### C++ Control Structures, If Else and Switch-Case Statement:

In this series of our C++ tutorials, we will visualize the control structure, if-else, and switch statements in the C++ language in this lecture. In our last lesson, we discussed the constant, manipulators and operator precedence in C++.

In this C++ tutorial, the topics which we are going to cover today are given below:

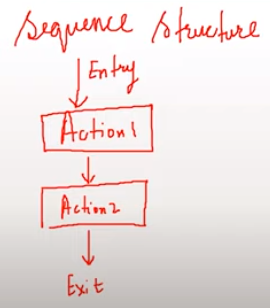
* **Control Structures in C++**
* **IF Else in C++**
* **Switch Statement in C++**

#### Control Structures in C++

The work of control structures is to give flow and logic to a program. There are three types of basic control structures in C++.

##### **Sequence Structure:**

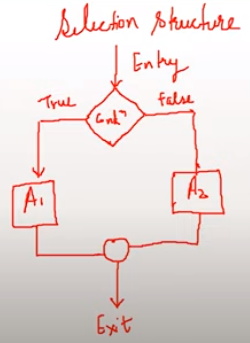
Sequence structure refers to the sequence in which program execute instructions one after another. An example diagram for the sequence structure is shown in figure 1.



***Figure 1: Sequence Structure***

##### **Selection Structure:**

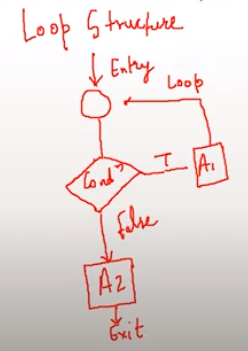
Selection structure refers to the execution of instruction according to the selected condition, which can be either true or false. There are two ways to implement selection structures, by “**if-else statements**” or by “**switch case statements**”. An example diagram for selection structure is shown in figure 2.



***Figure 2: Selection Structure***

##### **Loop Structure:**

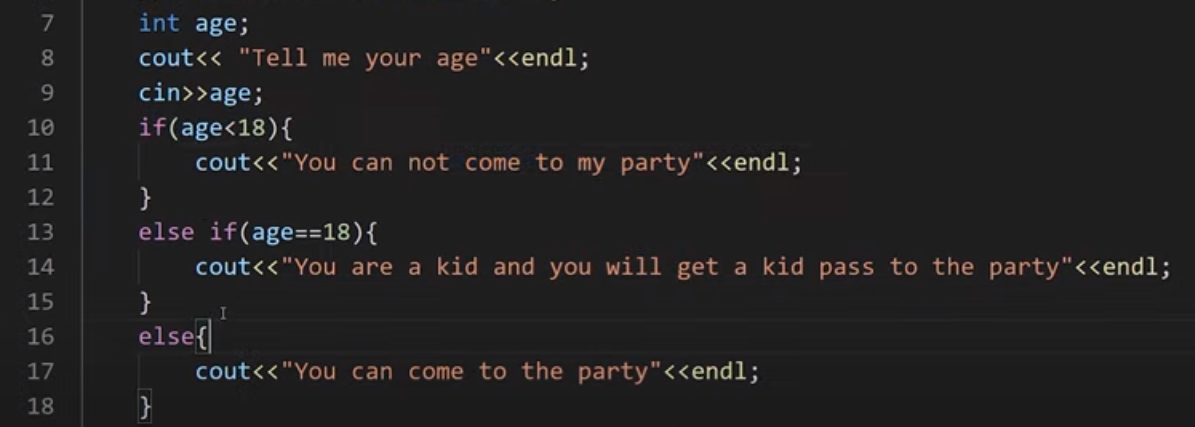
Loop structure refers to the execution of an instruction in a loop until the condition gets false. An example diagram for loop structure is shown in figure 3.



***Figure 3: Loop Structure***

#### If Else Statements in C++:

As we have discussed the concepts of the different control structure, If else statements are used to implement a selection structure. An example program for if-else is shown in figure 4.



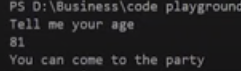
***Figure 4: If-Else program in C++***

As shown in figure 4, we declared a variable “**age**” and used “**cin**" function to gets its value from the user at run time. At line 10 we have used "**if**” statement and give a condition “**(age<18)**” which means that if the age entered by the user is smaller than "**18**” the output will be “**you cannot come to my party**” but if the age is not smaller than “**18**” the compiler will move to the next condition.

At line 13 we have used “**else if**” statement and given another condition “**age==18**" which means that if the age entered by the user is equal to "**18**” the output will be “**you are a kid and you will get a kid pass to the party**” but if the age is not equal to the “**18**” the compiler will move to the next condition.

At line 16 we have used “**else**" condition which means that if none of the above condition is "true" the output will be "**you can come to the party**”.

The output for the following program is shown in figure 5.

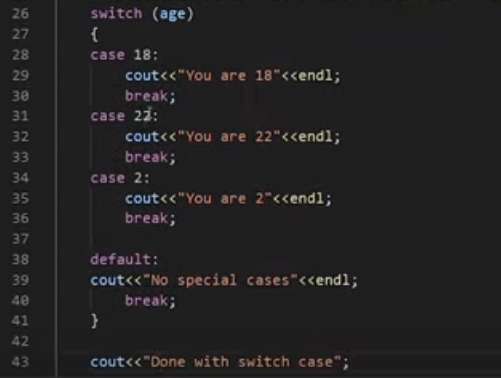


***Figure 5: If-Else Program Output***

As can be seen in figure 5, that when we entered the age "**81**" which was greater than 18, so it gives us the output "**you can come to the party**”. The main thing to note here is that we can use as many “**else if**” statements as we want.

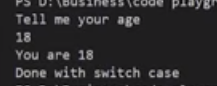
#### Switch Case Statements in C++:

In switch-case statements, the value of the variable is tested with all the cases. An example program for the switch case statement is shown in figure 6.



***Figure 6: Switch Case Statement Program***

As shown in figure 4, we passed a variable “**age**” to the switch statement. The switch statement will compare the value of variable “**age**" with all cases. For example, if the entered value for variable "**age**” is “**18**”, the case with value “**18**” will be executed and prints “**you are 18**”. The keyword “**break**" will let the compiler skips all other cases and goes out of the switch case statement. An output of the following program is shown in figure 6.



***Figure 7: Switch Case Statement Program Output***

As shown in figure 7, we entered the value “**18**” for the variable “**age**", and it gives us an output "**you are 18**” and “**Done with switch case**”. The main thing to note here is that after running the “**case 18**” is skips all the other cases due to the “**break**” statement and printed “**Done with switch case**” which was outside of the switch case statement.

**Code as described/written in the video**

#include<iostream>

using namespace std;

int main(){

// cout<<"This is tutorial 9";

int age;

cout<< "Tell me your age"<<endl;

cin>>age;

// 1. Selection control structure: If else-if else ladder

// if((age<18) && (age>0)){

// cout<<"You can not come to my party"<<endl;

// }

// else if(age==18){

// cout<<"You are a kid and you will get a kid pass to the party"<<endl;

// }

// else if(age<1){

// cout<<"You are not yet born"<<endl;

// }

// else{

// cout<<"You can come to the party"<<endl;

// }

// 2. Selection control structure: Switch Case statements

switch (age)

{

case 18:

cout<<"You are 18"<<endl;

break;

case 22:

cout<<"You are 22"<<endl;

break;

case 2:

cout<<"You are 2"<<endl;

break;

default:

cout<<"No special cases"<<endl;

break;

}

cout<<"Done with switch case";

return 0;

}

### For, While and do-while loops in C++

In this series of our C++ tutorials, we will visualize for loop, while loop, and do-while loop in C++ language in this lecture. In our last lesson, we discussed the control structures, If-else statements, and switch statements in C++.

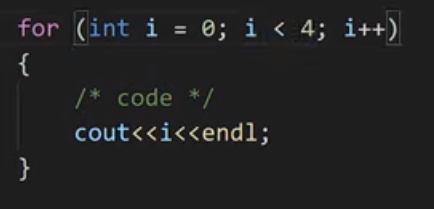
#### Loops in C++:

Loops are block statements, which keeps on repeatedly executing until a specified condition is met. There are three types of loops in C++

* **For loop in C++**
* **While loop in C++**
* **Do While in C++**

#### 1.For Loop in C++

For loop help us to run some specific code repeatedly until the specified condition is met. An example program **for the loop**is shown in figure 1.

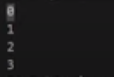


***Figure 1: For Loop Program***

As shown in figure 1, we created **for loop,**and inside its condition, there are three statements separated by a semicolon. The 1st statement is called “**initialization**”, the 2nd statement is called “**condition**”, and the 3rd statement is called “**updation**". After that, there is a loop body in which code is written, which needs to be repeated.  Here is how our for loop will be executed:

* Initialize integer variable “**i**” with value “**0**”
* Check the condition if the value of the variable "**i**” is smaller than “**4**”
* If the condition is true go into loop body and execute the code
* Update the value of “**i**” by one
* Keep repeating this step until the condition gets false

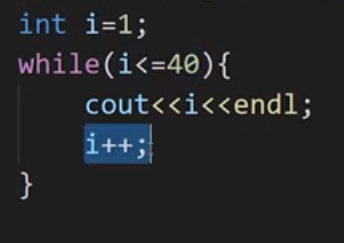
The output for the following program is shown in figure 2.



***Figure 2: For Loop Program Output***

#### 2.While Loop in C++:

While loop helps us to run some specific code repeatedly until the specified condition is met. An example program of **while loop**is shown in figure 3.



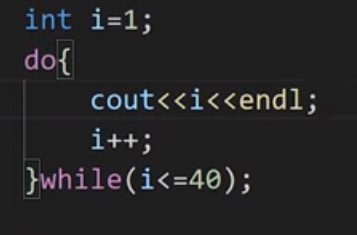
***Figure 3: While Loop Program***

As shown in figure 3, we created **a while loop,**and inside its condition, there is one statement. The statement is called "**condition**”. Here is how our while loop will be executed:

* Initialize integer variable “**i**” with value “**1**”
* Check the condition if the value of the variable "**i**” is smaller or equal to "40."
* If the condition is true to go into loop body and execute the code
* Update the value of “**i**” by one
* Keep repeating this step until the condition gets false.

#### 3.Do-While Loop in C++:

The do-while loop helps us to run some specific code repeatedly until a specified condition is met. An example program of the **do-while loop**is shown in figure 1.



***Figure 4: Do-While Loop Program***

As shown in figure 4, we created a **do-while loop,**and the syntax of the do-while loop is like write body with "**do**” keyword and at the end of body write “**while**" keyword with the condition. Here is how our do-while loop will be executed:

* Initialize integer variable “**i**” with value “**1**”
* Go into loop body and execute the code
* Check the condition if the value of the variable "**i**" is smaller or equal to "**40**”
* If the condition is true - go into loop body and execute the code
* Keep repeating this step until the condition gets false

Code as described/written in the video

#include <iostream>

using namespace std;

int main()

{

/\*Loops in C++:

There are three types of loops in C++:

1. For loop

2. While Loop

3. do-While Loop

\*/

/\*For loop in C++\*/

// int i=1;

// cout<<i;

// i++;

// Syntax for loop

// for(initialization; condition; updation)

// {

// loop body(C++ code);

// }

// for (int i = 1; i <= 40; i++)

// {

// /\* code \*/

// cout<<i<<endl;

// }

// Example of infinite for loop

// for (int i = 1; 34 <= 40; i++)

// {

// /\* code \*/

// cout<<i<<endl;

// }

/\*While loop in C++\*/

// Syntax:

// while(condition)

// {

// C++ statements;

// }

// Printing 1 to 40 using while loop

// int i=1;

// while(i<=40){

// cout<<i<<endl;

// i++;

// }

// Example of infinite while loop

// int i = 1;

// while (true)

// {

// cout << i << endl;

// i++;

// }

/\* do While loop in C++\*/

// Syntax:

// do

// {

// C++ statements;

// }while(condition);

// Printing 1 to 40 using while loop

// int i=1;

// do{

// cout<<i<<endl;

// i++;

// }while(false);

return 0;

}

sss

### Break and Continue Statements in C++

In this series of our C++ tutorials, we will visualize Break and continue statements in C++ language in this lecture. In our last lesson, we discussed for loop, while loop and do-while loop structures in C++.

In this C++ tutorial, the topics which we are going to cover today are given below:

* **Break Statements in C++**
* **Continue Statements in C++**

#### 1.Break Statements:

We had already discussed a little bit about break statements in switch statements. Today we will see the working of break statements in loops. Break statements in loops are used to terminate the loop. An example program for Break's statement is shown in figure 1.

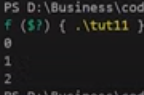


***Figure 1: Break Statement Program***

As shown in figure 1, this is how the break statement program will be executed:

* Initialize integer variable “**i**” with value “**0**”
* Check the condition if the value of the variable "**i**” is smaller than “**40**”
* If the condition is true go into the loop body
* Execute “**cout**” function
* Check the condition if the value of the variable "**i**” is equal to “**2**”, if it is equal terminate the loop and get out of loop body
* Update the value of “**i**” by one
* Keep repeating these steps until the loop condition gets false, or the “if” condition inside the loop body gets true.

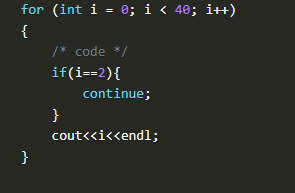
The output of the following program is shown in figure 2.



***Figure 2: Break Statement Program output***

#### 2.Continue Statements in C++:

Continue statements are somewhat similar to break statements. The main difference is that the break statement entirely terminates the loop, but the continue statement only terminates the current iteration. An example program for continue statements is shown in figure 3.



***Figure 3: Continue Statement Program***

As shown in figure 3, this is how the continue statement program will be executed:

* Initialize integer variable “**i**” with value “**0**”
* Check the condition if the value of the variable "**i**” is smaller than “**40**”
* If the condition is true go into the loop body
* Check the condition if the value of the variable "**i**” is equal to “**2**", if it is equal terminate the loop for the current iteration and go to the next iteration
* Execute “**cout**” function
* Update the value of “**i**” by one
* Keep repeating these steps until the loop condition gets false.

#### Code as described/written in the video

#include<iostream>

using namespace std;

int main(){

// for (int i = 0; i < 40; i++)

// {

// /\* code \*/

// if(i==2){

// break;

// }

// cout<<i<<endl;

// }

for (int i = 0; i < 40; i++)

{

/\* code \*/

if(i==2){

continue;

}

cout<<i<<endl;

}

return 0;

}