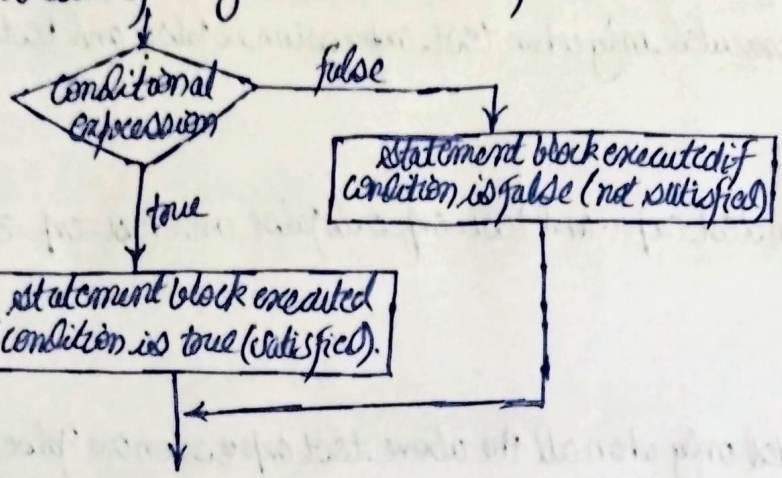
Conditionals and loops:

Conditionals are used to execute a certain section of code only if some specific condition is fulfilled, and optionally execute other statements if given condition is false.

The result of the given conditional expression must be either true or false.



- So, Different variations of this conditional statement are :-
  - (i) if statement
  - (ii) if-else statement
  - (iii) if-else-if
  - (iv) Nested if statement

(i) if-statement :-

if statement evaluates the given test expression. If it is evaluated to true, then statement inside the if block will be executed. Otherwise, statements inside if block is skipped.

Syntax:

```
if(test_expression){
        //statements to be executed only when test_expression is true.
}
```

(ii) if-else statement :-

if statement evaluates the given test expression. If it is evaluated to true, then statements inside the if block will be executed. Otherwise, statements inside else block will be executed. After that, rest of the statements will be executed normally.

Syntax:

```
if (test-expression){
        //statements to be executed when test-expression is true.
}
else{
        //statements to be executed when test-expression is false.
}
```

(iii) if-else-if :-

Using this we can execute statements based on multiple conditions.

Syntax:

```
if(test_expression){
    //statements will be executed only when test-expression is true
}
else if(test_expression_2){
    //statements to be executed only when test_expression is false and test_exp_2 is
    true.
}
else if(test_expression_3){
    //only executed when test_expr. and test_exp_2 is false and test_exp_3 is true.
}
.
.
.
else{
    //statements to be executed only when all the above test expression are false
}
```

Out of all block of statements, only one will be executed based on the given test expression, all others will be skipped. As soon as any expression evaluates to true, that block of statement will be executed and rest will be skipped. If none of the expression evaluates to true, then the statements inside else block will be executed.

(iv) Nested-if statement:-

We can put another if-else statement inside an if.

Syntax:

```
if(test-expression){
    //statements to be executed when test-expression is true.
    if(test_exp_2){
        //statements
    }
    else{
        //statements to be executed only when test-exp-2 is false.
    }
}
```

Ex:

```
int main(){
    int a=15;
    if(a>10){
        if(a>20){
            cout<<"Hello"<<endl;
        }else{
            cout<<"Hi"<<endl;
        }
    }
}
```

Output:
Hi

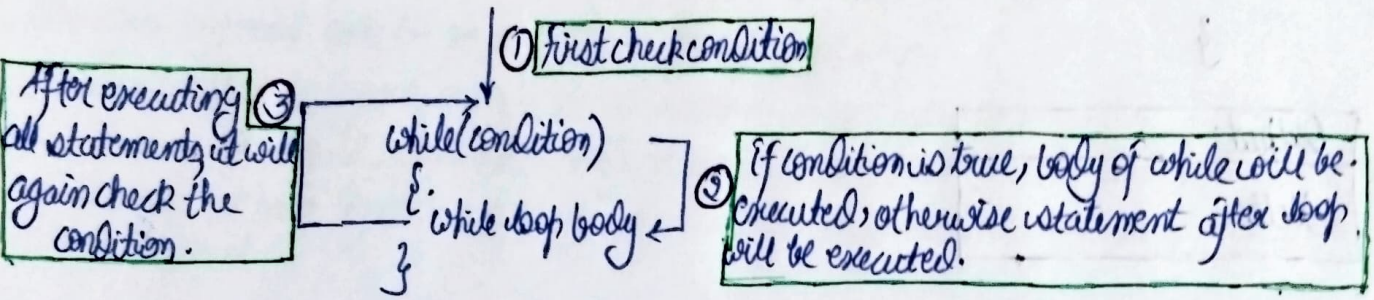
● While loop:

Loop statements allows us to execute a block of statements several no. of times depending on certain condition. while is one kind of loop that we can use.

When executing, if the test expression results is true, then the actions inside the loop will be executed. This will continue as long as expression result is true.

Syntax:-

```
while (test_expression){
    //statements to be executed till test_expression is true.
}
```



Ex:

```
int main() {
    int i=1;
    while(i<=5){
        cout<<i<<endl;
        i++;
    }
}
```

Output
1
2
3
4
5

```
int main(){
    int n;
    cin>>n;
    int i=2;
    bool divided=false;
    while(i<n/2)
        if(n%i==0){
            divided=true;
        }   i++;
    }
    if(!divided){
        cout<<"Prime"<<endl;
    }
    else{
        cout<<"Not Prime"<<endl;
    }
}
```

return keyword:

return is a special keyword, when encountered ends the main. That means, no statement will be executed after the return statement.

Code:-

```
int main(){
    int a=10;
    if(a>5){
        cout<<"Hello"<<endl;
        return 0;
    }
    cout<<"Hi"<<endl;
}
```

* and it also tells to our system that there is no error and successful execution of program

Output:
Hello

Fahrenhite to Celcius = 5.0/9*(cel-32);

Do-while loops:

```
int main(){
    int i=1; //initialization
    do{
        cout<<"Akshat"<<i<<endl;
        i=i+1; //updation
    }
    while(i<=5); //condition
```

* No condition check during 1st iteration in do-while loop means that it will get/be executed at least once whether the condition is true or not.

* Some Questions:-

```
int i;
if(cin>>i){
    cout<<i<<endl;
}
```

Output:-
10
>10

```
if(cout<<"Hi"<<endl){
    cout<<"Hello"<<endl;
}
```

Output:-
Hi
Hello

```
if(a>10); {
    cout<<"Ok"<<endl;
}
```

Output:-
Ok

Patterns:

There are 2 popular types of patterns
- Square Patterns
- Triangular Patterns

Patterns are good application of loops.

Suppose, we want to print the following pattern:

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

→ Here, you can see that we have 4 rows.

→ Now, moving to find the generic no. of columns, it's clear that the 'ith' row (where i=1,2--) has i columns. Hence, the no. of columns for i-th rows is i.

→ Visually, it's also clear that we want to print no. starting from 1 to the row-th no. like for the first row, we print no. starting from 1 and ending at 1, for second row no. are starting from 1 and ending at 2 and so on....

The above approach can be generalized in the following way:-

(i) We start to figure out the no. of rows that our pattern requires.

(ii) Now, we should know how many columns do we have to print in the generic ith row.

(iii) Once, we have figured out the no. of rows and columns, then we should focus on what to print.

Code Example:

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

```
#include<iostream>
using namespace std;
int main(){
    int n;
    cin>>n;
    int i=1;
    while(i<=n){
        int j=1;
        while(j<=n){
            cout<<"*";
            j++;
        }
        cout<<endl;
        i++;
    }
}
```

```
abcde
abcde
abcde
abcde
abcde
```

```
#include<iostream>
using namespace std;
int main(){
    int n;
    cin>>n;
    int i=1;
    while(i<=n){
        int j=1;
        while(j<=n){
            char ch='a'+j-1;
            cout<<ch;
            j++;
        }
        cout<<endl;
        i++;
    }
}
```

```
ABCDE
ABCD
ABC
AB
A
```

```
#include<iostream>
using namespace std;
int main(){
    int n;
    cin>>n;
    int i=1;
    while(i<=n){
        int j=n-i+1;
        ~~char ch='A'+j-1;~~
        while(j>0){
            char ch='A'+j-1;
            cout<<ch;
            j--;
        }
        cout<<endl;
        i++;
    }
}
```

```
55555
45555
34555
23455
12345
```

```
#include<iostream>
using namespace std;
int main(){ int n; i=1;
    cin>>n;
    while(i<=n){
        int j=n-(i+1); l=1
        while(j<=i){
            cout<<j;
            j++;
        } l++;
        int k=n-i;
        while(k>0){
            cout<<n;
            k--;
        }
        cout<<endl;
        i++;
    }
}
```

```
1234 4321
123**321
12****21
1******1
```

```
#include<iostream>
using namespace std;
int main(){
    int n;
    cin>>n;
    int i=1;
    while(i<=n){
        int j=n-i+1;
        int k=1;
        while(j>0){
            cout<<k;
            k++;
        } j--;
        int l=1;
        while(l<=2(i-2)){
            cout<<"*";
        } l++;
        int m=n-l+1;
        while(m>0){
            cout<<m;
        } m--;
        cout<<endl;
        i++;
    }
}
```