**C**

**1. Basics of C Programming**

|---- Intro and Code Compilation

|---- Syntax, Variables, Data Types

|---- Operators

|\_\_\_ Conditions and Loops

|---- If-else Statements

|---- Loops (for, while, do-while)

**2. Functions and Pointers**

|---- Function Prototypes

|---- Parameters and Return Types

|---- Recursion

|---- Pointers

|---- Arrays

|—- Strings

**3. Memory Management**

|---- Static vs. Dynamic Memory Allocation

|---- malloc(), calloc(), realloc(), free()

|---- Stack vs. Heap Memory

**4. Data Structures in C**

|---- Arrays, Structs, Union

|---- Linked Lists \*

|---- Stacks and Queues \*

|---- Trees and Graphs \*

|\_\_\_ Algorithms (Searching, Sorting, Traversals) \*

**5. Bit Manipulation**

|---- Bitwise Operators

|\_\_\_ Applications in Electronics

**1.Basics of C Programming**

**1.1 Intro**

* C is a medium level language
* Among other programming languages c is fastest
* Advantage: Fastest, closest to hardware
* Disadvantage: MLL, no OOPS

**Code Compilation**

* **Steps of Compilation**

**Pre-processor:** Removal of Comments, Expansion of Macros,Expansion of the included files, Conditional compilation, .C to .i

gcc -E main.c>main.i

**Compiler:** .i to .s convert the preprocessed code to assembly

gcc -S main.i

**Assembler:** .s to .o convert assembly code to object code

gcc -c main.i

**Linker:** .o to .out process the object file to executable code

gcc -o main.out main.o

**1.2 Syntax, Variables, Data Types**

#include <stdio.h> / / import libaries

int main() { // beging of main function it is the. Every c code has main function

printf("Hello, world!\n"); //

return 0; // return zero for code executed without errors; 1 for errors

}

**Variables**

* containers that holds value
* Can be of many type based on the data type of value
* Int %d, float %f, double &lf, char %c
* Input and output
* printf is used for inputs
* scanf is used for outputs

**Data types**

**Basic data Types**

* int: Integer data type. It typically represents whole numbers.
* char: Character data type. It represents a single character.
* float: Floating-point data type. It represents single-precision floating-point numbers.
* double: Double-precision floating-point data type. It represents double-precision floating-point numbers.
* void: Void data type. It represents the absence of type. It is commonly used for functions that do not return a value.

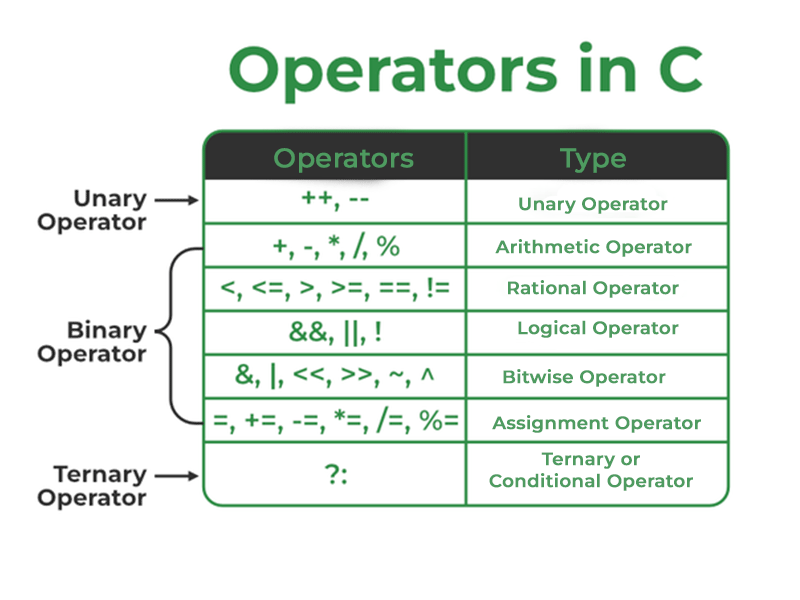
**Derived data types**

* Arrays: A collection of elements of the same data type, accessed using an index.
* Pointers: Variables that store memory addresses. They point to another variable.
* Structures: A user-defined data type that groups related data items under one name.
* Union: A special data type that allows storing different data types in the same memory location.

**Modifiers**

* signed: Specifies that a data type can hold both positive and negative values (default for int).
* unsigned: Specifies that a data type can hold only non-negative values.
* short: Specifies that a data type will take less memory (typically 2 bytes) but can represent a smaller range of values.
* long: Specifies that a data type will take more memory (typically 4 or 8 bytes) and can represent a wider range of values.

**Operator**



Unary operator: operator that need only single operand

Binary operator: operator that need two operand

Ternary operator: operator that need three operand

Operator in c:

* **Arithmetic Operations** in C
  are used to perform arithmetic/mathematical operations on operands. There are 9 arithmetic operators in C language:
  + |+ plus| - minus| / divide | \* multiply |% modulus |++ increment |— decrement |+unary plus(Used to specify the positive values.) |- unary minus (flips sign)|
* **The relational operators** in C are used for the comparison of the two operands. All these operators are binary operators that return true or false values as the result of comparison.The relational operators in C are used for the comparison of the two operands. All these operators are binary operators that return true or false values as the result of comparison.
  + |<|>|<=|>=|==|!=|
* **Logical Operator** in C are used to combine two or more conditions/constraints or to complement the evaluation of the original condition in consideration. The result of the operation of a logical operator is a Boolean value either true or false.
  + && and ,|| or ,! Not
* **Bitwise operators** are used to perform bit-level operations on the operands. The operators are first converted to bit-level and then the calculation is performed on the operands. Mathematical operations such as addition, subtraction, multiplication, etc. can be performed at the bit level for faster processing.
  + &,|,^,~,<<,>>

Well, above all the basic operator in c, well discuss bitwise operators later as it is very crucial.

**Conditions and Loops:**

**loops:** Loops in programming are used to repeat a block of code until the specified condition is met.

There are three types of loops in c

**For loop:** first Initializes, then condition check, then executes the body and at last, the update is done.

**Syntax:**

for (initialize expression; test expression; update expression){

// body of for loop

}

**while loop:** first Initializes, then condition checks, and then executes the body, and updating can be inside the body.

**Syntax:**

initialization\_expression;

while (test\_expression){

// body of the while loop

update\_expression;

}

**do-while loop:** do-while first executes the body and then the condition check is done Among the loops do while is only one that executes and once even it the conditions don’t met.

**Syntax:**

initialization\_expression;

do

{

// body of do-while loop

update\_expression;

} while (test\_expression);

**Conditions statements**

* **if**: if the if statement is true only then it will execute block of the statement
* **else if :** if the above if statement is false and else-if statement is true only then it will execute block of the statement
* **else**: if the above statements are false only then it will execute block of the statement

**Syntax:**

if (condition)

{

// Executes this block if

// condition is true

}

else if (condition)

{

// Executes this block if

// condition is true

}

else

{

// Executes this block if

// condition is false

}