LMNs-C Programming

Last Updated : 30 Jan, 2025

C programming is a powerful and widely-used programming language that forms the backbone of many modern technologies. Known for its simplicity and efficiency, it is the foundation for learning advanced programming concepts. C programming is a powerful and widely-used programming language that forms the backbone of many modern technologies. Known for its simplicity and efficiency, it is the foundation for learning advanced programming concepts. This "Last Minute Notes" article is designed to provide a quick and concise revision of the key topics in C programming, including data types, operators, control flow statements, functions, and storage classes.

Table of Content

Data Types and Operators

Data Types

1. Primitive Data Type

(a) Integer Types:

- short int, unsigned short int
- int, unsigned int
- long int, unsigned long int
- long long int, unsigned long long int

(b) Character Types:

- char, unsigned char

(c) Floating Types:

- float, double, long double

(d) Other:

- void, bool

2. Non – Primitive Data Type

(a) Derived data type:

- Array
- Pointer
- String

(b) User-defined data type:

- Structure
- Union
- Enum
- Typedef

Note:

- C standard does not specify how many bits/bytes to allocate for every type. Compilers may choose different ranges.
- Short and int types are at least 16 bits, long types are at least 32 bits.

read more about - Data Types

Operators

Arithmetic Operators

Used for mathematical calculations:

- `+` **(Addition):** Adds two operands. Example: `a + b`
- `-` **(Subtraction)**: Subtracts second operand from the first. Example: `a - b`
- `*` **(Multiplication):** Multiplies two operands. Example: `a * b`
- `/` **(Division):** Divides first operand by the second. Example: `a / b`
- `%` **(Modulus):** Gives the remainder of division (works with integers only). Example: `a % b`

2. Relational Operators

Used to compare two values, returning `true (1)` or `false (0)`:

- `<` (Less than)
- `>` (Greater than)
- `<=` (Less than or equal to)
- `>=` (Greater than or equal to)
- `==` (Equal to)
- `!=` (Not equal to)

3. Logical Operators

Used for logical conditions, returning `true (1)` or `false (0)`:

- `&&` **(Logical AND):** Returns true if both conditions are true. Example: `a > 0 && b > 0`
- `||` **(Logical OR):** Returns true if at least one condition is true. Example: `a > 0 || b > 0`
- `!` **(Logical NOT):** Inverts the boolean value. Example: `!(a > b)`

4. Assignment Operator

Used to assign values:

- `=`

  Example: int a = 5;

5. Increment and Decrement Operators

Used to increase or decrease a value by 1:

- `++x` (Pre-increment): Increments the value of `x` before using it.
- `x++` (Post-increment): Uses the current value of `x` and then increments it.
- `--x` (Pre-decrement): Decrements the value of `x` before using it.
- `x--` (Post-decrement): Uses the current value of `x` and then decrements it.

Example:

```
int x = 5;
printf("%d", ++x); // Outputs 6
```

6. Bitwise Operators

Operate at the bit level:

- `&` (AND): Sets each bit to 1 if both bits are 1.

- | (OR): Sets each bit to 1 if at least one bit is 1.
- ^ (XOR): Sets each bit to 1 if the bits are different.
- ~ (NOT): Inverts all the bits.
- << (Left Shift): Shifts bits to the left, adding 0s on the right.
- >> (Right Shift): Shifts bits to the right, removing bits on the right.

Example:

```
int a = 5; // Binary: 0101
int b = a << 1; // Result: 1010 (Decimal 10)
```

7. Ternary Operator

A shorthand for if-else:

- Syntax:

```
condition ? value_if_true : value_if_false
```

Example:

```
int a = 10, b = 5;
int max = (a > b) ? a : b; // Assigns the larger value to max
```

Operators	Associativity
() [] -> .	Left to right
! ~ ++ -- + - * (type) size of	Right to left
* / %	Left to right
+ -	Left to right
<< >>	Left to right
< <= > > >=	Left to right
== !=	Left to right
&	Left to right
^	Left to right
|	Left to right
&&	Left to right
||	Left to right
? :	Right to left
= += -= *= /= %= &= ^= |= <<= >>=	Right to left
,	Left to right

C Operators and Their Associativity.

read more about - Operators

Control Flow Statements

A) Decision-Making Statements

1. If Statement

Executes a block of code if a condition is true.

Syntax:

```
if (condition) {
  // statements to execute if condition is true
}
```

2. If-Else Statement

Executes one block if the condition is true; another block if false.

Syntax:

```
if (condition) {
  // statements if condition is true
}
 else {
 // statements if condition is false
}
```

3. Else-If Ladder

Used when multiple conditions need to be tested.

Syntax:

```
if (condition1) {
    // statements
} else if (condition2) {
     // statements
} else {
    // default statements
}
```

4. Switch Statement

Selects one of many blocks of code to execute based on a specific value.

Syntax:

```
switch (expression) {
    case value1:        // statements
break;
    case value2:        // statements
break;
    default:        // default statements
}
```

B) Looping Statements

1. For Loop

The for statement evaluates 3 expressions and executes the loop body until second controlling expression executes to false. It is recommended to use for loop when the number of iterations is known in advance.
Syntax:

```
for (initialization; condition; increment/decrement) {
   // statements
}
```

- expression – 1 is evaluated once before the first iteration of the loop.
- expression – 2 is a expression that determines whether to terminate the loop. expression – 2 is evaluated before every iteration. If the expression is true (non - zero), the loop body executed. If the expression is false (zero), execution of the for statement is terminated.
- expression – 3 is evaluated after each iteration.

2. While Loop

- The while statement evaluates a controlling expression before every execution of the loop body.
- If the controlling expression is true (non-zero), the loop body is executed. If the controlling expression is false (zero), then the while statement terminates
- Syntax:

```
while (condition) {
   // statements
}
```

3. Do-While Loop

- Both for and while loop checks the loop termination condition before every iteration but do while loop check the condition after executing the loop body.
- do while loop body executes at least 1 time, no matter whether the loop termination condition is false or true.
  Syntax:

```
do {
    // statements
```

```
} while (condition);
```

C) Jump Statements

1. Break Statement

Exits the nearest enclosing loop or switch statement.

Example:

```
for (int i = 1; i <= 5; i++) {
    if (i == 3) break;
    printf("%d\n", i);
}
```

2. Continue Statement

Skips the rest of the current loop iteration and moves to the next iteration.

Example:

```
for (int i = 1; i <= 5; i++) {
    if (i == 3) continue;
    printf("%d\n", i);
}
```

3. Goto Statement

Transfers control to a labeled statement.

Example:

```
int i = 1;
 start: if (i <= 5) {
    printf("%d\n", i);
    i++;
   goto start;
}
```

read more about - Control Flow Statements

Storage Class & Function

1. Memory Organization of a C Program

A C program's memory is divided into several segments:

1. Code Segment:

- Contains executable instructions of the program.
- Typically read-only and sharable.

2. Data Segment:

- **Initialized Data Segment**: Stores global/static variables initialized by the programmer.
- **Uninitialized Data Segment (BSS)**: Stores uninitialized global/static variables; initialized to zero by default.

3. Stack:

- Stores local variables and function call information.
- Follows the Last In, First Out (LIFO) principle.

4. Heap:

Used for dynamic memory allocation during runtime (e.g., using `malloc()` or `calloc()`).

2. Storage Classes in C

Storage classes determine the characteristics of a variable such as **scope**, **lifetime**, and **default value**.

Storage Class	Scope	Lifetime	Default Value	Storage Location
auto	Local to function	Till function ends	Garbage value	Stack
static	Local to function	Till the program ends	Zero (0)	Data Segment
extern	Global	Till the program ends	Zero(0)	Data Segment
register	Local to function	Till function ends	Garbage value	CPU Registers (if available)

3. Functions in C

Function Declaration

Functions are blocks of code designed to perform specific tasks. They improve modularity and reusability in a program.

Syntax:

```
return_type function_name(parameters);
```

Function Definition

```
return_type function_name(parameters) {     // function body }
```

4. Recursion

- A function that calls itself either directly or indirectly.
- Requires a base condition to prevent infinite recursion.
- Example (Factorial):

```
int factorial(int n) {
    if (n == 0) return 1; // Base condition
return n * factorial(n - 1); // Recursive call
}
```

5. Static and Dynamic Scoping

Static Scooping

- Variable binding is determined at compile-time.
- The scope of the variable depends on its declaration.

Example:

```
int a = 10;
void main() {
   {
      int a = 1;
      {
          int b;
         printf("%d", a);
      }
   }
}
```

Dynamic Scooping

- Variable binding is determined at runtime.
- Scope is based on the call stack of the program.

Example:

```
int i;
program main() {
    i = 10;
    call f();
}
procedure f() {
    int c = 20;
    call g();
}
procedure g() {
    print i;
}
```

read more about - Storage Class and Function

Comment | More info | Placement Training Program

Next Article

C++ Programming Examples

Similar Reads

Benefits of C language over other programming languages

C is a middle-level programming language developed by Dennis Ritchie during the early 1970s while working at AT&T Bell Labs in the USA. The objective of its...

3 min read

Use of FLAG in programming

Flag variable is used as a signal in programming to let the program know that a certain condition has met. It usually acts as a boolean variable indicating a condition to be...

6 min read

Structured Programming Approach with Advantages and Disadvantages

Structured Programming Approach , as the word suggests, can be defined as a programming approach in which the program is made as a single structure. It means...

2 min read

Web Programming in C++

CGI(COMMON GATEWAY INTERFACE) may be a set of standards that outline however data is changed from the online server, passing the online user's request to Associate...

5 min read

Which C++ libraries are useful for competitive programming?

C++ is one of the most recommended languages in competitive programming (please refer our previous article for the reason) C++ STL contains lots of containers which ar...

3 min read

Draw a Chess Board using Graphics Programming in C

Prerequisite: graphics.h, How to include graphics.h in CodeBlocks? In Computer Graphics, we use graphics.h which provide direct functions to draw different coordinat...

4 min read

Customized Debugging in Sublime Text using C++ for Competitive Programming

Competitive Programming is a mental sport that enables us to code a given problem under provided constraints. The purpose of this article is to guide every individual on...

15+ min read

Setting up Sublime Text For Competitive Programming (C++) Using Fast Olympic...

Competitive Programming is a mind sport where time matters the most. For that, a fast and easy-to-use tool is required for writing code. Sublime text along with the...

4 min read

10 C++ Programming Tricks That You Should Know

C++ Programming Language is a powerful, versatile, and compiled language, which means the source code is converted into machine language before the execution. In...

10 min read

C++ Programming Examples

Writing C++ programs yourself is the best way to learn the C++ language. C++ programs are also asked in the interviews. This article covers the top practice proble...

9 min read

Corporate & Communications Address:
A-143, 7th Floor, Sovereign Corporate Tower, Sector- 136, Noida, Uttar Pradesh (201305)

Registered Address:
K 061, Tower K, Gulshan Vivante Apartment, Sector 137, Noida, Gautam Buddh Nagar, Uttar Pradesh, 201305

Android Tutorial

Data Science & ML

Data Science With Python
Data Science For Beginner
Machine Learning
ML Maths
Data Visualisation
Pandas
NumPy
NLP
Deep Learning

Python Tutorial

Python Programming Examples
Django Tutorial
Python Projects
Python Tkinter
Web Scraping
OpenCV Tutorial
Python Interview Question

DevOps

Git
AWS
Docker
Kubernetes
Azure
GCP
DevOps Roadmap

School Subjects

Mathematics
Physics
Chemistry
Biology
Social Science
English Grammar

Databases

SQL
MYSQL
PostgreSQL
PL/SQL
MongoDB

Competitive Exams

Web Technologies

HTML
CSS
JavaScript
TypeScript
ReactJS
NextJS
NodeJs
Bootstrap
Tailwind CSS

Computer Science

GATE CS Notes
Operating Systems
Computer Network
Database Management System
Software Engineering
Digital Logic Design
Engineering Maths

System Design

High Level Design
Low Level Design
UML Diagrams
Interview Guide
Design Patterns
OOAD
System Design Bootcamp
Interview Questions

Commerce

Accountancy
Business Studies
Economics
Management
HR Management
Finance
Income Tax

Preparation Corner

Company-Wise Recruitment Process
Resume Templates
Aptitude Preparation
Puzzles
Company-Wise Preparation
Companies
Colleges

More Tutorials

JEE Advanced
UGC NET
UPSC
SSC CGL
SBI PO
SBI Clerk
IBPS PO
IBPS Clerk

Software Development
Software Testing
Product Management
Project Management
Linux
Excel
All Cheat Sheets
Recent Articles

Free Online Tools

Typing Test
Image Editor
Code Formatters
Code Converters
Currency Converter
Random Number Generator
Random Password Generator

Write & Earn

Write an Article
Improve an Article
Pick Topics to Write
Share your Experiences
Internships

DSA/Placements

DSA - Self Paced Course
DSA in JavaScript - Self Paced Course
DSA in Python - Self Paced
C Programming Course Online - Learn C with Data Structures
Complete Interview Preparation
Master Competitive Programming
Core CS Subject for Interview Preparation
Mastering System Design: LLD to HLD
Tech Interview 101 - From DSA to System Design [LIVE]
DSA to Development [HYBRID]
Placement Preparation Crash Course [LIVE]

Development/Testing

JavaScript Full Course
React JS Course
React Native Course
Django Web Development Course
Complete Bootstrap Course
Full Stack Development - [LIVE]
JAVA Backend Development - [LIVE]
Complete Software Testing Course [LIVE]
Android Mastery with Kotlin [LIVE]

Machine Learning/Data Science

Complete Machine Learning & Data Science Program - [LIVE]
Data Analytics Training using Excel, SQL, Python & PowerBI - [LIVE]
Data Science Training Program - [LIVE]
Mastering Generative AI and ChatGPT
Data Science Course with IBM Certification

Programming Languages

C Programming with Data Structures
C++ Programming Course
Java Programming Course
Python Full Course

Clouds/Devops

DevOps Engineering
AWS Solutions Architect Certification
Salesforce Certified Administrator Course

GATE

GATE CS & IT Test Series - 2025
GATE DA Test Series 2025
GATE CS & IT Course - 2025
GATE DA Course 2025
GATE Rank Predictor