

The C Programming Language was originally created to write the UNIX operating system. It quickly turned into a multi-purpose language used by all types of programmers for a wide variety of applications. C is a small language that can be learned quickly. It is highly-structured and modular, supporting both small and large programs equally well.

Batky-Howell's C Programming course has been introducing students to the power and flexibility of this language for years. You will develop the ability to design and write programs in the ANSI Standard C programming language. Concepts such as arrays, functions, control flow, pointers, and many others will quickly prepare you to successfully code your own C applications.

Course Objectives:

- Describe the basic elements of C.
- Write C programs using all the major features of the language.
- Define and use C datatypes.
- Write variable declarations for programs.
- Apply the unique notations that C uses for assignments, incrementing, and decrementing.
- Control the flow of program execution.
- Write modular programs consisting of functions.
- Describe the purpose and functioning of a preprocessor.
- Define the relationship between arrays and pointers.
- Use structure variables for data storage and manipulation.

Audience: Programmers new to the ANSI C language.

Prerequisites: Programming skill in a language such as Pascal, COBOL, BASIC, or assembler.

Number of Days: 5 days

<p>1. Course Introduction Course Objectives Overview Suggested References</p> <p>2. Introduction to C What is C ? Features of C Why Program in C ? History of C Current Status and Future</p> <p>3. An Overview of C The First Program (hello.c) How to Compile and Run a C Program An Arithmetic Program (roof.c) Execution Flow Control (mph.c) The for Loop</p>	<p>The for Loop - Diagram Character I/O A File Copier Program (cp2.c) A Character Counter (wc2.c) A Look at Arrays Stock Values (stock1.c) The char Data Type Strings (Character Arrays) A String Copy Program (stringcp.c) A Look at Functions A Functional Program (func1.c) A Review of printf()</p> <p>4. Data Types and Variables Fundamental Data Types Data Type Values and Sizes</p>
--	---

- Variable Declarations
- Variable Names
- Constants
- Character Constants
- String Constants
- 5. Operators and Expressions**
 - What are Expressions?
 - Arithmetic Operators
 - Relational Operators
 - Assignment Operator
 - Expressions Have Resulting Values
 - True and False
 - Logical Operators
 - Increment and Decrement Operators
 - (++ and --)
 - Increment and Decrement Operators:
 - Examples
 - 'Operate-Assign' Operators (+=, *=, ...)
 - Conditional Expression
 - Operator Precedence
 - Precedence and Order of Evaluation
 - Evaluation of Logical Operators
 - Type Conversions
 - The Cast Operator
 - Bitwise Logical Operators
- 6. Control Flow**
 - Statements
 - if - else
 - if() - else if()
 - switch()
 - while()
 - do - while()
 - for()
 - The for Loop - Diagram
 - Example: for() Loop
 - Another Example: for() Loop
 - The break Statement
 - The continue Statement
- 7. Functions**
 - What is a Function?
 - Example: findbig3()
 - Why Use Functions?
 - Anatomy of a Function
 - Example: find_big_int()
 - Arguments Passed by Value
 - Addresses of Arguments Can Be Passed
- A Picture of Addresses and Values
- When to Use the Return Statement
- Returning Non-Integer Values
- Functions in Multiple Source Files
- A Simple make File
- The Concept of Variable Scope
- Automatic Variables
- Global (External) Variables
- Static Variables
- External Static Variables
- 8. The C Preprocessor**
 - Symbolic Constants
 - Macro Substitution
 - File Inclusion
- 9. Pointers and Arrays**
 - What is a Pointer?
 - Pointer Operators
 - Example: Pointers
 - Why Use Pointers?
 - Arrays
 - Arrays (a Picture)
 - The & Operator
 - Pointers and Arrays
 - Pointer Arithmetic
 - Pointer Arithmetic (a Picture)
 - Arrays and Pointers
 - Array Names are Constant Pointers
 - Passing Arrays to Functions
 - Initializing Arrays
- 10. Advanced Pointers**
 - Pointer Initialization
 - Command-Line Arguments
 - Strings and Character Pointers
 - Arrays of Pointers
 - Command-Line Arguments
 - Access Through Pointers
 - Functions and Pointers
 - Example: Functions and Pointers
- 11. Structures**
 - Structures
 - Comparison of Structures and Arrays

- Structure Definitions
- Structure Declarations
- Structure Parameter Passing by Reference
- Pointers to Structures
- Structure Parameter Passing Again
- Arrays of Structures
- The malloc Routine

12. Appendix - File I/O in C

- File Streams
- Predefined Streams
- The fprintf Function
- The fscanf Function
- fscanf() Examples
- The fputs and fgets Functions
- The fwrite and fread Functions
- System I/O