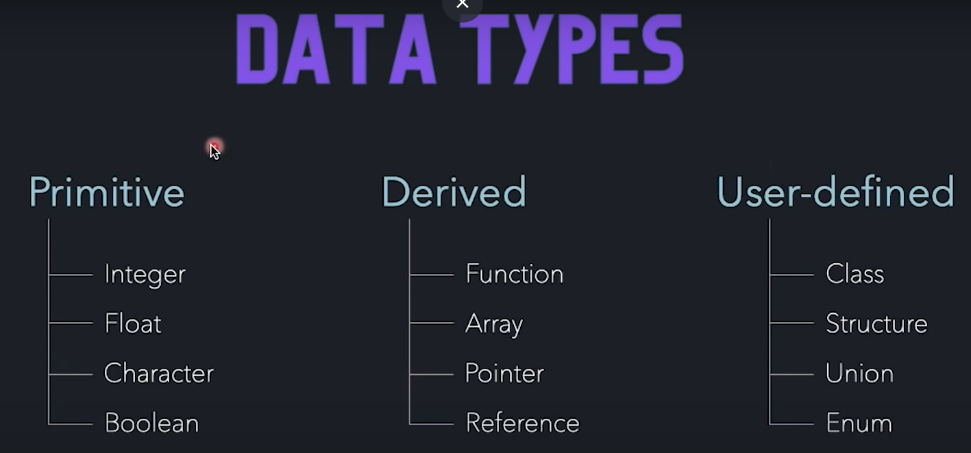
**C++ BASICS**



Memory in form of blocks.

1 byte = 8 bits

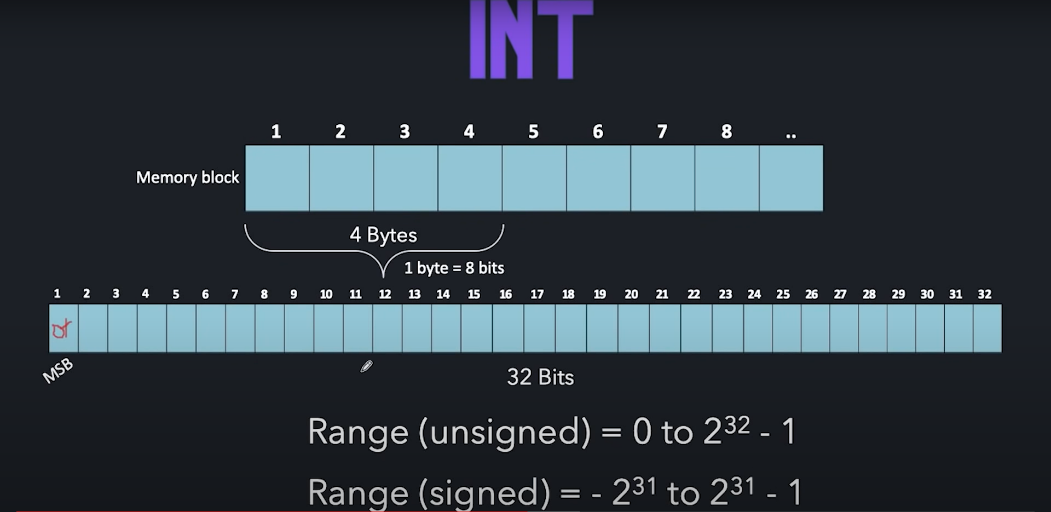
Int = 4 bytes = 32 bits

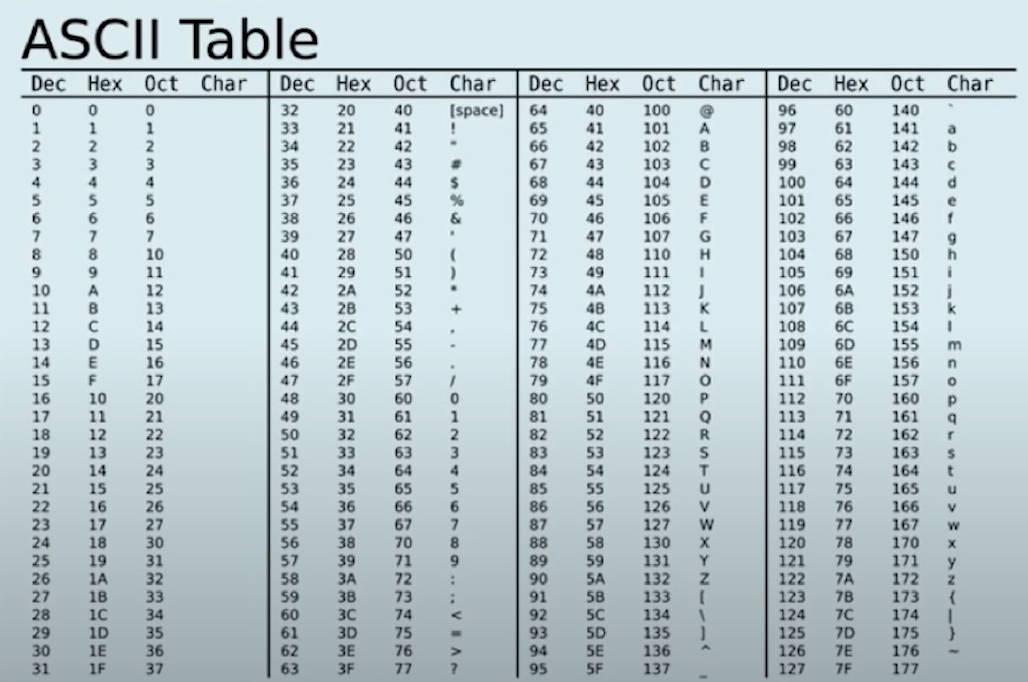
In case signed integers, 1st block [MSB bit] is allocated 0/1 depending if positive or negative.

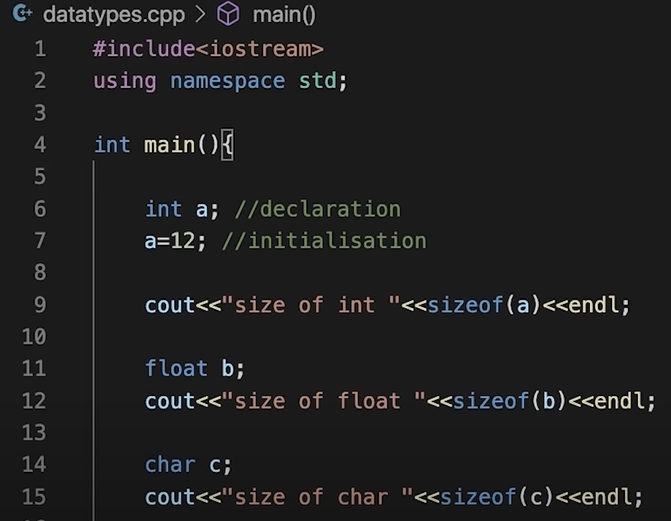
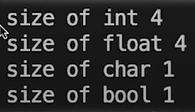
* 0 -> negative
* 1 -> positive

Char = 1 byte = 8 bits

signof() is an inbuilt function used to get the size if the variable.

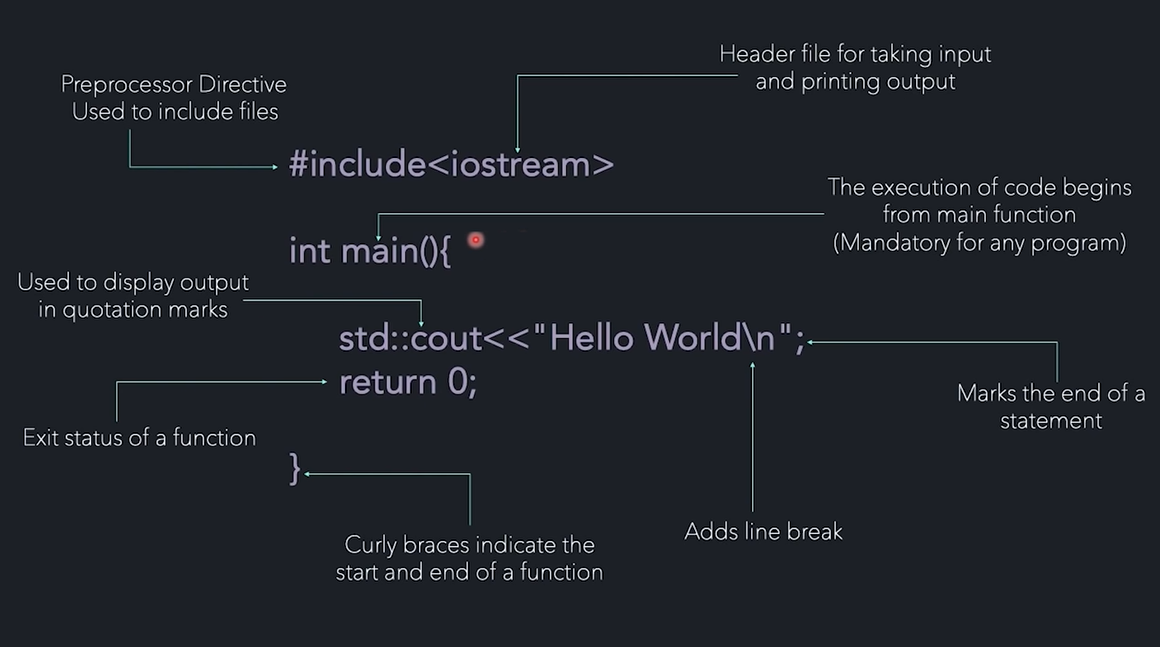


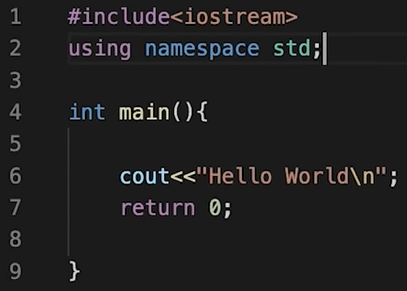
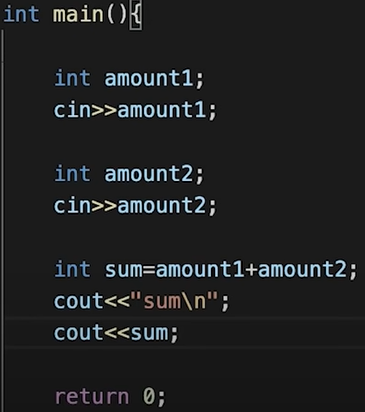




**Output:**

**Input/Output:**



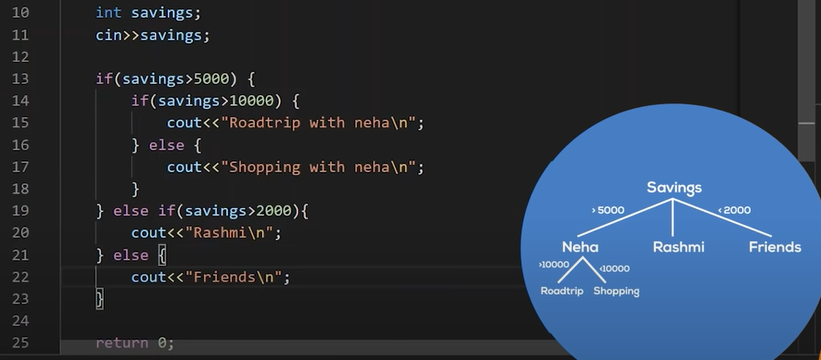


**Output:**

“sum” will print 🡪 sum

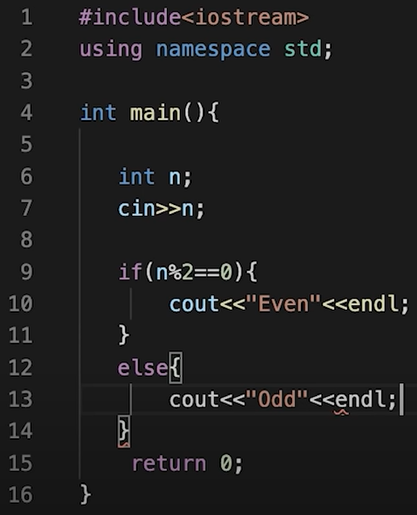
And other sum will print the addition output i.e., total amount after taking both inputs and performing operation

**If/else statements:**



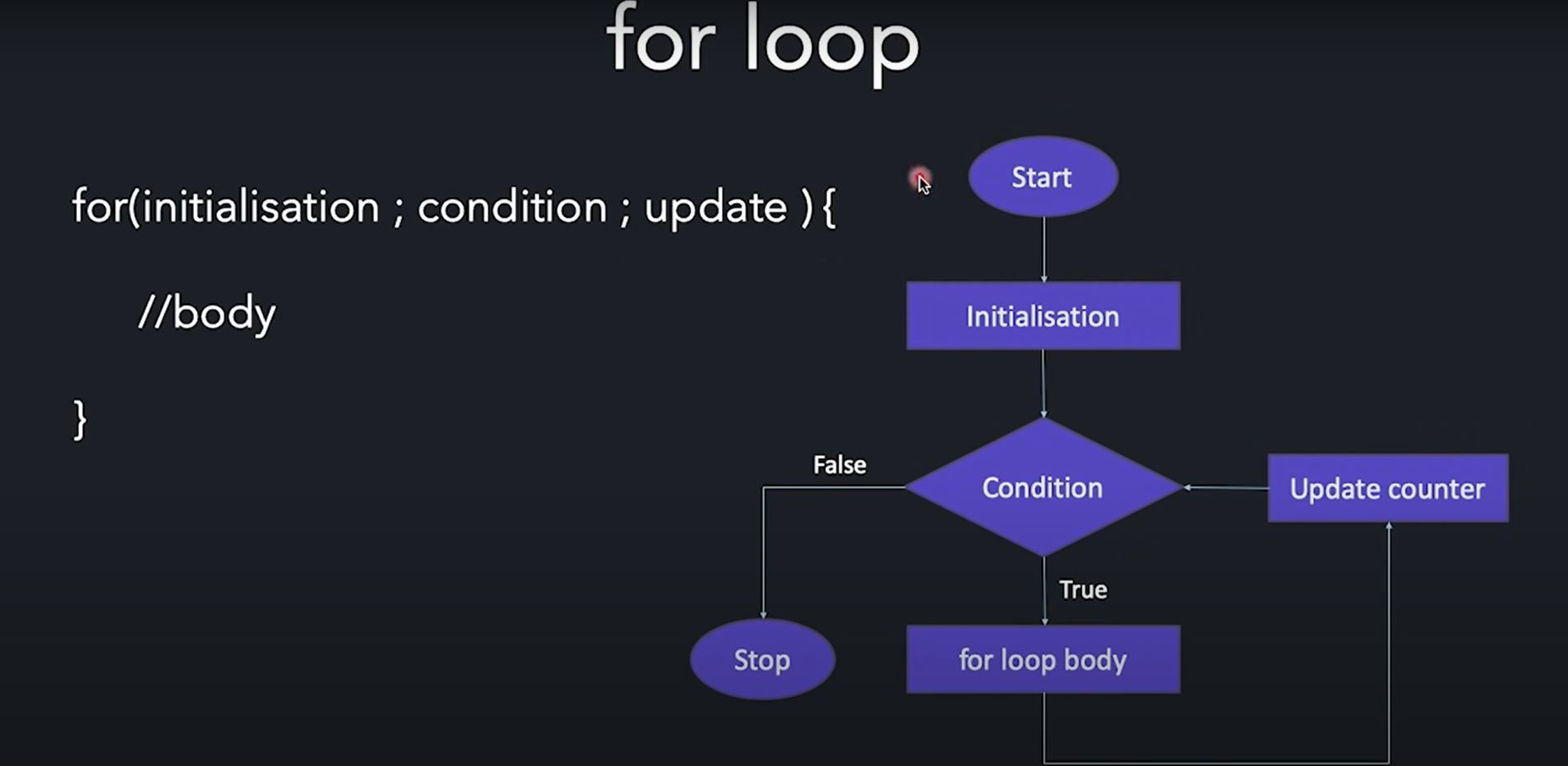
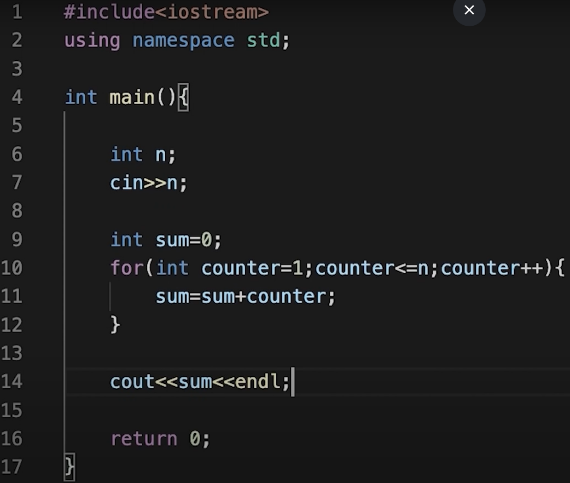
Above example is of nested if/else in which we put an if/else condition inside the if/else condition. Hence the name “nested”.

**Q.** To find largest of three numbers a, b, c: **Q.** To check number is even or odd:

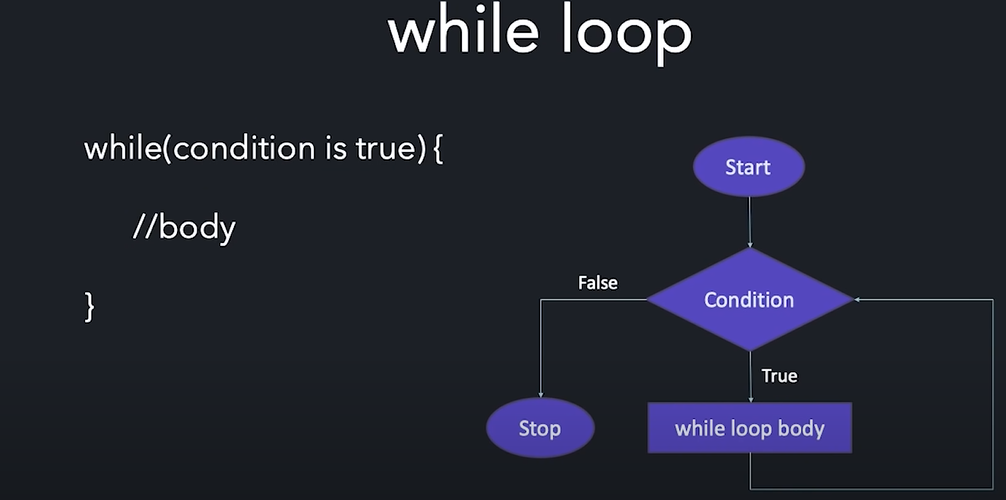
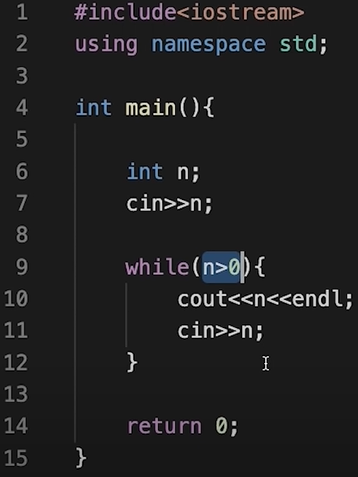


**Loops:**

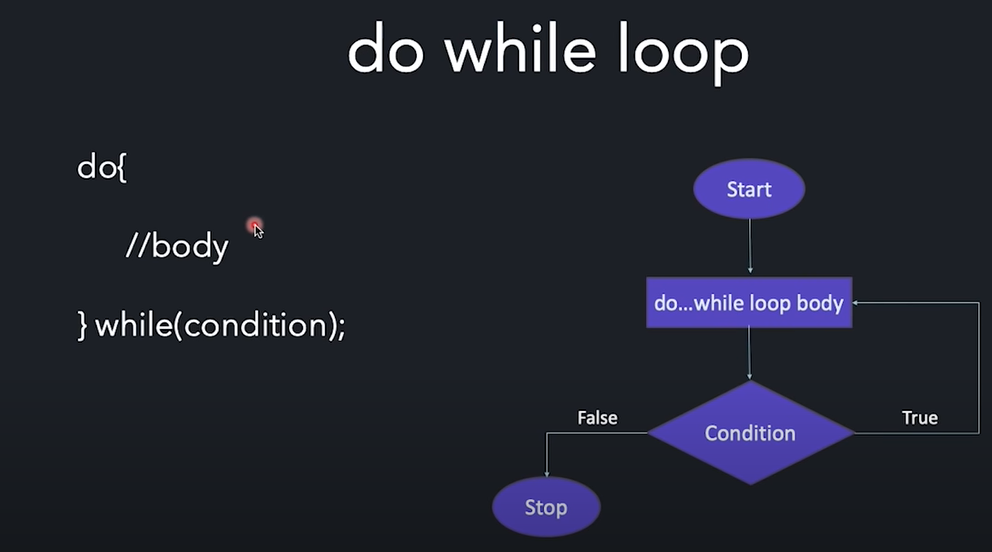
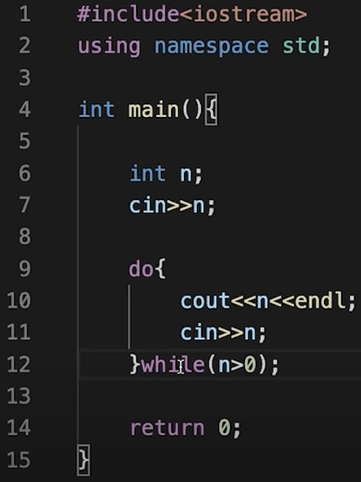
1. **For loop** 🡪



1. **While loop** 🡪



1. **Do-while loop** 🡪



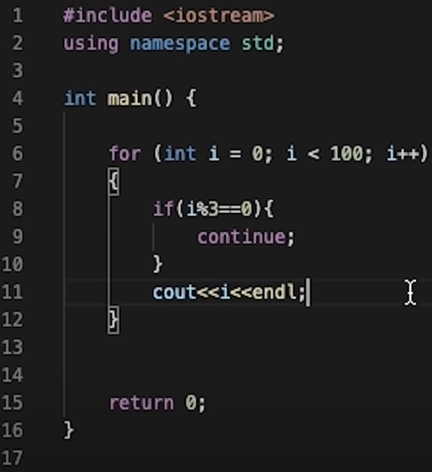
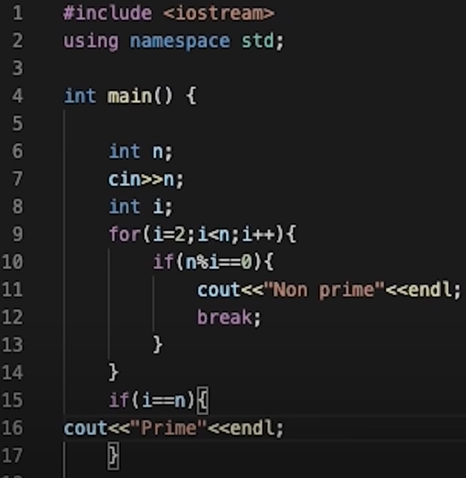
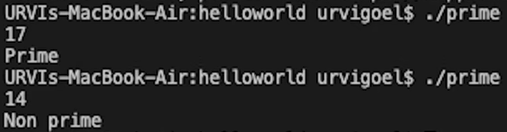
In case of while loop, the statement will run only if the condition holds true.

Whereas, in case do-while, even if the condition is false, the loop will execute once.

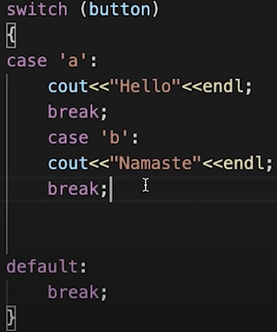
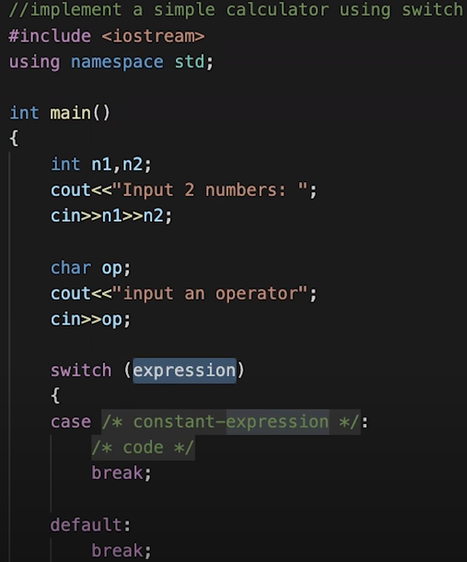
**Break/Continue:**



**Q**. To print numbers 1 to 100 exclude multiples of 3: **Q**. Check if number is prime or not:



**Switch-Case:**



The variable in switch should have a constant value.

The break statement is optional. It terminates the switch statement and moves control to the next line after switch.

Every case value should be unique.

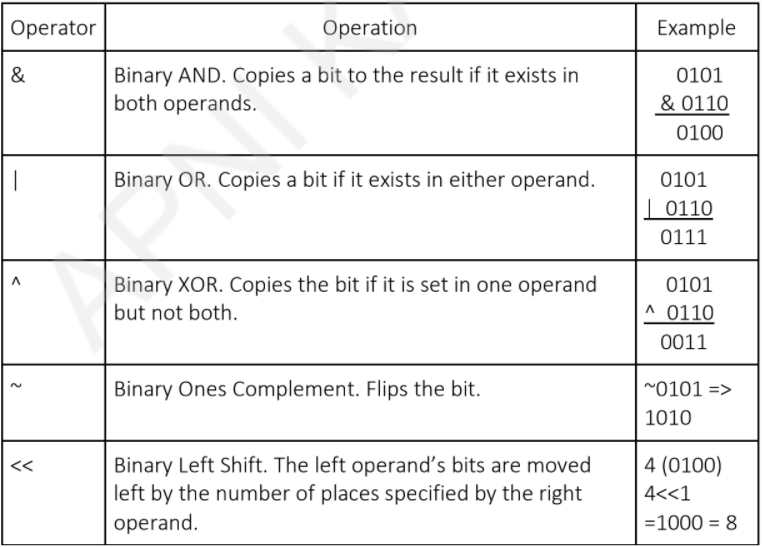
Default case is optional, but it is important as it is executed when no case value could be matched.

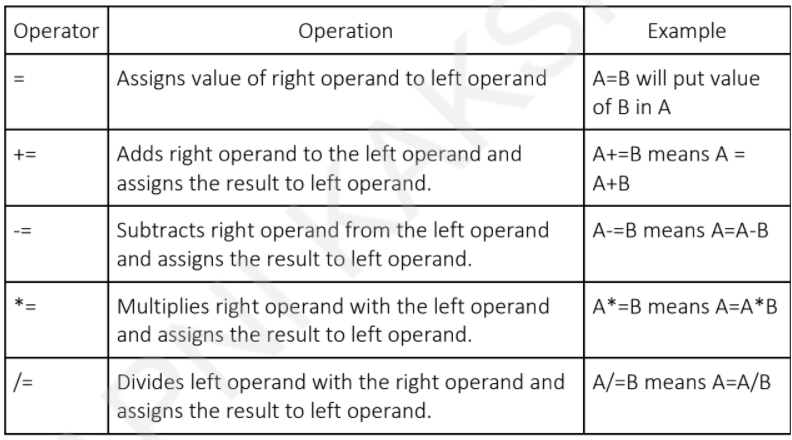
**Operators:**

|  |  |  |
| --- | --- | --- |
| Operator | Operation | Example |
| + | Adds two operands | A+B = 15 |
| - | Subtracts right operand from left operand | B-A = 5 |
| \* | Multiplies two operands | A\*B = 50 |
| / | Divides left operand by right operand | B/A = 2 |
| % | Finds the remainder after integer division | B%A = 0 |
| ++ | Incrementer | A++ = 6 |
| -- | Decrementer | A-- = 4 |

|  |  |  |
| --- | --- | --- |
| Operator | Operation | Example |
| == | Gives true if two operands are equal | A==B is not true |
| != | Gives true if two operands are not equal | A!=B is true |
| > | Gives true if left operand is more than right operand | A>B is not true |
| < | Gives true if left operand is less than right operand | A<B is true |
| >= | Gives true if left operand is more than right operand or equal to it | A>=B is not true |
| <= | Gives true if left operand is more than right operand or equal to it | A<=B is true |

|  |  |  |
| --- | --- | --- |
| Operator | Operation | Example |
| && | AND operator. Gives true if both operands are non-zero | (A && B) is false |
| || | OR operator. Gives true if at least one of the two operands are non-zero. | (A || B) is true |
| ! | NOT operator. Reverse the logical state of operand | !A is true |





|  |  |  |
| --- | --- | --- |
| Operator | Operation | Example |
| sizeof() | Returns the size of variable | If a is integer  then sizeof(a) will return 4 |

|  |  |  |
| --- | --- | --- |
| Condition?  X:Y | Conditional operator. If condition is true, then returns value of X or else value of Y | A+=B means A = A+B |
| Cast | Casting operators convert one data type to another. | int(4.350) would return 4. |
| Comma (,) | Comma operator causes a sequence of operations to be performed. The value of the entire comma expression is the value of the last expression of the comma-separated list. |  |

|  |  |  |
| --- | --- | --- |
| Category | Operator | Associativity |
| Postfix | () [] -> . ++ - - | Left to right |
| Unary | (type)\* & sizeof | Right to left |
| Multiplicative | \* / % | Left to right |
| Additive | + - | Left to right |
| Shift | << >> | Left to right |
| Relational | < <= > >= | Left to right |
| Equality | == != | Left to right |
| Bitwise AND | & | Left to right |
| Bitwise XOR | ^ | Left to right |
| Bitwise OR | | | Left to right |
| Logical AND | && | Left to right |
| Logical OR | || | Left to right |

|  |  |  |
| --- | --- | --- |
| Conditional | ?: | Right to left |
| Assignment | = += -= \*= /= %=>>= <<= &= ^= |= | Right to left |
| Comma | , | Left to right |

**Patterns:**

