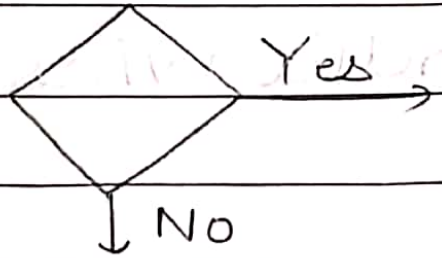
29/12023

Conditionals

This is similar to the decision making box that we studied in flowcharts.



if statement

if (condition) {

}

If the condition is true, then execute the line of code within the scope of if block.

Ex →
```
if (marks > 95) {          ← condition true, then A is printed
    cout << "A";
}
```

Ques Predict the output of following code

```
if (score < 300) {
    cout << "India wins";
}
cout << "Pak wins";
}
```

(i) score = 323 Pak wins

(ii) score = 123 India wins Pak wins
(no new line)

Note → If condition is false, then the line just next to the scope of if will be executed.

But this looks illogical & hence if else can be used.

```
if else
if (score < 300) {
    cout << "India wins";
}
else {
    cout << "Pak wins";
}
```

The control will go inside the else block

only when condition is false.

(i) condition → score = 323 Pak wins
(ii) condition → score = 123 India wins

What to do if there are multiple conditions.
We will be using if else if else statements
Other solution can be the nested if else.

```
if (cond1) {      }
else {
    if (cond2) {      }
    else {      }
}
```

one within another → Nested if else for multiple conditions.

But more readable will be using if else if and else blocks together.

```
if else if else
1. if (cond1) {      }
2. else if (cond2) {      }
3. else {      }
4.
```

Flow of above code execute the block &
✓ If cond1 is true, then go to line 4 but if cond1 is false, then check for cond2.

✓ If cond2 is true, then execute the block & go to line 4 but if cond2 is false, then we will have to execute the else block.

Note → We don't have to write any condition with the else block

We have to write the condition with if and else if blocks.

Also we have to remember, else block is optional as if we don't write it then we won't be getting any syntax error.

Note
```
if ( ) {      }
else if ( ) {   }   } Can be multiple else if
else if ( ) {   }   } blocks.
else {        } → Optional
```

Also there can be multiple if blocks.

Loops

We have various types of loops such as for loop, while loop, do while loop and for each loop but for now we will be studying for loop.

Loops are used when we want to do same task multiple times.
Ex → Printing name 5 times, counting from 1 to 5 etc.

for loop syntax

initialization ↑ condition ↑ updation (increment) ↑

```
for (int i=0 ; i < 5 ; i=i+1) {
        // Scope of for loop
}
```

Flow of for loop

1) The initialization of variable takes place.

2) The condition will be checked & if the condition is true then flow goes inside for loop.
3) After execution, updation of variable takes place & then if the condition is true, then only flow goes inside for loop.
4) This will go on until the condition becomes false

Also note that there will be only one time initialization.

Ques → Predict the o/p of below code

```
for (int i=0 ; i<3 ; i++) {
        cout << i ;
}
```

Output
0 1 2

Note → Updation can be done in any way say

```
i = i+2        i = i-2
i = 2 * i      i = i/2
```

There are many other ways of updation as well.

Ques → Predict the o/p of below code

```
for (int i = 5 ; (i>=0 && i<=10) ; i=i+1) {
        cout << i ;
}
```

Output
5 6 7 8 9 10

From the above question, we get to know

that there can be many conditions.

Ques→ Out of initialization, updation & condition, which of them are optional & which are mandatory ?

Initialization
Condition
Updation
} All are optional

```
for ( ; ; ) {
    if (i<5) {
        cout << i;
        i = i+1;
    }
}
```

} This will not give any syntax error.

Ques→ What will be o/p of following code?

(i)
```
int n;
if (cin >> n) {
    cout << "Bhavya";
}
```

This will always print Bhavya for all values.

(ii)
```
int n;    cin>>n;
if (cout<<n) {
    cout <<"Bhavya";
}
```

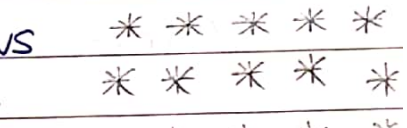
This will first print the value of n & then print Bhavya for all value be it -ve, +ve or 0.

Patterns

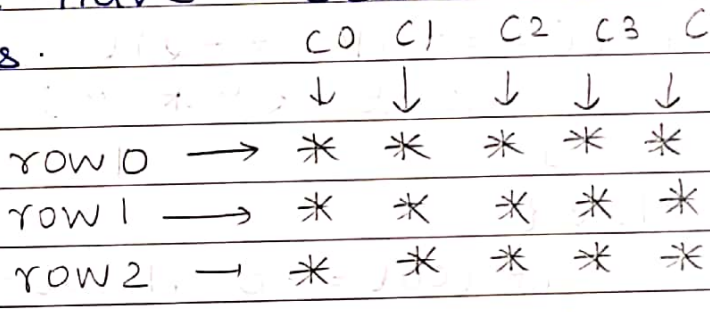
These provide warm up for logic building & will improve our concept of loops.

Steps to print patterns

1) First observe the number of rows in the pattern. These have 3 rows.



2) Now we have to observe the no. of columns & build a relation blw no. of rows and no. of columns. In this pattern, every column have 3 stars and there are 5 columns.



Patterns are mostly coded using nested loops. There will be outer & inner for loop. Outer for loop will be for rows & inner loop will be for columns.

Code

```
for (int row = 0 ; row < 3 ; row = row + 1) {
    for (int col = 0 ; col < 5 ; col = col + 1) {
        cout << "*" ;
    }
    cout << endl ; → To go to next line
}
```

Note → row = row + 1 is similar to row += 1

Square pattern

In this no. of rows & no. of columns will be same.

Hollow rectangular pattern

→
```
* * * * *
*       *
* * * * *
```
- → spaces

1st and last row → 5 stars
Other rows → 1st space (↓ Star) last (↓ Star)

```
for (int i=0 ; i<3 ; i=i+1) {
    for (int j=0; j<5; j=j+1) {
        if (i==0 || i==2) {
            cout << "* ";
        }
        else {
            if (col==0 || col==4) {
                cout << "* ";
            }
            else {
                cout << "  ";
            }
        }
    }
    cout <<endl;
}
```

Here i is row
j is column

→ Half pyramid pattern

```
*          row-0
* *        row-1
* *        row-2
* * *      row-3
```

Number of stars to print is one more than row number.
Also no. of stars are equal to the no. of columns in that particular row.

```
for (int row = 0 ; row < 4 ; row = row + 1) {
    for (int col = 0 ; col < row + 1 ; col++) {
        cout << " * " ;
    }
    cout << endl;
}
```

Discussed the logic above.

→ Inverted (half) pyramid pattern

```
* * * *    → row-0
* * *      → row-1
* *        → row-2
*          → row-3
```

In the code of previous code we just have to change condition in inner for loop i.e

```
col < n - row
```

n → no. of rows. In this question n = 4

→ Numeric half pyramid pattern

```
1                  row-0 → 0+1 count
1 2                row-1 → 1+1 = 2 count
1 2 3              row-2 → 2+1 = 3 count
1 2 3 4            row-3 → 3+1 = 4 count
↑ col0 ↑ col1 ↑ col2 ↑ col3
```

In the half pyramid pattern just change
cout << "*" to
cout << col+1;

→ Numeric inverted half pyramid

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

In the inverted half pyramid pattern just change cout << "*" to
cout << col+1;