C++ Course Notes : Fundamentals of Programming 1 - Week 2

Hello everyone welcome to the weekly lecture notes.

Topics to be covered:

1. If statement
2. If-else statement
3. Nested if-else statement
4. Conditional operators
5. Switch statement
6. Introduction to Iterative statements/Loops
7. The for loop
8. The while loop
9. The do-while loop

If statement

1. An if statement is the most common conditional statement. It executes when the condition being checked evaluates to true.
2. Syntax:

```
if(condition) {
  //statements
}
```

3. Example:

```
if(x > 20) {
  cout << "Value of x is greater than 20\n";
}
```

Output:

CASE 1: Let value of x be 30

`Value of x is greater than 20`

CASE 2: Let value of x be 10

No output in this case

If-else statement

1. An if-else statement is designed to give us this functionality in our code. It executes statements if some condition is true or false. If the condition is true, the if part is executed, otherwise the else part of the statement is executed.
2. Syntax:

```
if(condition) {
  //statements
} else {
  // statements
}
```

3. Example:

```
if(x > 20) {
  cout << "Value of x is greater than 20\n";
} else {
  cout << "Value of x is less than 20"
}
```

Output:

CASE 1: Let value of x be 30

`Value of x is greater than 20`

CASE 2: Let value of x be 10

`Value of x is less than 20`

Nested if-else

1. It is simply an if-else statement inside another if-else statement.
2. Syntax:

```
if (condition – 1)
{
  if (condition – 2)
  {
    //statements
  }
  else
  {
  //statements
  }
}
else
{
  //statements
}
```

3. Example:

```
cout << "Value of x is "
if (x > 20)
{
  if(x > 100)
  {
    cout << “very much”;
  }
    cout << “greater\n”;
}
else
{
    cout << “smaller\n”;
}
```

Output:

CASE 1: Let value of x be 150

```
Value of x is very much greater than 20
```

CASE 2: Let value of x be 50

```
Value of x is greater than 20
```

CASE 3: Let value of x be 10

```
Value of x is less than 20
```

Conditional operators

1. These operators are used when a condition comprises of more than one Boolean expression/ condition check.
2. We have following types of conditional operators - logical-and, logical-or and ternary operator.

1. **Logical-and operator (&&):** It is used when we want the condition to be true if both the expressions are true.

   Syntax:

```
if(condition – 1 && condition – 2)
{
  // statements
}
```

   Example:

```
if (val >= 10 && val <= 20)
{
  cout << "Value is in range 10 to 20."
}
```

   Case - 1: val = 3

   Output: No output

   Case - 2: val = 11

   Output: `Value is in range 10 to 20.`

   Case - 3: val = 40

   Output: No output

2. **Logical-or operator (||) :** This operator is used when any one of the Boolean expressions is evaluated as true.

Syntax:

```
if(condition - 1 || condition - 2)
{
    statement;
}
```

Example:

```
if (val < 10 || val > 20)
{
    cout << "Value is either greater than 20 or less than 10."
}
```

Case - 1: val = 3

Output: `Value is either greater than 20 or less than 10.`

Case - 2: val = 40

Output: `Value is either greater than 20 or less than 10.`

Case - 3: val = 11

Output: No output

3. **Ternary operator (?:) :** It is a smaller version for the if-else statement. If the condition is true then statement - 1 is executed else the statement - 2 is executed.

   Syntax:

   ```
   condition ? statement - 1 : statement - 2;
   ```

   Example:

```
//Without ternary operator

if (val == 10)
{
    cout << "The current value is 10\n";
}
else
{
    cout << "The current value is not 10\n";
}



//With ternary operator

val == 10 ? cout << "The current value is 10\n" : cout << "The current value is not 10\n";
```

Case - 1: val = 10

Output: `The current value is 10`

Case - 2: val = 20

Output: `The current value is not 10`

NOTE : “and” can also be written instead of “&&” and “or” can also be written as “||”.

Switch statement

1. Switch Statement is like an if-else ladder with multiple conditions, where we check for equality of a variable with different values.
2. **Syntax:**

```
switch (expression)
{
case x:
    // code
    break;
case y:
    // code
    break;
.
.
.
default:
    // code
}
```

3. Example:

```
switch (ch) {
case 'a':
    cout << "Vowel\n";
    break;
case 'e':
    cout << "Vowel\n";
    break;
case 'i':
    cout << "Vowel\n";
    break;
case 'o':
    cout << "Vowel\n";
    break;
case 'u':
    cout << "Vowel\n";
    break;
default:
    cout << "Consonant\n";
}
```

Output:

Case - 1: ch = 'a'

Output - Vowel

Case - 2: ch = 'x'

Output - Consonant

Introduction to Iterative statements/Loops

The simple dictionary meaning of loop is a structure, series, or process, the end of which is connected to the beginning. We have the following types of iterative statements -

1. The while loop
2. The for loop
3. The do-while loop

The For Loop

1. The advantage of a for loop is we know exactly how many times the loop will execute even before the actual loop starts executing.
2. **Syntax:**

```
for (init-statement; condition; final-expression)
{
statement
// logic
}
```

   1. **Init-statement:** This statement is used to initialize or assign a starting value to a variable which may be altered over the course of loop (we will see this while solving examples). It is used/referred only once at the start of the loop.
   2. **Condition:** This condition serves as a loop control statement. The loop block is executed until the condition evaluates to true.
   3. **Final-expression:** It is evaluated after each iteration of the loop. It is generally to update the values of the loop variables.

3. **Example:**

```
for(int i = 1; i < 6; i++ ){
  cout << i << " ";
}
```

Output:

```
1 2 3 4 5
```

The while loop

1. A while loop is a loop that runs through its body, known as a while statement, as long as a predetermined condition is evaluated as true.
2. **Syntax:**

```
while (condition)
    statement;
```

3. **Example:**

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

Output:

```
1 2 3 4 5
```

The do-while loop

1. Do-while loop tests the condition at the end of each execution for the next iteration. In other words, the loop is executed at least once before the condition is checked. Rest everything is the same as while loop.
2. **Syntax:**

```
do
{
    statement;
} while (condition);
```

3. **Example:**

```
int idx = 1;
do
{
    cout << idx << " ";
  idx++;
} while (idx <= 5);
```

Output:

```
1 2 3 4 5
```
