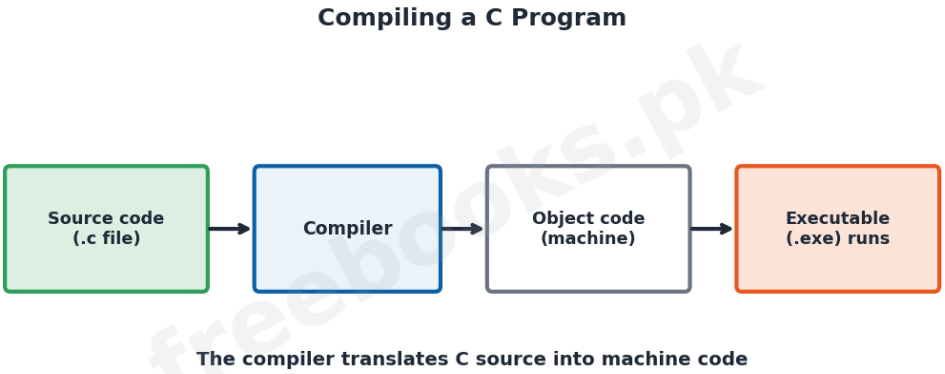
Diagrams & Illustrations

Structure of a C program: the structure of a C program: preprocessor directives, the main() function, declarations, statements and a return.



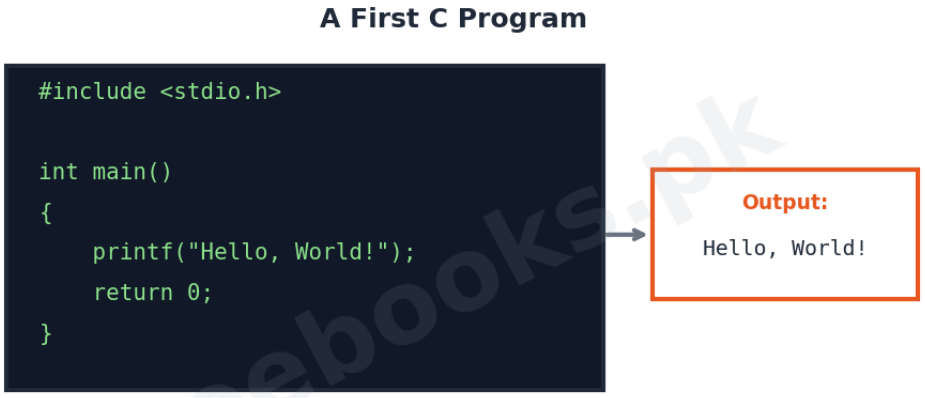
Structure of a C program

Compiling a C program: the compilation process turning C source code into object code and then an executable file.



Compiling a C program

A first C program: a simple C program that prints Hello, World! and its output.



A first C program

Solved Examples & Practice

Include a header

To use printf and scanf, the program must include the header with the line #include <stdio.h>.

Print a message

printf("Pakistan"); displays the word Pakistan on the screen.

Valid identifier

'marks1' is a valid identifier, but '1marks' is not because it starts with a digit, and 'int' is not because it is a keyword.

Compile then run

A .c source file is first compiled into an .exe file, which is then run to produce the output.

Short Questions & Answers

Q1. Who developed C and when?
Ans. C was developed by Dennis Ritchie in the early 1970s (around 1972).

Q2. What is the main() function?
Ans. The function where the execution of every C program begins.

Q3. What is a header file? Give an example.
Ans. A file included to provide ready-made declarations, such as stdio.h for input/output functions.

Q4. What is a comment in C?
Ans. An explanatory note added to the code that is ignored by the compiler, written between /* and */ or after //.

Q5. What is the role of the compiler?
Ans. To translate the C source code into machine code (an executable) that the computer can run.

Q6. Differentiate a keyword and an identifier.
Ans. A keyword is a reserved word with a fixed meaning (e.g. int); an identifier is a name the programmer gives to a variable or function.

Long Questions & Answers

Q1: Describe the C language and the structure of a C program.

C is a general-purpose, high-level programming language that was developed by Dennis Ritchie at Bell Laboratories in the early 1970s. It became extremely popular because it is powerful and efficient, giving the programmer a great deal of control, and because it is portable, which means that a program written in C can be run on many different kinds of computer with little or no change. C has been used to write operating systems, compilers, application programs and much other software, and because it clearly teaches the fundamental ideas of programming it is very widely used as a first serious programming language for students. Every C program follows a definite structure. It usually begins with one or more preprocessor directives, lines that start with a hash sign, the most common being #include, which brings a header file into the program. After the directives come the functions of the program, and every C program must contain one special function called main(), because this is the point at which the program starts running when it is executed. The body of a function is written between an opening brace and a closing brace, and inside it come the declarations, which introduce the variables the program will use, followed by the statements, which are the instructions that actually do the work, such as printing output or reading input; each statement normally ends with a semicolon. A typical simple program

therefore consists of a header line, the main() function, some declarations, some statements and a return statement at the end. Understanding this structure is the first step in learning to write C programs.

Q2: Explain header files, comments and the compilation process in C.

Three ideas are important when beginning to write C programs: header files, comments and compilation. A header file is a file that contains ready-made declarations which a program needs in order to use certain built-in functions, and it is brought into the program by a preprocessor directive of the form #include. For example, the line #include <stdio.h> includes the standard input/output header, which makes available the functions printf, used to display output, and scanf, used to read input; without including this header, a program could not use these functions. Comments are explanatory notes that the programmer writes inside the source code to describe what the program, or a part of it, does; they are meant only for human readers and are completely ignored by the compiler, and they are written either between the symbols /* and */, which can span several lines, or after //, which comments out the rest of a single line. Good comments make a program much easier to read, understand and correct later. Finally, although the program is written in C as source code and saved in a file with the extension .c, the computer cannot run this source code directly, because it understands only machine code. A special program called a compiler is therefore used to translate the source code into machine code: it first produces object code and then, after a step called linking, an executable file, which on Windows has the extension .exe and which the computer can actually run. If the source code contains any errors, the compiler detects them and reports them to the programmer, who must correct them before the program will successfully compile and run. Thus header files supply needed functions, comments explain the code, and the compiler turns the finished source into a working program.

Q3: Explain, with an example, a simple C program and describe the C character set, keywords and identifiers.

A good way to understand C is to look at a simple complete program and the basic elements from which it is built. A first program usually just prints a message on the screen. Such a program begins with the directive #include <stdio.h> to make the output function available; it then defines the function main(); inside the braces of main() it uses the statement printf("Hello, World!"); which displays the words Hello, World! on the screen; and it ends with the statement return 0; before the closing brace. When this program is compiled and run, the words Hello, World! appear as output, showing clearly how the instructions in the source code produce a visible result. The elements used to write such programs come from the C character set, which is the collection of characters allowed in a C program; it includes the capital letters A to Z and small letters a to z, the digits 0 to 9, special symbols such as the plus, minus, multiply, divide, equals, brackets, braces and semicolon, and white-space characters such as the space, the tab and the new line. From these characters two important kinds of word are formed. Keywords are reserved words that have a fixed, special meaning in the language, such as int, float, char, if, else, while, for and return, and because C already uses them for a definite purpose they may not be used for anything else. Identifiers, on the

other hand, are the names that the programmer invents for things such as variables and functions; an identifier must begin with a letter or an underscore, may then contain letters, digits and underscores, and must not be the same as any keyword. Choosing meaningful identifiers, such as marks or total, makes a program much easier to read and understand.

MCQs with Answers

1. C was developed by:
(a) Bill Gates (b) Dennis Ritchie (c) Charles Babbage (d) James Gosling
Correct Answer: (b) Dennis Ritchie. Dennis Ritchie.

2. Execution of a C program begins at:
(a) the first line (b) main() (c) the last function (d) printf
Correct Answer: (b) main(). The main() function.

3. A line starting with # is a:
(a) comment (b) preprocessor directive (c) statement (d) keyword
Correct Answer: (b) preprocessor directive. A preprocessor directive.

4. The header for input/output is:
(a) conio.h (b) stdio.h (c) math.h (d) string.h
Correct Answer: (b) stdio.h. stdio.h.

5. The function used to print output is:
(a) scanf (b) printf (c) gets (d) main
Correct Answer: (b) printf. printf.

6. A comment in C is ignored by the:
(a) user (b) compiler (c) printer (d) keyboard
Correct Answer: (b) compiler. The compiler.

7. A program that translates C into machine code is a:
(a) editor (b) compiler (c) browser (d) modem
Correct Answer: (b) compiler. A compiler.

8. Which is a keyword in C?
(a) marks (b) int (c) total (d) sum
Correct Answer: (b) int. int.

9. Most C statements end with a:
(a) comma (b) semicolon (c) full stop (d) colon
Correct Answer: (b) semicolon. Semicolon.

10. Which is a valid identifier?
(a) 1num (b) int (c) num1 (d) float
Correct Answer: (c) num1. num1.

Quick Revision Summary

- C = high-level, portable language by Dennis Ritchie (~1972).

- Structure: #include directives, main() function, declarations, statements, return.
- Header files (e.g. stdio.h) supply functions (printf, scanf); comments /* */ // ignored by compiler.
- Compilation: source (.c) -> object -> executable (.exe); compiler reports errors.
- printf("..."); prints output; every statement ends with a semicolon.
- Keywords (int, if, while...) are reserved; identifiers = programmer's names (start with letter/underscore). Notes by freebooks.pk.

Exam Tips

- State who developed C and its key features (portable, efficient).
- Draw the structure of a C program.
- Explain #include and give a header example.
- Describe the compilation process (source -> executable).
- Write a simple printf program.
- Distinguish keywords from identifiers (with valid/invalid examples).