



TECH NEXTGEN ACADEMY

C PROGRAMMING

Think. Code. Solve. Build. Grow.

Foundation to Professional Level

Practical Classroom Training

Project-Based Learning

Duration: 4 Months



BUILD YOUR PROGRAMMING FOUNDATION WITH C

C is one of the most powerful and foundational programming languages in computer science. It teaches you how programs actually work—from **variables and memory to functions, pointers, data structures and file handling**.

At **Tech NextGen Academy**, C Programming is designed to take you from **complete beginner to confident programmer** through structured concepts, hands-on coding, problem-solving exercises, debugging challenges and real-world programming projects.

This course focuses not only on **learning C syntax**, but on developing the ability to **think logically, solve problems and write efficient programs**.

 **OUR GOAL: From Programming Beginner → Confident C Programmer.**

WHO IS THIS COURSE FOR?

- School & College Students
- BCA / B.Sc. / B.Tech Students
- Commerce & Non-IT Students
- Job Seekers & Freshers
- Aspiring Programmers
- Beginners in Programming
- Students preparing for programming subjects
- Students preparing for coding interviews
- Aspiring Software Developers
- Anyone who wants to build a strong programming foundation

PREREQUISITES

- No prior programming experience is required.
- Basic computer knowledge is recommended.
- A willingness to practice and solve problems is essential.

LEARNING METHODOLOGY

**LEARN → THINK → CODE → TEST → DEBUG →
BUILD → PRESENT → GROW**



WHAT YOU WILL MASTER — COURSE CURRICULUM

MODULE 01

PROGRAMMING FUNDAMENTALS & LOGICAL THINKING

LEARN HOW PROGRAMS ACTUALLY WORK

- What is programming? & Programming language
- Why C? Features & Applications of C
- Program development life cycle
- Source code & executable code
- Compiler & interpreter concepts
- Compilation process | Syntax & semantics
- Algorithms, Flowcharts & Pseudocode
- Problem-solving techniques
- Input → Process → Output
- Breaking a problem into smaller problems

PRACTICAL: Convert real-world problems into: **Problem** → **Algorithm** → **Flowchart** → **C Program** → **Output**

MODULE 02

C PROGRAM STRUCTURE & DEVELOPMENT ENVIRONMENT

START WRITING YOUR FIRST C PROGRAM

- C program structure & Header files
- `main()` function & Statements
- Comments, Keywords & Identifiers
- Constants, Variables & C tokens
- Standard libraries
- Compilation & Execution
- Error messages & Basic debugging

CODING PRACTICE: First C program | Hello World | User input/output | Simple calculations | Basic interactive programs

MODULE 03

VARIABLES, DATA TYPES & OPERATORS

BUILD THE CORE OF YOUR PROGRAMS

DATA TYPES

- `int`, `float`, `double`, `char`
- Understanding `void`
- Signed & unsigned data
- Type modifiers
- Type conversion & casting

VARIABLES & CONSTANTS

- Variable declaration
- Initialization & Assignment
- Constants & Scope basics
- Naming conventions

OPERATORS

- Arithmetic, Relational, Logical
- Assignment & Bitwise
- Increment & Decrement
- Conditional & Special operators
- Precedence & Associativity

PRACTICAL: Build calculators and mathematical problem-solving programs using operators.

MODULE 04

INPUT, OUTPUT & FORMATTED DATA

MAKE PROGRAMS INTERACTIVE

- `printf()` and `scanf()`
- Character I/O: `getchar()`, `putchar()`
- Formatted input & Formatted output
- Format specifiers & Escape sequences
- Multiple inputs handling
- Input validation basics

PRACTICAL: Create Student information program, Employee information program, Simple billing program, Salary calculator, and Mark calculation program.

MODULE 05

DECISION MAKING & CONDITIONAL STATEMENTS

TEACH YOUR PROGRAM HOW TO MAKE DECISIONS

CONCEPTS

- `if`, `if-else`, Nested `if`
- `else-if` ladder
- `switch`, `case`, `default`, `break`
- Conditional operator & Multiple conditions

PROBLEM-SOLVING

- Even/Odd | Positive/Negative
- Largest/Smallest of numbers
- Grade calculator | Eligibility checker
- Electricity bill calculation | Menu programs

CHALLENGE: Solve real-world decision-making problems without copying solutions.

MODULE 06

LOOPS & ITERATION

AUTOMATE REPETITIVE TASKS

CONCEPTS

- Why loops? | `for`, `while`, `do-while`
- Nested loops & Loop control (`break`, `continue`)
- Infinite loops & Loop tracing
- Counter-controlled & Condition-controlled loops

PROGRAMMING PRACTICE

- Number patterns & Character patterns
- Tables, Factorial, Fibonacci series
- Prime, Armstrong, Palindrome numbers
- Reverse numbers, Sum of digits, Counting

CHALLENGE: Build pattern and mathematical programs using nested loops.

MODULE 07

FUNCTIONS & MODULAR PROGRAMMING

WRITE CLEANER & REUSABLE CODE

- What is a function? & Advantages
- Function declaration, definition & call
- Parameters, Arguments & Return values
- `void` functions & Prototypes
- Passing values to functions
- Local vs Global variables | Scope
- Lifetime basics & Modular programming
- Function Design: Breaking large problems down

PRACTICAL: Build a menu-driven application using multiple user-defined functions.

MODULE 08

ARRAYS & COLLECTION OF DATA

STORE AND PROCESS MULTIPLE VALUES

1D ARRAYS & OPERATIONS

- Array declaration, initialization, indexing
- Array traversal, Input & Output
- Searching, Sorting, Updating elements
- Sum, average, min/max, frequency, reversing

MULTI-DIMENSIONAL ARRAYS

- Two-dimensional arrays & Matrix representation
- Matrix input/output
- Matrix addition, subtraction, multiplication
- Matrix transpose

PRACTICAL: Build a **Student Marks Analysis System** using arrays.

MODULE 09

STRINGS & CHARACTER HANDLING

WORK WITH TEXT DATA

CONCEPTS & FUNCTIONS

- Character vs string | Character arrays
- String initialization & Null character (`\0`)
- String I/O & String traversal
- `strlen()`, `strcpy()`, `strcmp()`, `strcat()`

PROBLEM-SOLVING

- Reverse string | Palindrome check
- Count vowels, consonants, words
- Search characters | Compare strings
- Remove spaces | Character frequency

PROJECT: Build a simple **Text Analysis Program**.

MODULE 10

POINTERS & MEMORY CONCEPTS

UNDERSTAND ONE OF C'S MOST POWERFUL FEATURES

POINTER BASICS

- Address operator (`&`) & Dereference (`*`)
- Pointer declaration, initialization & arithmetic
- Pointers with variables, arrays & strings
- Passing addresses | Call by value vs reference
- Multiple pointers, Null pointers, Pointer safety

MEMORY CONCEPTS

- Memory addresses
- Stack basics vs Heap basics
- Static vs Dynamic memory concepts
- Common pointer mistakes

PRACTICAL: Solve problems using pointers and memory addresses.

MODULE 11

DYNAMIC MEMORY ALLOCATION

WORK WITH MEMORY AT RUNTIME

- Static vs Dynamic memory
- Heap memory management
- `malloc()`, `calloc()`, `realloc()`, `free()`
- Dynamic arrays & Dynamic memory with pointers
- Memory allocation workflow
- Memory leaks, Dangling pointers & Safe handling

PRACTICAL: Create dynamically sized data structures and applications.

MODULE 12

STRUCTURES, UNIONS & USER-DEFINED DATA TYPES

CREATE CUSTOM DATA MODELS

STRUCTURES

- Introduction & Structure declaration
- Structure variables & Accessing members
- Array of structures & Nested structures
- Structures with functions & Pointers to structures

OTHER DATA TYPES

- `typedef`
- `enum`
- `union`
- Structure vs Union comparison

PROJECT: Create a **Student / Employee Record Management System** using structures.

MODULE 13

POINTERS, ARRAYS & STRUCTURES TOGETHER

COMBINE CORE C CONCEPTS

- Pointers with arrays & strings
- Pointers with structures
- Array of structures & Structure pointers
- Dynamic structures
- Passing & returning structures from functions
- Memory-aware programming

PRACTICAL: Build a structured record-management application using: **Structures + Pointers + Arrays + Functions**

MODULE 14

FILE HANDLING

STORE DATA PERMANENTLY

FILE CONCEPTS

- File handling fundamentals & File pointers
- Opening, closing, reading, writing, appending
- Text files vs Binary files
- File modes & Error handling

FILE FUNCTIONS

- `fopen()`, `fclose()`
- `fprintf()`, `fscanf()`
- `fgetc()`, `fputc()`, `fgets()`, `fputs()`
- `fread()`, `fwrite()`

PRACTICAL: Create a **File-Based Student / Employee Management System**.

MODULE 15

DEBUGGING, ERROR HANDLING & CODE QUALITY

DON'T JUST WRITE CODE. LEARN TO FIX IT.

ERROR TYPES

- Syntax errors
- Compilation errors
- Runtime errors
- Logical errors
- Input-related errors

DEBUGGING

- Reading compiler messages
- Tracing execution
- Testing individual sections
- Identifying logical mistakes
- Boundary testing & validation

CODE QUALITY

- Meaningful variable names
- Indentation & Comments
- Modular & reusable code
- Avoiding redundancy
- Defensive programming

PRACTICAL: Debug intentionally broken C programs and identify the cause of each error.

MODULE 16

PROBLEM-SOLVING & CODING CHALLENGES

TURN LOGIC INTO CODE

Students solve progressively difficult problems covering Number problems, Pattern problems, Mathematical logic, Arrays, Strings, Functions, Pointers, Structures, Searching, Sorting, File handling, and Data processing.

Coding Challenges Process (Beginner → Intermediate → Advanced):
Understand → Plan → Code → Test → Debug → Optimize

MODULE 17

REAL-WORLD C PROGRAMMING SIMULATION

BUILD A COMPLETE APPLICATION

STEP 01 — REQUIREMENT ANALYSIS: Understand the problem and define required features.

STEP 02 — ALGORITHM: Design the step-by-step solution.

STEP 03 — FLOWCHART: Visualize the program logic visually.

STEP 04 — PROGRAM STRUCTURE: Break the application into functions and modules.

STEP 05 — DATA MANAGEMENT: Use arrays, strings and structures.

STEP 06 — MEMORY MANAGEMENT: Apply pointers and dynamic memory where required.

STEP 07 — FILE STORAGE: Store and retrieve application data.

STEP 08 — VALIDATION: Test different inputs and edge cases.

STEP 09 — DEBUGGING: Identify and correct errors.

STEP 10 — OPTIMIZATION: Improve code quality and efficiency.

STEP 11 — FINAL TESTING: Test the complete application end-to-end.

STEP 12 — PRESENTATION: Explain the logic, features and implementation.



CAPSTONE PROJECT: COMPLETE MANAGEMENT SYSTEM IN C

Students will develop a complete menu-driven application using the concepts learned throughout the course.

PROJECT OPTIONS (CHOOSE ONE):

- Student Management System
- Employee Management System
- Library Management System
- Inventory Management System
- Banking Management System
- Simple Billing & Sales Management System

PROJECT FEATURES:

- Add, View, Search, Update & Delete records
- Sort records & Calculate totals/summaries
- Validate input & Store/Retrieve file data
- Menu-driven interface with modular functions
- Structures, Pointers & Error handling

FINAL PROJECT FLOW: Requirement → Algorithm → Flowchart → Design → Coding → Testing → Debugging → File Storage → Optimization → Presentation

FINAL DELIVERABLES:

- 📁 Complete C Source Code
- 📝 Algorithm & Flowchart
- 📖 Project Documentation
- 🖋️ Test Cases
- 🐛 Debugging & Error-Fixing Record
- 💾 File-Based Data Storage
- 📊 Project Report
- 🎤 Final Viva & Presentation



YOUR LEARNING JOURNEY

FOUNDATION (Logic) ↓ SYNTAX (Structure) ↓ CORE (Variables & Operators) ↓ INTERACTION (I/O) ↓ LOGIC (Conditionals) ↓ AUTOMATION (Loops) ↓ MODULARITY (Functions) ↓ DATA (Arrays) ↓ TEXT (Strings) ↓ MEMORY (Pointers) ↓ DYNAMIC PROGRAMMING (Allocation) ↓ DATA MODELING (Structures) ↓ STORAGE (Files) ↓ QUALITY (Debugging) ↓ PROBLEM-SOLVING (Challenges) ↓ APPLICATION (Project) ↓ ACHIEVEMENT (Certification)



WHAT YOU WILL BE ABLE TO DO

- ✓ Understand programming fundamentals
- ✓ Develop logical and computational thinking
- ✓ Write C programs from scratch
- ✓ Understand C program structure
- ✓ Work with variables and data types
- ✓ Use operators effectively
- ✓ Take and process user input
- ✓ Implement decision-making logic
- ✓ Use loops to automate repetitive tasks
- ✓ Design reusable functions
- ✓ Work with 1D and 2D arrays
- ✓ Manipulate strings and character data
- ✓ Understand memory addresses and pointers
- ✓ Work with dynamic memory allocation
- ✓ Create custom data structures using structures
- ✓ Work with unions, enums and typedef
- ✓ Read and write files permanently
- ✓ Build menu-driven applications
- ✓ Debug syntax, runtime and logical errors
- ✓ Write cleaner and more maintainable code
- ✓ Understand recursion
- ✓ Use preprocessor directives and macros
- ✓ Understand command-line arguments
- ✓ Implement basic searching and sorting algorithms
- ✓ Understand fundamental data structures
- ✓ Build file-based applications
- ✓ Solve programming problems independently
- ✓ Build strong foundation for C++, Java, Python, DSA



ASSESSMENT & CERTIFICATION

EVALUATION PROCESS

Your progress is evaluated continuously through:

Programming Assignments → Logic Challenges → C Coding Exercises → Conditional Problems → Loop Challenges → Function Tasks → Array Projects → String Challenges → Pointer Exercises → Structure Projects → File Handling Tasks → Debugging Challenges → Practical Tests → Capstone Project → Viva & Presentation

CERTIFICATION GRADING

Score	Grade
90–100%	A+
80–89%	A
70–79%	B+
60–69%	B
Below 60%	Needs Improvement



**TECH NEXTGEN ACADEMY C Programming Course
Completion Certificate**



WHY TECH NEXTGEN ACADEMY?

NOT JUST C CLASSES. A COMPLETE PROGRAMMING SKILL-BUILDING EXPERIENCE.

-  **Logic-Building Focus:** Develop the ability to think before you code.
-  **Hands-On Programming:** Write programs during classroom sessions.
-  **Problem-Solving Practice:** Solve progressively challenging coding problems.
-  **Strong Fundamentals:** Build concepts that remain useful across programming languages.
-  **Modular Programming:** Learn to structure programs professionally.
-  **Arrays & Data Handling:** Process and analyze collections of information.
-  **Pointers & Memory:** Understand how C works closer to the system level.

-  **File Handling:** Build applications that permanently store data.
-  **Debugging Skills:** Learn to identify and fix programming errors.
-  **Real-World Projects:** Apply concepts to practical management systems.
-  **Project-Based Learning:** Learn by building—not just watching.
-  **Assessment-Based Certification:** Measure your actual programming skills.
-  **Viva & Presentation:** Learn to explain your code and logic confidently.



CONTINUE YOUR PROGRAMMING JOURNEY



YOUR NEXT STEP: DON'T JUST LEARN SYNTAX. LEARN TO THINK LIKE A PROGRAMMER.

After completing **C Programming (From Logic → Code → Problem Solving → Real-World Applications)**, progress toward:

ADVANCED PROGRAMMING & SOFTWARE DEVELOPMENT:

- | | |
|----------------------------------|-------------------------------------|
| • → C++ Programming | • → Database & SQL |
| • → Object-Oriented Programming | • → Web Development |
| • → Java Programming | • → Full Stack Development |
| • → Python Programming | • → Software Development |
| • → Data Structures & Algorithms | • → System Programming Fundamentals |
| • → Competitive Programming | |



JOIN TECH NEXTGEN ACADEMY

Learn Today. Lead Tomorrow.



Address: Jadavpur, Kolkata



Phone: +91 6292009436 / +91 6292009536



WhatsApp: +91 6292009436



Email: technextgenacademy@gmail.com



Website: technextgenacademy.com

ADMISSIONS OPEN

LIMITED SEATS AVAILABLE

**START CODING. START BUILDING.
START YOUR PROGRAMMING JOURNEY.**

Duration: 4 Months