Syllabus of Foundation Level C Programming Course

Absolute basics

Languages: natural and artificial
Machine languages
High-level programming languages
Obtaining the machine code: compilation process
Your first program
Variable – why?
Integer literals in "C"

Data types

Float literals in "C"
Arithmetic operators
Priority and binding
Post- and pre -incrementation and -decrementation
Operators of type op=
Char type and ASCII code, char literals
Comparison operators
Printf() and scanf() functions

Flow control

Conditional execution - if keyword and else branch
More integer and float types
Conversions - typecast and its operators
Loops – while, do and for
Controlling the loop execution – break and continue
Logical and bitwise operators

Arrays

Switch: different faces of 'if'
Arrays (vectors) – why do you need them?
Sorting in real life and in a computer memory
Pointers: another kind of data in "C"
An address, a reference, a dereference and the sizeof operator
Pointer to nothing (NULL)
Pointers arithmetic
Pointers vs. arrays: different forms of the same phenomenon
Using strings and basic functions dedicated to string manipulation

Memory management and structures

The meaning of array indexing
The usage of pointers: perils and disadvantages
Void type
Arrays of arrays and multidimensional arrays
Memory allocation and deallocation: malloc() and free() functions
Arrays of pointers vs. multidimensional arrays
Structures – why? and declaring, using and initializing structures
Pointers to structures and arrays of structures
Basics of recursive data collections

Functions

Functions – why?
How to declare, define and invoke a function
Variables' scope, local variables and function parameters
Pointers, arrays and structures as function parameters
Function result and return statement
Void as a parameter, pointer and result
Parameterizing the main function
External function and the extern declarator
Header files and their role

Files and streams

Files vs. streams: where does the difference lie?
Header files needed for stream operations
FILE structure
Opening and closing a stream, open modes, errno variable
Reading and writing to/from a stream
Predefined streams: stdin, stdout and stderr
Stream manipulation: fgetc(), fputc(), fgets() and fputs() functions
Raw input/output: fread() and fwrite() functions

Preprocessor and complex declarations

Preprocessor – why?
#include: how to make use of a header file
#define: simple and parameterized macros
#undef directive
Predefined preprocessor symbols
Macro operators: # and ##
Conditional compilation: #if and #ifdef directives
Avoiding multiple compilations of the same header files
Storage classes
User defined types – why?
Pointers to functions
Analyzing and creating complex declarations

Syllabus of Intermediate Level C Programming Course

The C Language

C Program Compilation
Execution Process
Tokens of C Program
C Instructions
Constants, Variables
Identifiers and Keywords
Primitive Data Types
Structures – The Definition
Structures – Declaration & Type
Accessing Elements of Structure
Range of Signed/Unsigned Data-types
Efficient way of Printing Pointer
Compiler Memory Allocation for Data-types
Compiler Memory Allocation for Structures
Data-type Alignments
Compiler Memory Allocation for Unions
Union – Data Corruption
Practical Usage of Unions
Practical Usage of Bitfields
Bitfields Overflow
Printing every byte of an Integer
Enumeration
Typedef Statements
Practical example of Typedef Usage
typedef'ing a Function Pointer
Bit-Fields in Structure

Practical examples of Bitfield Usage
Structure Padding & Pitfalls
Programming Model & Memory Sizes

Why Sizeof Int and Long is 4 or 8?
Use of long long in 32-bit Architecture

Practical Example of long long
IA-32, IA-64, ILP-32, LP64, x86-64

Array – Representation
Array – Memory Allocation
Array – Declaration & Initialization
Two Dimensional Arrays

Pointers

Accessing a Variable Through Pointer

Pointer – Memory Allocation
Pointer – Declaration & Initialization
Pointer – Dereferencing
Pointers & Arrays
Character Arrays using Pointers
Array of Character Pointers
Memory Diagram – Array of Char Pointers
Arrays as Pointers – a[i] == i[a]?
Constant Pointers
Pointer Arithmetic
String Handling Functions
String Conversion Functions
Efficient usage of sscanf()/sprintf()
Computing Basic
Binary & Octal Systems
Decimal & Hexadecimal Systems
Signed Representations in Memory
Binary Shifts – Right & Left
Sign Bits and Bit-Shift Operations
Right Shift – Logical Vs Arithmetic Shift
Bit-Shift Overflow
ASCII Representations
Endian-ness – Little Vs Big
Endian-ness – Portability Issues
Operators
Bitwise Operations
Logical Operators – Short Circuit
Bitwise Vs Logical Operations
sizeof() operator
Pitfalls/Issues with sizeof() usage
Pointer Increment & Scaling
Operator Precedence
Operator Associativity
True meaning of Associativity
Examples of Precedence & Associativity
Ternary Operator Associativity Rule
Data-type Conversion Rules
Float to Int to Float Conversions
Variadic functions & default promotion rules
Printf Idiosynchracies
Pointer Format Specifiers
Signed Vs Unsigned – Pitfalls
Evaluation of i = ++i + ++i
Evaluation of i = ++i + ++i + ++i
Concept of Sequence Points
Example of Sequence Points
Storage Classes
Storage Class Specifiers
Scope of a Variable
Register, Auto, Static, Extern
Why Register Class and Practical Examples
Automatic Variables and Stack
Static Variables and Functions
True meaning of Extern
How to Use extern across Multiple Files with Examples
Best Practices for Extern Usage
Local/Block/Global Scope
Nesting of Scope
Lifetime of a Variable
Linkage of a Variable
What is Const?
Practical Examples of Const Qualifier
Usage of Constant in library functions (libc)
What is Volatile?
Practical Examples of Volatile Qualifier
Const Volatile Together?
Register Vs Volatile Performance?
Practical Examples of Const Volatile
Pointer Aliasing
What is Restrict Qualifier?
Restrict Keyword and Compiler Optimization
Examples of Restrict Qualifier
Memory
Dynamic Memory Allocations
malloc, calloc, realloc, free
malloc Vs calloc
Heap Memory
Stack Memory – Pitfalls
Dangling Pointers
DMA – Errors
Best Practices for malloc() & free()
DMA – Unspecified Behaviour
Functions & Pointers
Invoking Functions
Passing Arguments to Functions
Call by Value & Reference
Is C call by Value?
Is C call by Reference?
Array as Function Argument
Rules for Array Argument Passing
Multi-dimensional Array Argument Passing
Structure as Function Argument
Static Vs Dynamic Runtime Environment
Function Call and Runtime Stack
Rules for Evaluation of Function Arguments
Memory Organization
Code Segment
Data Segment
Heap Segment
Stack Segment
free space
register space
Stack Frames
Calling Sequence
View of Runtime Stack with Example
Access to Local Variable in Stack
Local Temporaries
Function Pointers
Declaration and Usage of Function Pointers
Function Pointers as Function Parameters
Practical Example of Function Pointers
Pointer to an Integer Array
C Pointers Complexity Chart
int **p
int (*p)()
int (*p)[] int *p()
int *(*p[])()
int *(**p)[]
Preprocessor
Preprocessor – #include statements
Multiple Inclusion of a Header File?
Preprocessor – #define statements
Preprocessor – Conditional Compilation
Preprocessor – Nested Macros
Preprocessor – Multiline Macros
Preprocessor – Stringizer
Preprocessor – Token Concatenation
Preprocessor – Useful Directives
Conditional Directives for Debugging
Where Macros are Heavily Used
Practical Examples of Macros
Macros Pitfalls

Macros Vs Enums
Inline Functions
Macros Vs Inline
Inline Recursive Functions
Command Line Argument
Environment Variables in C Programs
Recursion Example
Recursion Vs Iteration
Code/Space/Time Complexity

Standard I/O Library

Files & Streams
Streams Buffers
IO Buffers – Line Vs Full Vs No-Buffers
Setting & Flushing Buffers
File Access
File Access Modes
Sequential Vs Random Access
Concept of File Offsets
File Operation Errors
End-of-File Condition?
Return Values and Error Values
Character Based File I/O
Line Based File I/O
Formatted File I/O
Block File I/O
Dangerous – gets() Vs fgets()
File Random Access Methods

Syllabus of System Level C Programming Course

Logic circuits

Registers basic

Architecture of processors

Types of memories like ROM, RAM, and Cache

BIOS(Basic Input Output System)

CMOS
Header files detail study and modifications

Advanced system functions

BIOS related functions
Graphics related functions
Processor related functions

Creation of child process and parent process

Software interrupts
Hardware interrupts

DMA controlling

Serial ports and Parallel ports in depth study

Advanced pointers

Near pointers
Far pointers
Huge pointers

Advanced pre-processor directives

Optimization of C programming performance
Understanding Little and big Endian architecture
Common errors and their reasons

Advanced command level arguments

Virtual memory organization
Advanced storage class manipulations
Undefined behaviour of C
Buffer overflow
Memory overflow and Memory underflow
Segmentation fault

DOS programming

TSR(Terminate & Stay Resident programs)
How to develop Virus programs
How to develop Anti-virus programs
How to develop scanning programs
How to develop Defragmentation programs
How to develop Memory dumping programs
How to write assembly programs in C
How to program video RAM
How to program serial and parallel ports
Mode 13h programming
How to program Mouse
How to program Keyboard
How to program CMOS RAM