CONTROL FLOW

Prerequisites:

1. Basic structure of a C++ program.

DECISION MAKING STATEMENTS

Many a time we need to take decisions depending on situation. Let us take the following example:
You will decide what activity you will do today depending on the weather. If the weather is sunny you will go for a picnic. Otherwise you will stay at home and play indoor games.

How will you write a computer program which takes as input the weather type(“SUNNY” or “RAINY” or “CLOUDY” or “FOGGY”) and output your activity?

Here comes the need of decision making statements. The syntax is as simple as:

```
string weather;
cin>>weather;//Inputting the weather as a String
if(weather==”SUNNY”)
{
        cout<<”Go for a picnic!”;
}
else
{
        cout<<”Stay at home and play indoors”;
}
```

****For the time being you can assume cin>>weather is used to input a string and store it in variable weather. We will go into the details of these in the tutorial on I/O.
****cout<<”Some Text” is used to print Some Text on the screen.

The code is fairly easy to understand isn’t it. Let's read it. If weather is SUNNY print go for a picnic else print stay indoors. Understandable? Refer to the question. It asks us to do exactly this.
But be careful! Observe we write if and not If or IF. This is the syntax! Do not forget the brackets () surrounding the condition. In Python we did not have these surrounding the conditions.
Notice the use of {} . All the statements that are to be executed if the ‘if condition’ is satisfied is put within these {}, curly braces. Remember, in Python how we used to indent those statements one level deeper than the if statement to make the Python Language understand that these are the statements to be executed if the condition is true?

Let us change the activities slightly.

If the weather is SUNNY you go for a picnic, else if the weather is RAINY you play football, else if the weather is CLOUDY you go for a walk else if the weather is FOGGY you sleep.

Let's see the syntax:

```
string weather;
cin>>weather; //Inputting weather
if(weather == "SUNNY")
{
        cout<<"Go for a picnic";
}
else if(weather == "RAINY")
{
        cout<<"Play Football";
}
else if(weather == "CLOUDY")
{
        cout<<"Go for a walk";
}
else if(weather == "FOGGY")
{
        cout<<"Sleep!";
}
```

Isn't this easy to understand as well? If weather is SUNNY do something, else if weather is RAINY do something, else if weather is CLOUDY do something and so on…
Notice the use of else followed by if-condition in order to make different decisions for different conditions.

Now let's say the input is SNOWY or WINDY What do we do? I want you to print "invalid weather". What change do we make to the above code?
This is left as an exercise for you.
Solution: https://ideone.com/876AOa
***It is a good habit to enclose the statements to be executed if the condition is true by {}. Not putting {} will work only if there is one statement to be executed if the if-condition is true. Having multiple statements to be executed if the condition is true calls for the statements to be enclosed by {}. As a golden rule always put the {}.
Let's see an example:

```
if(weather == "SUNNY") //Here you have to put the braces as there are two print statements
{
        cout<<"Go for a picnic";
        cout<<"Have Lunch outside";
}
else if(weather == "RAINY")
```

```
cout<<"Play Football"; //This will work.
else if(weather == "CLOUDY")
cout<<"Go for a walk"; //This will work
else if(weather == "FOGGY")
cout<<"Sleep!"; //This will work
```

***Wondering what is // . Well this helps us to make a comment. Remember how we used to comment in Python using #.

AND OPERATOR:

Many a time we need to check multiple conditions before taking a decision.
Let's take the following problem:
I want you to input a number and print it only if it is divisible by 2 and greater than 45.

In such cases you have to use something called the AND operator(written as && in C++) to join the two conditions. Remember what was this operator in Python? Yes, we used to write 'and', to join conditions . The program goes as follows: https://ideone.com/bGC858
****You can have more than two conditions joined by and(&&) as well.

OR OPERATOR:

We may also have to check a number of conditions and if any one of them is true we have to execute a set of statements.
In such situations we have to use the OR operator(written as || in C++).
Let us take a simple problem:
Input the weather and predict the activity to be done as follows. If the weather is SUNNY or CLOUDY or WINDY, print "Go for a picnic" else print "Stay at home".
Go ahead and try to write your own code using the || operator.
Solution: https://ideone.com/dX7DbZ

NOT OPERATOR

Another problem: Input a number and print if it is not equal to 0.
And it turns out that programming languages offers us something called the not operator(!).
So you can use !=(read as not equal to). Can you write the solution now?
Solution: https://ideone.com/9v8OVp

Exercise:
Can you write the following statement using ! operator?

```
bool ok = true; //This is some variable
if(ok==true)cout<<"Hello!"<<endl; //You have to rewrite the condition in this statement using !
else cout<<"False!"<<endl;
```

LOOPS:

We learnt while loops in Python. In case you are not familiar with loops have a look at this video to get a motivation about loops: https://www.youtube.com/watch?v=mgooqyWMTxk

while loop:

while loop is same as it was in Python with slight difference in syntax.
Here is the general syntax:

```
while(condition)
{
        /* Statements to be repeated*/
}
```

As long as the condition specified in the brackets is true the statements enclosed by {} are repeated. We learnt about counters as well. How these variables keep track of how many times the loop ran or how many times the statements inside {} are repeated.

Let us solve a simple problem to recall the essential components of a loop.
Input N and print the product of the numbers from 1 to N.
In case you are unable to write the solution, refer to this commented code:
https://ideone.com/yt5PqB

***Don't understand endl? It just takes the cursor to the new line when printing on the screen.

You should have got that!

for loop:

There is another type of loop called the for loop. In this loop we initialize the counter, put the condition which as long as holds true the loop keeps on executing the statements enclosed by {}. And we have the update operation of the counter in the same line, at the beginning of the loop. Then we enclose the statements to be repeated with in {}.

The syntax of this loop is as follows:

```
for(declaring and initializing the counter; condition which if true will execute the statements in the loop; Update operation of the counter)
{
        //Statements to be repeated
}
```

Let's convert our previous code using while loops to one which uses for loops.
What did we initialize our cnt to? int cnt =1;
Let's put that in it's place in the for loop syntax:

```
for(int cnt=1;   ;   )
{
        //Statements to be executed
}
```
What was the condition which as long as it was true we wanted to execute the statements in the loop?
Refer to the while loop condition: cnt<=N
Let's put that in place as well:

```
for(int cnt=1; cnt<=N; )
{
        //Some statements to be repeated
}
```
What was the update operation we performed on our counter? We incremented the counter by 1 each time after executing the statements in the loop. So let's put that in place:
```
for(int cnt=1; cnt<=N; cnt++)
{
        //Some statements to be repeated
}
```
Notice the short hand for incrementing a variable by 1. It is cnt++.
Great! Now what are the statements to be repeated. Whatever is in the while loop block? But hang on do we need to put the counter update again? No. The counter update is already being done(in the third parameter).
[The way for loop works is the last parameter(counter update) will always be used to update the counter after each time the loop statements are executed.]
So we put only the 1 statement.
```
for(int cnt=1; cnt<=N; cnt++)
{
        prod = prod*cnt;
}
```
And now again the braces{} are not important in case we have 1 statement. So you can write the above as:
```
for(int cnt=1; cnt<=N; cnt++)
prod = prod*cnt;
```
But in case there are multiple statements it is a compulsion to put the braces.

Think:
What would have happened if we would have put cnt = cnt+1; in the loop statements.
Answer: The counter cnt would have incremented by 2 instead of 1 and as a result we would have ended up calculating prod = 1*3*5*7*.....less than equal to N. In other words we would have calculated the product of all the odd numbers less than equal to N.

So we saw how for loop helps us to take care of the 3 most important elements of a loop in 1 line!