Introduction to C Programming

Course outline

Module 1: Basic Syntax and Structure of C

Module 1 of the Introduction to C Programming course covers the basic syntax and structure of the C programming language. It introduces the fundamentals of the language, including variables, data types, operators, and control flow. It also covers the basics of writing and debugging C programs.

Lessons

- Overview of C Programming
- Data Types and Variables
- Operators and Expressions
- Control Flow Statements
- Functions
- Arrays
- Pointers
- Structures
- Input/Output
- Preprocessor Directives

After completing this module, students will be able to:

- Understand the basic syntax and structure of C programming language.
- Write basic C programs using variables, constants, operators, and control statements.
- Compile and execute C programs using a compiler.
- Debug and troubleshoot C programs.

Module 2: Data Types and Variables

Module 2 of the Introduction to C Programming course covers the fundamentals of data types and variables. Students will learn about the different types of data, such as integers, floats, and characters, and how to declare and use variables. They will also learn about the different operators and how to use them to manipulate data. Finally, students will learn about the different data structures available in C and how to use them to store and manipulate data.

Lessons

- Introduction to Data Types
- Primitive Data Types

- Declaring Variables
- Naming Variables
- Assigning Values to Variables
- Understanding Variable Scope
- Working with Strings
- Working with Arrays
- Working with Structures
- Working with Pointers

After completing this module, students will be able to:

- Understand the different data types available in C programming and how to declare and initialize variables.
- Utilize the various arithmetic and logical operators to perform calculations and comparisons.
- Use the if-else statement to control the flow of a program.
- Utilize the switch statement to control the flow of a program.

Module 3: Operators and Expressions

Module 3: Operators and Expressions introduces students to the fundamentals of C programming, including the use of operators and expressions. Students will learn how to use arithmetic, relational, logical, and bitwise operators to create expressions and statements. They will also learn how to use the various types of data in C programming, such as integers, floats, and characters. Finally, students will learn how to use the various control structures in C programming, such as if-else statements, switch statements, and loops.

Lessons

- Introduction to Operators
- Arithmetic Operators
- Relational Operators
- Logical Operators
- Bitwise Operators
- Assignment Operators
- Increment and Decrement Operators
- Conditional Operators
- Precedence of Operators
- Type Conversion in Expressions
- Evaluating Expressions
- Operator Precedence and Associativity
- Operator Overloading
- Expressions and Statements
- Writing Simple C Programs using Operators and Expressions

After completing this module, students will be able to:

- Understand the different types of operators and how to use them in C programming.

- Be able to write expressions using arithmetic, relational, logical, and assignment operators.
- Be able to use the increment and decrement operators in C programming.
- Be able to use the ternary operator to create conditional expressions.

Module 4: Control Flow Statements

Module 4: Control Flow Statements introduces students to the fundamentals of control flow statements in C programming. Students will learn how to use if-else statements, switch statements, and loops to control the flow of their programs. They will also learn how to use break and continue statements to modify the behavior of loops. By the end of the module, students will have a better understanding of how to use control flow statements to create more efficient and effective programs.

Lessons

- Understanding the if-else Statement
- Working with the switch Statement
- Using the for Loop
- Exploring the while and do-while Loops
- Working with the break and continue Statements
- Understanding the goto Statement
- Working with Logical Operators
- Using Conditional Operators
- Writing Nested Control Flow Statements
- Debugging Control Flow Statements

After completing this module, students will be able to:

- Understand the concept of control flow statements and how they are used to control the flow of a program.
- Utilize if-else statements to create conditional logic in a program.
- Use switch statements to create multiple branches of logic in a program.
- Implement looping statements such as for, while, and do-while to create repetitive logic in a program.

Module 5: Functions

Module 5 of the Introduction to C Programming course covers the fundamentals of functions in C. Students will learn how to create and use functions, as well as how to pass parameters and return values. They will also learn about the scope of variables and how to debug functions.

Lessons

- Understanding the Basics of Functions
- Writing and Calling Functions
- Passing Arguments to Functions
- Returning Values from Functions
- Scope of Variables in Functions

- Recursive Functions
- Function Pointers
- Inline Functions
- Function Overloading
- Preprocessor Directives for Functions

After completing this module, students will be able to:

- Understand the concept of functions and how to use them in C programming.
- Create and call functions with parameters and return values.
- Utilize the library functions available in C programming.
- Debug and troubleshoot errors related to functions.

Module 6: Arrays

Module 6: Arrays introduces students to the concept of arrays in C programming. It covers topics such as declaring and initializing arrays, accessing array elements, and manipulating arrays. It also covers topics such as multi-dimensional arrays, passing arrays to functions, and sorting and searching arrays. This module provides a comprehensive overview of the array data structure and its uses in C programming.

Lessons

- Introduction to Arrays
- Declaring and Initializing Arrays
- Accessing Array Elements
- Array Operations
- Multi-dimensional Arrays
- Array Sorting
- Array Searching
- Array Manipulation
- Array Pointers
- Array Applications

After completing this module, students will be able to:

- Understand the concept of an array and how to declare and initialize an array in C.
- Be able to use array subscripts to access individual elements of an array.
- Be able to use loops to iterate through an array and perform operations on each element.
- Be able to use functions to pass arrays as arguments and return arrays as values.

Module 7: Pointers

Module 7: Pointers introduces students to the concept of pointers in C programming. It covers topics such as memory addresses, pointer variables, pointer arithmetic, and dynamic memory allocation. Students will learn how to use pointers to manipulate data and create efficient programs.

Lessons

- What is a Pointer?
- Memory Allocation and Pointers
- Pointer Arithmetic
- Pointers and Arrays
- Pointers and Strings
- Pointers and Structures
- Pointers and Functions
- Dynamic Memory Allocation
- Pointer to Pointer
- Pointers and Linked Lists

After completing this module, students will be able to:

- Understand the concept of pointers and how to use them in C programming.
- Be able to declare and initialize pointers.
- Be able to use pointers to access and manipulate data in memory.
- Be able to use pointers to pass arguments to functions and return values from functions.

Module 8: Structures and Unions

Module 8: Structures and Unions introduces students to the concept of data structures and unions in C programming. It covers topics such as declaring and initializing structures, accessing structure members, unions, and bit fields. It also covers the use of structures and unions in functions and pointers. This module provides a comprehensive overview of the fundamentals of data structures and unions in C programming.

Lessons

- Overview of Structures and Unions
- Declaring and Initializing Structures and Unions
- Accessing Structure and Union Members
- Nesting Structures and Unions
- Pointers to Structures and Unions
- Array of Structures and Unions
- Bit Fields in Structures and Unions
- Unions vs Structures
- Applications of Structures and Unions

After completing this module, students will be able to:

- Understand the concept of structures and unions in C programming.
- Create and manipulate structures and unions in C programming.
- Utilize structures and unions to store and access data in C programming.
- Implement structures and unions to optimize memory usage in C programming.

Module 9: Input/Output

Module 9 of the Introduction to C Programming course covers the basics of input and output in C. It covers topics such as reading and writing data from files, using the standard input and output streams, and formatting output. It also covers the use of the printf and scanf functions for formatted input and output.

Lessons

- Understanding Input/Output Streams
- Working with Files
- Reading and Writing Text Files
- Working with Binary Files
- Understanding Standard Input/Output Streams
- Working with Formatted Input/Output
- Understanding Error Handling
- Working with Command Line Arguments
- Understanding File System Operations
- Working with Directories

After completing this module, students will be able to:

- Understand the concept of input/output operations in C programming.
- Be able to read and write data from/to files.
- Be able to use the standard I/O library functions such as fopen(), fclose(), fgetc(), fputc(), fscanf(), fprintf(), etc.
- Be able to use the command line arguments to read data from the user.

Module 10: Dynamic Memory Allocation

Module 10 of the Introduction to C Programming course covers the concept of dynamic memory allocation. This module will teach students how to use the malloc() and calloc() functions to allocate memory dynamically, as well as how to free memory with the free() function. Students will also learn about the differences between static and dynamic memory allocation, and how to use pointers to access dynamically allocated memory.

Lessons

- Overview of Dynamic Memory Allocation
- Allocating Memory with malloc()
- Releasing Memory with free()
- Working with Pointers and Dynamic Memory Allocation
- Memory Leaks and How to Avoid Them
- Dynamic Arrays and Structures
- Memory Fragmentation and How to Avoid It
- Dynamic Memory Allocation in Multithreaded Applications
- Dynamic Memory Allocation Performance Considerations
- Best Practices for Dynamic Memory Allocation

After completing this module, students will be able to:

- Understand the concept of dynamic memory allocation and its importance in C programming.
- Utilize the malloc(), calloc(), realloc(), and free() functions to allocate and deallocate memory dynamically.
- Create and manipulate dynamic data structures such as linked lists, stacks, and queues.
- Analyze the time and space complexity of programs that use dynamic memory allocation.

Module 11: Preprocessor Directives

Module 11: Preprocessor Directives introduces students to the concept of preprocessor directives in C programming. It covers topics such as #include, #define, #ifdef, #ifndef, and #pragma. Students will learn how to use these directives to control the compilation process and how to write their own preprocessor directives.

Lessons

- What is a Preprocessor Directive?
- Understanding the #include Directive
- Exploring the #define Directive
- Working with the #ifdef Directive
- Utilizing the #pragma Directive
- Using the #error Directive
- Exploring the #undef Directive
- Working with the #ifndef Directive
- Understanding the #line Directive
- Utilizing the #warning Directive

After completing this module, students will be able to:

- Understand the purpose of preprocessor directives and how they are used in C programming.
- Utilize preprocessor directives to include header files and define macros.
- Recognize the various types of preprocessor directives and their syntax.
- Implement preprocessor directives to control the compilation process.

Module 12: Error Handling

Module 12: Error Handling introduces students to the fundamentals of debugging and error handling in C programming. Students will learn how to identify and debug errors in their code, as well as how to use the C language's built-in error handling functions. Additionally, students will learn how to use the debugger to trace and debug their code.

Lessons

- Understanding Error Messages

- Debugging Techniques
- Exception Handling
- Error Prevention Strategies
- Common Error Types
- Error Logging
- Error Recovery
- Error Handling Best Practices
- Error Handling in C
- Error Handling in C++

After completing this module, students will be able to:

- Understand the concept of error handling and how to use it in C programming.
- Identify and debug common errors in C programs.
- Utilize techniques such as exception handling and assertions to handle errors.
- Implement error handling strategies to ensure the reliability of C programs.

Module 13: Advanced C Programming Topics

Module 13: Advanced C Programming Topics is an introduction to more complex topics in C programming. It covers topics such as memory management, pointers, dynamic memory allocation, and data structures. It also covers advanced topics such as multi-threading, debugging, and optimization. This module is designed to give students a deeper understanding of the C language and its capabilities.

Lessons

- Memory Management
- Pointers and Dynamic Memory Allocation
- Structures and Unions
- Bitwise Operators
- Preprocessor Directives
- File I/O
- Command Line Arguments
- Multi-Threading
- Network Programming
- Advanced Debugging Techniques
- Advanced Compiler Optimization
- Advanced Data Structures
- Advanced Algorithms
- Advanced Memory Management
- Advanced Linked Lists
- Advanced Sorting Algorithms
- Advanced Search Algorithms
- Advanced Graph Algorithms
- Advanced String Manipulation
- Advanced Recursion

After completing this module, students will be able to:

- Understand the fundamentals of memory management in C programming.
- Develop proficiency in using pointers and dynamic memory allocation.
- Utilize advanced features of the C language such as structures, unions, and bit fields.
- Implement complex algorithms and data structures in C programming.