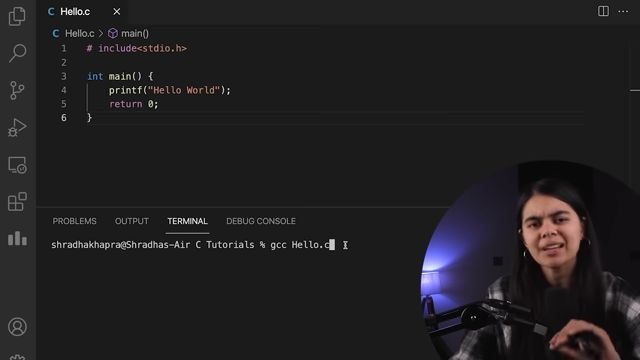
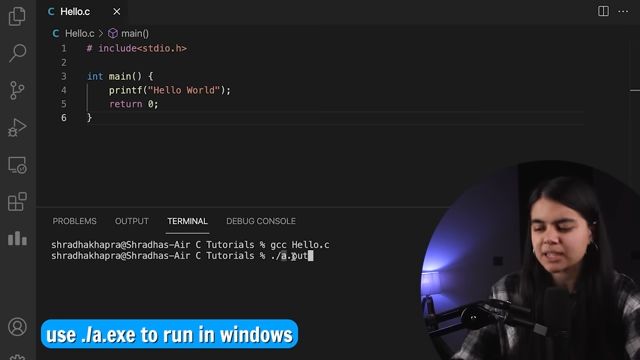
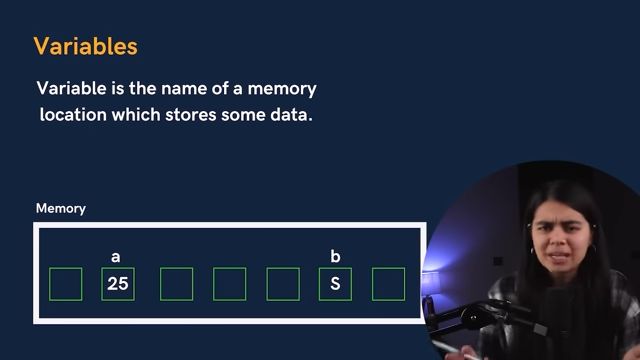
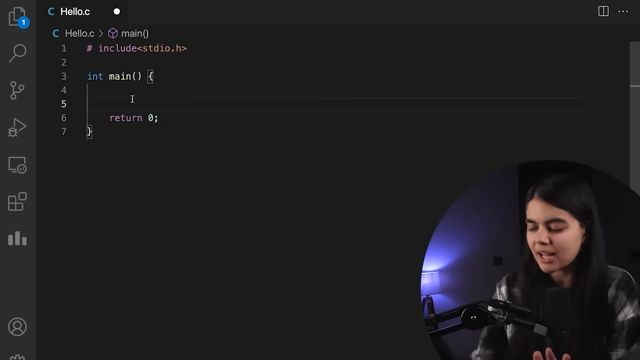
# Compiling and Running

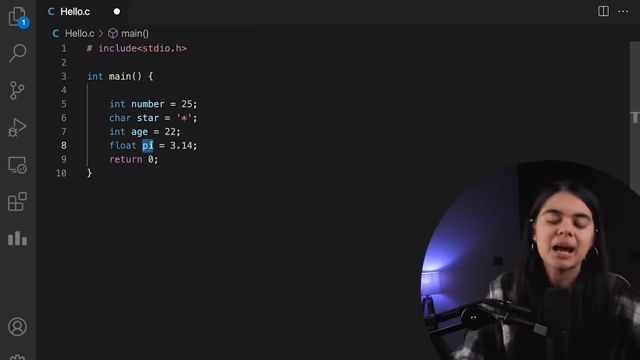
[12:59](https://youtu.be/irqbmMNs2Bo?t=779)  
  
We make a file named Hello.c  
We open terminal to write **gcc Hello.c**  
Here gcc is the compiler for c lang.  
[13:16](https://youtu.be/irqbmMNs2Bo?t=796)  
  
After compilation, to run the code we write **./a.exe** . .out is for Linux or MacOS devices while this is for Windows.

# **Chapter 01**

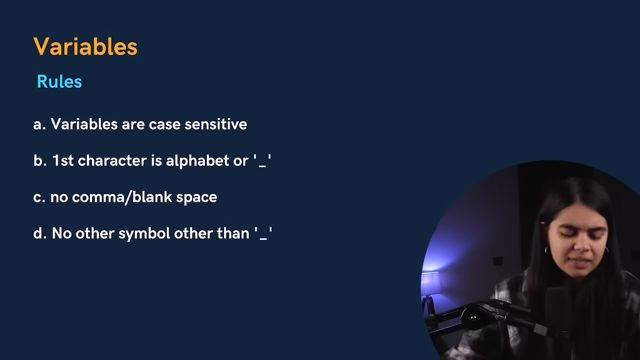
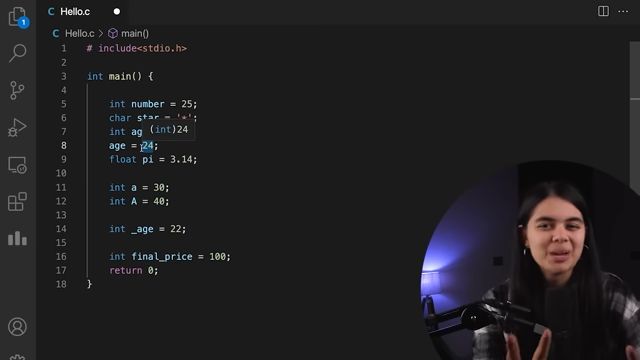
[14:39](https://youtu.be/irqbmMNs2Bo?t=879)  


# Variables :

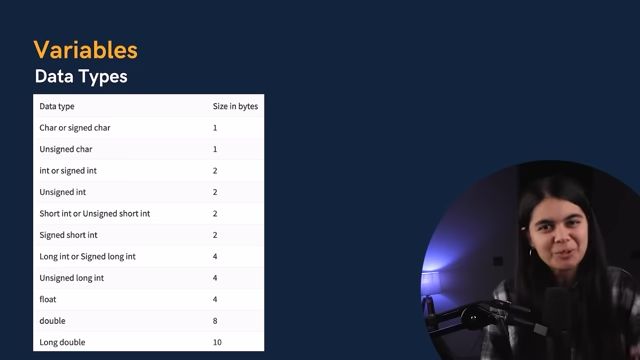
[15:01](https://youtu.be/irqbmMNs2Bo?t=901)  
a is variable for the value 25  
b is variable for the value S  
[16:43](https://youtu.be/irqbmMNs2Bo?t=1003)  
Above is the by default code of C lang. Just like there is a main class and a psvm in java.

[17:53](https://youtu.be/irqbmMNs2Bo?t=1073)  
Format to store variables in C lang. int for integer, char for characters, float for decimals

## Rules of Storing Variables :

[18:06](https://youtu.be/irqbmMNs2Bo?t=1086)  
**Note :** Value of variables can change. They are not fixed.  
[20:52](https://youtu.be/irqbmMNs2Bo?t=1252)  
**Note :** Variable names should be meaningful like you enter Ashutosh, you keep it in a 'name' variable.

## Data Types of Variables:

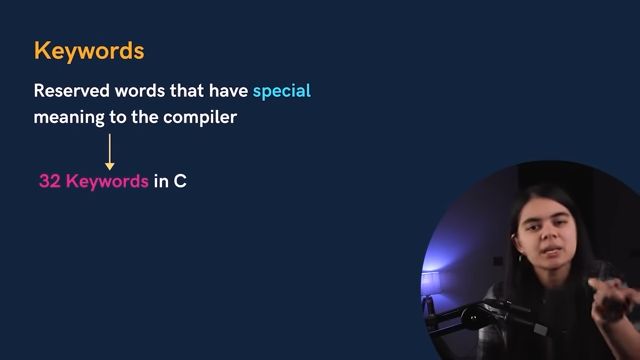
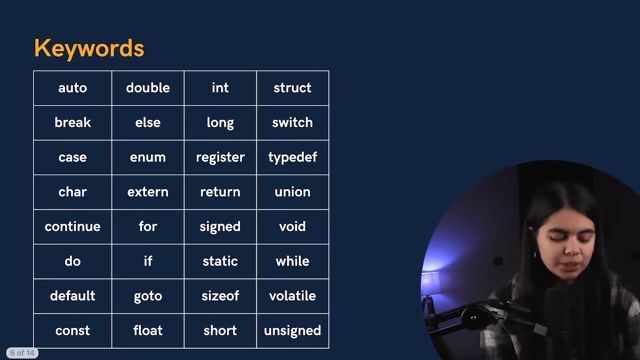
[22:10](https://youtu.be/irqbmMNs2Bo?t=1330)  
Types of data we are storing and how much size it consumes.  
**Note** there is no **boolean** or **String** type data types because it's too old and came before Java and python.

**Int :** Stores integer type data  
**float :** Stores decimal type data  
**char :** Stores single character type data enclosed within ' ' i.e. two single quotes

# Constants :

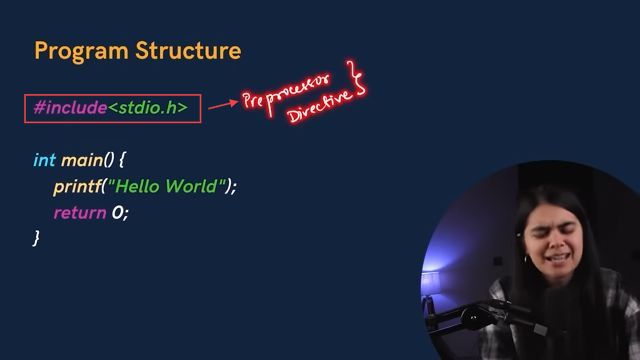
[25:55](https://youtu.be/irqbmMNs2Bo?t=1555)  


# Keywords :

[27:55](https://youtu.be/irqbmMNs2Bo?t=1675)  
  
32 Keywords are :  
[28:14](https://youtu.be/irqbmMNs2Bo?t=1694)  
**Note :** Return is also a keyword, fixed terms that compiler knows has a specific function

# Program Structure :

[29:33](https://youtu.be/irqbmMNs2Bo?t=1773)

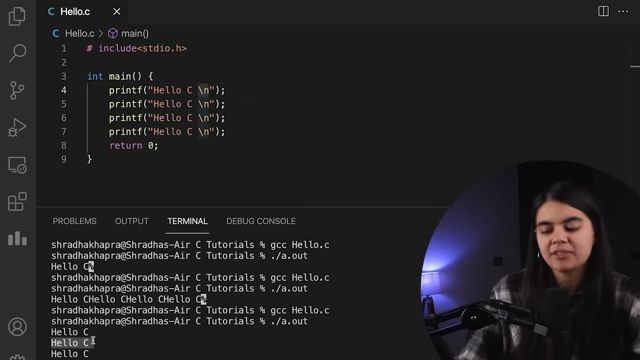
* Execution of program starts with the main() function and is **executed line by line**
* We use semi colon (;) at end of each line as it acts as point of termination of that code line.
* This structure is case sensitive i.e. printf is not the same as Printf.
* In the end we write a return 0

﻿

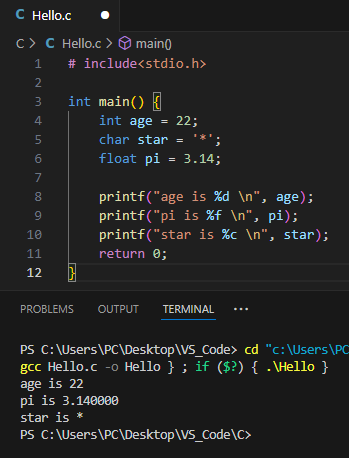
# Comments :

[31:53](https://youtu.be/irqbmMNs2Bo?t=1913)  
Somethings that is not a part of code but extra instructions. Some codes also support multiple languages.  
[33:05](https://youtu.be/irqbmMNs2Bo?t=1985)  


# Output :

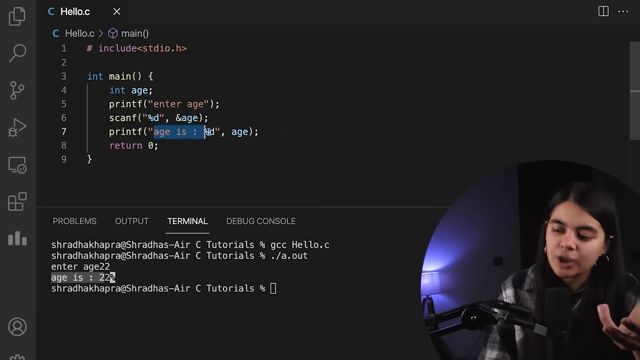
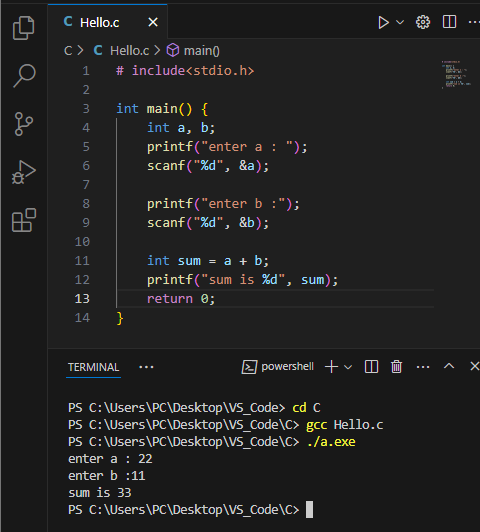
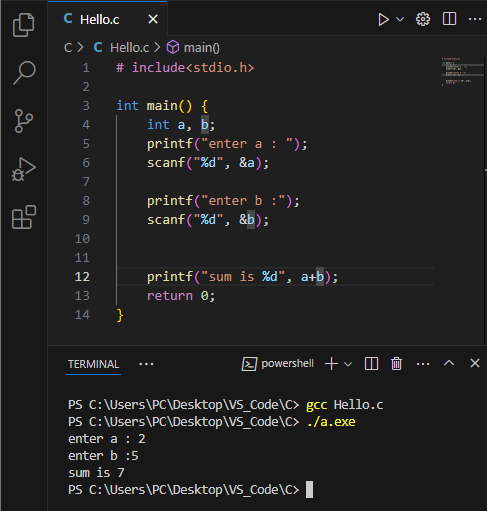
[33:59](https://youtu.be/irqbmMNs2Bo?t=2039)  
Anything written in a " " i.e. double quote is taken as String and printed with the printf function.  
[35:18](https://youtu.be/irqbmMNs2Bo?t=2118)  
But **we want in keep printing in next line** ? Solution **\n**  
[36:05](https://youtu.be/irqbmMNs2Bo?t=2165)  
**We want to print value contained in a variable ?**

# Cases to print in C :

[36:54](https://youtu.be/irqbmMNs2Bo?t=2214)  
\n to print each in new line

# Input :

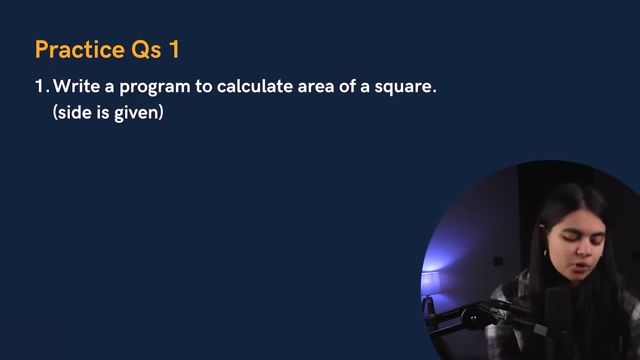
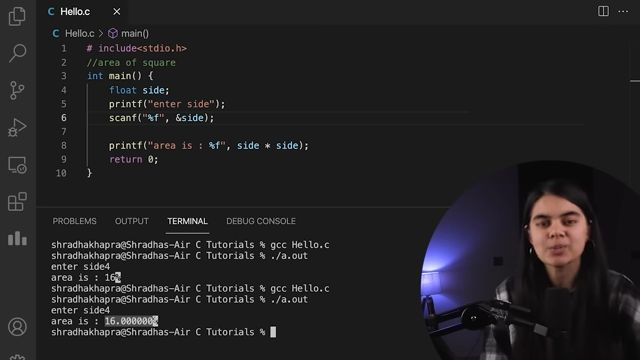
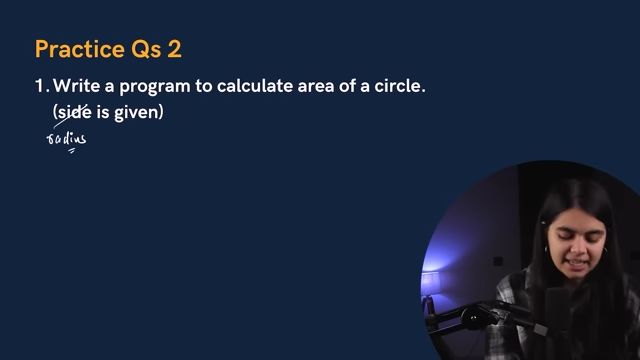
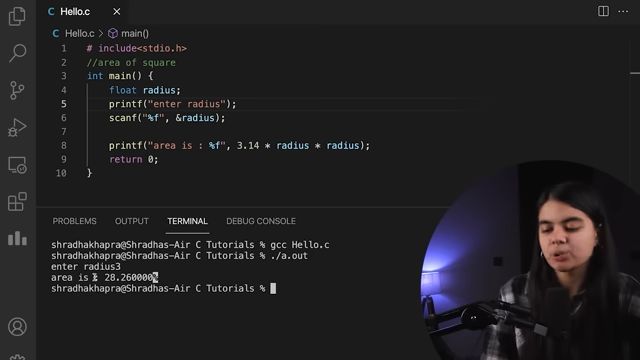
[40:10](https://youtu.be/irqbmMNs2Bo?t=2410)  
first the type of input to be taken is specified i.e. a **Format Specifier** inside a double quote. In this case it is %d i.e. integer value

**&** indicates the address of the variable written adjacent to it to store the value taken there. In this case it is **&age**  
[42:09](https://youtu.be/irqbmMNs2Bo?t=2529)  
  
**Q : Write a code to take two integer inputs, sum them and print the results !!**  
Even without a third variable, we can do such operations :  
  
The above total process is called**\_ \_**

# Compilation :

[45:29](https://youtu.be/irqbmMNs2Bo?t=2729)  
because computers only know 0s and 1s i.e. Binary - Low voltage (0) and High Voltage (1)

# Practice Questions :

[47:57](https://youtu.be/irqbmMNs2Bo?t=2877)  
[49:03](https://youtu.be/irqbmMNs2Bo?t=2943)  
It isn't necessary that our side is an Integer, can be other data types  
[49:29](https://youtu.be/irqbmMNs2Bo?t=2969)  
  
[49:45](https://youtu.be/irqbmMNs2Bo?t=2985)  
[50:55](https://youtu.be/irqbmMNs2Bo?t=3055)  
  
XXXXXXXX CHAPTER 01 XXXXXXXX

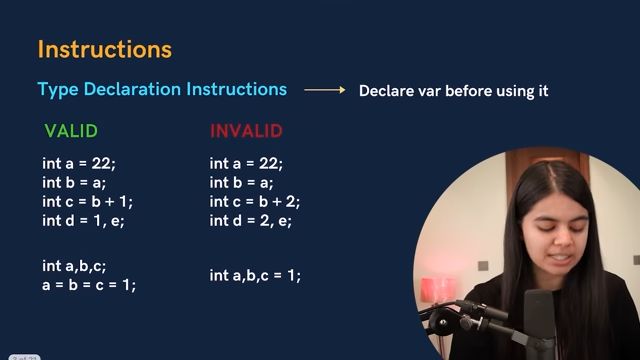
# **CHAPTER 02 :**

[52:01](https://youtu.be/irqbmMNs2Bo?t=3121)  

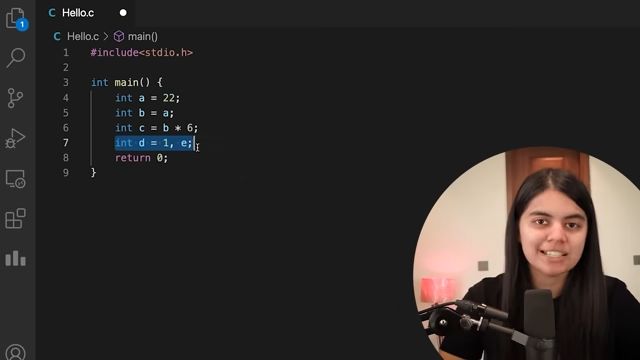

# i) Instructions :

[52:45](https://youtu.be/irqbmMNs2Bo?t=3165)  
Sequence of Instructions tells the program what to do.

## **-> TYPE DECLARATION INSTRUCTIONS :**

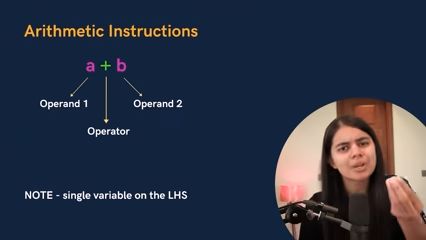
[53:53](https://youtu.be/irqbmMNs2Bo?t=3233)  
Variable declared before use as shown above

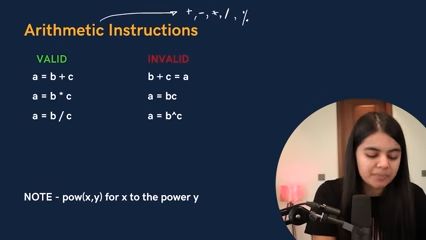
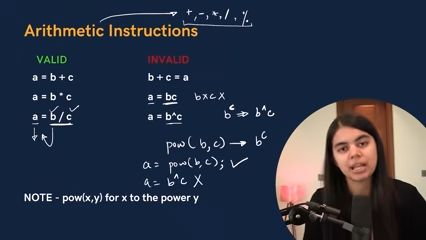
[55:35](https://youtu.be/irqbmMNs2Bo?t=3335)  
Variable is declared then assigned and then performed arithmetic operations upon is also valid as shown above

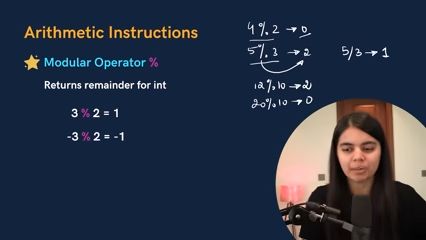
[56:36](https://youtu.be/irqbmMNs2Bo?t=3396)  
d is declared and initialized as 1 but e is just declared which is also valid statement.

[58:00](https://youtu.be/irqbmMNs2Bo?t=3480)  
We can also declare multiple variables at a time.  
But we can't do the below:  
[59:20](https://youtu.be/irqbmMNs2Bo?t=3560)  
You can't use and declare variables at a time. First declare then initialize.

## **-> ARITHMETIC INSTRUCTIONS :**

[59:47](https://youtu.be/irqbmMNs2Bo?t=3587)  
[01:01:22](https://youtu.be/irqbmMNs2Bo?t=3682)  
 All values of Arithmetic operations stored in a single variable.  
[01:02:50](https://youtu.be/irqbmMNs2Bo?t=3770)  
Line no. 8 in above not possible.

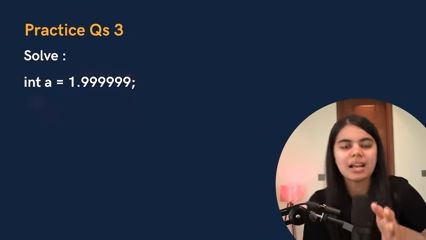
[01:03:13](https://youtu.be/irqbmMNs2Bo?t=3793)  
  
**Note :** To perform power function :  
[01:05:52](https://youtu.be/irqbmMNs2Bo?t=3952)  
[01:06:10](https://youtu.be/irqbmMNs2Bo?t=3970)  
You have to include another header file **#include<math.h> to use power function as pow(b,c) i.e. b^c. ^** is called **XOR** operator but it goes inside bitwise operator giving not an error but a different and wrong result.

-> -> Modulo Operator :  
[01:08:40](https://youtu.be/irqbmMNs2Bo?t=4120)  
  
**Note :** Modulo operator doesn't work in float values but just integer values

**Note :** If numerator is +ve we get remainder +ve, and if numerator is -ve we get remainder -ve  
[01:09:56](https://youtu.be/irqbmMNs2Bo?t=4196)  
**negative remainder**  
[01:10:14](https://youtu.be/irqbmMNs2Bo?t=4214)  
**positive remainder**

**-> ->Type Conversion :**  
[01:10:22](https://youtu.be/irqbmMNs2Bo?t=4222)  
Simple logic small with a big type will result in big type, so we need a big container type to store it !!

## Practice Questions :

[01:14:53](https://youtu.be/irqbmMNs2Bo?t=4493)  
Assigning such will remove the numbers past the decimal point. So **what should it be ? 1 or 2 ?**

[01:15:31](https://youtu.be/irqbmMNs2Bo?t=4531)  
There are 2 types of conversions : Implicit & Explicit

**Implicit**

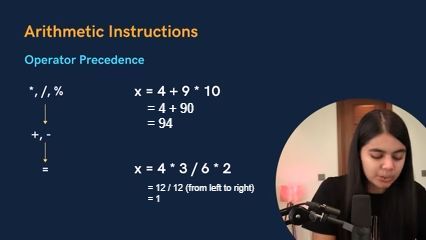
* Done by Compiler
* Small types can be converted to larger types

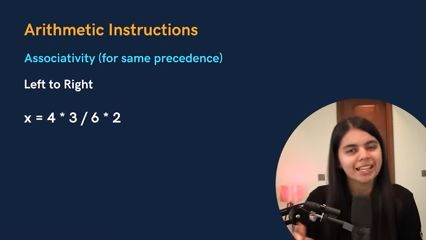
**Explicit**

* Done by User
* It is done by giving a datatype you want the value to be converted in inside a parenthesis as shown below :

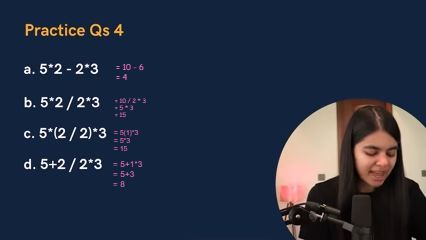
[01:16:25](https://youtu.be/irqbmMNs2Bo?t=4585)  
  
Now again if we try :  
[01:16:48](https://youtu.be/irqbmMNs2Bo?t=4608)  
**Note :** By default all decimal values are counted as double datatype.

**Compiler doesn't rounds off, it completely removes the numbers past decimal point.**

-> -> Operator Precedence :  
[01:17:38](https://youtu.be/irqbmMNs2Bo?t=4658)  
Priority as shown in above picture in **Descending order**

[01:20:24](https://youtu.be/irqbmMNs2Bo?t=4824)  
Questions containing operators of same precedence, we use **Associativity (left to right)**

# **Practice Questions :**

[01:22:28](https://youtu.be/irqbmMNs2Bo?t=4948)  


## -> Control Instructions :

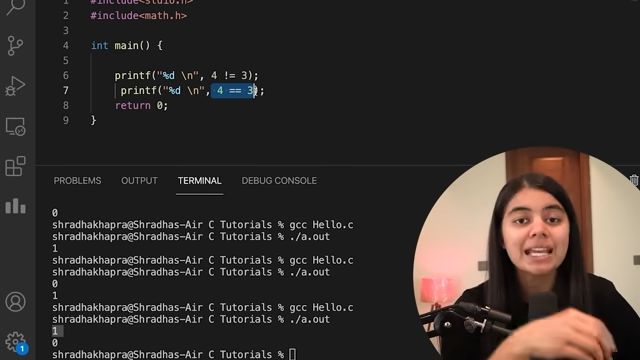
[01:27:06](https://youtu.be/irqbmMNs2Bo?t=5226)  
  
**a. Sequence Control :** Sequence wise one after another  
**b. Decision Control :** Includes if-else  
**c. Loop Control :** For loop, while loop  
**d. Case Control :** Includes cases

# ii) OPERTATORS :

[01:30:42](https://youtu.be/irqbmMNs2Bo?t=5442)  

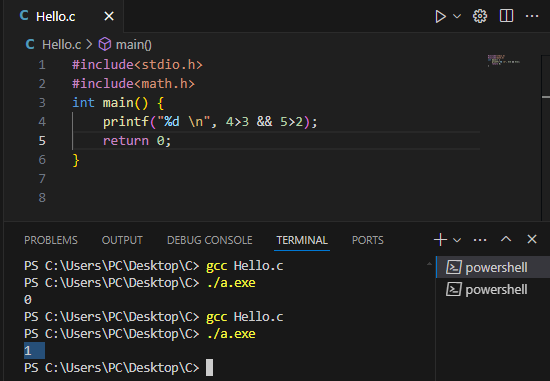
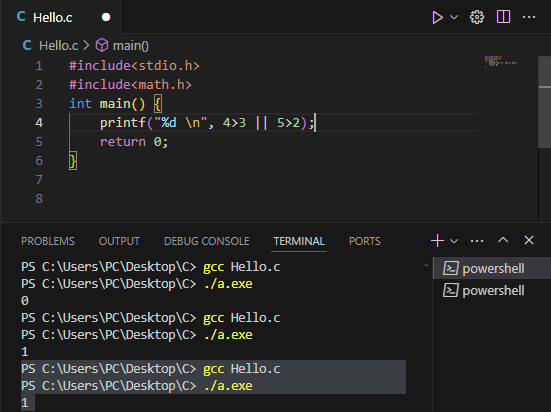
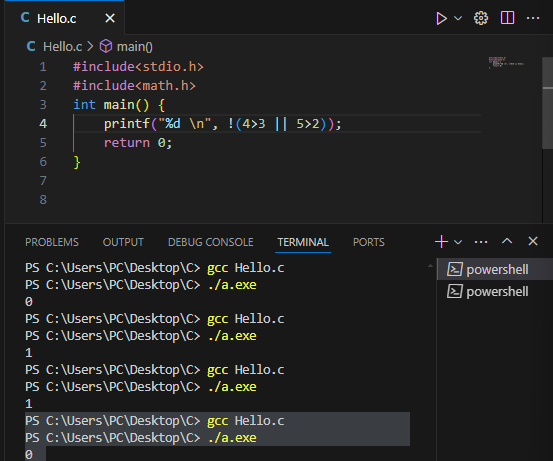
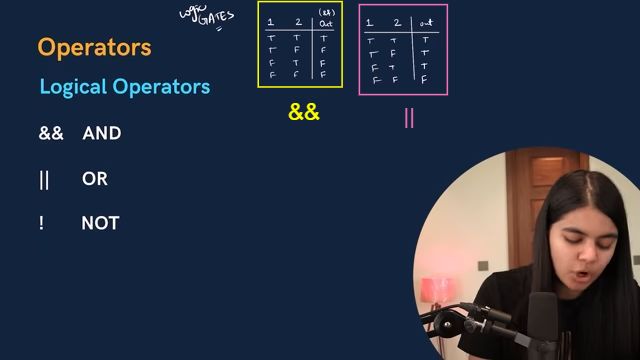

## b. Relational Operators :

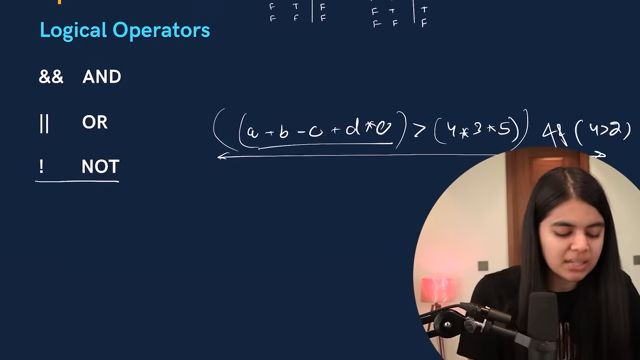
[01:31:14](https://youtu.be/irqbmMNs2Bo?t=5474)  
We are checking the relation between both side whether **True** or **False**  
C is a special language where instead of **True** or **False** there is **1** or **0** respectively.

[01:33:04](https://youtu.be/irqbmMNs2Bo?t=5584)  
**Any number is a True, but only 0 is False**  
[01:35:45](https://youtu.be/irqbmMNs2Bo?t=5745)  
! -> Not operator  
!= -> Not equal to  
[01:37:02](https://youtu.be/irqbmMNs2Bo?t=5822)  


## c. Logical Operators :

[01:37:18](https://youtu.be/irqbmMNs2Bo?t=5838)  
Required when we are comparing or analyzing two conditions for a result

**1) AND (&&)**  
  
**2) OR (||)**  
  
**3) NOT (!)**  
  
[01:42:10](https://youtu.be/irqbmMNs2Bo?t=6130)  
**Note :**  
& -> Logical AND  
&& -> Bitwise AND  
| -> Logical OR  
|| -> Bitwise OR

**Writing expressions and comparing them using Logical Operators :**  
[01:44:16](https://youtu.be/irqbmMNs2Bo?t=6256)  


## Operator Precedence

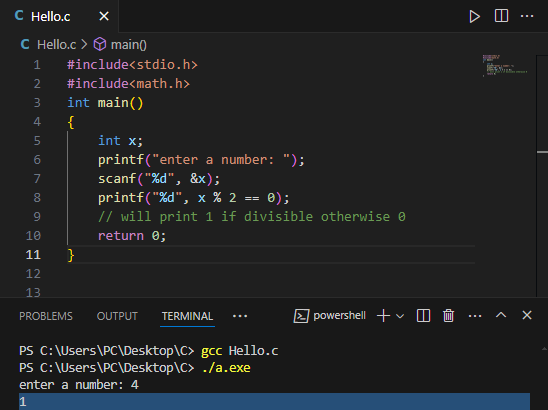
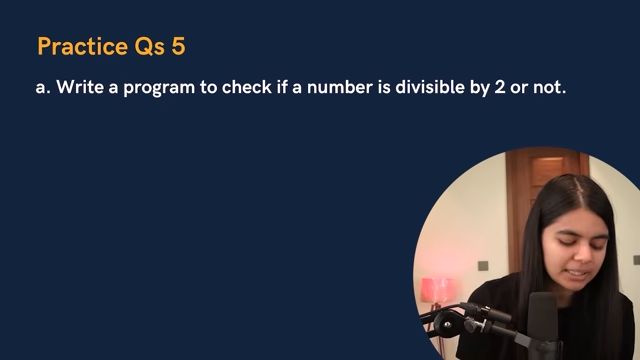
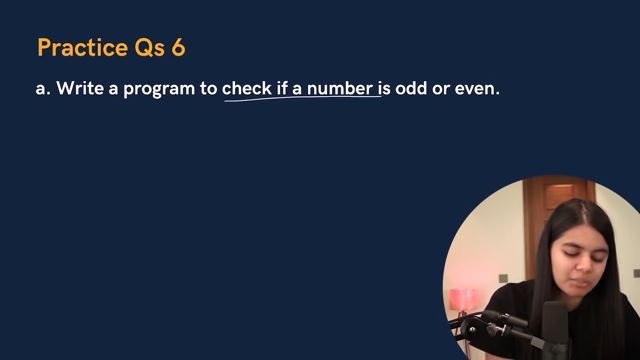
[01:44:37](https://youtu.be/irqbmMNs2Bo?t=6277)  


## Assignment Operators

[01:45:31](https://youtu.be/irqbmMNs2Bo?t=6331)  


#

# Practice Questions :

[01:48:19](https://youtu.be/irqbmMNs2Bo?t=6499)  
  
[01:51:43](https://youtu.be/irqbmMNs2Bo?t=6703)  
Same solution as before