Programming Fundamentals through C

Muhammad Iqbal Bhat

Assistant Professor (Computer App.)

Government Degree College Beerwah

Course Objectives

- ✓ To demonstrate the use of flowcharts and algorithms for problem-solving
- ✓ To introduce the concepts of structured programming
- ✓ To familiarize the student with the syntactic constructs of C
- ✓ To enable students to translate algorithms into C programs

Syllabus

THEORY (4 CREDITS)

UNIT – I **(15 Lectures)**

Programming Languages: History and Role of Programming Languages, Syntax and semantics, source code and object code, datatypes, variables, constants, declaration, Structured Data Types. Sequence Control: Implicit and Explicit. Sequence control between Statements. Subprogram Control: Simple call return and recursive subprogram. Language Paradigms: Simple Procedural Languages, Block Structured Programming Languages, Object Based Languages, Functional Languages, Logic Programming Languages.

Flowcharts, Flowchart Elements, Problem Solving Through Flowcharts.

Algorithms, Characteristics of an Algorithm, Algorithms for basic problems.

UNIT – II

Introduction to C Programming **(15 Lectures)**

History and overview of C, Basic structure of a C Program, Compilation, Execution and Debugging of programs in C.

Keywords, Identifiers and Datatypes. Variables and Constants. Comments. Console I/O using printf() and scanf(). Typecasting.

Operators – Arithmetic, Logical, Relational, Increment Decrement and Assignment Operators. Expressions. Operator Precedence.

Conditional Statements (If, If-Else, If-Else If, Nested If, Switch).

UNIT – III (15 Lectures)

Looping, Functions and Pointers.

Loops (while, do-while, for). Break and Continue. Nested Loops.

Functions: Declaring, Defining and Calling. Call by Value, Call by Reference. Function Arguments and Return Values.

Pointers: Declaring and Initializing. Accessing value of a pointer variable. Pointer Expressions. Pointer Increments and Scale Factors. Pointers and Arrays. Passing Pointers to Functions.

UNIT – IV

Arrays, Strings, Structures and Unions

Declaring, Initializing 1-D arrays and 2-D arrays. Accessing Elements of an Array, Memory Layout of Arrays. Passing Arrays to Functions, Command Line Arguments.

Character Arrays and String. Declaring and Initializing Strings, Reading and Writing Strings, String Handling Function (strlen, strcat, strcmp, strcpy).

Structures and Unions: Declaring, initializing and using simple structures and unions, Manipulating individual members of structures and unions, Array of structures, Passing structures to functions.

Dynamic Memory Allocation using malloc and free.

Programming Languages – an Intro

Important of Programming languages

- ✓ Programming languages are the backbone of software development.
- ✓ They provide a way for humans to communicate with computers and create complex programs that solve real-world problems.
- ✓ Programming languages offer a wide range of flexibility, from low-level languages like C and Assembly to high-level languages like Python and Java.
- ✓ This flexibility allows developers to choose the best language for a given task, making programming more efficient and effective.
- ✓ Choosing the right programming language can greatly improve the efficiency of software development.
- ✓ For example, C is a popular choice for developing operating systems because of its efficiency, while Python is commonly used for web development due to its ease of use and rapid prototyping capabilities.

Important of Programming languages

- ✓ Compatibility is a key factor in programming language selection.
- ✓ Many programming languages are designed to work on specific platforms, such as Windows or Linux, while others are more platform-agnostic.
- ✓ Developers must consider compatibility when selecting a language to ensure that their code works seamlessly with the desired platforms.
- ✓ The programming language landscape is constantly evolving, with new languages and tools emerging regularly.
- ✓ Staying up to date with new programming languages and tools is important for developers to remain competitive and continue producing high-quality software.
- ✓ Programming languages are essential for software development, providing a way for developers to communicate with computers and create complex programs.
- ✓ Choosing the right programming language is critical for efficient and effective development, and staying up to date with new languages and tools is important for remaining competitive in the field.

Modern World & Programming languages

- ✓ Programming languages are the backbone of modern technology, powering everything from smartphones and laptops to cars and airplanes.
- ✓ In today's world, programming languages are an essential tool for almost every industry, including healthcare, finance, transportation, entertainment, and education.
- ✓ The rise of the internet and mobile devices has made programming languages even more crucial, as they enable us to connect with people and businesses around the world.
- ✓ Without programming languages, we wouldn't have modern-day conveniences like online shopping, social media, or even email.
- ✓ In the era of big data, programming languages like Python and R are essential for processing and analyzing vast amounts of information, powering everything from scientific research to business intelligence.
- ✓ The importance of programming languages will only continue to grow as we move towards a more interconnected and technology-driven world.
- ✓ As such, it's essential for anyone interested in technology, whether it's as a career or a hobby, to have at least a basic understanding of programming languages and how they work.

Real World Examples:

- ✓ In healthcare, programming languages are used to develop medical software and systems that can help doctors diagnose and treat patients more effectively.
- ✓ In finance, programming languages are used to develop trading algorithms, risk management tools, and other financial software that can help businesses make better decisions.
- ✓ In transportation, programming languages are used to develop self-driving cars, traffic management systems, and other innovations that are revolutionizing the way we travel.
- ✓ In entertainment, programming languages are used to develop video games, special effects, and other multimedia experiences that immerse us in new worlds.
- ✓ In education, programming languages are used to teach students critical thinking skills, problem-solving, and other essential skills that will prepare them for the future.
- ✓ These examples are just a small fraction of the many ways in which programming languages are transforming our world and making it a better place for everyone.

The popularity of various programming Languages

Programming Language	Use in Web Development	Use in Data Science	Use in Mobile Development	Popularity Ranking (as per TIOBE Index)
JavaScript	High	Medium-High	Medium-High	1
Python	Medium	High	Medium	2
Java	Medium	Medium	High	3
C++	Low-Medium	Medium-High	Low-Medium	4
C#	Medium	Low	Medium	5
PHP	High	Low	Low	6
TypeScript	High	Low	Medium	7
Ruby	Low-Medium	Low-Medium	Low-Medium	8
Swift	Low	Low	High	9
Kotlin	Low	Low	High	10

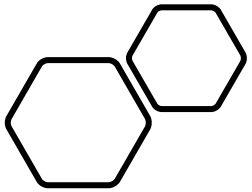
Programming Languages and Job Opportunities

Programming Language	Average Salary	Job Openings
Python	$116,000	18,000+
Java	$104,000	17,000+
JavaScript	$97,000	16,000+
C#	$91,000	13,000+
PHP	$85,000	7,000+
Ruby	$85,000	1,000+
Swift	$82,000	3,000+
Objective-C	$78,000	1,000+
SQL	$76,000	15,000+
C++	$73,000	8,000+

C in Modern World

Industry/Field	Use of C Language
Operating Systems	C is widely used for developing operating systems like Windows, Linux, Unix, and MacOS.
System Programming	C is the primary language used for system programming.
Embedded Systems	C is used in the development of firmware and device drivers for embedded systems.
Game Development	C is used in the development of game engines and game logic.
Financial Industry	C is used in the development of high-performance trading systems and financial software.
Scientific Computing	C is used in scientific computing to develop simulations and numerical analysis applications.

Evolution of Programing Languages



Historical timeline of programming languages

Programming Language	Year	Evolution
First Programming language	1843	The first computer program was written for a general-purpose machine by Ada Lovelace and Charles Babbage
Machine Language	1940s	Directly representing binary instructions that can be executed by a computer's CPU.
Assembly Language	1950s	Using mnemonic codes to represent machine instructions, making it easier for humans to write programs.
Fortran	1957	The first high-level programming language, designed for scientific and engineering calculations.
COBOL	1959	Designed for business applications and data processing, and still used in some legacy systems.
BASIC	1964	Designed to be simple and accessible for beginners, and popularized the use of interactive computing.
C	1972	A powerful and efficient programming language with a low-level syntax, still widely used for systems programming.
Smalltalk	1972	The first object-oriented programming language, with a focus on objects and message passing.
Pascal	1972	Designed for teaching programming and structured programming concepts.
C++	1983	An extension of C with added object-oriented features, still widely used for systems programming and game development.
Objective-C	1984	Adds object-oriented features to C, and popularized for developing software on Apple's macOS and iOS.
Perl	1987	A scripting language with a focus on text processing and system administration tasks.
Python	1991	A high-level, easy-to-learn programming language used for web development, scientific computing, and more.
Java	1995	A cross-platform programming language designed for building robust applications, especially for the web.
JavaScript	1995	A scripting language used to create interactive web applications, and now used for server-side programming with Node.js.
Ruby	1995	A flexible and easy-to-learn language used for web development, automation, and more.
C#	2000	Microsoft's object-oriented language designed for building Windows applications, and now used for web development with ASP.NET.
Swift	2014	Apple's modern programming language for building macOS, iOS, and other Apple platform applications.

Machine and Assembly Language:

The earliest programming languages were machine language and assembly language.

Machine language is a low-level programming language that consists of binary code that can be executed directly by a computer.

Assembly language is a low-level programming language that uses mnemonic codes to represent machine language instructions.

Early programming languages such as FORTRAN and COBOL were developed in the 1950s and were used primarily for scientific and business applications.

These languages were procedural and focused on the sequence of steps needed to perform a task.

Key Characteristics of Early Programming Languages

Benefits of Early Programming Languages:

- Enabled the development of complex algorithms and systems.
- Paved the way for the development of high-level programming languages.

Limitations of Early Programming Languages:

Difficult to learn and use.

Limited functionality and portability.

Required extensive knowledge of computer architecture.

Note: The limitations of early programming languages led to the development of high-level programming languages.

High-Level Programming Language

High-Level Programming Languages:

- High-level programming languages are designed to be more user-friendly and abstract away details of the computer architecture.
- They use English-like syntax and provide a high level of abstraction.
- Some popular high-level programming languages include C++, Java, and Python.

Benefits of High-Level Programming Languages:

- Easier to learn and use.
- Increased productivity.
- Portability across different computer systems.
- Less error-prone.

Characteristics of High-Level Programming Languages:

- Use English-like syntax.
- Provide a high level of abstraction.
- Support a wide range of programming paradigms.

Note: High-level programming languages paved the way for object-oriented programming, functional programming, and scripting languages.

Object-Oriented Programming

Object-Oriented Programming:

- Object-oriented programming (OOP) is a programming paradigm that focuses on objects and their interactions.
- It uses classes to define objects and their properties and methods to define their behavior.
- OOP languages include Java, C++, and Python.

Key Concepts of OOP:

- Classes: Blueprint for creating objects.
- Objects: Instances of a class.
- Inheritance: Ability for a class to inherit properties and methods from another class.
- Polymorphism: Ability for objects of different classes to be used interchangeably.

Benefits of OOP:

- Encapsulation: Keeps data and methods related to an object in one place.
- Modularity: Allows for easy maintenance and testing.
- Reusability: Enables the reuse of code through inheritance.

Note: OOP is Widely used in the development of software applications and is a popular programming paradigm.

Functional Programming

Functional Programming:

- Functional programming is a programming paradigm that focuses on the use of functions to solve problems.
- It emphasizes the use of pure functions, higher-order functions, and immutability.
- Functional programming languages include Haskell and Lisp.

Key Concepts of Functional Programming:

- Pure functions: Functions that do not have side effects and always return the same output for a given input.
- Higher-order functions: Functions that take other functions as input or return functions as output.
- Immutability: Data cannot be modified once it is created.

Benefits of Functional Programming:

- Easier to reason about and test.
- Parallelizable: Functions can be executed concurrently.
- Resilient to errors: Pure functions do not have side effects

Note: Functional programming is gaining popularity due to its benefits in developing concurrent and distributed systems.

Scripting Languages

Scripting Languages:

- Scripting languages are programming languages that are interpreted at runtime.
- They are often used for automation, web development, and system administration.
- Examples of scripting languages include JavaScript, Python, Perl, and Ruby.

Key Characteristics of Scripting Languages:

- Interpreted at runtime.
- Dynamic typing: Types are inferred at runtime.
- High-level of abstraction.

Benefits of Scripting Languages:

- Rapid development: Scripts can be written quickly and easily.
- Cross-platform: Can run on different operating systems without recompilation.
- Integration: Can easily interact with other languages and systems

Note: Scripting languages are useful for tasks such as web development and automation where rapid development and cross-platform compatibility are important.

Questions?