Lecture 1: Intro to Programming & Flowcharts

03 January 2022 15:26

Topics covered:

→ Course Flow

→ Flow charts

→ Pseudocode

Q. How to solve a programming problem?

→ Given the problem,

① Understand the problem. → Add 2 numbers

② Check the given values. → 2 variables, Data Types?

③ Figure out an approach → a+b = my answer.

- This comes from practice and past coding experience.

④ Code! → `int ans = a+b; cout << ans << endl;`

WARNING ⚠

Agar code samajh na aaye to

Aage sab kuch cover hoga.

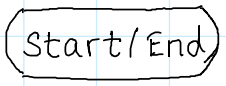
Given some problem [P], say you 'think' of some solution, ki aise aise karenge, etc. Now write down this crude solution on paper, not necessarily in correct syntax (code ki bhasha). Now your idea is on paper. Convert this rough work, also called 'pseudocode' into a program in a programming language of your choice, say C++.

Pseudocode : A very simple and high-level (upar-upar ka) form of computer language that is used in program design.

A Flowchart is a diagramatic representation of an approach. This draws out all the steps of your approach in order.

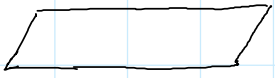
Components :

① Terminator : Specifies the start and end of a program.



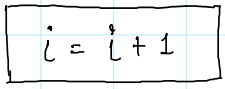
→ Terminator

② Parallelogram : For taking input or showing output.

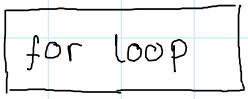


Input / Output

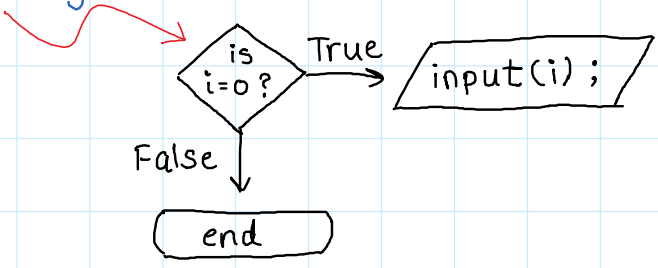
③ Process : Operations and processes ke liye.



or



④ Decision Making : (Diamond Shape)

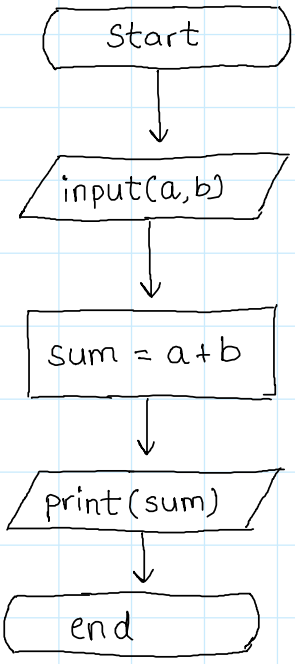


⑤ Circle : Connectors (To be covered when we discuss functions / methods)

⑥ Arrows : Code ka pravaah dikhane ke liye.
(Upar flowchart dekho ↑)

EXAMPLE :

Flow chart for adding 2 numbers.



Pseudocode for adding 2 numbers:

→ input 2 numbers a and b.
→ Let sum = a+b
→ Print out sum

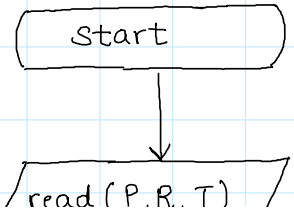
→ Read a and read b.
→ Sum variable is a+b
→ sum chaapo

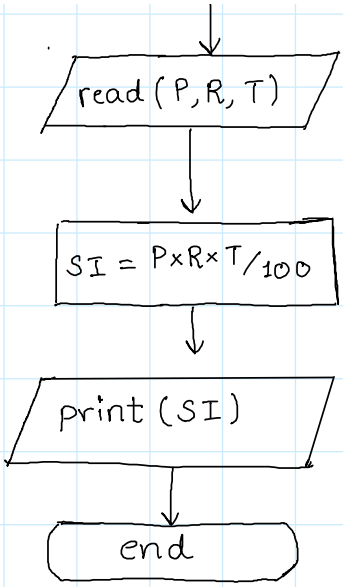
Both pseudocodes are OK. No pseudocode is wrong as long as the logic is same / similar.

EXAMPLE: Calculate simple interest.

$$SI = \frac{P \times R \times T}{100}$$

(P - Principal Amount
R - Rate of Interest
T - Time)





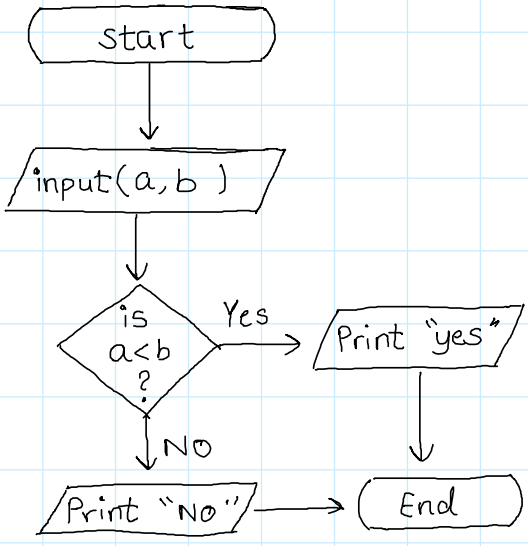
Pseudocode

→ read P, R and T

→ Make SI = P×R×T/100

→ Print SI

EXAMPLE: Determine if a < b



PSEUDOCODE:

→ Read a and b.

→ if a<b
then print Yes
else
print No

NEW CONCEPT : % (modulo) operator

Gives the remainder after division a/b.

∴ a % b = Remainder of a/b.

Eg : 5 % 2 = 1

6 % 4 = 2

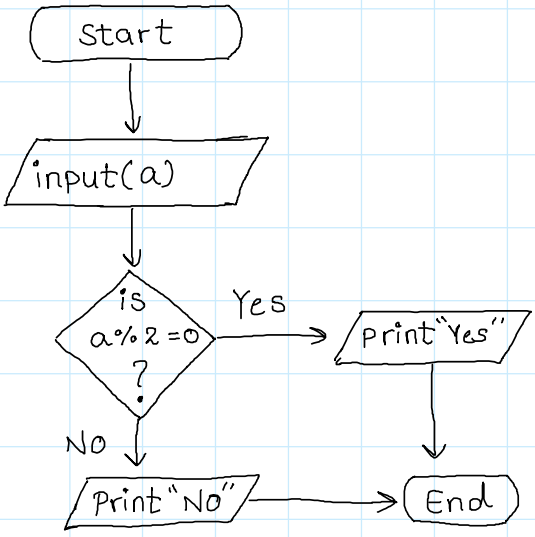
8 % 4 = 0

* 4 % 9 = 4 when a > b, a % b = a.

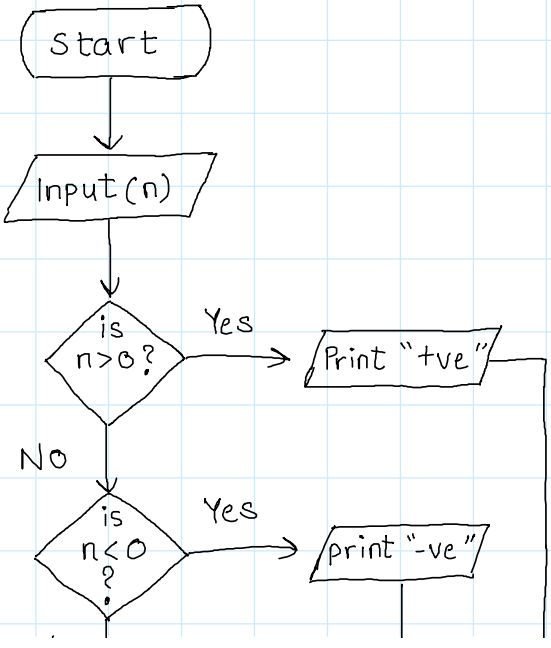
∴ If n % 2 = 0, n is even

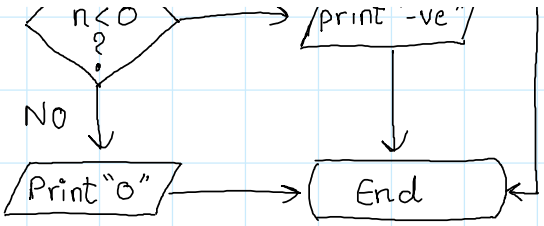
else n is odd.

EXAMPLE : Check if n is even or odd.



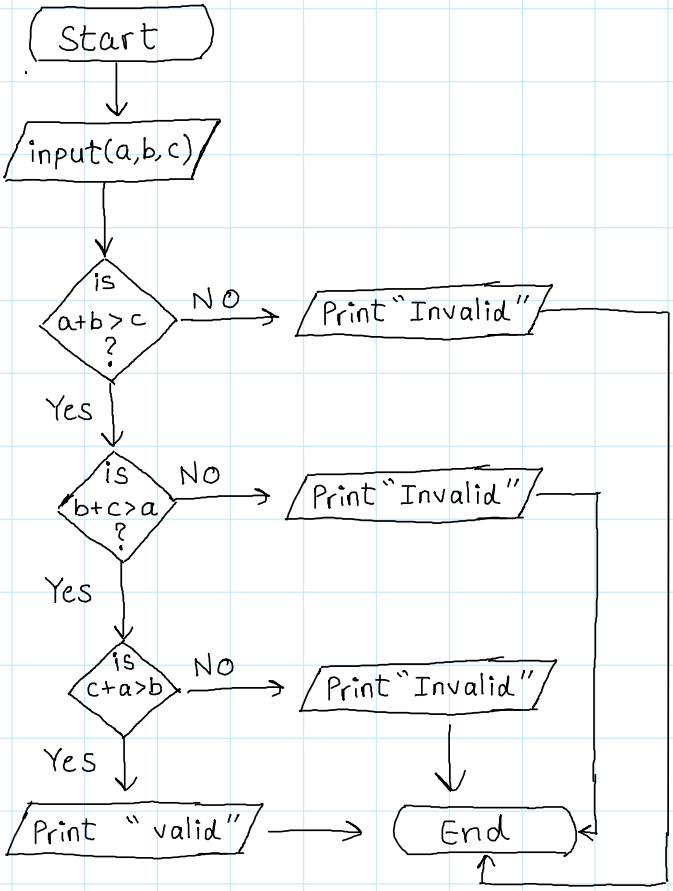
EXAMPLE : Is n positive, negative or zero.





Homework : Check if a given triangle is valid.

Hint - $a+b>c$, $b+c>a$ & $c+a>b$



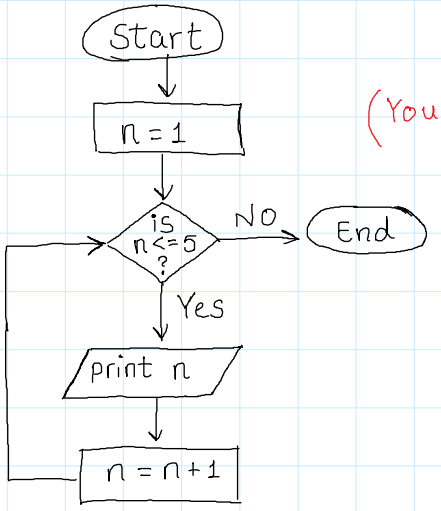
Loops

03 January 2022 17:39

Let Variable n = 1.

Now make n go from 1 to 5.

Soln.

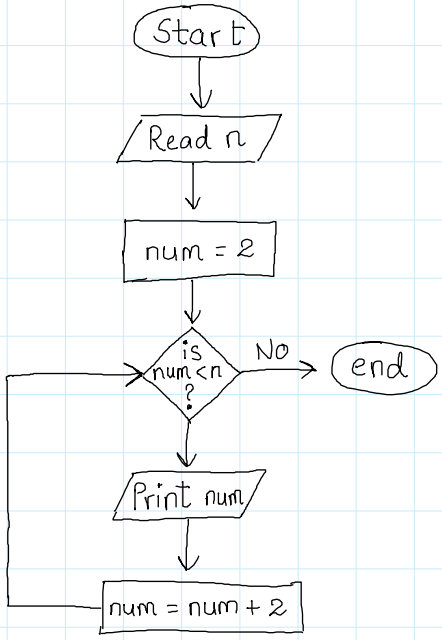


(You can also give input n = 1)

This is where we use loops to continuously perform some action while updating some value.

EXAMPLE: Print even numbers b/w 1 and n. (exclusive)

Hint - Even numbers start from 2 and occur alternatively.



Pseudocode

→ Input(n)

→ Let num = 2

→ while num < n,
print num
num = num+2

→ End

EXAMPLE: Print all odd numbers from 1 to n (inclusive)
(Homework hai)

Pseudocode

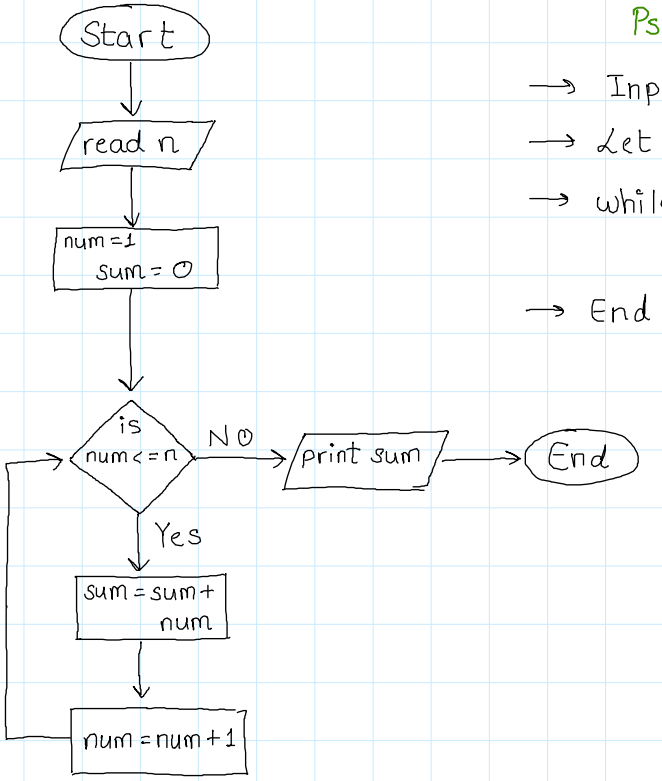
→ Input (n)

→ Let a = 1

→ while a < 1, print(a)
a = a + 2.

→ End

EXAMPLE: Find sum from 1 to n.



Pseudocode

→ Input (n)

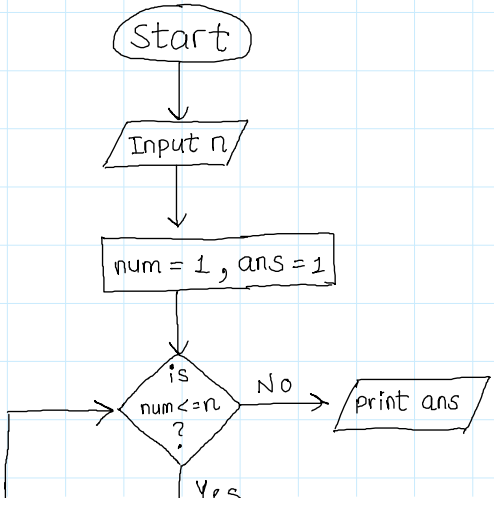
→ Let num = 1 and sum = 0

→ while num <= n,
sum = sum + num
num = num + 1

→ End

Homework: Find n!

Hint - $n! = n \times (n-1) \times (n-2) \times \ldots \times 3 \times 2 \times 1$



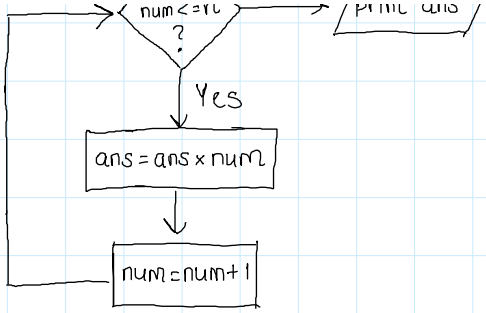
Pseudocode

→ Input (n)

→ Let num = 1 and ans = 1

→ while num <= n,
ans = ans × num
num = num + 1

→ End

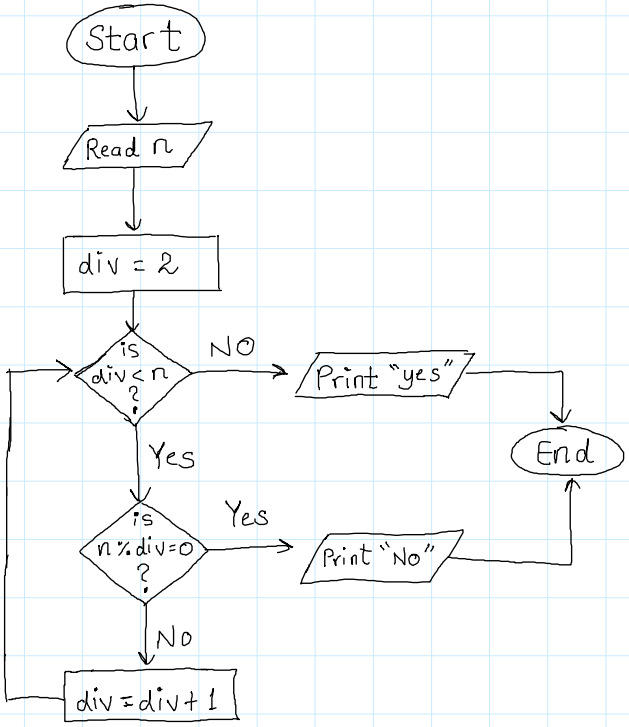


Example: Check if n is prime.

Input: 5 Input: 9

Output: Yes Output: NO

Prime n : n % (any number from 2 to n-1) $\neq 0$ → Written as != in C++



Pseudocode

→ Input (n)

→ Let div = 2

→ while div < 2,

if n % div = 0

Print "No"

exit

else

div = div + 1

→ Print "Yes"

Programming Language

03 January 2022 18:33

What is a Programming Language ?

→ Jaise khaana khaane ke liye mummy/papa ko bolna padhta hai, vaise hi we must instruct our computer to perform some task for us.

→ More formally, a programming language is a way to communicate with a computer. It is a formal language which consists of sets of strings that produce various kinds of machine output.

Eg: C, C++, Java, Python, R, GO, etc.

```
#include<iostream>

using namespace std;

// main function -
// where the execution of program begins
int main()
{
    // prints hello world
    cout<<"Hello World";

    return 0;
}
```

Output:

```
Hello World
```

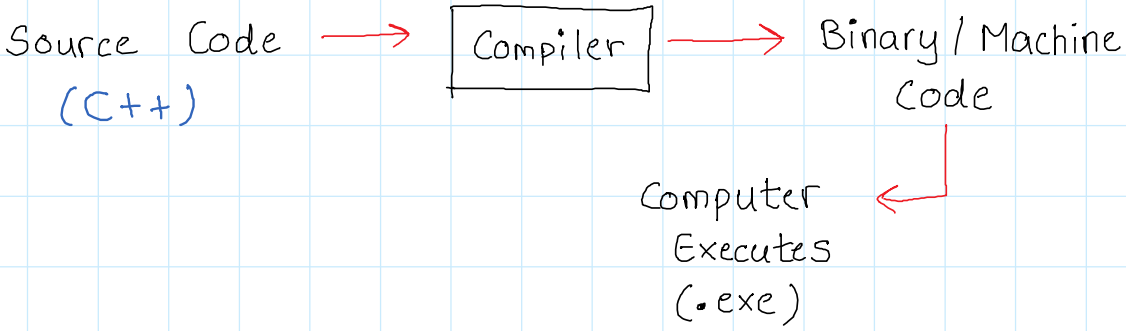
⇒ This code in C++ instructs our computer to print "Hello World" on our screen.

Source: https://www.geeksforgeeks.org/writing-first-c-program-hello-world-example/

→ Every language must be written following some rules called syntax of that language.

→ A computer essentially only understands binary codes of 0s and 1s. A compiler processes the statements of a programming language into Machine Code (Binary).

Source Code → [Compiler] → Binary / Machine

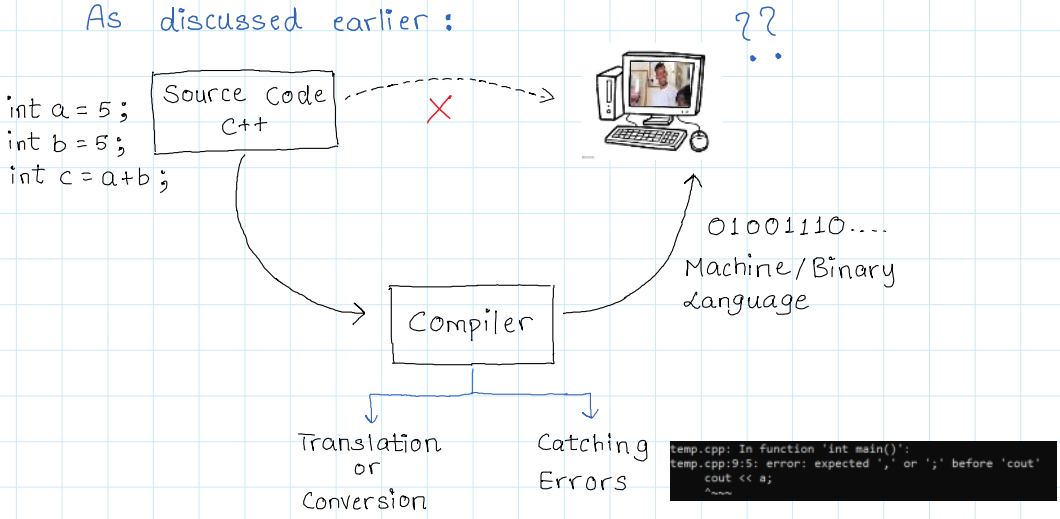


Read more about Compilers here: https://www.geeksforgeeks.org/introduction-to-compilers/

First Program in C++

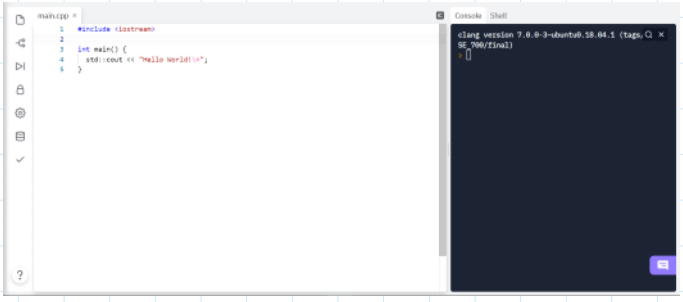
04 January 2022 20:53

As discussed earlier :



IDEs (Integrated Development Environment) :

An environment that helps you write, run and even debug (in some cases) code in a programming language.
Eg: VS CODE, Code Studio, Eclipse, NetBeans, etc.



First Program in C++ : (Namaste Duniya)

① Our program will find `int main(){ }` and start executing from there.

② `int main(){ }` These brackets show the 'scope' of the main function i.e. the code which belongs to / is defined within int main() function.

More about scope of a function

③ In C++, we use `cout` to print something.
Eg: `cout << "Namaste Duniya";`

④ This cout is already defined in a file (header file)

which must be included before using cout.

⇒ `#include <file_name>` is a preprocessor directive which runs before the program is compiled and includes the file to be used later in the source code.

A file called iostream has cout defined in it so:

`#include <iostream>`

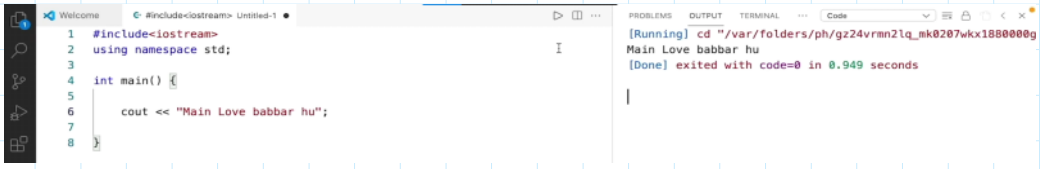
Hint: i/o means input/output.

⑤ Namespaces : Stack Overflow Question

`using namespace std;`

⑥ Using cout :

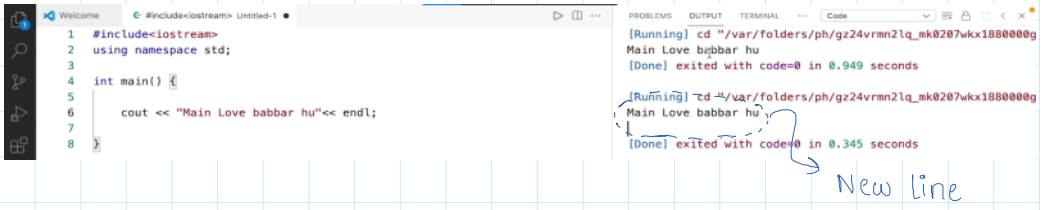
We use `<<` after `cout` to display something to Standard Output (your screen) within std namespace.



⑦ endl : Used to enter new line. Just like ENTER.

endl is like "\n" which is a new line escape sequence character used in various languages including C++.

cout << " Namaste Duniya " << endl ;



⑧ endl; ; is used to terminate statements.

DATA TYPES: Different types of data to be stored in memory. Eg- integer, float, character, double, etc.

Eg- int: Stores integers like -5, 0, 8, etc.
char: Single character like 'a', '+', '$', '7', etc.
float: Floating point values like -2.014, 1.0000, 6.7800

Different data types use different amounts of memory. Amount of memory used also depends on the architecture of your CPU.

Data Type	Meaning	Size (in Bytes)
int	Integer	2 or 4
float	Floating-point	4
double	Double Floating-point	8
char	Character	1
wchar_t	Wide Character	2
bool	Boolean	1
void	Empty	0

Source: Programiz

Character : A 1-byte (= 8 bits) data type that takes 1 character.

```
char ch = 'a';
```

Boolean : True / False. Take 1 bit and 1 : True
0 : False

```
bool isGood = 1;
bool isBad = false;
```

Float / Double : Float takes 4 / 8 bytes
Double takes 8 bytes.

```
float num1 = 1.2;
double num2 = 2.4;
```

Know more about C++ Data Types

Variable Naming / Nomenclature :

1. Can contain alphabets, numbers and underscores.
2. Cannot start with a number.
3. Cannot be keywords like int, cout, double, bool, etc.
4. Case sensitive
5. Cannot contain special symbols like %, $, !, #, etc.

WARNING : Don't use same variable name multiple times.

```
6   int a = 123;
7
8   cout<< a << endl;
9
10  char a = 'v';
11
12  cout<< a <<endl;
```

```
ning] cd "/var/folders/ph/gz24vrmn2lq_mk0207wkx1880000gn/T/"
CodeRunnerFile.cpp:10:10: error: redefinition of 'a' with a
char a = 'v';
         ^
CodeRunnerFile.cpp:6:9: note: previous definition is here
int a = 123;
    ^
```

Using different data types in code :

```cpp
#include<iostream>
using namespace std;

int main() {

    int a = 123;

    cout<< a << endl;

    char b = 'v';

    cout<< b <<endl;

    bool bl = true;
    cout<< bl << endl;

    float f = 1.2;
    cout<< f << endl;

    double d = 1.23;
    cout<< d << endl;

}
```

```
[Running] cd "/var/folders/ph/gz24vrmn2lq_mk0207wkx1880000g
123
v
1

[Done] exited with code=0 in 0.595 seconds

[Running] cd "/var/folders/ph/gz24vrmn2lq_mk0207wkx1880000g
123
v
1
1.2

[Done] exited with code=0 in 0.551 seconds

[Running] cd "/var/folders/ph/gz24vrmn2lq_mk0207wkx1880000g
123
v
1
1.2
1.23

[Done] exited with code=0 in 0.537 seconds
```

Check the size of different Data Types for your system using sizeof(variable_name);

```cpp
#include <iostream>
using namespace std;

int main() {
    int a = 4;
    double b = 1.90;
    char c = '+';
    int sizeInteger = sizeof(a), sizeDouble = sizeof(b), sizeChar = sizeof(c);
    cout << "Size of an integer is " << sizeInteger << " bytes" << endl;
    cout << "Size of a double is " << sizeDouble  << " bytes" << endl;
    cout << "Size of a char is " << sizeChar  << " bytes" << endl;
}
```

```
clang++-7 -pthread -std=c++17 -o main main.cpp
./main
Size of an integer is 4 bytes
Size of a double is 8 bytes
Size of a char is 1 bytes
```

HOW IS DATA STORED IN MEMORY ?

Eg: int a = 8; // int takes 4 bytes = 32 bits

In binary, 8 = 1000 (4 bits needed)

∴ int a → 00000000 00000000 00000000 00001000 } 32 bits

Eg: int b = 5;

b

5

4 bytes.

address = 100 (assume)

100, 101, 102, 103

4 bytes are consumed

Eg: char c = 'a';

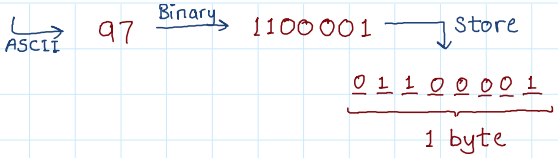
Characters are mapped to the standard ASCII values

'a' → 97	'A' → 65
'b' → 98	'B' → 66
'c' → 99	'C' → 67
⋮	⋮
'z' → 122	'Z' → 90

ASCII Table

(Homework)

char c = 'a';


ASCII
97
Binary
1100001
Store
0 1 1 0 0 0 1
1 byte

Conversion of one data type to another (if it is valid) is called Type Casting.

Example :

```
int a = 'a';
```

variable a will store the ASCII value of 'a' = 97.

Example :

```
char ch = 98;
```

98 will automatically get typecasted to its corresponding character.

This automatic typecasting is called implicit typecasting.

```
int a = 'a';
cout<< a << endl;

char ch = 98;
cout << ch <<endl;
```

→

```
[Running] cd "/var/folders/ph/gz24vrmn2lq_mk0207wkx1880000g
97
b
```

What if we type cast from int to char but the value is too large to be stored in char ?

Soln : A warning is thrown and the last byte from the original value is given to the character.

```
char ch1 = 123456;
cout<< ch1 <<endl;
```

→

```
[Running] cd "/var/folders/ph/gz24vrmn2lq_mk0207wkx1880000g
tempCodeRunnerFile.cpp:33:16: warning: implicit conversion
    char ch1 = 123456;
           ~~   ^~~~~~
1 warning generated.
```

How are negative numbers stored ?

Soln : The first bit tells us whether the number is positive or negative.

First Bit → 0 means Positive
→ 1 means Negative

Steps to store -5 in binary format :

1. Ignore the -ve sign. (5)
2. Write the binary representation of 5.
3. Take its 2's complement and store it.

Example : a = -5.

1. -5 → 5 (Ignore the -ve sign)
2. 5 → 0101 → 00.....0101 (Binary)
   (29 Zeros)
3. 2's complement = 1's Complement + 1.

```
        5  →  00.....0101
1's compl. →  11.....1010   (Flip the bits)
            +             1
2's Compl. →  11.....1011
```

2's Compl. → 11.....1011 (29 Ones) + 1

Displaying Negative Number :

① Take 2's complement of the stored number

Stored : 11.....1011

This shows -ve

11....1011

1's comp. 00....0100

2's comp. 00...0101 = 5

-5 print ho gaya !

Why 2's Complement ?

If we stored numbers as it is without using 2's complement, then

1	0	0	-----	0	0

and

0	0	0	-----	0	0

will be equal & thus waste space.

Store only Positive Integers

The default signed representation allows us to store both positive & negative values.
To store only positive integers, we use unsigned..

Eg: unsigned int a = 10;

What if we store a negative value in an unsigned number ?

Example : unsigned int a = -112;
cout << a << endl;

Output :
4294967184 ??

Explaination :
We tried to store -112.

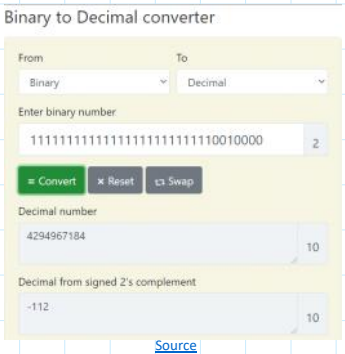
-112 = 2's Complement of 112.

112 = 00.....01110000.
25 Zeros.

1's Compl. = 11.....10001111
+ 1

2's Compl. = 11.....10010000

Unsigned int uses all 32 bits to store the value and the MSB (=1) will make the value.
An unsigned int does not use the 2's complement again to display the number.
Thus, 11....10010000. gets printed as it is in decimal.
25 ones



Source

Therefore,

```
unsigned int a = -112;

cout<< a << endl;
```

→

```
[Running] cd "/var/folders/ph/gz24vrmn2lq_mk0207wkx1880000g
4294967184
```

Operators

05 January 2022 19:05

Basic Arithmetic Operators:

+ , - , * , / , %

Caution ⚠

① int / int = int (Floor value of answer)

Examples: 5/2 = 1.

3/5 = 0

9/2 = 4

② int / float } float
float / int }

double / int } double
int / double }

cout << 5.0/2 << endl;

Output:

2.5

```
42  float a = 2.0/5;
43
44  cout << a <<endl;
45
46  cout<< 2.0/5 <<endl;
47
```

```
[Running] cd "/var/folders/ph/gz24vrmn2lq_mk0207wkx1880000g
0.4
0.4

[Done] exited with code=0 in 0.591 seconds
```

Relational Operators:

== , >= , <= , > , < , !=

① Is a = b ?

a==b → Yes → 1
No → 0

② Is a greater than or equal to b ?

a >= b → Yes → 1
No → 0

```
49
50  int a = 2;
51  int b = 3;
52
53  bool first = (a==b);
54  cout<< first <<endl;
55
56  bool second = (a>b);
57  cout<< second <<endl;
58
59  bool third = (a<b);
60  cout<< third <<endl;
61
62  bool fourth = (a>=b);
63  cout<< fourth <<endl;
64
65  bool fifth = (a<=b);
66  cout<< fifth <<endl;
67
68  bool sixth = (a!=b);
69  cout<< sixth <<endl;
```

```
[Running] cd "/var/folders/ph/gz24vrmn2lq_mk0207wkx1880000g
0

[Done] exited with code=0 in 0.851 seconds

[Running] cd "/var/folders/ph/gz24vrmn2lq_mk0207wkx1880000g
0
0
1
0
1
1

[Done] exited with code=0 in 1.124 seconds
```

Logical Operators:

&& , || , !

(AND) (OR) (NOT)

① Logical AND:

All conditions must be true for the output to be true.

Example: int a = 5, b = 10, c = 15;

```
cout <<((a>0) && (b!=0) && (c<=15));
```

Output :

1

② Logical OR :

At least 1 condition must be true for the output to be true.

Example :
```
int a=5, b=10, c=15;
cout <<((a>5) || (b<10) || (c>=15));
```

Output :

1

③ Logical NOT :

Inverts the logic. True ⇌ False

Non-zero ⇌ Zero

Example :
```
int a=10, b=0;
cout << (!a) << endl;
cout << (!b) << endl;
```

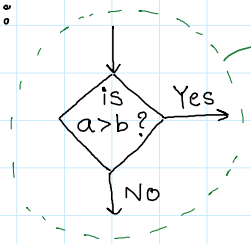
Output :

0

1

Point out what was not covered in Typecasting and try to code yourself.

Recall :



→ In code is called a conditional statement.

Example : I/P : a, b

O/P : $\begin{cases} a & \text{if } a > b \\ b & \text{otherwise} \end{cases}$

Conditionals like these are solved using if statements.

```
if (~~~~) {
    ~~~~
    ~~~~
    ~~~~
}
```

If the condition within ( ) is true, then execute the entire code block within { }.

Thus,

```
if ( a > b ) {
    cout << a << endl;
}
if ( a <= b ) {
    cout << b << endl;
}
```

In the above example, instead of checking again for b > a in the second if block, we know that if the first if condition does not get fulfilled, then the second block must be executed no matter what. This can be achieved using an if - else block.

If this is fulfilled, else block won't execute

```
if ( a > b ) {
    cout << a << endl;
}
else {
    cout << b << endl;
}
```

If if block doesn't execute, then this else block will execute.

NEW CONCEPT :

`cin >> n;` → Waits for user to give input and assigns it to n at its address.

Example- `int a;`
`cin >> a;`

① variable a will get initialized with a random, 'garbage' value.

② Program waits for user to give input (an integer) to a and stores a with that value.

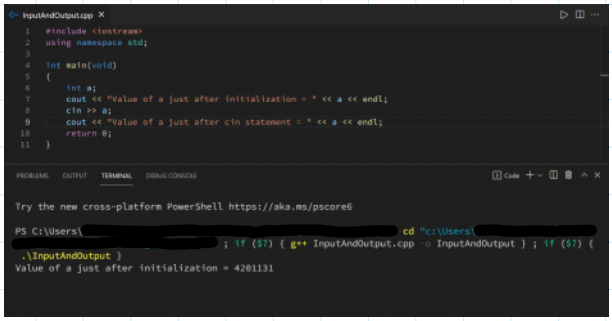


Fig ①

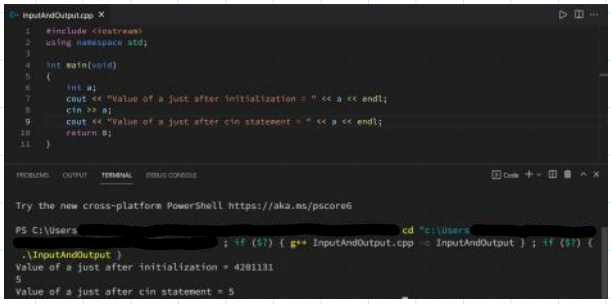


Fig ②

Back to if-else :

Here, the if block doesn't execute and the program exits without printing anything

①

InputAndOutput.cpp

```cpp
#include <iostream>
using namespace std;

int main(void)
{
    int a;
    cin >> a;

    if(a > 0) {
        cout << "a is positive" << endl;
    }
    return 0;
}
```

```
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE
{ .\InputAndOutput }
5
a is positive
```

②

InputAndOutput.cpp

```cpp
#include <iostream>
using namespace std;

int main(void)
{
    int a;
    cin >> a;

    if(a > 0) {
        cout << "a is positive" << endl;
    }
    return 0;
}
```

```
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE
-1
PS [redacted]
```

③

InputAndOutput.cpp

```cpp
#include <iostream>
using namespace std;

int main(void)
{
    int a;
    cin >> a;
```

④

InputAndOutput.cpp

```cpp
#include <iostream>
using namespace std;

int main(void)
{
    int a;
    cin >> a;
```

```cpp
int main(void)
{
    int a;
    cin >> a;

    if(a > 0) {
        cout << "a is positive" << endl;
    }
    else {
        cout << "a is not positive" << endl;
    }
    return 0;
}
```

```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE
 .\InputAndOutput }
5
a is positive
```

```cpp
int main(void)
{
    int a;
    cin >> a;

    if(a > 0) {
        cout << "a is positive" << endl;
    }
    else {
        cout << "a is not positive" << endl;
    }
    return 0;
}
```

```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE
 .\InputAndOutput }
-2
a is not positive
```

Program to compare two numbers :

InputAndOutput.cpp

```cpp
#include <iostream>
using namespace std;

int main()
{
    int a, b;
    cout << "Enter the value of a: ";
    cin >> a;
    cout << "Enter the value of b: ";
    cin >> b;

    if(a > b) {
        cout << "a is greater than b" << endl;
    }
    if (b > a) {
        cout << "b is greater than a" << endl;
    }
    return 0;
}
```

```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE
 .\InputAndOutput }
Enter the value of a: 6
Enter the value of b: 4
a is greater than b
```

InputAndOutput.cpp

```cpp
#include <iostream>
using namespace std;

int main()
{
    int a, b;
    cout << "Enter the value of a: ";
    cin >> a;
    cout << "Enter the value of b: ";
    cin >> b;

    if(a > b) {
        cout << "a is greater than b" << endl;
    }
    if (b > a) {
        cout << "b is greater than a" << endl;
    }
    return 0;
}
```

```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE
 .\InputAndOutput }
Enter the value of a: 3
Enter the value of b: 6
b is greater than a
```

Note : cin ignores ENTER (\n), TAB (\t) and SPACE ( ) while taking input. These are called whitespace characters. Use cin.get() to read these whitespace characters.

Using nested if - else :

```cpp
#include <iostream>
using namespace std;

int main()
{
    int a;
    cin >> a;

    if(a > 0) {
        cout << "a is positive" << endl;
    }
    else {
        cout << "a is not positive" << endl;
    }
    return 0;
}
```

→ In this else block, we can see that either a will be negative or 0. So we can further break down this else block.

```cpp
#include <iostream>
using namespace std;

int main()
{
    int a;
    cin >> a;
    if(a > 0) {
        cout << "a is positive" << endl;
    }
    else {
        if (a == 0) {
            cout << "a is 0" << endl;
        }
        else {
            cout << "a is negative" << endl;
        }
    }
    return 0;
}
```

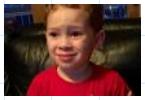
```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE
 .\InputAndOutput }
5
a is positive
```

```cpp
#include <iostream>
using namespace std;

int main()
{
    int a;
    cin >> a;
    if(a > 0) {
        cout << "a is positive" << endl;
    }
    else {
        if (a == 0) {
            cout << "a is 0" << endl;
        }
        else {
            cout << "a is negative" << endl;
        }
    }
    return 0;
}
```

```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE
 .\InputAndOutput }
0
a is 0
```

```cpp
#include <iostream>
using namespace std;

int main()
{
    int a;
    cin >> a;
    if(a > 0) {
        cout << "a is positive" << endl;
    }
    else {
        if (a == 0) {
            cout << "a is 0" << endl;
        }
        else {
            cout << "a is negative" << endl;
        }
    }
    return 0;
}
```

```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE
 .\InputAndOutput }
-9
a is negative
```



Code ganda hote jaa raha hai. Ek-do baar aur nested if-else kar diya to coding chorni padhegi.

Solution : else if

```
if ( ~~~~ ) {

}
else if ( ~~~~ ) {

}
else {

}
```

```cpp
int a ;
cout<<" enter the value of a "<<endl;
cin>>a;


if(a>0) {
    cout<<" A is positive"<< endl;
}
else if(a<0) {
    cout<<" A is negative"<<endl;
}
else {
    cout<<" A is 0"<<endl;
}
```

Note : else if and else are optional.
else can be used as a default case.

Homework : Output ??

(1)

```cpp
#include <iostream>
using namespace std;

int main() {

        int a = 9;
        if(a == 9) {
                cout<< "NINEY";
        }

        if(a > 0){
                cout<<"POSITIVE";
        }
        else
        {
                cout<<"NEGATIVE";
        }
}
```

Answer : NINEYPOSITIVE

(2)

```cpp
#include <iostream>
using namespace std;

int main() {

        int a = 2;
        int b = a+1;

        if((a=3)==b) {
                cout<<a;
        }
        else
        {
                cout<<a+1;
        }
}
```

Answer : 3

a is assigned 3.
Now, a == b is true.

(3)

```cpp
#include <iostream>
using namespace std;

int main() {

        int a = 24;

        if(a > 20){
                cout<< "Love ";
        }
        else if(a == 24) {
                cout<<"Lovely";
        }
        else
        {
                cout<<"Babbar";
        }
        cout<<a;
}
```

Answer : Love24

(4) Code Homework :

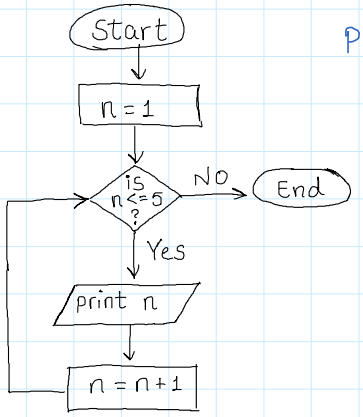
```cpp
#include <iostream>
using namespace std;

int main()
{
    char a;
    cin >> a;
    // 'A' is 65
    // 'a' is 97
    // '0' is 48
    if(a >= 'A' && a <= 'Z') {
        cout << "This is upper case" << endl;
    }
    else if(a >= 'a' && a <= 'z') {
        cout << "This is lower case" << endl;
    }
    else if(a >= '0' && a <= '9') {
        cout << "This is a digit" << endl;
    }
    return 0;
}
```

We can also use ASCII values

Loops

07 January 2022 20:16

Recollect from flowcharts :



Print from 1 to 5.

while loop :

```
while (condition) {
}
```

जब तक ये तुरू हैं

तब तक ये करते रहो

while the condition is true, keep on executing the block

Example :

Print 1 to N.

```cpp
#include <iostream>
using namespace std;

int main()
{
    int n;
    cin >> n;
    int i = 1;
    while(i <= n) {
        cout << i << " ";
        i = i + 1;
    }
    return 0;
}
```

```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE
.\InputAndOutput }
8
1 2 3 4 5 6 7 8
```

Example : Find sum from 1 to n.

```cpp
#include <iostream>
using namespace std;

int main()
{
    int n;
    cin >> n;
    int sum = 0;
    int i = 1;
    while(i <= n) {
        sum = sum + i;
        i = i + 1;
    }
    cout << "Sum from " << 1 << " to " << n << " = " << sum << endl;
    return 0;
}
```

```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE
 { .\InputAndOutput }
8
Sum from 1 to 8 = 36
```

Example: Find sum of all even numbers from 1 to n.

```cpp
#include <iostream>
using namespace std;

int main()
{
    int n;
    cin >> n;
    int sum = 0;
    int i = 2;
    while(i <= n) {
        if(i % 2 == 0)
            sum = sum + i;
        i = i + 1;
    }
    cout << "Sum from " << 1 << " to " << n << " = " << sum << endl;
    return 0;
}
```

```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE
8
Sum from 1 to 8 = 20
```

Homework: Fahrenheit to Celsius.

$$C = \frac{5}{9}(F - 32)$$

```cpp
#include <iostream>
using namespace std;

int main()
{
    float fahrenheit;
    cin >> fahrenheit;
    float celsius = (5.0/9) * (fahrenheit - 32);
    cout << fahrenheit << " F = " << celsius << " C" << endl;

    return 0;
}
```

```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE
 { .\InputAndOutput }
100
100 F = 37.7778 C
```

Example: Prime or not ?

```cpp
#include <iostream>
using namespace std;

int main()
{
    int n;
    cin >> n;
    int i = 2;
    while(i < n) {
        if(n % i == 0) {
            cout << "Not Prime" << endl;
            return 0;
        }
        i = i + 1;
    }
    cout << "Prime" << endl;
    return 0;
}
```

```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE
 { .\InputAndOutput }
14
Not Prime
```

```cpp
#include <iostream>
using namespace std;

int main()
{
    int n;
    cin >> n;
    int i = 2;
    while(i < n) {
        if(n % i == 0) {
            cout << "Not Prime" << endl;
            return 0;
        }
        i = i + 1;
    }
    cout << "Prime" << endl;
    return 0;
}
```

```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE
 { .\InputAndOutput }
11
Prime
```

Patterns

07 January 2022 20:46

Example :

n = 4 (4 rows, 4 columns)

For every row, print 'row' num. of columns or stars.

here n = 3

```cpp
#include <iostream>
using namespace std;

int main() {
    int n;
    cin >> n;
    int i = 0;
    while(i < n) { // i = 0, 1, 2, 3, ..., n-1  which is basically n times
        int j = 0;
        while(j < n) {
            cout << "* ";
            j = j + 1;
        }
        cout << endl; // after printing one row, we need to enter the next row so endl
        i = i + 1;
    }
    return 0;
}
```

```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE

 .\InputAndOutput }
3
* * *
* * *
* * *
```

Ye code samajh nahi aa raha

Don't worry, initially thoda difficult lag sakta hai BUT the next video has 18 problems solved end to end so keep going

Example : For n = 3,

```
1 1 1
2 2 2
3 3 3
```

```cpp
#include <iostream>
using namespace std;

int main() {
    int n;
    cin >> n;
    int i = 1;
    while(i <= n) {
        int j = 1;
        while(j <= n) {
            cout << i << " ";
            j = j + 1;
        }
        cout << endl;
        i = i + 1;
    }
    return 0;
}
```

```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE

4
1 1 1 1
2 2 2 2
3 3 3 3
4 4 4 4
```

Pattern Questions (Part- 2)

09 January 2022 19:06

Q1. Print

```
1 2 3 4
1 2 3 4
1 2 3 4
1 2 3 4
```

for n = 4

Answer :

```cpp
#include <iostream>
using namespace std;

int main() {
    int n;
    cin >> n;
    int i = 1; // Try printing in reverse (solved in the video)
    while(i <= n) {
        int j = 1;
        while(j <= n) {
            cout << j << " ";
            j = j + 1;
        }
        cout << endl;
        i = i + 1;
    }
    return 0;
}
```

```
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

4
1 2 3 4
1 2 3 4
1 2 3 4
1 2 3 4
```

```cpp
#include <iostream>
using namespace std;

int main() {
    int n;
    cin >> n;
    int i = 1;
    // Try printing in reverse (solved in the video)
    while(i <= n) {
        int j = 1;
        while(j <= n) {
            cout << j << " ";
            j = j + 1;
        }
        cout << endl;
        i = i + 1;
    }
    return 0;
}
```

```
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

rk\Coding\Patterns\" ; if ($?) { g++ Pattern.cpp -o Pattern
3
1 2 3
1 2 3
1 2 3
```

Homework : For n = 4,

```
4 3 2 1
4 3 2 1
4 3 2 1
4 3 2 1
```

Q2. Print :

```
1 2 3
4 5 6
7 8 9
```

for n = 3

Logic - maintain a variable, incrementing it by 1 after every cout.

Answer :

```cpp
#include <iostream>
using namespace std;

int main() {
    int n;
    cin >> n;
    int i = 1;
    int toPrint = 1;
    while(i <= n) {
        int j = 1;
        while(j <= n) {
            cout << toPrint << " ";
            toPrint = toPrint + 1;
            j = j + 1;
        }
        cout << endl;
        i = i + 1;
    }
    return 0;
}
```

```
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

\Work\Coding\Patterns\" ; if ($?) { g++ P
3
1 2 3
4 5 6
7 8 9
```

```cpp
#include <iostream>
using namespace std;

int main() {
    int n;
    cin >> n;
    int i = 1;
    int toPrint = 1;
    while(i <= n) {
        int j = 1;
        while(j <= n) {
            cout << toPrint << " ";
            toPrint = toPrint + 1;
            j = j + 1;
        }
        cout << endl;
        i = i + 1;
    }
    return 0;
}
```

```
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

rk\Coding\Patterns\" ; if ($?) { g++ Pa
4
1 2 3 4
5 6 7 8
9 10 11 12
13 14 15 16
```

Try printing :

```
9 8 7
6 5 4
3 2 1
```

for n = 3.

3 2 1

16 15 14 13
12 11 10 9
8 7 6 5
4 3 2 1

{Hint : Starting point has a relation with n

Q3. Print :

for n = 4.

{Hint : We are printing '*', row number of times where row = 1, 2, ...

Answer :

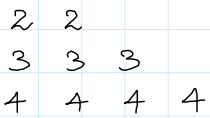
```cpp
#include <iostream>
using namespace std;

int main() {
    int n;
    cin >> n;
    int row = 1;
    while(row <= n) {
        int column = 1;
        while(column <= row) {
            cout << "* ";
            column = column + 1;
        }
        cout << endl;
        row = row + 1;
    }
    return 0;
}
```

```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE
rk\Coding\Patterns\" ; if ($?) { g++ Patter
4
*
* *
* * *
* * * *
```

```cpp
#include <iostream>
using namespace std;

int main() {
    int n;
    cin >> n;
    int row = 1;
    while(row <= n) {
        int column = 1;
        while(column <= row) {
            cout << "* ";
            column = column + 1;
        }
        cout << endl;
        row = row + 1;
    }
    return 0;
}
```

```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE
5
*
* *
* * *
* * * *
* * * * *
```

Q4. Print :



- for n - 4.

(Easy)

Answer :

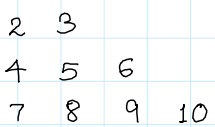
```cpp
#include <iostream>
using namespace std;

int main() {
    int n;
    cin >> n;
    int row = 1;
    while(row <= n) {
        int column = 1;
        while(column <= row) {
            cout << row << " ";
            column = column + 1;
        }
        cout << endl;
        row = row + 1;
    }
    return 0;
}
```

```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE
rk\Coding\Patterns\" ; if ($?) { g++ Pattern.cpp
4
1
2 2
3 3 3
4 4 4 4
```

```cpp
#include <iostream>
using namespace std;

int main() {
    int n;
    cin >> n;
    int row = 1;
    while(row <= n) {
        int column = 1;
        while(column <= row) {
            cout << row << " ";
            column = column + 1;
        }
        cout << endl;
        row = row + 1;
    }
    return 0;
}
```

```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE
5
1
2 2
3 3 3
4 4 4 4
5 5 5 5 5
```

Homework - Print :



for n = 4.

Answer :

```cpp
#include <iostream>
```

7 8 9 10

Answer :

```cpp
#include <iostream>
using namespace std;

int main() {
    int n;
    cin >> n;
    int row = 1, toPrint = 1;
    while(row <= n) {
        int column = 1;
        while(column <= row) {
            cout << toPrint << " ";
            toPrint = toPrint + 1;
            column = column + 1;
        }
        cout << endl;
        row = row + 1;
    }
    return 0;
}
```

```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE

4
1
2 3
4 5 6
7 8 9 10
```

q5 Print

```
1
2 3
3 4 5
4 5 6 7
```

for n = 4.

Logic - We are starting from i for every row i.
Keep incrementing i times for each row.

Answer :

```cpp
#include <iostream>
using namespace std;

int main() {
    int n;
    cin >> n;
    int row = 1, toPrint = 1;
    while(row <= n) {
        int column = 1;
        toPrint = row; // start printing from row
        while(column <= row) {
            cout << toPrint << " ";
            toPrint = toPrint + 1;
            column = column + 1;
        }
        cout << endl;
        row = row + 1;
    }
    return 0;
}
```

```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE

4
1
2 3
3 4 5
4 5 6 7
```

Homework : Solve the above question without using the extra variable toPrint.

Logic - Column starts from 1 for every row. Change it to start from row and go till row + row = row * 2 (exclusive)

Answer :

```cpp
#include <iostream>
using namespace std;

int main() {
    int n;
    cin >> n;
    int row = 1;
    while(row <= n) {
        int column = row;
        while(column < row * 2) {
            cout << column << " ";
            column = column + 1;
        }
        cout << endl;
        row = row + 1;
    }
    return 0;
}
```

```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE
rk\Coding\Patterns\" ; if ($?) { g++ Pattern.cp
4
1
2 3
3 4 5
4 5 6 7
```

Q6. Print

```
1
2 1
3 2 1
4 3 2 1
```

Hint: starting from row number and decrementing.

Logic :

	column = 1	2	3	4
row = 1	1			
row = 2	2	1		
row = 3	3	2	1	
row = 4	4	3	2	1

Subtract column number from row number and add 1.

(row - column + 1)

Answer :

```cpp
#include <iostream>
using namespace std;

int main() {
    int n;
    cin >> n;
    int row = 1;
    while(row <= n) {
        int column = 1;
        while(column <= row) {
            cout << (row - column + 1) << " ";
            column = column + 1;
        }
        cout << endl;
        row = row + 1;
    }
    return 0;
}
```

```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE
rk\Coding\Patterns\" ; if ($?) { g++ Pattern.cpp -o Pat
5
1
2 1
3 2 1
4 3 2 1
5 4 3 2 1
```

q7 Print : A A A for n = 3.
B B B
C C C

Answer :

```cpp
int main() {
    int n;
    cin >> n;
    int i = 1;
    while(i <= n) {
        int j = 1;
        while(j <= n) {
            // you can also print 'A' + i - 1 because with an int
            // 'A' will get typecasted to 65
            cout << (char)(65 + i - 1) << " ";
            j = j + 1;
        }
        i = i + 1;
        cout << endl;
    }
    return 0;
}
```

```
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE
rk\Coding\Patterns\" ; if ($?) { g++ Pattern.cpp -o Pattern } ; if ($?) {
4
A A A A
B B B B
C C C C
D D D D
```

q8. Print : A B C for n = 3
A B C
A B C (Homework Code)

Answer :

```cpp
int main() {
    int n;
    cin >> n;
    int i = 1;
    while(i <= n) {
        int j = 1;
        char ch = 'A';
        while(j <= n) {
            cout << ch << " ";
            ch = ch + 1;
            j = j + 1;
        }
        i = i + 1;
        cout << endl;
    }
    return 0;
}
```

```
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE
rk\Coding\Patterns\" ; if ($?) { g++ Pattern.cpp -o
3
A B C
A B C
A B C
```

q9 Print : A B C for n = 3
D E F
G H I

Logic is same as with numbers

Answer :

```cpp
int main() {
    int n;
    cin >> n;
    int i = 1;
    char ch = 'A';
    while(i <= n) {
        int j = 1;
        while(j <= n) {
            cout << ch << " ";
            ch = ch + 1;
            j = j + 1;
        }
        i = i + 1;
        cout << endl;
    }
    return 0;
}
```

```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE
rk\Coding\Patterns\" ; if ($?) { g++ Pattern.cpp -o Pa
3
A B C
D E F
G H I
```

Q10 Print :

A B C
B C D
C D E

Logic is same as with numbers

{ 'A' + row - 1
for row = 1, 2, ...
increment this

Answer :

```cpp
int main() {
    int n;
    cin >> n;
    int i = 1;
    while(i <= n) {
        char ch = 'A' + i - 1;
        int j = 1;
        while(j <= n) {
            cout << ch << " ";
            ch = ch + 1;
            j = j + 1;
        }
        i = i + 1;
        cout << endl;
    }
    return 0;
}
```

```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE
4
A B C D
B C D E
C D E F
D E F G
```

```cpp
int main() {
    int n;
    cin >> n;
    int i = 1;
    while(i <= n) {
        int j = 1;
        while(j <= n) {
            char ch = 'A' + i + j - 2;
            cout << ch << " ";
            j = j + 1;
        }
        i = i + 1;
        cout << endl;
    }
    return 0;
}
```

```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE
rk\Coding\Patterns\" ; if ($?) { g++ Pattern.cpp -o Patterr
4
A B C D
B C D E
C D E F
D E F G
```

Q11. Print :

A
B B
C C C

for n = 3

Answer :

```
int main() {
    int n;
    cin >> n;
    int i = 1;
    while(i <= n) {
        int j = 1;
        char ch = 'A' + i - 1;
        while(j <= i) {
            cout << ch << " ";
            j = j + 1;
        }
        i = i + 1;
        cout << endl;
    }
    return 0;
}
```

```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE
rk\Coding\Patterns\" ; if ($?) { g++ Pattern.cpp -o
4
A
B B
C C C
D D D D
```

Q12 Print :

```
A
B C
D E F
G H I J
```

(Try yourself)

Logic same as with numbers homework question.

Answer :

```
int main() {
    int n;
    cin >> n;
    int i = 1;
    char ch = 'A';
    while(i <= n) {
        int j = 1;
        while(j <= i) {
            cout << ch << " ";
            ch = ch + 1;
            j = j + 1;
        }
        i = i + 1;
        cout << endl;
    }
    return 0;
}
```

```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE
4
A
B C
D E F
G H I J
```

Q13 Print :

```
A
B C
C D E
D E F G
```

for n = 4

Logic is same as numbers.

Answer:

```
int main() {
    int n;
    cin >> n;
    int i = 1;
    while(i <= n) {
        char ch = 'A' + i - 1;
        int j = 1;
        while(j <= i) {
            cout << ch << " ";
            ch = ch + 1;
            j = j + 1;
        }
        i = i + 1;
        cout << endl;
    }
    return 0;
}
```

```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE
rk\Coding\Patterns\" ; if ($?) { g++ Pattern.cpp
4
```

The Babbar Answer

```
#include<iostream>
using namespace std;

int main() {

    int n;
    cin>>n;

    int row = 1;

    while(row <= n) {

        int col = 1;

        while(col <= row) {
            char ch = ('A' + row + col - 2);
            cout<<ch;
            col = col + 1;
        }
        cout<<endl;
        row = row + 1;

    }
```

```
20  }
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

```
rk\Coding\Patterns\" ; if ($?) { g++ Pattern.cpp
4
A
B C
C D E
D E F G
```

```
18          col = col + 1;
19      }
20      cout<<endl;
21      row = row + 1;
22
23  }
24
25  }
```

(Homework)

Q 14 Print :

```
D
C D
B C D
A B C D
```

Logic - We are starting from ch = ('A' + rows - 1) in row 1.
ch = ('A' + rows - 2) in row 2.
ch = ('A' + rows - 3) in row 3.
⋮
ch = ('A' + rows - i) for row i.

Answer :

```
4   int main() {
5       int n;
6       cin >> n;
7       int i = 1;
8       while(i <= n) {
9           char ch = 'A' + n - i;
10          int j = 1;
11          while(j <= i) {
12              cout << ch << " ";
13              ch = ch + 1;
14              j = j + 1;
15          }
16          i = i + 1;
17          cout << endl;
18      }
19      return 0;
20  }
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

```
rk\Coding\Patterns\" ; if ($?) { g++ Pattern.cpp
4
D
C D
B C D
A B C D
```

Q 15 Print :

```
      *
    * *
  * * *
* * * *
```

for n = 4 (Notice the number of spaces and its relation with the row number, i)

Logic - for row i, we are first printing n-i spaces and then printing i stars.

Answer :

```
int main() {
    int n;
    cin >> n;
    int i = 1;
    while(i <= n) {
        int space = 1;
        while(space <= n - i) {
            cout << " ";
            space = space + 1;
        }
        int j = 1;
        while(j <= i) {
            cout << "*";
            j = j + 1;
        }
        i = i + 1;
        cout << endl;
    }
    return 0;
}
```

```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE
\Work\Coding\Patterns\" ; if ($?) { g++
4
   *
  **
 ***
****
```

Q16. Print :

```
* * * *
* * *
* *
*
```

(Homework)

Answer :

```
#include <iostream>
using namespace std;

int main() {
    int n;
    cin >> n;
    int i = 1;
    while(i <= n) {
        int j = 1;
        while(j <= n - i + 1) {
            cout << "* ";
            j = j + 1;
        }
        i = i + 1;
        cout << endl;
    }
    return 0;
}
```

```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE
\Work\Coding\Patterns\" ; if ($?) { g++ Patte
4
* * * *
* * *
* *
*
```

Q17. Print :

```
* * * *
  * * *
    * *
      *
```

for n = 4

(Homework)

Logic : Printing i-1 spaces & n-i+1 stars for i^{th} row.

Answer :

```
int main() {
    int n;
    cin >> n;
    int i = 1;
    while(i <= n) {
        int j = 1, space = 1;
        while(space < i) {
            cout << " ";
            space = space + 1; // space++ is also used
        }
        while(j <= n - i + 1) {
            cout << "*";
            j = j + 1;
        }
        i = i + 1;
        cout << endl;
    }
    return 0;
}
```

```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE
\Work\Coding\Patterns\" ; if ($?) { g++ Pattern.cpp -o Pattern }
4
****
 ***
  **
   *
```

Q18 Print :

```
1 1 1 1
  2 2 2
    3 3
      4
```

for n = 4 (Homework)

Answer :

```
4  int main() {
5      int n;
6      cin >> n;
7      int i = 1;
8      while(i <= n) {
9          int j = 1, space = 1;
10         while(space < i) {
11             cout << " ";
12             space = space + 1; // space++ is also used
13         }
14         while(j <= n - i + 1) {
15             cout << i;
16             j = j + 1;
17         }
18         i = i + 1;
19         cout << endl;
20     }
21     return 0;
22 }
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

```
\Work\Coding\Patterns\" ; if ($?) { g++ Pattern.cpp -o Pat
4
1111
 222
  33
   4
```

Q19 Print :

```
      1
    2 2
  3 3 3
4 4 4 4
```

for n = 4 (Homework)

Answer :

```
4  int main() {
5      int n;
6      cin >> n;
7      int i = 1;
8      while(i <= n) {
9          int j = 1, space = 1;
10         while(space <= n - i) {
11             cout << " ";
12             space = space + 1; // space++ is also used
13         }
14         while(j <= i) {
15             cout << i;
16             j = j + 1;
17         }
18         i = i + 1;
19         cout << endl;
20     }
21     return 0;
22 }
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

```
\Work\Coding\Patterns\" ; if ($?) { g++ Pattern.cpp -o Pat
4
   1
  22
 333
4444
```

Q20 Print :

```
      1
    2 3
  4 5 6
7 8 9 10
```

for n = 4

Logic - Print space (n-i) times and print num. Then increment num.

Answer -

```cpp
int main() {
    int n;
    cin >> n;
    int i = 1, num = 1;
    while(i <= n) {
        int j = 1, space = 1;
        while(space <= n - i) {
            cout << "  "; // Notice two spaces
            space = space + 1;
        }
        while(j <= i) {
            cout << num << " ";
            num = num + 1;
            j = j + 1;
        }
        i = i + 1;
        cout << endl;
    }
    return 0;
}
```

```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE
4
      1
    2 3
  4 5 6
7 8 9 10
```

Q 21 Print :

```
      1
    1 2 1
  1 2 3 2 1
1 2 3 4 3 2 1
```

for n = 4

Logic - Use 2 while loops for each row i.
One for printing (n-i) "__" (double spaces)
One for printing from 1 to i.
Other for printing from (i-1) to 1.

Answer :

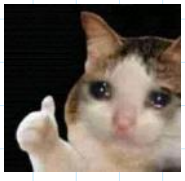
```cpp
#include <iostream>
using namespace std;

int main() {
    int n;
    cin >> n;
    int i = 1;
    while(i <= n) {
        int j = 1, space = 1;
        while(space <= n - i) {
            cout << "  "; // Notice two spaces
            space = space + 1;
        }
        while(j <= i) {
            cout << j << " ";
            j = j + 1;
        }
        j = i - 1;
        while(j >= 1) {
            cout << j << " ";
            j = j - 1;
        }
        i = i + 1;
        cout << endl;
    }
    return 0;
}
```

TAGDA HOMEWORK :

Print :

```
1 2 3 4 5 5 4 3 2 1
1 2 3 4 ☆ ☆ 4 3 2 1
1 2 3 ☆ ☆ ☆ ☆ 3 2 1
1 2 ☆ ☆ ☆ ☆ ☆ ☆ 2 1
1 ☆ ☆ ☆ ☆ ☆ ☆ ☆ ☆ 1
```

for n = 5



Answer :

```cpp
int main() {
    int n;
    cin >> n;
    int i = 1;
    while(i <= n) {
        // part 1: numbers from 1 to n - i + 1
        int j = 1;
        while(j <= n - i + 1) {
            cout << j << " ";
            j = j + 1;
        }
        // part 2: Stars (i-1)*2 times
        j = 1;
        while(j <= (i-1)*2) {
            cout << "* ";
            j = j + 1;
        }
        // part 3: numbers from n - i + 1 to 1
        j = n - i + 1;
        while(j >= 1) {
            cout << j << " ";
            j = j - 1;
        }
        i = i + 1;
        cout << endl;
    }
    return 0;
}
```

Bitwise Operators

11 January 2022 17:39

① Bitwise AND : (&)

Eg: 2 & 3 = 010 & 011 = 010

Both bits must be 1 to get 1.

x	y	AND
0	0	0
0	1	0
1	0	0
1	1	1

Eg: 5 & 7 :

```
  101
& 111
-----
  101   = (5)
```

② Bitwise OR : (|)

Eg: 2 | 5 :

```
  010
  101
-----
  111
```

If at least one of the bits is 1 then the output will be 1.

x	y	OR
0	0	0
0	1	1
1	0	1
1	1	1

Eg: 3 | 6 :

```
  011
  110
-----
  111
```

③ Bitwise NOT : (~)

Eg: (consider 4 byte representation)

a = 2 = 000...0010 (30 bits)

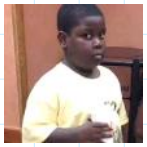
~a = ~2 = 111...1101

The leading 1 is the signed bit.

Two's complement :

```
  000...0010
+          1
------------
```

x	NOT
0	1
1	0



000....0011 = -3

④ Bitwise XOR : (^)

x	y	XOR
0	0	0
0	1	1
1	0	1
1	1	0

If different bits → 1
If same bits → 0

Eg: 5 ^ 7 = 101
^ 111
010 = 2

```
1  #include<iostream>
2  using namespace std;
3
4
5  int main() {
6
7      int a = 4;
8      int b = 6;
9
10     cout<<" a&b " << (a&b) << endl;
11     cout<<" a|b " << (a|b) << endl;
12     cout<<" ~a " << ~a << endl;
13     cout<<" a^b " << (a^b) << endl;
14
15 }
```

```
BitwiseOperators && "/Users/lovebabbar/"BitwiseOperators
 a&b 4
 a|b 6
 ~a -5
 a^b 2
lovebabbar@192 ~ %
```

⑤ Left Shift : (<<)

Eg: 5 << 1 → Left shift 5, 1 time
= 101 << 1 → 00....0101 << 1
= 00....1010 = 10 = 5×2^1

Eg: 3 << 2 → 00....011 << 2
= 00....1100 = 12 = 3×2^2

In most cases we multiply with power of 2.
But in some cases, this isn't true.

Eg: 0100...00 << 1 = 1000...00
Positive → Negative ??

So << is ok for smaller numbers.

⑥ Right Shift : (>>)

Eg: 15 >> 1 = 000....1111 >> 1
= 000....0111 = 7

Eg: 5 >> 2 = 000....0101 >> 2
000....0001 = (1)

Padding in << and >> for POSITIVE numbers is done with 0.

For NEGATIVE numbers, padding is compiler dependent.

```
cout<< (17>>1)<<endl;
cout<< (17>>2) <<endl;
cout<< (19<<1) <<endl;
cout<< (21<<2) <<endl;
```

```
8
4
38
84
lovebabbar@192 ~ %
```

(7) Increment / Decrement :

→ We can write $i = i+1$ as $i++$ or $++i$.

$i++$ is called Post - Increment
$++i$ is called Pre - Increment.

→ We can write $i = i-1$ as $i--$ or $--i$.

$i--$ is called Post - decrement
$--i$ is called Pre - decrement.

→ $a = a+b$ is same as $a += b$.

→ $a = a-b$ is same as $a -= b$.

Post - Increment : The value gets used first and then increments.

Eg: int i = 3, a = 2;

```
int sum = a + (i++);
sum = 2 + 3;
sum = 5;
```
Now i is 4.

Pre - Increment : The value gets incremented first and then gets used.

Eg: int i = 3, a = 2;

```
int sum = a + (++i);     i has become 4
sum = 2 + 4 ;            first.
sum = 6;
```

Post - Decrement : The value gets used first and then rements.

Eg: int i = 3, a = 2;

```
int sum = a + (i--);
sum = 2 + 3 ;
sum = 5;
  Now i is 2.
```

Pre - Decrement : The value gets remented first and then gets used.

Eg: int i = 3, a = 2;

```
int sum = a + (--i);     i has become 2
sum = 2 + 2 ;            first.
sum = 4 ;
```

```
20    int i = 7;
21
22    cout<< (++i) <<endl;
23    // 8
24    cout<< (i++) <<endl;
25    // 8 , i = 9
26    cout<< (i--) <<endl;
27    //9 , i = 8
28    cout<< (--i) <<endl;
29    // 7, i =7
```

```
8
8
9
7
lovebabbar@192 ~ %
```

Q1

```
1  #include <iostream>
2  using namespace std;
3  int main()
4  {
5          int a, b = 1;
6          a = 10;
7          if (++a)
8             cout << b;
9          else
10             cout << ++b;
11 }
```

Answer : 1

Q2

```
#include <iostream>
using namespace std;
int main()
{
    int a = 1;
    int b = 2;

    if(a-- > 0 && ++b > 2 ){
    cout << "Stage1 - Inside If ";
    } else{
    cout << "Stage2 - Inside else ";
    }
    cout << a << " " << b << endl;
}
```

Answer : Stage1 - Inside If 0 3

Q3

```
#include <iostream>
using namespace std;
int main()
{
    int a = 1;
    int b = 2;

    if(a-- > 0 || ++b > 2 ){
    cout << "Stage1 - Inside If ";
    } else{
    cout << "Stage2 - Inside else ";
    }
    cout << a << " " << b << endl;
}
```

Answer : Stage1 - Inside If 0 2

Hint: Only one of the conditions must be true for || so it won't check ++b > 2.

Q3

```
#include <iostream>
using namespace std;
int main()
{
        int number = 3;
        cout << (25 * (++number) );
}
```

Answer : 100

Q4

```
#include <iostream>
using namespace std;
int main()
{
    int a = 1;
    int b = a++;
    int c = ++a;
    cout << b ;
    cout << c;
}
```

Answer : 13

Example Code :

```cpp
#include<iostream>
using namespace std;

int main() {

    int n ;
    cout<<" enter the value of n" <<endl;
    cin >> n;

    cout<<"printing count from 1 to n" << endl;

    for(int i = 1; i<=n; i++) {
        cout<< i << endl;
    }


}
```

```
& "/Users/lovebabbar/"forLoop
 enter the value of n
5
printing count from 1 to n
1
2
3
4
5
lovebabbar@192 ~ %
```

Structure : `for ( int i = 0 ; i < n ; i++ )`

- `int i = 0` — initialization
- `i < n` — cond. to check before every loop/iteration
- `i++` — updation

```
{
  ~~~~
  ~~~~
}
```

Not Necessary to specify.

Example : for(;;;) = ?

```cpp
#include<iostream>
using namespace std;

int main() {

    int n ;
    cout<<" enter the value of n" <<endl;
    cin >> n;

    cout<<"printing count from 1 to n" << endl;
    int i = 1;
    for(; i<=n; ) {
        cout<< i << endl;
        i++;
    }


}
```

→

```
lovebabbar@192 ~ % cd "/Users/lovebabbar/" && g++ forLoop.cpp -o forLoop &
& "/Users/lovebabbar/"forLoop
 enter the value of n
5
printing count from 1 to n
1
2
3
4
5
lovebabbar@192 ~ %
```

You can declare all the 3 parts outside the paranthesis ().

WRONG :

```cpp
int i = 1;
for(;;; ) {
    if(i<=n) {
        cout<< i << endl;
    }

    i++;
}
```

→ 

The for loop doesn't know when to stop.

Solution : break ;

↳ Gets you out of the current loop.

```cpp
int n ;
cout<<" enter the value of n" <<endl;
cin >> n;

cout<<"printing count from 1 to n" << endl;
int i = 1;
for(; ; ) {
    if(i<=n) {
        cout<< i << endl;
    }
    else{
        break;
    }

    i++;
}
```

```
 enter the value of n
5
printing count from 1 to n
1
2
3
4
5
lovebabbar@192 ~ %
```

Example: Using multiple inits., conditions and updations.

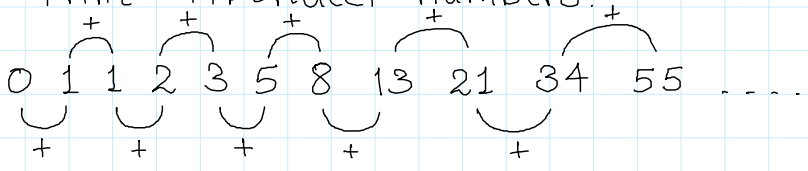
```cpp
for(int a = 0 , b =1, c = 2; a>=0 && b>=1 && c>=2; a--,b--, c-- ){
    cout<<a <<" "<< b << " "<<c << endl;
}
```

Question: Print 1 to n.

```cpp
#include<iostream>
using namespace std;

int main() {

    int n ;
    cout<<" enter the value of n" <<endl;
    cin >> n;

    int sum = 0;

    for(int i=1; i<=n; i++ ) {
        //sum = sum + i;
        sum += i;
    }

    cout<< sum <<endl;

}
```

```
lovebabbar@192 ~ % cd "/Users/lovebabbar/" && g++ forSum.cpp -o forSum &&
"/Users/lovebabbar/"forSum
 enter the value of n
5
15
lovebabbar@192 ~ %
```

Question: Print Fibonacci numbers.



Solution :

```cpp
#include<iostream>
using namespace std;

int main() {

    int n = 10;

    int a = 0;
    int b = 1;
    cout<<a <<" " <<b<<" ";
    for(int i = 1; i<=n; i++ ) {

        int nextNumber = a+b;
        cout<<nextNumber<<" ";

        a = b;
        b = nextNumber;
    }


}
```

```
lovebabbar@192 ~ % cd "/Users/lovebabbar/" && g++ forFib.cpp -o forFib &&
"/Users/lovebabbar/"forFib
0 1 1 2 3 5 8 13 21 34 55 89
lovebabbar@192 ~ %
```

Question: Print if prime. (Logic already covered)

Ex: I/P - 7
O/P - Yes

Solution:

```cpp
#include<iostream>
using namespace std;

int main() {

    int n ;
    cout<<" enter the value of n" <<endl;
    cin >> n;

    bool isPrime = 1 ;

    for(int i = 2; i<n; i++) {

        //rem = 0, Not a Prime
        if(n%i == 0) {
            //cout<<" Not a Prime Number" << endl;
            isPrime = 0;
            break;
        }
    }

    if(isPrime == 0) {
        cout<<" Not a Prime Number"<<endl;
    }
    else
    {
        cout<<"is a Prime Number "<<endl;
    }


}
```

```
lovebabbar@192 ~ % cd "/Users/lovebabbar/" && g++ forPrime.cpp -o forPrime
 && "/Users/lovebabbar/"forPrime
 enter the value of n
101
is a Prime Number
```

break ka brother continue

continue: Used to skip an iteration of the current loop.

It skips the remaining block of code for that iteration.

code example:

```cpp
#include<iostream>
using namespace std;

int main() {

    for(int i=0; i<5; i++) {

        cout<< " HI " << endl;
        cout<< " Hey " << endl;
        continue;

        cout<< "Reply toh karde " <<endl;

    }


}
```

```
lovebabbar@192 ~ % cd "/Users/lovebabbar/" && g++ continue.cpp -o continue
 && "/Users/lovebabbar/"continue
 HI
 Hey
 HI
 Hey
 HI
 Hey
 HI
 Hey
 HI
 Hey
lovebabbar@192 ~ %
```

Output Questions:

Q1.

```cpp
#include<iostream>
using namespace std;
int main() {

    for(int i = 0; i<=5; i++) {
        cout<< i << " ";
        i++;
    }
```

Output: 0 2 4

Q1.

```cpp
#include<iostream>
using namespace std;
int main() {

    for(int i = 0; i<=5; i++) {
        cout<< i << " ";
        i++;
    }

}
```

Output : 0 2 4

Q2.

```cpp
#include<iostream>
using namespace std;
int main() {

    for(int i = 0; i<=5; i--) {
        cout<< i << " ";
        i++;
    }

}
```

Output : 0 0 0 0 0 ..... (infinite times)

Q3.

```cpp
#include<iostream>
using namespace std;
int main() {

    for(int i = 0; i<=15; i += 2 ) {
        cout<< i << " ";

        if( i&1 ){
            continue;
        }

        i++;
    }
}
```

Output : 0 3 5 7 9 11 13 15

Q4

```cpp
#include<iostream>
using namespace std;
int main() {

    for(int i=0;i<5;i++) {
        for(int j=i;j<=5;j++) {
            cout<<i << " " << j <<endl;
        }
    }
}
```

Output :

```
0 0
0 1
0 2
0 3
0 4
0 5
1 1
1 2
1 3
1 4
1 5
2 2
2 3
2 4
2 5
3 3
3 4
3 5
4 4
4 5
```

Q5

```
#include<iostream>
using namespace std;
int main() {

    for(int i=0;i<5;i++) {
        for(int j=i;j<=5;j++) {
            if(i+j == 10) {
                break;
            }
            cout<<i << " " << j <<endl;
        }
    }
}
```

Output:

```
0 0
0 1
0 2
0 3
0 4
0 5
1 1
1 2
1 3
1 4
1 5
2 2
2 3
2 4
2 5
3 3
3 4
3 5
4 4
4 5
```

Scope: The life-time of a variable. Where does the variable exist and after what line of code will it get destroyed. In short, its accessibility.

Example:

```cpp
#include<iostream>
using namespace std;


int main() {

    int a = 3;
    cout<< a << endl;

    if(true) {
        cout<< a <<endl;
    }


}
```

a is accessible throughout the main() function, after it is declared.

Example:

```cpp
#include<iostream>
using namespace std;


int main() {

    int a = 3;
    cout<< a << endl;

    if(true) {
        int a = 5;
        cout << a <<endl;
    }


}
```

This is 'local' to int main().

This is 'local' to if block, and is only accessible within the if block.

This will print out 3
5

Example:

```cpp
#include<iostream>
using namespace std;


int main() {

    int a = 3;
    cout<< a << endl;

    if(true) {
        int b = 5;
        cout << b <<endl;
    }

    cout<< b << endl;

}
```

??

→ 

Example:

```cpp
int i = 8;
cout << b << endl;

for(; i<8; i++) {
    cout<<" HI " << endl;
}
```

→ Kuch Nahi (Empty)

```
for(; i<8; i++) {
    cout<<" HI " << endl;
}
```

The i outside the for loop gets used.

Precedence Table:

(No need to mug up)

Precedence order	Operator	Associativity
1	() [] →	Left to right
2	++ -- - (unary) ! ~ * & sizeof	Right to left
3	* / %	Left to right
4	+ -	Left to right
5	<< >>	Left to right
6	< <= > >=	Left to right
7	== !=	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	? : (conditional)	Right to right
14	= += -= *= /= %= (assignment operators)	Right to left
15	, (comma Operator)	Left to right

Just like BODMAS prioritizes Brackets, we can also prioritize calculations using Brackets.

Q1. Subtract the product and sum of the digits of an integer.

Soln :

```cpp
class Solution {
public:
    int subtractProductAndSum(int n) {
        int product = 1, sum = 0;
        while(n) {
            product = product * (n % 10); // Can also use *=
            sum += (n % 10);
            n /= 10; // same as n = n / 10;
        }
        return product - sum;
    }
};
```

Success Details >

Runtime: 0 ms, faster than 100.00% of C++ online submissions for Subtract the Product and Sum of Digits of an Integer.

Memory Usage: 5.9 MB, less than 23.16% of C++ online submissions for Subtract the Product and Sum of Digits of an Integer.

Q2. Count number of 1 bits.

Soln :

```cpp
class Solution {
public:
    int hammingWeight(uint32_t n) {
        int ans = 0;
        while(n) {
            if(n & 1) {
                ans++;
            }
            n = n >> 1;
        }
        return ans;
    }
};
```

Q3. Reverse integer.

Soln :

```cpp
class Solution {
public:
    int reverse(int x) {
        int rev = 0;
        while (x != 0) {
            int pop = x % 10;
            x /= 10;
            if (rev > INT_MAX/10 || (rev == INT_MAX / 10 && pop > 7)) return 0;
            if (rev < INT_MIN/10 || (rev == INT_MIN / 10 && pop < -8)) return 0;
            rev = rev * 10 + pop;
        }
        return rev;
    }
};
```

Q4. Complement of Base 10

Soln :

```cpp
class Solution {
public:
    int bitwiseComplement(int n) {
        if(n == 0) return 1;

        int ans = 0, fac = 1;

        while(n != 0){
            int bit = n % 2 == 0;
            ans += fac * bit;
            fac *= 2;
            n /= 2;
        }
        return ans;
    }
};
```

Q5 Number Compliment.

Soln :

```cpp
class Solution {
public:
    int findComplement(int num) {
        int msb = (int)(log2(num));
        if(num ==0) {
            return 1;
        }

        int sum = 0;
        for(int i=msb; i>=0;i--) {
            if(num &(1<<i)){
                continue;
            }
            else {
                sum+=pow(2,i);
            }
        }
        return sum;
    }
};
```

Q6. Binary to Decimal.

Soln :

```cpp
#include <iostream>
using namespace std;

int main()
{
    int num = 1010001;
    int dec_value = 0;
    int base = 1;
    int temp = num;

    while (temp) {
        int last_digit = temp % 10;
        temp = temp / 10;

        dec_value += last_digit * base;

        base = base * 2;
    }
    cout << dec_value << endl;
    return 0;
}
```

Decimal to Binary :

(1) 10 →

$\frac{10}{2} = 5$, $\frac{5}{2} = 2$, $\frac{2}{2} = 1$, $\frac{1}{2} = 0$

(Remainder) 0 1 0 1

Finish ← Start

$2^3\ 2^2\ 2^1\ 2^0$

10 → 1010

Steps : i) Divide number by 2
ii) Store the remainder.
iii) Repeat for the dividend.

Division	Remainder
10/2	0
5/2	1
2/2	0
1/2	1

Reverse = 1010

g. n = 7 to Binary

Division	Remainder
7/2	1
3/2	1
1/2	1

Reverse = 111

(2) n = 5 → Binary ?

Logic : n & 1 = 0 if n is even.

n & 1 = 1 if n is odd

Thus, a bit is 1 if bit & 1 = 1
a bit is 0 if bit & 1 = 0

Thus, (any_bit) & 1 = (any_bit)

∴ Last bit of 5 = 5 & 1 = 1

Check 5 = 101 ⇒

```
  101
& 001
  ---
  (1)
```

Right shift 5 by 1 ⇒ n = 2 (010)

Second last bit of 5 = 2 & 1 = 0

Right shift 2 by 1 ⇒ n = 1 (001)

First bit of 5 = 1 & 1 = 1

Right shift 1 by 1 ⇒ n = 0 (000)

Khatam !

```cpp
#include<iostream>
#include<math.h>
using namespace std;


int main() {

    int n;
    cin >> n;


    int ans  = 0;
    int i = 0;
    while(n != 0 ) {

        int bit  = n & 1;

        ans = (bit * pow(10, i) ) + ans;

        n = n >> 1;
        i++;

    }

    cout<<" Answer is " << ans << endl;


}
```

Example : n = 6

① ans = 0 , i = 0

② bit = 6 & 1 = 0 (even)

③ ans = 10^0 x 0 + ans = 0 , n = 3 , i = 1

④ bit = 3 & 1 = 1 (odd)

⑤ ans = (10^1 x 1) + ans = 10 , n = 1 , i = 2

⑥ bit = 1 & 1 = 1 (odd)

⑦ ans = $(10^2 \times 1)$ + ans = 100 + 10 = 110 , n = 0

ans = 110

Negative Decimal to Binary: (Homework)

-6 ⟶ Binary ??

We have discussed the logic about the storage & display of negative numbers earlier.

$$-6 \rightarrow 6 \rightarrow 000\ldots00110 \rightarrow \begin{array}{r} 111\ldots11001 \\ +\quad\quad\quad\quad 1 \\ \hline 111\ldots11010 \end{array}$$

Think what is 111....11010 equal to if it is unsigned?

$\underbrace{111\ldots11}_{29\text{ bits}}010 = 4294967290$

$= 2^{32} - 6$

$\therefore -6 \xrightarrow{\text{Binary}}$ Binary of $(2^{32} - 6) \rightarrow \underbrace{111\ldots11}_{29}010$

We can't express this in any data type.

Assuming we have 2 byte (= 16 bit) integers:

(Might need online IDE)

```cpp
#include <iostream>
#include <math.h>
using namespace std;

int main(void)
{
    long long int n;
    cin >> n;
    unsigned long long int i = 0, ans = 0;
    if(n < 0) {
        n = pow(2, 16) + n;
    }
    cout << n << endl;
    while(n) {
        int lastBit = n & 1;
        ans = (pow(10, i) * lastBit) + ans;
        n = n >> 1;
        i++;
        cout << ans << endl;
    }
    cout << ans << endl;
    return 0;
}
```

Binary to Decimal:

$101011 = 2^0 \times 1 + 2^1 \times 1 + 2^2 \times 0 +$

$2^5\ 2^4\ 2^3\ 2^2\ 2^1\ 2^0 \quad 2^3 \times 1 + 2^4 \times 0 + 2^5 \times 1$

$= 1 + 2 + 0 + 8 + 0 + 32$

$= \boxed{43}$

Logic: $n \& 1 = \begin{cases} 0, & \text{don't do anything} \\ 1, & \text{multiply with } 2^i \end{cases}$

$i++, \quad n >> 1.$

We will give int n = 10010; but it is actually a decimal number so last bit = n%10;

```
#include<iostream>
#include<math.h>
using namespace std;


int main() {

    int n;
    cin >> n;

    int i = 0, ans = 0 ;

    while( n != 0) {

        int digit = n % 10;

        if( digit == 1) {
            ans = ans + pow(2, i);
        }

        n = n/10;
        i++;

    }
    cout<< ans << endl;


}
```