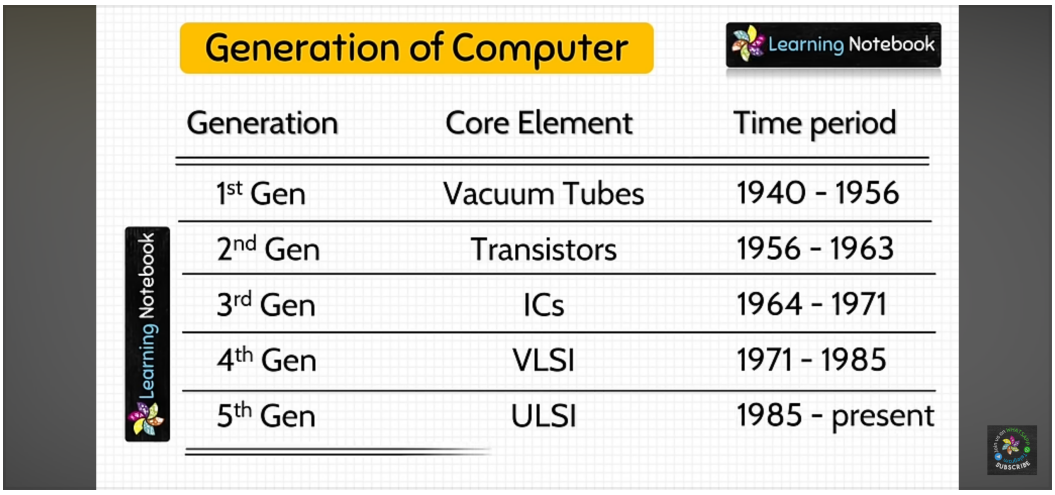
Unit - 1

Computer - A computer is a device that accepts information (in the form of digitalized data) and manipulates it for some result based on a program, software, or sequence of instructions on how the data is to be processed.



Generation	Core Element	Time period
1st Gen	Vacuum Tubes	1940 - 1956
2nd Gen	Transistors	1956 - 1963
3rd Gen	ICs	1964 - 1971
4th Gen	VLSI	1971 - 1985
5th Gen	ULSI	1985 - present

Generations of Computers – Computer Fundamentals

Generations of Computer: The modern computer took its shape with the arrival of your time. It had been around the 16th century when the evolution of the computer started. The initial computer faced many changes, obviously for the betterment. It continuously improved itself in terms of speed, accuracy, size, and price to urge the form of the fashionable day computer.

Basic Terms Related to Computers

The basic terms related to generations of computers are listed below.

1. **Vacuum Tube:** Vacuum tubes have the functionality of controlling the flow of electronics in a vacuum. Generally, it is used in switches, amplifiers, radios, televisions, etc.
2. **Transistor:** A transistor helps in controlling the flow of electricity in devices, it works as an amplifier or a switch.
3. **Integrated Circuit (IC):** Integrated circuits are silicon chips that contain their circuit elements like transistors, resistors, etc.
4. **Microprocessors:** Microprocessors are the components that contain the CPU and its circuits and are present in the Integrated Circuit.
5. **Central Processing Unit (CPU):** The CPU is called the brain

of the computer. CPU performs processing and operations work.

6. **Magnetic Drum:** Magnetic Drum is like a cylinder that stores data and cylinder.
7. **Magnetic Core:** Magnetic cores are used to store information. These are arrays of small rings.
8. **Machine Language:** Machine Language is the language that a computer accepts (in the form of binary digits). It is also called low-level programming language.
9. **Memory:** Memory is used to store data, information, and program in a computer.
10. **Artificial Intelligence:** Artificial Intelligence deals with creating intelligent machines and behaviors.

Phases of Computer Generations

This long period is often conveniently divided into the subsequent phases called computer generations.

- First Generation Computers (1940-1956)
- Second Generation Computers (1956-1963)
- Third Generation Computers (1964-1971)
- Fourth Generation Computers (1971-Present)
- Fifth Generation Computers (Present and Beyond)

Generations of Computer	Time-Period	Evolving Hardware
First Generation	1940s – 1950s	Vacuum Tube Based
Second Generation	1950s – 1960s	Transistor Based
Third Generation	1960s – 1970s	Integrated Circuit Based
Fourth Generation	1970s – Present	Microprocessor Based
Fifth Generation	Present – Future	Artificial Intelligence Based

Before the generation of computers, we used calculators, spreadsheets, and computer algebra systems, mathematicians and inventors searched for solutions to ease the burden of calculation. Below are the 8 **Mechanical Calculators** before modern computers were invented.

1. Abacus (ca. 2700 BC)
2. Pascal's Calculator (1652)

3. Stepped Reckoner (1694)
4. Arithmometer (1820)
5. Comptometer (1887) and Comptograph (1889)
6. The Difference Engine (1822)
7. Analytical Engine (1834)
8. The Millionaire (1893)

First Generation Computers

The technology behind the primary generation computers was a fragile glass device, which was called a vacuum tube. These computers were very heavy and really large. These weren't very reliable and programming on them was a tedious task as they used low-level programming language and used no OS. First-generation computers were used for calculation, storage, and control purpose. They were too bulky and large that they needed a full room and consume a lot of electricity. Punch cards were used for improving the information for external storage. Magnetic card used . Machine and assembly language is developed.
Examples of some main first-generation computers are mentioned below.

- **ENIAC:** Electronic Numerical Integrator and Computer, built by J. Presper Eckert and John V. Mauchly was a general-purpose computer. It had been cumbersome, and large, and contained 18,000 vacuum tubes.
- **EDVAC:** Electronic Discrete Variable Automatic Computer was designed by von Neumann. It could store data also as instruction and thus the speed was enhanced.
- **UNIVAC:** Universal Automatic Computer was developed in 1952 by Eckert and Mauchly.

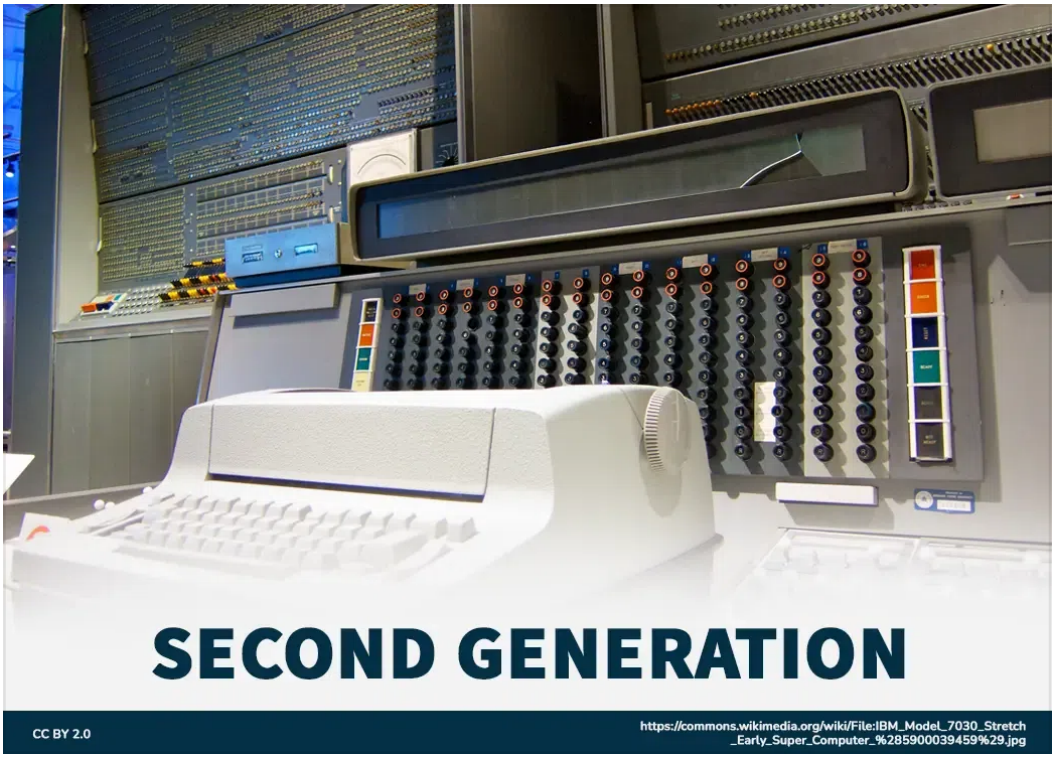
Vacuum Tube

Characteristics of First-Generation Computers

Characteristics	Components
Main electronic component	Vacuum tube.
Programming language	Machine language.
Main memory	Magnetic tapes and magnetic drums.
Input/output devices	Paper tape and punched cards.
Speed and size	Very slow and very large (often taking up an entire room).
Examples of the first generation	IBM 650, IBM 701, ENIAC, UNIVAC1, etc.

Second Generation Computers

Second-generation computers used the technology of transistors rather than bulky vacuum tubes. Another feature was the core storage. A transistor may be a device composed of semiconductor material that amplifies a sign or opens or closes a circuit.



Second Generation Computer

Transistors were invented in Bell Labs. The use of transistors made it possible to perform powerfully and with due speed. It reduced the dimensions and price and thankfully the warmth too, which was generated by vacuum tubes. Central Processing Unit (CPU), memory, programming language, and input, and output units also came into the force within the second generation.

The programming language was shifted from high level to programming language and made programming comparatively a simple task for programmers. Languages used for programming during this era were FORTRAN (1956), ALGOL (1958), and COBOL (1959).

Transistor

Characteristics of Second-Generation Computers

Characteristics	Components
Main electronic component	Transistor.
Programming language	Machine language and assembly language.
Memory	Magnetic core and magnetic tape/disk.
Input/output devices	Magnetic tape and punched cards.
Power and size	Smaller in size, had low power consumption, and generated less heat (in comparison with the first-generation computers).
Examples of the second generation	PDP-8, IBM1400 series, IBM 7090 and 7094, UNIVAC 1107, CDC 3600, etc.

Third Generation Computers

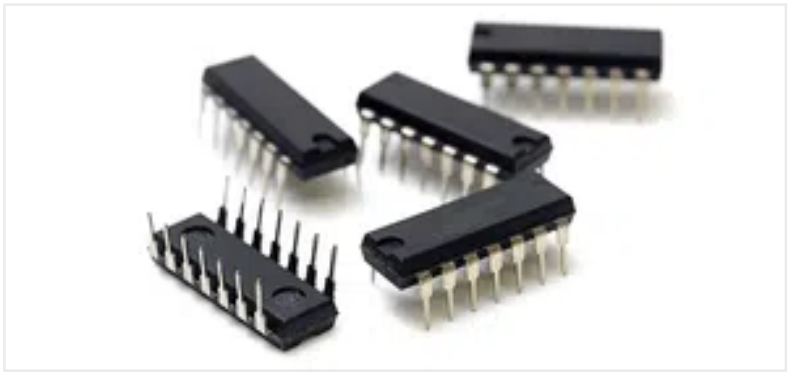
During the third generation, technology envisaged a shift from huge transistors to integrated circuits, also referred to as IC. Here a variety of transistors were placed on silicon chips, called semiconductors.

The most feature of this era’s computer was speed and reliability. IC was made from silicon and also called silicon chips.

Third Generation Computers

The computer programs was designed to make the machine work. Operating system was a program designed to handle a machine completely. Because of the operating system machine could execute multiple jobs simultaneously. Integrated circuits were used to replace many transistors used in the second generation.

A single IC has many transistors, registers, and capacitors built on one thin slice of silicon. The value size was reduced and memory space and dealing efficiency were increased during this generation. Programming was now wiped out Higher level languages like BASIC (Beginners All-purpose Symbolic Instruction Code). Minicomputers find their shape during this era.



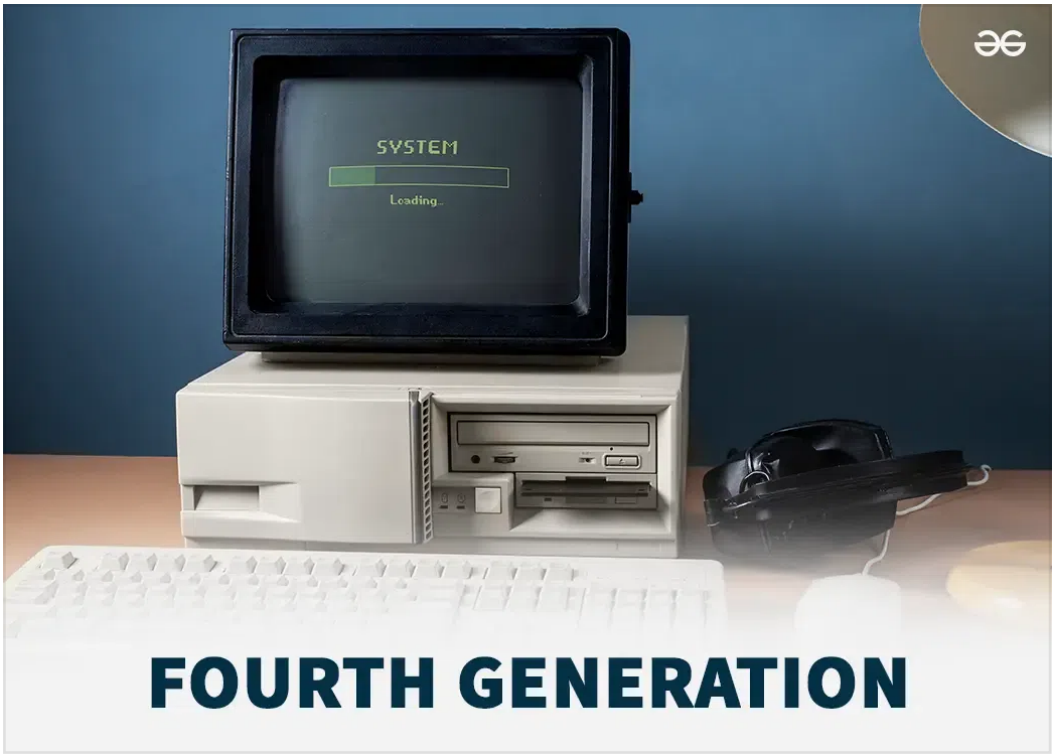
Integrated Circuit

Characteristics of Third-Generation Computers

Characteristics	Components
Main electronic component	Integrated circuits (ICs).
Programming language	High-level language.
Memory	Large magnetic core, magnetic tape/disk.
Input/output devices	Magnetic tape, monitor, keyboard, printer, etc.
Examples of the third generation	IBM 360, IBM 370, PDP-11, NCR 395, B6500, UNIVAC 1108, etc.

Fourth Generation Computers

In 1971 First microprocessors were used, the large-scale of integration LSI circuits built on one chip called microprocessors. The advantage of this technology is that one microprocessor can contain all the circuits required to perform arithmetic, logic, and control functions on one chip. LSI placed thousands of transistors onto a single chip.



Fourth Generation Computer

The computers using microchips were called microcomputers. This generation provided even smaller size of computers, with larger capacities. That's not enough, then Very Large Scale Integrated (VLSI) circuits replaced LSI circuits. The Intel 4004 chip, developed in 1971, located all the components of the pc from the central

processing unit and memory to input/ output controls on one chip and allowed the dimensions to reduce drastically. VLSI placed several hundred thousand transistors on a single silicon chip. This silicon chip is known as the micro processor.
Technologies like multiprocessing, multiprogramming, time-sharing, operating speed, and virtual memory made it a more user-friendly and customary device. The concept of private computers and computer networks came into being within the fourth generation.

Microprocessor

Characteristics of Fourth-Generation Computers

Characteristics	Components
Main electronic component	Very-large-scale integration (VLSI) and the microprocessor (VLSI has thousands of transistors on a single microchip).
Memory	semiconductor memory (such as RAM, ROM, etc.).
Input/output devices	pointing devices, optical scanning, keyboard, monitor, printer, etc.
Examples of the fourth generation	IBM PC, STAR 1000, APPLE II, Apple Macintosh, Alter 8800, etc.

Fifth Generation Computers

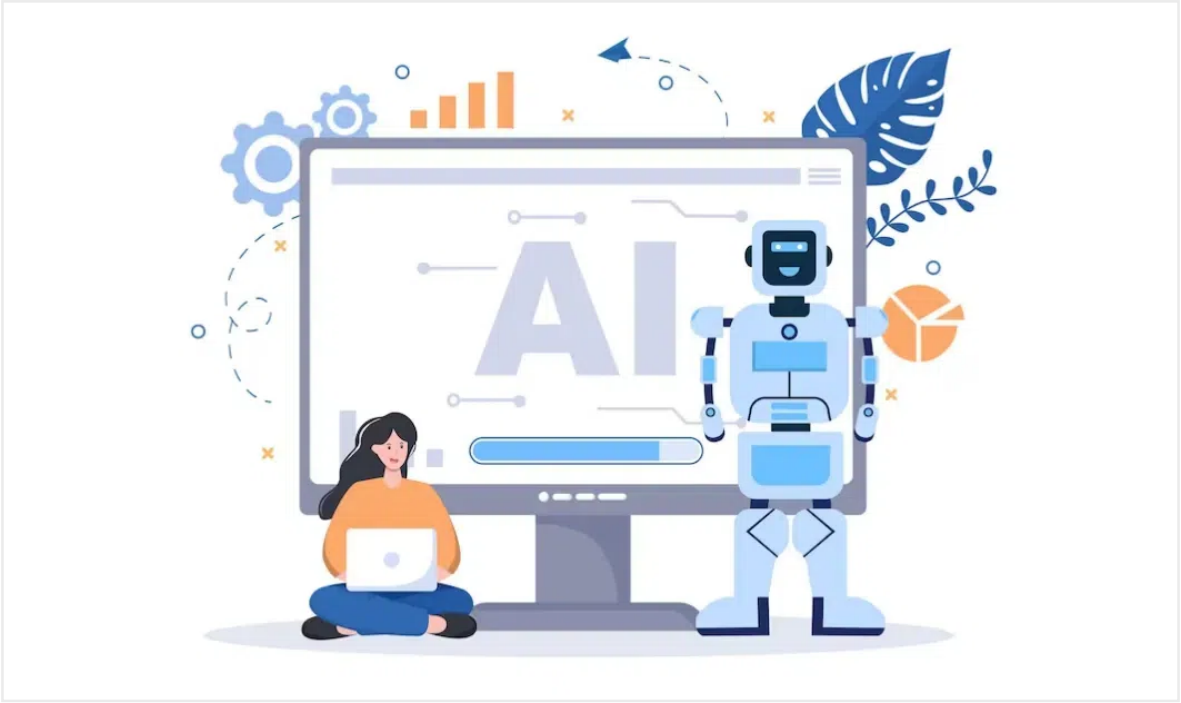
The technology behind the fifth generation of computers is AI. It allows computers to behave like humans. It is often seen in programs

like voice recognition, area of medicine, and entertainment. Within the field of game playing also it's shown remarkable performance where computers are capable of beating human competitors.

Fifth-Generation-Computers

The speed is the highest, size is the smallest and area of use has remarkably increased within the fifth generation computers. Though not a hundred percent AI has been achieved to date but keeping in sight the present developments, it is often said that this dream also will become a reality very soon.

To summarize the features of varied generations of computers, it is often said that a big improvement has been seen so far because of the speed and accuracy of functioning care, but if we mention the dimensions, it's been small over the years. The value is additionally diminishing and reliability is increasing.



AI-Based Computers

Characteristics of Fifth-Generation Computers

Characteristics	Components
Main electronic component	Based on artificial intelligence, uses the Ultra Large-Scale Integration (ULSI) technology and parallel processing method (ULSI has millions of transistors on a single microchip and the Parallel processing method use two or more microprocessors to run tasks simultaneously).
Language	Understand natural language (human language).
Size	Portable and small in size.
Input/output device	Trackpad (or touchpad), touchscreen, pen, speech input (recognize voice/ speech), light scanner, printer, keyboard, monitor, mouse, etc.
Example of the fifth generation	Desktops, laptops, tablets, smartphones, etc.

FAQs on Generations of Computer

What are the 5 types of generation of computer?

The five generations of computers are:

1. First Generation (1940s-1950s):

- *Characterized by vacuum tubes and punched cards.*
- *Examples: ENIAC, UNIVAC.*

2. Second Generation (1950s-1960s):

- *Transistors replaced vacuum tubes, allowing smaller and more efficient computers.*
- *Introduction of high-level programming languages.*
- *Examples: IBM 1401, IBM 7094.*

3. Third Generation (1960s-1970s):

- *Integrated circuits (ICs) replaced transistors, leading to smaller and faster computers.*
- *Introduction of operating systems.*
- *Examples: IBM System/360, DEC PDP-11.*

4. Fourth Generation (1970s-1980s):

- *Microprocessors brought computing power to individual users.*
- *Introduction of personal computers.*
- *Examples: IBM PC, Apple Macintosh.*

5. Fifth Generation (1980s-Present):

- *Focus on parallel processing, artificial intelligence (AI), and natural language processing.*
- *Development of supercomputers and expert systems.*
- *Ongoing advancements in AI and machine learning.*
- *Examples: IBM Watson, Google's DeepMind.*

What is Gen Z technology?

Gen Z technology encompasses the digital tools and platforms that define the experiences of individuals born roughly between the mid-1990s and early 2010s. This generation is characterized by its seamless integration of smartphones, social media, online collaboration, and video content into daily life, shaping their communication, learning, and entertainment habits.

What is Artificial Intelligence?

Artificial Intelligence (AI) is the simulation of human intelligence in machines. It involves programming computers to think, learn, and perform tasks that traditionally require human intelligence, such as problem-solving and decision-making. AI encompasses subfields like machine learning and natural language processing, with applications ranging from virtual assistants to autonomous vehicles.

What was the First Computer?

The ENIAC (Electronic Numerical Integrator and Computer), completed in 1945, is widely regarded as the first electronic general-purpose computer.

Who is Known as the Father of Computers?

Charles Babbage is known as the Father of Computers for his pioneering work on the concept of a programmable mechanical computer in the 19th century.

Operating System

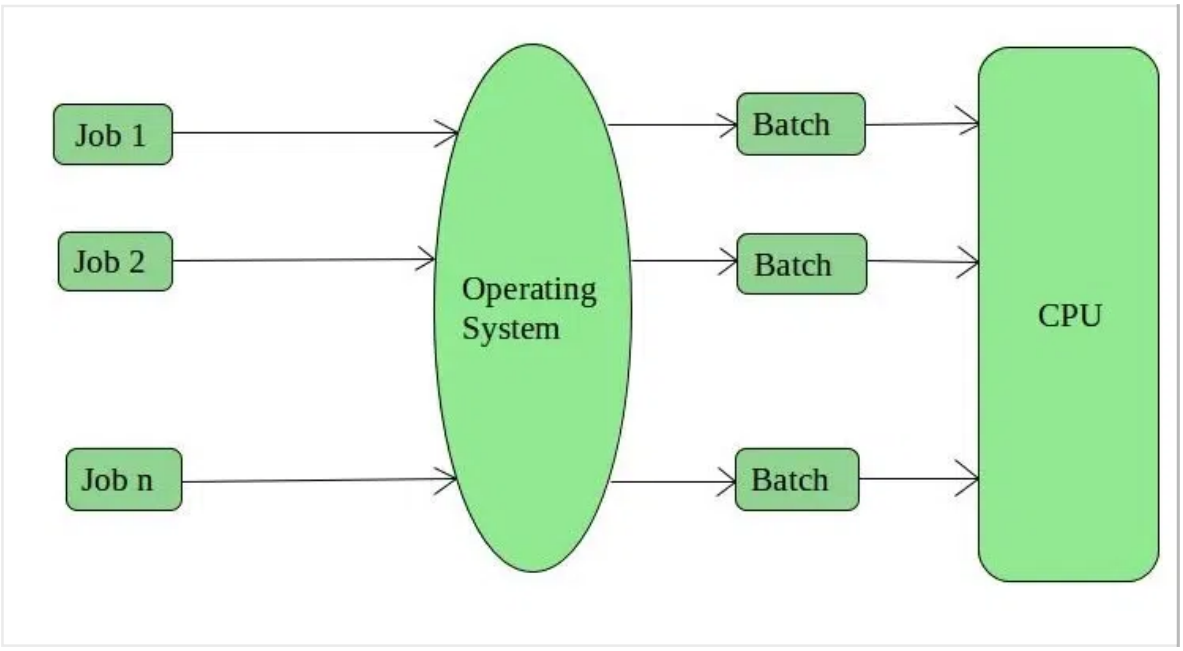
An Operating System performs all the basic tasks like managing files, processes, and memory. Thus, the operating system acts as the manager of all the resources, i.e. **resource manager**. Thus, the operating system becomes an interface between the user and the machine. It is one of the most required software that is present in the device.

Operating System is a type of software that works as an interface between the system program and the hardware. There are several types of Operating Systems many of which are mentioned below. Let's have a look at them.

TYPES OF OPERATING SYSTEM :

1. Batch Operating System

This type of operating system does not interact with the computer directly. There is an operator which takes similar jobs having the same requirements and groups them into batches. It is the responsibility of the operator to sort jobs with similar needs. Batch Operating System is designed to manage and execute a large number of jobs efficiently by processing them in groups.



Batch Operating System

Understanding the various types of operating systems is crucial for exams like GATE, where operating systems form a core part of the syllabus. To gain a deeper understanding and enhance your preparation, check out the GATE CS Self-Paced Course . This course provides detailed insights into different operating systems, helping you build a solid foundation for your exams.

Advantages of Batch Operating System

- Multiple users can share the batch systems.
- The idle time for the batch system is very less.
- It is easy to manage large work repeatedly in batch systems.

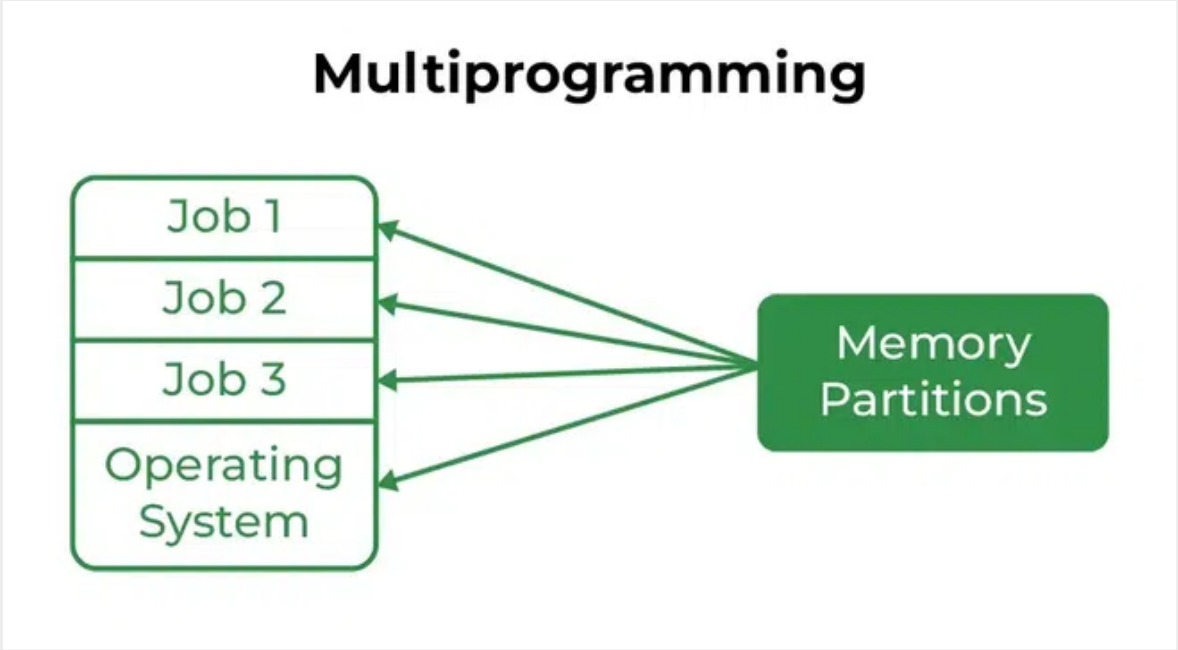
Disadvantages of Batch Operating System

- Batch systems are hard to debug.
- It is sometimes costly.
- The other jobs will have to wait for an unknown time if any job fails.
- In batch operating system the processing time for jobs is commonly difficult to accurately predict while they are in the queue.

Examples of Batch Operating Systems: Payroll Systems, Bank Statements, etc.

2. Multi-Programming Operating System

Multiprogramming Operating Systems can be simply illustrated as more than one program is present in the main memory and any one of them can be kept in execution. This is basically used for better utilization of resources.



MultiProgramming

Advantages of Multi-Programming Operating System

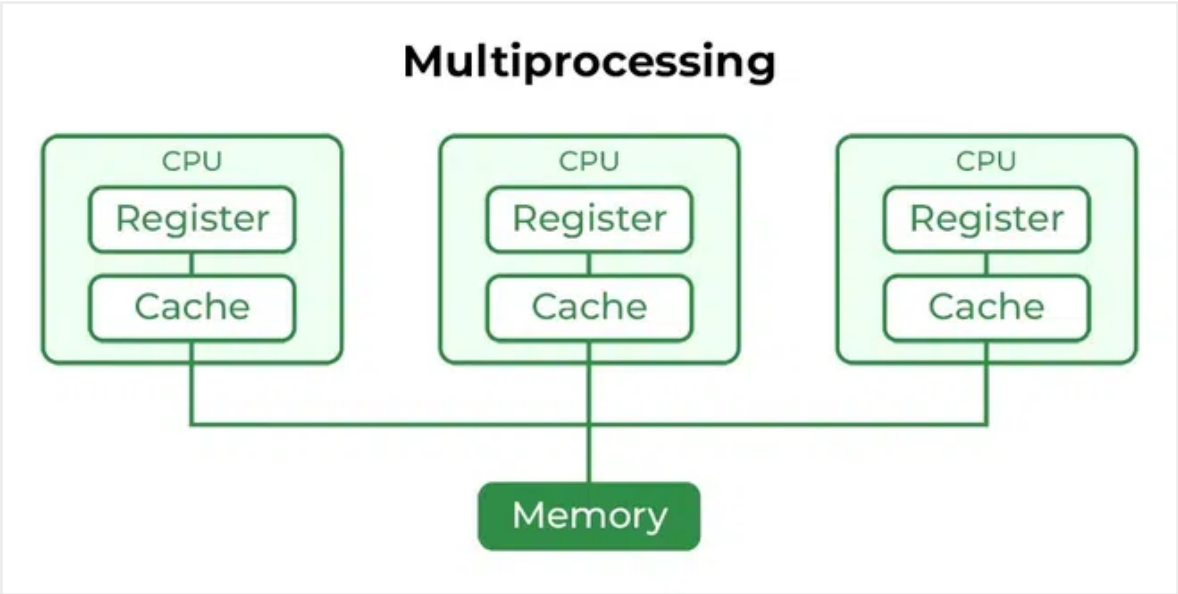
- Multi Programming increases the Throughput of the System.
- It helps in reducing the response time.

Disadvantages of Multi-Programming Operating System

- There is not any facility for user interaction of system resources with the system.

3. Multi-Processing Operating System

Multi-Processing Operating System is a type of Operating System in which more than one CPU is used for the execution of resources. It betters the throughput of the System.



Multiprocessing Operating System

Advantages of Multi-Processing Operating System

- It increases the throughput of the system.
- As it has several processors, so, if one processor fails, we can proceed with another processor.

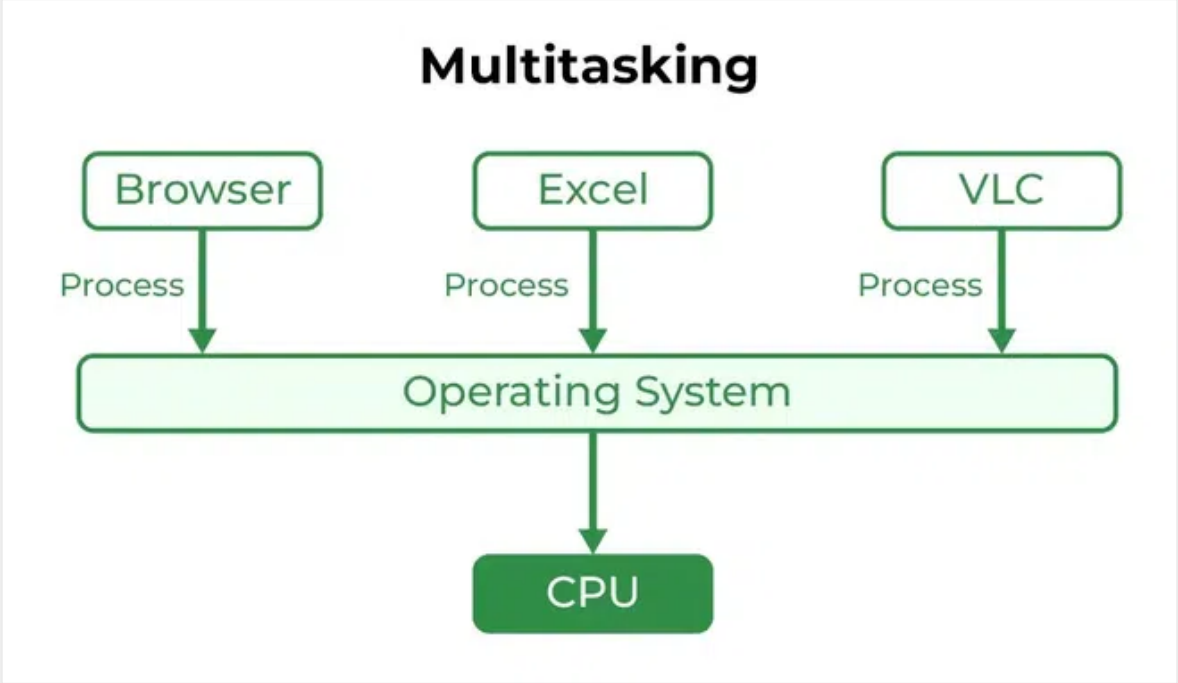
Disadvantages of Multi-Processing Operating System

- Due to the multiple CPU, it can be more complex and somehow difficult to understand.

4. Multi-Tasking Operating System

Multitasking Operating System is simply a multiprogramming Operating System with having facility of a Round-Robin Scheduling Algorithm. It can run multiple programs simultaneously.
There are two types of Multi-Tasking Systems which are listed below.

- Preemptive Multi-Tasking
- Cooperative Multi-Tasking



Multitasking Operating System

Advantages of Multi-Tasking Operating System

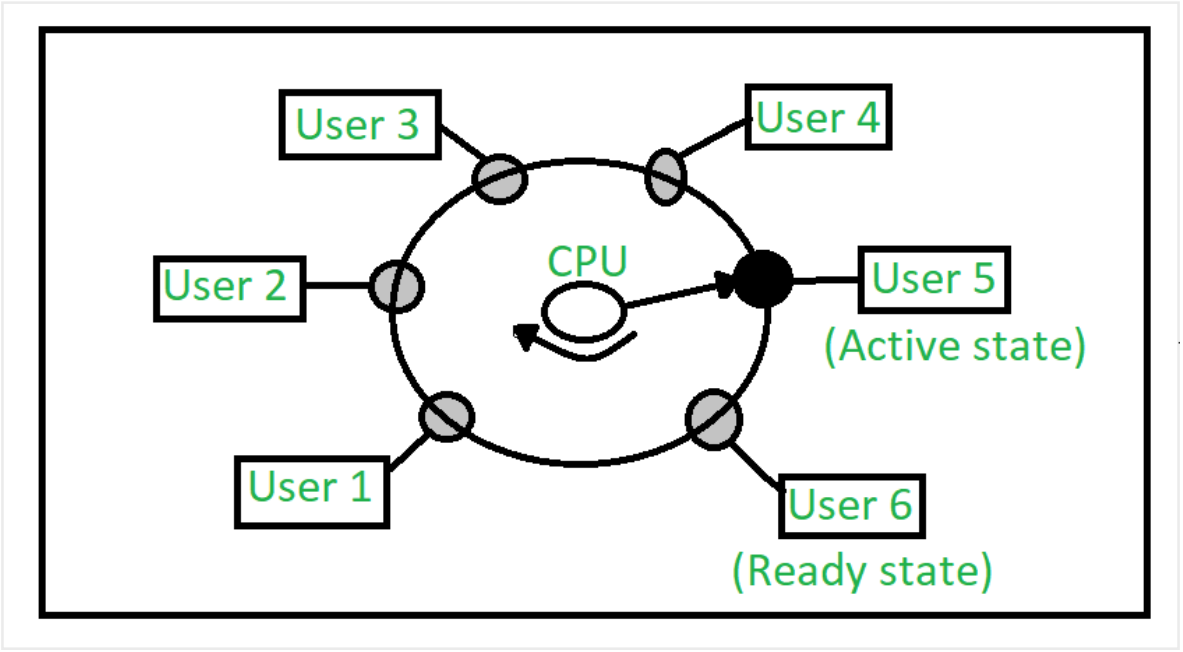
- Multiple Programs can be executed simultaneously in Multi-Tasking Operating System.
- It comes with proper memory management.

Disadvantages of Multi-Tasking Operating System

- The system gets heated in case of heavy programs multiple times.

5. Time-Sharing Operating Systems

Each task is given some time to execute so that all the tasks work smoothly. Each user gets the time of the CPU as they use a single system. These systems are also known as Multitasking Systems. The task can be from a single user or different users also. The time that each task gets to execute is called quantum. After this time interval is over OS switches over to the next task.



Time-Sharing OS

Advantages of Time-Sharing OS

- Each task gets an equal opportunity.
- Fewer chances of duplication of software.
- CPU idle time can be reduced.
- Resource Sharing: Time-sharing systems allow multiple users to share hardware resources such as the CPU, memory, and peripherals, reducing the cost of hardware and increasing efficiency.
- Improved Productivity: Time-sharing allows users to work concurrently, thereby reducing the waiting time for their turn to use the computer. This increased productivity translates to more work getting done in less time.
- Improved User Experience: Time-sharing provides an interactive environment that allows users to communicate with the computer in real time, providing a better user experience than batch processing.

Disadvantages of Time-Sharing OS

- Reliability problem.
- One must have to take care of the security and integrity of user programs and data.
- Data communication problem.
- High Overhead: Time-sharing systems have a higher overhead than other operating systems due to the need for scheduling, context switching, and other overheads that come with supporting multiple users.
- Complexity: Time-sharing systems are complex and require advanced software to manage multiple users simultaneously. This complexity increases the chance of bugs and errors.
- Security Risks: With multiple users sharing resources, the risk of security breaches increases. Time-sharing systems require careful management of user access, authentication, and authorization to ensure the security of data and software.

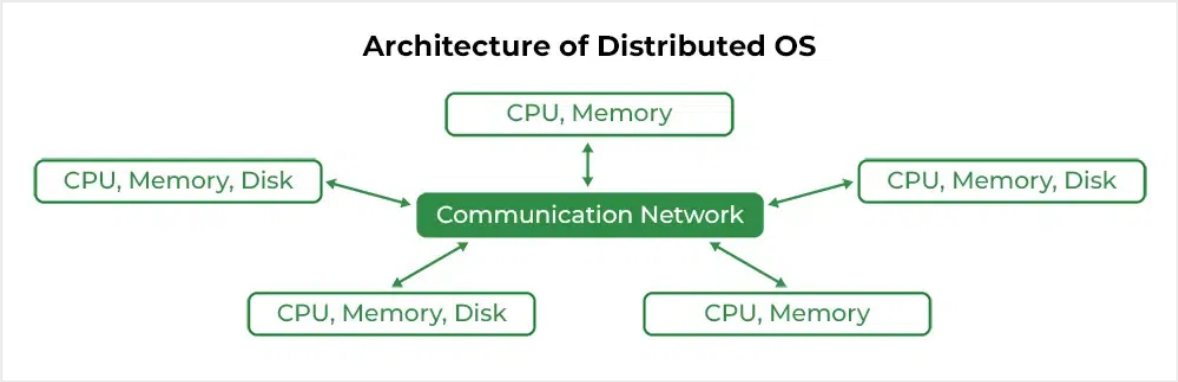
Examples of Time-Sharing OS with explanation

- IBM VM/CMS : IBM VM/CMS is a time-sharing operating system that was first introduced in 1972. It is still in use today, providing a virtual machine environment that allows multiple users to run their own instances of operating systems and applications.
- TSO (Time Sharing Option) : TSO is a time-sharing operating system that was first introduced in the 1960s by IBM for the IBM System/360 mainframe computer. It allowed multiple users to access the same computer simultaneously, running their own applications.
- Windows Terminal Services : Windows Terminal Services is a time-sharing operating system that allows multiple users to access a Windows server remotely. Users can run their own applications and access shared resources, such as printers and network storage, in real-time.

6. Distributed Operating System

These types of operating system is a recent advancement in the world of computer technology and are being widely accepted all over the world and, that too, at a great pace. Various autonomous interconnected computers communicate with each other using a shared communication network. Independent systems possess their own memory unit and CPU. These are referred to as loosely coupled

systems or distributed systems . These systems' processors differ in size and function. The major benefit of working with these types of the operating system is that it is always possible that one user can access the files or software which are not actually present on his system but some other system connected within this network i.e., remote access is enabled within the devices connected in that network.



Distributed OS

Advantages of Distributed Operating System

- Failure of one will not affect the other network communication, as all systems are independent of each other.
- Electronic mail increases the data exchange speed.
- Since resources are being shared, computation is highly fast and durable.
- Load on host computer reduces.
- These systems are easily scalable as many systems can be easily added to the network.
- Delay in data processing reduces.

Disadvantages of Distributed Operating System

- Failure of the main network will stop the entire communication.
- To establish distributed systems the language is used not well-defined yet.
- These types of systems are not readily available as they are very expensive. Not only that the underlying software is highly complex and not understood well yet.

Examples of Distributed Operating Systems are LOCUS, etc.

Issues With Distributed Operating Systems

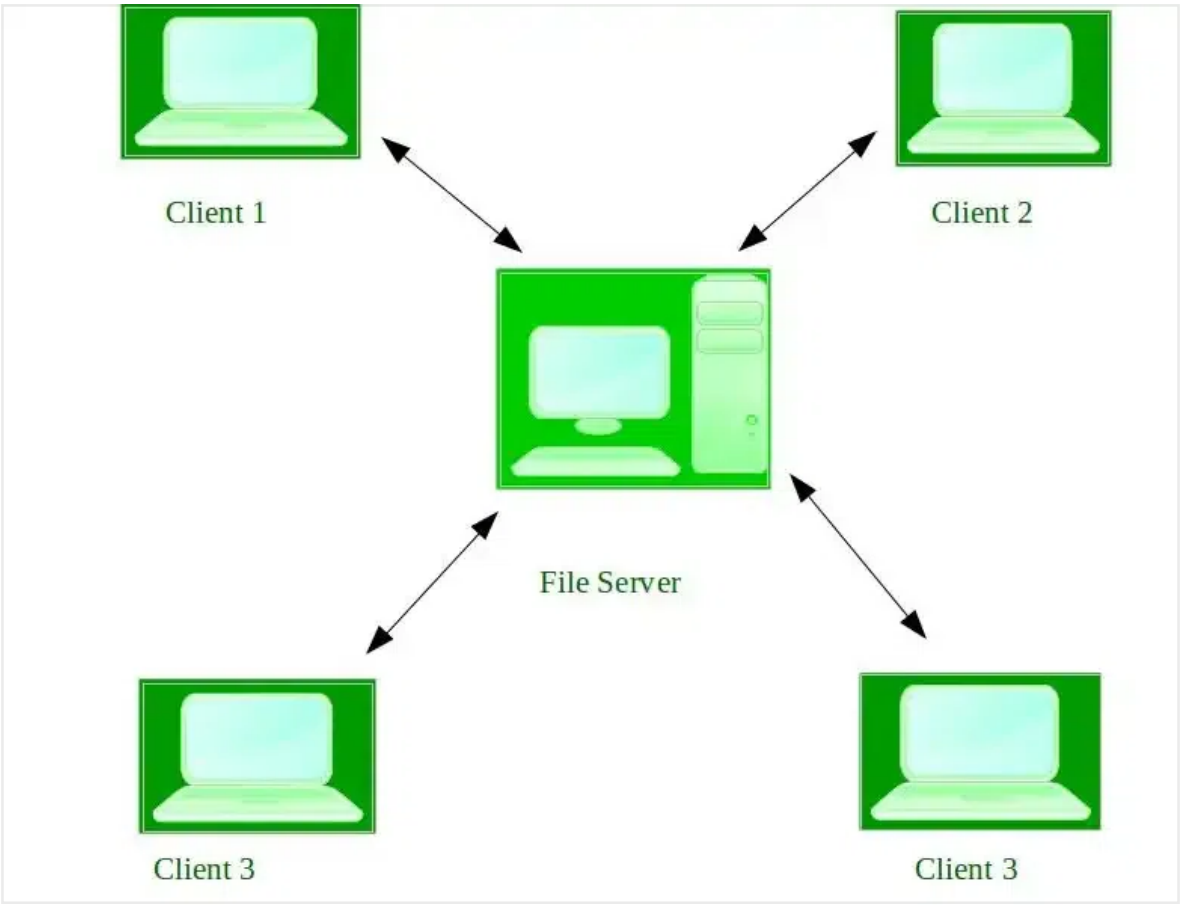
- Networking causes delays in the transfer of data between nodes of a distributed system. Such delays may lead to an inconsistent view of data located in different nodes, and make it difficult to know the chronological order in which events

occurred in the system.

- Control functions like scheduling, resource allocation, and deadlock detection have to be performed in several nodes to achieve computation speedup and provide reliable operation when computers or networking components fail.
- Messages exchanged by processes present in different nodes may travel over public networks and pass through computer systems that are not controlled by the distributed operating system. An intruder may exploit this feature to tamper with messages, or create fake messages to fool the authentication procedure and masquerade as a user of the system.

7. Network Operating System

These systems run on a server and provide the capability to manage data, users, groups, security, applications, and other networking functions. These types of operating systems allow shared access to files, printers, security, applications, and other networking functions over a small private network. One more important aspect of Network Operating Systems is that all the users are well aware of the underlying configuration, of all other users within the network, their individual connections, etc. and that's why these computers are popularly known as tightly coupled systems .



Network Operating System

Advantages of Network Operating System

- Highly stable centralized servers.
- Security concerns are handled through servers.
- New technologies and hardware up-gradation are easily integrated into the system.
- Server access is possible remotely from different locations and types of systems.

Disadvantages of Network Operating System

- Servers are costly.
- User has to depend on a central location for most operations.
- Maintenance and updates are required regularly.

Examples of Network Operating Systems are Microsoft Windows Server 2003, Microsoft Windows Server 2008, UNIX, Linux, Mac OS X, Novell NetWare, BSD, etc.

8. Real-Time Operating System

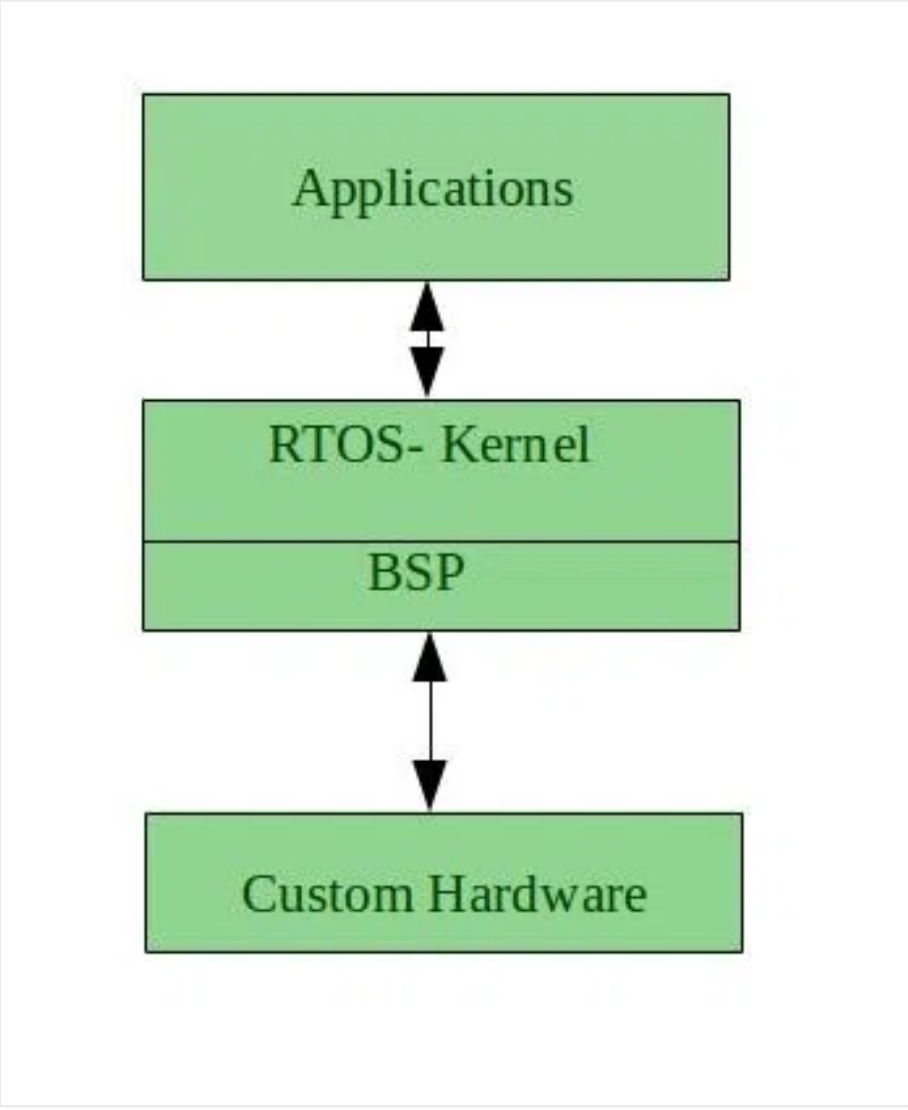
These types of OSs serve real-time systems. The time interval required to process and respond to inputs is very small. This time interval is called response time. Real-time systems are used when there are time requirements that are very strict like missile systems,

air traffic control systems, robots, etc.

Types of Real-Time Operating Systems

- Hard Real-Time Systems: Hard Real-Time OSs are meant for applications where time constraints are very strict and even the shortest possible delay is not acceptable. These systems are built for saving life like automatic parachutes or airbags which are required to be readily available in case of an accident. Virtual memory is rarely found in these systems.
- Soft Real-Time Systems: These OSs are for applications where time-constraint is less strict.

For more, refer to the Difference Between Hard Real-Time OS and Soft Real-Time OS .



Real-Time Operating System

Advantages of RTOS

- Maximum Consumption: Maximum utilization of devices and systems, thus more output from all the resources.
- Task Shifting: The time assigned for shifting tasks in these

systems is very less. For example, in older systems, it takes about 10 microseconds in shifting from one task to another, and in the latest systems, it takes 3 microseconds.
- Focus on Application: Focus on running applications and less importance on applications that are in the queue.
- Real-time operating system in the embedded system: Since the size of programs is small, RTOS can also be used in embedded systems like in transport and others.
- Error Free: These types of systems are error-free.
- Memory Allocation: Memory allocation is best managed in these types of systems.

Disadvantages of RTOS

- Limited Tasks: Very few tasks run at the same time and their concentration is very less on a few applications to avoid errors.
- Use heavy system resources: Sometimes the system resources are not so good and they are expensive as well.
- Complex Algorithms: The algorithms are very complex and difficult for the designer to write on.
- Device driver and interrupt signals: It needs specific device drivers and interrupts signal to respond earliest to interrupts.
- Thread Priority: It is not good to set thread priority as these systems are very less prone to switching tasks.

Examples of Real-Time Operating Systems are Scientific experiments, medical imaging systems, industrial control systems, weapon systems, robots, air traffic control systems, etc.

Conclusion

Operating systems come in various types, each used for specific needs. Whether it's managing large batches of jobs, enabling multiple users to work simultaneously, coordinating networked computers, or ensuring timely execution in critical systems. Understanding these types helps in choosing the right operating system for the right job, ensuring efficiency and effectiveness.

Structure of the C Program

The basic structure of a C program is divided into 6 parts which makes it easy to read, modify, document, and understand in a particular format. C program must follow the below-mentioned outline

in order to successfully compile and execute. Debugging is easier in a well-structured C program.
There are 6 basic sections responsible for the proper execution of a program. Sections are mentioned below:

1. Documentation
2. Preprocessor Section
3. Definition
4. Global Declaration
5. Main() Function
6. Sub Programs

1. Documentation

This section consists of the description of the program, the name of the program, and the creation date and time of the program. It is specified at the start of the program in the form of comments. Documentation can be represented as:
// description, name of the program, programmer name, date, time etc.
or

```
/*
   description, name of the program, programmer name, date, time etc.
*/
```

Anything written as comments will be treated as documentation of the program and this will not interfere with the given code. Basically, it gives an overview to the reader of the program.

2. Preprocessor Section

All the header files of the program will be declared in the preprocessor section of the program. Header files help us to access other's improved code into our code. A copy of these multiple files is inserted into our program before the process of compilation.
Example:

```
#include<stdio.h>
#include<math.h>
```

3. Definition

Preprocessors are the programs that process our source code before the process of compilation. There are multiple steps which are involved in the writing and execution of the program. Preprocessor directives start with the '#' symbol. The #define preprocessor is used to create a constant throughout the program. Whenever this name is encountered by the compiler, it is replaced by the actual piece of

defined code.
Example:
#define long long ll

4. Global Declaration

The global declaration section contains global variables, function declaration, and static variables. Variables and functions which are declared in this scope can be used anywhere in the program.
Example:
int num = 18;

5. Main() Function

Every C program must have a main function. The main() function of the program is written in this section. Operations like declaration and execution are performed inside the curly braces of the main program. The return type of the main() function can be int as well as void too. void() main tells the compiler that the program will not return any value. The int main() tells the compiler that the program will return an integer value.
Example:
void main()
or
int main()

6. Sub Programs

User-defined functions are called in this section of the program. The control of the program is shifted to the called function whenever they are called from the main or outside the main() function. These are specified as per the requirements of the programmer.
Example:

```
int sum(int x, int y)
{
   return x+y;
}
```

Structure of C Program with example

Example: Below C program to find the sum of 2 numbers:
C

```
// Documentation
/**
 * file: sum.c
 * author: you
 * description: program to find sum.
 */
```

```
// Link
#include <stdio.h>

// Definition
#define X 20

// Global Declaration
int sum(int y);

// Main() Function
int main(void)
{
  int y = 55;
  printf("Sum: %d", sum(y));
  return 0;
}

// Subprogram
int sum(int y)
{
  return y + X;
}
```

Explanation of the above Program

Below is the explanation of the above program. With a description explaining the program's meaning and use.

Sections	Description
/ *** file: sum.c** *** author: you** *** description: program to find sum.** ***/**	It is the comment section and is part of the description section of the code.
#include<stdio.h>	Header file which is used for standard input-output. This is the preprocessor section.
#define X 20	This is the definition section. It allows the use of constant X in the code.

int sum(int y)	This is the Global declaration section includes the function declaration that can be used anywhere in the program.
int main()	main() is the first function that is executed in the C program.
{...}	These curly braces mark the beginning and end of the main function.
printf("Sum: %d", sum(y));	printf() function is used to print the sum on the screen.
return 0;	We have used int as the return type so we have to return 0 which states that the given program is free from the error and it can be exited successfully.
int sum(int y) { return y + X; }	This is the subprogram section. It includes the user-defined functions that are called in the main() function.

Steps involved in the Compilation and execution of a C program:

- Program Creation
- Compilation of the program
- Execution of the program
- The output of the program

Read more about Compiling a C Program Compilation – Behind the Scenes

Conclusion

In this article points we learned about the structure of the C Program are mentioned below:

- *The basic structure of a C program is divided into 6 parts which makes it easy to read, modify, document, and understand in a particular format.*
- *Debugging is easier in a well-structured C program.*
- *There are 6 sections in a C Program that are Documentation, Preprocessor Section, Definition, Global Declaration, Main() Function, and Sub Programs.*
- *There are certain steps while compilation and executing of C*

program as mentioned below:

- *Creation of Program*
- *Compilation of the program*
- *Execution of the program*
- *Output of the program*

Structure of the C Program – FAQs

What is meant by the structure of a program?

The structure of a program is defined by its control flow, as structures are built up of blocks of codes. These blocks have a single entry and exit in the control flow.

What is the structure of C program syntax?

Any C Program can be divided into header, main() function, variable declaration, body, and return type of the program.

Why C is a structured program?

C is a structured programming language because it divides the programs into small modules called functions which makes the execution easier.

Character set

The character set in C is the collection of characters that can be used in a program's source code. It includes:
Alphabets: Uppercase and lowercase letters
Digits: The numbers 0–9
Special characters: Symbols like +, -, *, /, =, <, >, &, |, !, ^, ~, %, #, \, ;, :, ', " and more
White spaces: Spaces, tabs, newlines, and form feeds
Escape sequences: Special meaning characters prefixed with a backslash, such as \n (newline), \t (tab), \\ (backslash character), \' (single quote), and \" (double quote)

Tokens

Tokens are the smallest units in a C program, similar to Lego blocks. Some types of tokens include:
Identifiers: Names given to variables, functions, and arrays
Keywords: Reserved words in C that must be written in lowercase
Operators: Symbols that operate on one or more operands to give an output
Strings: Sequences of characters enclosed within single or double quotation marks
Special characters: Parentheses, braces, brackets, semicolons, and more

Constants: Fixed values that don't change during execution

Expressions & Operators :

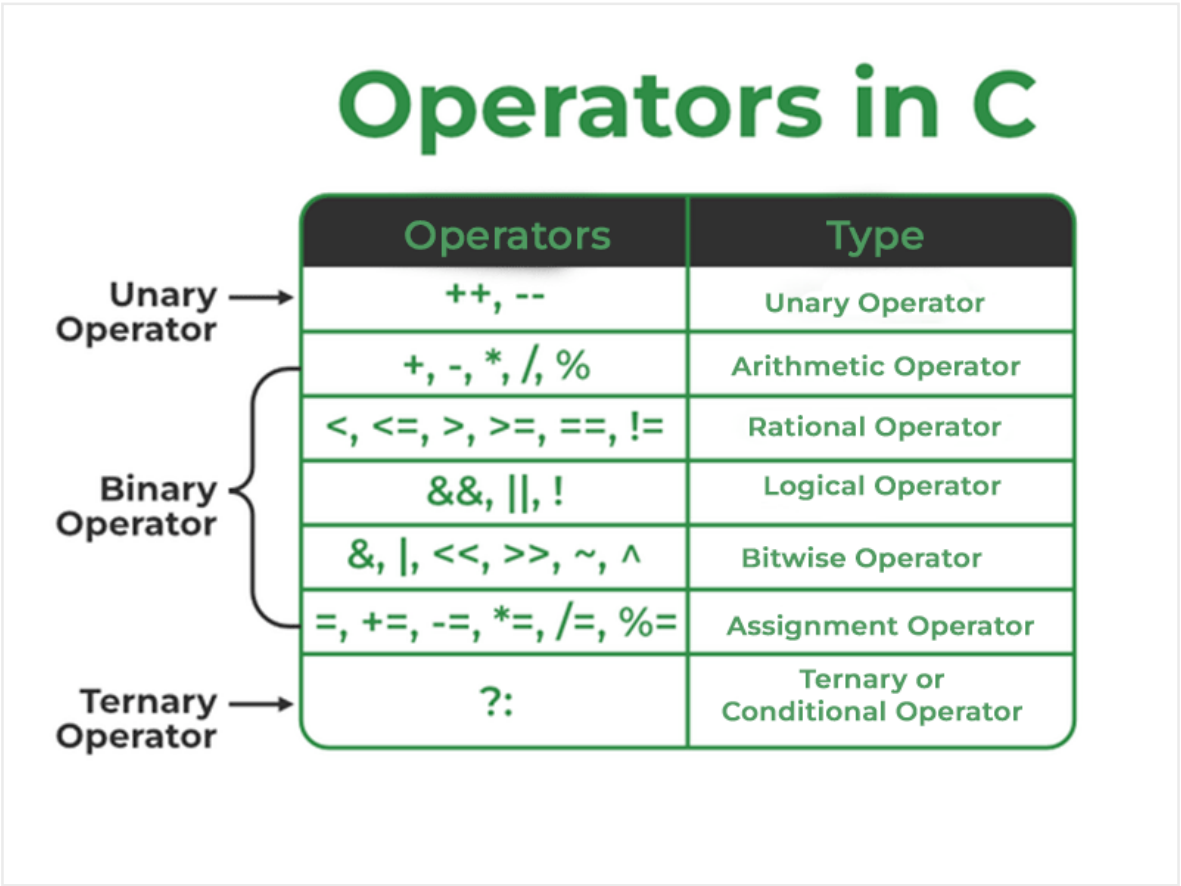
Expressions are the combination of variables, operands, and operators. The result would be stored in the variable once the expressions are processed based on the operator's precedence
Int a. = c + d; // is a expression

Operators in C

In C language, operators are symbols that represent operations to be performed on one or more operands. They are the basic components of the C programming. In this article, we will learn about all the built-in operators in C with examples.

What is a C Operator?

An operator in C can be defined as the symbol that helps us to perform some specific mathematical, relational, bitwise, conditional, or logical computations on values and variables. The values and variables used with operators are called operands. So we can say that the operators are the symbols that perform operations on operands.



For example,
c = a + b;

Here, ‘+’ is the operator known as the addition operator, and ‘a’ and ‘b’ are operands. The addition operator tells the compiler to add both of the operands ‘a’ and ‘b’. To dive deeper into how operators are used with data structures, the **C Programming Course Online with Data Structures** covers this topic thoroughly.

Types of Operators in C

C language provides a wide range of operators that can be classified into 6 types based on their functionality:

1. Arithmetic Operators
2. Relational Operators
3. Logical Operators
4. Bitwise Operators
5. Assignment Operators
6. Other Operators

1. Arithmetic Operations in C

The arithmetic operators are used to perform arithmetic/ mathematical operations on operands. There are 9 arithmetic operators in C language:

S. No.	Symbol	Operator	Description	Syntax
1	**+**	**Plus**	Adds two numeric values.	**a + b**
2	**–**	**Minus**	Subtracts right operand from left operand.	**a – b**
3	*****	**Multiply**	Multiply two numeric values.	**a * b**
4	**/**	**Divide**	Divide two numeric values.	**a / b**

5	**%**	**Modulus**	Returns the remainder after diving the left operand with the right operand.	**a % b**
6	**+**	**Unary Plus**	Used to specify the positive values.	**+a**
7	**–**	**Unary Minus**	Flips the sign of the value.	**-a**
8	**++**	**Increment**	Increases the value of the operand by 1.	a++
9	**—**	**Decrement**	Decreases the value of the operand by 1.	a–

2. Relational Operators in C

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.
These are a total of 6 relational operators in C:

S. No.	Symbol	Operator	Description	Syntax

1	**<**	**Less than**	Returns true if the left operand is less than the right operand. Else false	**a < b**
2	**>**	**Greater than**	Returns true if the left operand is greater than the right operand. Else false	**a > b**
3	**<=**	**Less than or equal to**	Returns true if the left operand is less than or equal to the right operand. Else false	**a <= b**
4	**>=**	**Greater than or equal to**	Returns true if the left operand is greater than or equal to right operand. Else false	**a >= b**

5	==	**Equal to**	Returns true if both the operands are equal.	**a == b**
6	!=	**Not equal to**	Returns true if both the operands are NOT equal.	**a != b**

3. Logical Operator in C

Logical Operators 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**.

S. No.	Symbol	Operator	Description	Syntax
1	&&	**Logical AND**	Returns true if both the operands are true.	**a && b**
2	||	**Logical OR**	Returns true if both or any of the operand is true.	**a || b**
3	!	**Logical NOT**	Returns true if the operand is false.	**!a**

4. Bitwise Operators in C

The 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.
There are 6 bitwise operators in C:

S. No.	Symbol	Operator	Description	Syntax
1	**&**	**Bitwise AND**	Performs bit-by-bit AND operation and returns the result.	**a & b**
2	**|**	**Bitwise OR**	Performs bit-by-bit OR operation and returns the result.	**a | b**
3	**^**	**Bitwise XOR**	Performs bit-by-bit XOR operation and returns the result.	**a ^ b**
4	**~**	**Bitwise First Complement**	Flips all the set and unset bits on the number.	**~a**

5	**<<**	**Bitwise Leftshift**	Shifts the number in binary form by one place in the operation and returns the result.	**a << b**
6	**>>**	**Bitwise Rightshilft**	Shifts the number in binary form by one place in the operation and returns the result.	**a >> b**

5. Assignment Operators in C

Assignment operators are used to assign value to a variable. The left side operand of the assignment operator is a variable and the right side operand of the assignment operator is a value. The value on the right side must be of the same data type as the variable on the left side otherwise the compiler will raise an error.
The assignment operators can be combined with some other operators in C to provide multiple operations using single operator. These operators are called compound operators.
In C, there are 11 assignment operators :

S. No.	Symbol	Operator	Description	Syntax
1	**=**	**Simple Assignment**	Assign the value of the right operand to the left operand.	**a = b**

2	**+=**	**Plus and assign**	Add the right operand and left operand and assign this value to the left operand.	**a += b**
3	**-=**	**Minus and assign**	Subtract the right operand and left operand and assign this value to the left operand.	**a -= b**
4	***=**	**Multiply and assign**	Multiply the right operand and left operand and assign this value to the left operand.	**a *= b**
5	**/=**	**Divide and assign**	Divide the left operand with the right operand and assign this value to the left operand.	**a /= b**

6	**%=**	**Modulus and assign**	Assign the remainder in the division of left operand with the right operand to the left operand.	**a %= b**
7	**&=**	**AND and assign**	Performs bitwise AND and assigns this value to the left operand.	**a &= b**
8	**|=**	**OR and assign**	Performs bitwise OR and assigns this value to the left operand.	**a |= b**
9	**^=**	**XOR and assign**	Performs bitwise XOR and assigns this value to the left operand.	**a ^= b**

10	**>>=**	**Rightshift and assign**	Performs bitwise Rightshift and assign this value to the left operand.	**a >>= b**
11	**<<=**	**Leftshift and assign**	Performs bitwise Leftshift and assign this value to the left operand.	**a <<= b**

6. Other Operators

Apart from the above operators, there are some other operators available in C used to perform some specific tasks. Some of them are discussed here:

sizeof Operator

- sizeof is much used in the C programming language.
- It is a compile-time unary operator which can be used to compute the size of its operand.
- The result of sizeof is of the unsigned integral type which is usually denoted by size_t.
- Basically, the sizeof the operator is used to compute the size of the variable or datatype.

Syntax

sizeof (operand)

To know more about the topic refer to this article.

Comma Operator (,)

- The comma operator (represented by the token) is a binary operator that evaluates its first operand and discards the result, it then evaluates the second operand and returns this value (and type).
- The comma operator has the lowest precedence of any C operator.
- Comma acts as both operator and separator.

Syntax

operand1 **,** operand2

To know more about the topic refer to this article.

Conditional Operator (? :)

- The conditional operator is the only ternary operator in C++.
- Here, Expression1 is the condition to be evaluated. If the condition(Expression1) is *True* then we will execute and return the result of Expression2 otherwise if the condition(Expression1) is *false* then we will execute and return the result of Expression3.
- We may replace the use of if..else statements with conditional operators.

Syntax

operand1 **?** operand2 **:** operand3;

To know more about the topic refer to this article.

dot (.) and arrow (->) Operators

- Member operators are used to reference individual members of classes, structures, and unions.
- The dot operator is applied to the actual object.
- The arrow operator is used with a pointer to an object.

Syntax

structure_variable **.** member;

and

structure_pointer **->** member;

To know more about dot operators refer to this article and to know more about arrow(->) operators refer to this article.

Cast Operator

- Casting operators convert one data type to another. For example, int(2.2000) would return 2.
- A cast is a special operator that forces one data type to be converted into another.
- The most general cast supported by most of the C compilers is as follows – **[(type) expression]**.

Syntax

(new_type) operand;

To know more about the topic refer to this article.

addressof (&) and Dereference (*) Operators

- Pointer operator & returns the address of a variable. For example &a; will give the actual address of the variable.
- The pointer operator * is a pointer to a variable. For example

*var; will pointer to a variable var.

To know more about the topic refer to this article.

Example of Other C Operators

C

```c
// C Program to demonstrate the use of Misc operators
#include <stdio.h>

int main()
{
    // integer variable
    int num = 10;
    int* add_of_num = &num;

    printf("sizeof(num) = %d bytes\n", sizeof(num));
    printf("&num = %p\n", &num);
    printf("*add_of_num = %d\n", *add_of_num);
    printf("(10 < 5) ? 10 : 20 = %d\n", (10 < 5) ? 10 : 20);
    printf("(float)num = %f\n", (float)num);

    return 0;
}
```

Output

```
sizeof(num) = 4 bytes
&num = 0×7ffe2b7bdf8c
*add_of_num = 10
(10 < 5) ? 10 : 20 = 20
(float)num = 10.000000
```

Unary, Binary and Ternary Operators in C

Operators can also be classified into three types on the basis of the number of operands they work on:

1. **Unary Operators:** Operators that work on single operand.
2. **Binary Operators:** Operators that work on two operands.
3. **Ternary Operators:** Operators that work on three operands.

Operator Precedence and Associativity in C

In C, it is very common for an expression or statement to have multiple operators and in these expression, there should be a fixed

order or priority of operator evaluation to avoid ambiguity.
Operator Precedence and Associativity *is the concept that decides which operator will be evaluated first in the case when there are multiple operators present in an expression.*
The below table describes the precedence order and associativity of operators in C. The precedence of the operator decreases from top to bottom.

Precedence	Operator	Description	Associativity
1	**()**	Parentheses (function call)	left-to-right
	[]	Brackets (array subscript)	left-to-right
	.	Member selection via object name	left-to-right
	->	Member selection via a pointer	left-to-right
	a++ , a–	Postfix increment/ decrement (a is a variable)	left-to-right
2	**++a , –a**	Prefix increment/ decrement (a is a variable)	right-to-left
	+ , –	Unary plus/ minus	right-to-left
	! , ~	Logical negation/ bitwise complement	right-to-left
	(type)	Cast (convert value to temporary value of type)	right-to-left
	*****	Dereference	right-to-left

	&	Address (of operand)	right-to-left
	sizeof	Determine size in bytes on this implementation	right-to-left
3	*** , / , %**	Multiplication/ division/ modulus	left-to-right
4	**+ , −**	Addition/ subtraction	left-to-right
5	**<< , >>**	Bitwise shift left, Bitwise shift right	left-to-right
6	**< , <=**	Relational less than/less than or equal to	left-to-right
	> , >=	Relational greater than/ greater than or equal to	left-to-right
7	**== , !=**	Relational is equal to/is not equal to	left-to-right
8	**&**	Bitwise AND	left-to-right
9	**^**	Bitwise XOR	left-to-right
10	**|**	Bitwise OR	left-to-right
11	**&&**	Logical AND	left-to-right
12	**||**	Logical OR	left-to-right
13	**?:**	Ternary conditional	right-to-left
14	**=**	Assignment	right-to-left
	+= , -=	Addition/ subtraction assignment	right-to-left

	***= , /=**	Multiplication/ division assignment	right-to-left
	%= , &=	Modulus/ bitwise AND assignment	right-to-left
	^= , |=	Bitwise exclusive/ inclusive OR assignment	right-to-left
	<<=, >>=	Bitwise shift left/right assignment	right-to-left
15	**,**	expression separator	left-to-right

To know more about operator precedence and associativity, refer to this article – Operator Precedence and Associativity in C

Conclusion

In this article, the points we learned about the operator are as follows:

- Operators are symbols used for performing some kind of operation in C.
- There are six types of operators, Arithmetic Operators, Relational Operators, Logical Operators, Bitwise Operators, Assignment Operators, and Miscellaneous Operators.
- Operators can also be of type unary, binary, and ternary according to the number of operators they are using.
- Every operator returns a numerical value except logical, relational, and conditional operator which returns a boolean value (true or false).
- There is a Precedence in the operators means the priority of using one operator is greater than another operator.

FAQs on C Operators

Q1. What are operators in C?

Answer:

Operators in C are certain symbols in C used for performing certain mathematical, relational, bitwise, conditional, or logical operations for the user.

Q2. What are the 7 types of operators in C?

Answer:

There are 7 types of operators in C as mentioned below:

- *Unary operator*
- *Arithmetic operator*
- *Relational operator*
- *Logical operator*
- *Bitwise operator*
- *Assignment operator*
- *Conditional operator*

Q3. What is the difference between the '=' and '==' operators?

Answer:

'=' is a type of assignment operator that places the value in right to the variable on left, Whereas '==' is a type of relational operator that is used to compare two elements if the elements are equal or not.

Q4. What is the difference between prefix and postfix operators in C?

Answer:

In prefix operations, the value of a variable is incremented/ decremented first and then the new value is used in the operation, whereas, in postfix operations first the value of the variable is used in the operation and then the value is incremented/decremented.

Example:

```
b=c=10;
a=b++;    // a==10
a=++c;    // a==11
```

Q5. What is the Modulo operator?

Answer:

The Modulo operator(%) is used to find the remainder if one element is divided by another.

Example:

a % b ***(a divided by b)***

5 % 2 == 1