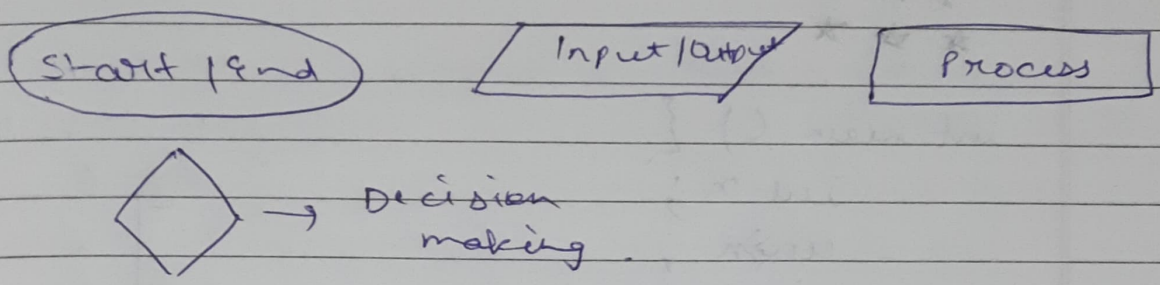
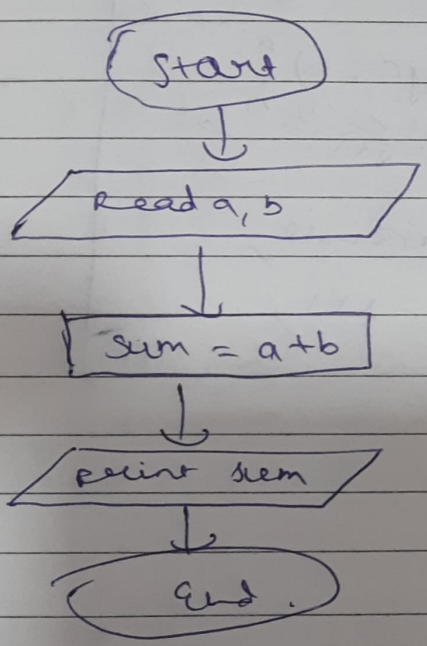
DSA in C++

Flowchart:



eg: Sum of 2 nos



Pseudocode:

eg: Sum of 2 no.s

→ Read a
→ " b
→ Let sum = 0
→ sum = a+b
→ print sum.

Patterns:

eg 1.) ★ ★ ★
★ ★ ★
★ ★ ★

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

eg 2.) 1 1 1
2 2 2
3 3 3

Same as above except cout << i; instead of cout << "★" ;

Date
DELTA Pg No.

Eg 3.)
1 2 3
1 2 3
1 2 3

cout << j ; instead of cout << "*";

Eg 4.)
3 2 1
3 2 1
3 2 1

cout << n-j+1 ; instead of cout << "*".

Eg 5.)
1 2 3
4 5 6
7 8 9

After int i=1; make a count variable with value = 1 and print count instead of * and in next line count = count + 1.

Eg 6.)
*
* *
* * *

Take while (j <= i) instead of (j <= n).

Eg 7.)
1
2 2
3 3 3

Take (j <= i) and cout << i instead of "*".

Eg 8.)
1
2 3
4 5 6

initially take count = 1;
while(j <= i) and cout << count;
count = count + 1;

Ans : In laptop view of upside pattern:

Eg 10.)
```
1
2 3
3 4 5
4 5 6 7
```

Eg 11.)
```
1
2 1
3 2 1
4 3 2 1
```

Eg 12.)
```
A A A
B B B
C C C
```

Eg 13.)
```
A B C
A B C
A B C
```

Eg 14.)
```
A B C
D E F
G H I
```

Eg 15.)
```
A B C
B C D
C D E
```

Eg 16.)
```
A
B B
C C C
```

Eg 17.)
```
A
B C
C D E
D E F G
```

Eg 18.)
```
D
C D
B C D
A B C D
```

Eg 19.)
```
   *
  * *
 * * *
* * * *
```

Eg 20.)
```
X X X X
X X X
X X
X
```

Eg 21.)
```
X X X X
 X X X
  X X
   X
```

Eg 22.)
```
1 1 1
 2 2 2
  3 3
   4
```

Eg 23.)
```
   1
  2 2
 3 3 3
4 4 4 4
```

Eg 24.)
```
1 2 3 4
  2 3 4
    3 4
      4
```

Eg 25.)
```
    1
   2 3
  4 5 6
 7 8 9 10
```

Eg 26.)
```
      1
    1 2 1
  1 2 3 2 1
1 2 3 4 3 2 1
```

Eg 27.)
```
1 2 3 4 5 5 4 3 2 1
1 2 3 4 * * 4 3 2 1
1 2 1 * * * * 3 2 1
1 2 * * * * * * 2 1
1 * * * * * * * * 1
```

Bitwise Operators:

AND → &
OR → |
NOT → ~
XOR → ^

AND:

x	y	z
0	0	0
0	1	0
1	0	0
1	1	1

Eg: a = 5 → 101 (if a & b asked)
b = 7 → 111
101 = 5

OR:

x	y	z
0	0	0
0	1	1
1	0	1
1	1	1

Eg: a = 2 → 10
b = 4 → 100
110 = 6

NOT:

x	z
0	1
1	0

Eg: a = 2 = 10 int = 4 bytes = 32 bit
so, 000---------010

Now, ~a = 1111 ---- 1101

(To print): means - a number

1's complement: 00 ---- 0010

2's: +1

00 ---- 0011

↓

3

So, ~a = -3

XOR:

x	y	z
0	0	0
0	1	1
1	0	1
1	1	0

eg: a = 2 0 1 0

b = 4 1 1 0

1 1 0 = 6 = a^b

Left Shift Operators:

eg: 5 << 1

000 ---- 0101

000 ---- 01010

↓

10

3 << 2

000 ---- 0110

000 ---- 001100

↓

12

Generally in left shift operator ans is multiplied by 2 no.s in small no. it works like in 010 - - - - 010

<<1

100 - - 0100

it becomes -ve so it does not work in big no.s.

Right shift operators:

eg: i) 5 >>1

ii) 5 >>2

000 - - - 00101

000 - - - 0001

It means 5 >>2 = 1. So divide it by 2 no. of time it is written to shift.

If +ve no: <<, >> → padding with 0

If -ve no: padding is Complier dependent.

Fibonacci Series:

0, 1, 1, 2, 3, 5, 8, 13, 21 - - -

$n = (n-1)_{fib} + (n-2)$

Eg: To print 10 no.s

```
int main(){
    int n = 10;
    int a = 0,
```

```
int b=1;
cout<<a<<" "<<b;
for(int i=1;i<=n;i++){
    nextno = a+b
    cout<<nextno.;
    a=b
    b=nextno.
}
```

Decimals and Binary:

(10, 15, 16, ..) (0, 1)

1) Decimal to Binary:

i) 10 = 1010 ? How

ii) 1st Approach

~~n=10~~

→ divide by 2

→ store remainder in arr.

→ repeat above two steps until n!=0

→ reverse arr.

n = 10

divisor = 2

Division	remainder
10/2 → 5	0
5/2 → 2	1
2/2 → 1	0
1/2 → 0	1

= 0101

= now reverse,

So, 1010 becomes which is value of 10 in binary

Date
DELTA Pg No.

n = 7

divisor = 2

Division	remainder
7/2 → 3	1
3/2 → 1	1
1/2 → 0	1

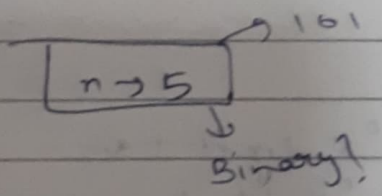
= 111

Now rev ans

⇒ 111 = 7

$2^2+2^1+2^0 = 4+2+1 = 7$

2nd Approach:



num & 1

1 → odd

0 → even

x x x & 1 → if x = 1 → 1

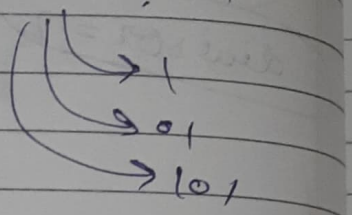
x = 0 → 0

n = 5

```
n != 0
{ bit = n & 1;
  n = n >> 1;
}
```

```
while (n!=0)
{ bit = n&1;
  n = n>>1;
}
```

answer = ? 101




```
int ans=0;
ans = (10^0 x digit) + ans
    = 10^0 x 1 + 0
    = 1
ans = (10^1 x digit) + ans
    = 10 x 0 + 1
    = 1
ans = (10^2 x digit) + ans
    = 100 x 1 + 1
    = 101.
```

FOR 123

```
ans = 10^0 x digit + ans
    = 10^0 x 1 + 0
    = 1
ans = 10^1 x 2 + 1
    = 21
ans = 10^2 x 3 + 21
    = 321
```

$$ans = digit \times 10^{i} + ans$$

(makes reverse)

For straight order,

$$ans = ans \times 10^{i} + digit$$

Date
DELTA Pg No

Binary to Decimal:

10101
$2^4 2^3 2^2 2^1 2^0$

$2^4 + 2^2 + 2^1 = 21$

Div	Rem
$21/2 \to 10$	1
$10/2 \to 5$	0
$5/2 \to 2$	1
$2/2 \to 1$	0
$1/2 \to 0$	1

21 | 2) 10101

```
int ans=0, i=0.
n = 110
while (n!=0)
{
    int digit = n%10
    if(digit==1) {
        ans = ans + pow(2,i)
        i++
        n = n/10.
    }
}
cout<<ans;
```

Switch statements:

```
switch (exp.)
{
    case 1:
        break;
```

→ can be int / char
cannot be float / string.

→ (Yes / No)

Date
DELTA Pg No.

case 2 const 2 :

break ; → Yes / No.

default :

not mandatory.

→ Bulky Code
→ Baggy code
→ Not readable code

Arrays

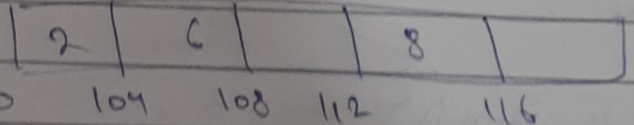
→ Similar type of items
→ contiguous location.
→ "index" : access.

Why array : 10000 values → 10000 variables
Difficult
Easy
variable

Declaration :

→ int a[10];

→ int v[5]



Date
DELTA Pg No.

v[0] → 1st location.

↳ 2.

cout << v[0]

→ 2.

v[1] → 100 + 1×4 = 104

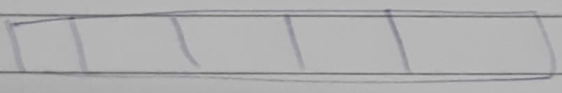
cout << v[1] → 6.

v[3] → 112

cout << v[3] = 8

Accessing:

int [a[5]



5th location → a[4].

array (n size.

↓

index

↓↓

0 - (n-1)

Initialize:

int n[3] = {5, 7, 11};

int a[100000] = {0};

↓

[0|0|0|0|0|0|0|0|0|0] → Everywhere 0. But we can do this with two ways.

int a[100000];

[| | | | | | | | | |] → Garbage value everywhere