

# DAILY ASSESSMENT FORMAT

Date:	22-06-2020	Name:	Rajeshwari Gadagi
Course:	C++ Programming	USN:	4AL17EC076
Topic:	Module 1	Semester & Section:	6th SEM & 'B' SEC
Github Repository:	Rajeshwari-gadagi		

## FORENOON SESSION DETAILS

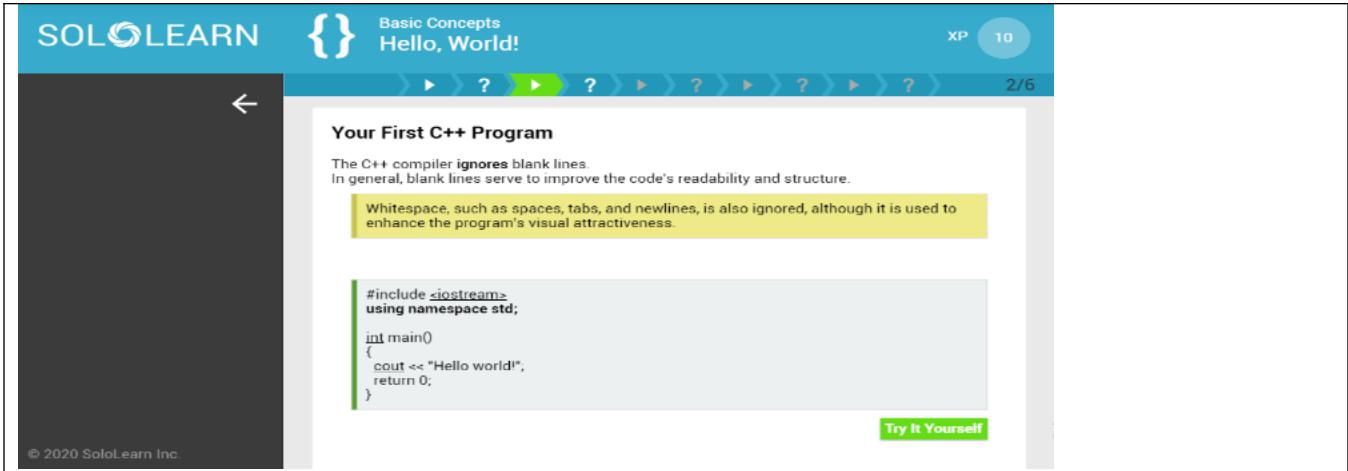
### Image of session

The screenshot shows a mobile application interface for SoloLearn. At the top, it says "Basic Concepts" and "What is C++". On the right, there's an "XP" button with a value of 0 and a progress bar showing 1/1. Below the title, the text "Welcome to C++" is displayed. It states: "C++ is a general-purpose programming language. C++ is used to create computer programs. Anything from art applications, music players and even video games!" A yellow callout box highlights the sentence "C++ was derived from C, and is largely based on it.". Below this, there are 1733 comments and a green "Q&A" button. The bottom left corner shows the copyright notice "© 2020 SoloLearn Inc."

The screenshot shows a mobile application interface for SoloLearn. At the top, it says "Basic Concepts" and "Hello, World!". On the right, there's an "XP" button with a value of 10 and a progress bar showing 1/6. Below the title, the text "Your First C++ Program" is displayed. It states: "A C++ program is a collection of commands or statements. Below is a simple code that has "Hello world!" as its output." A code block shows the following C++ code:

```
#include <iostream>
using namespace std;
int main()
{
    cout << "Hello world!";
    return 0;
}
```

A green "Try It Yourself" button is located at the bottom right of the code block. The bottom left corner shows the copyright notice "© 2020 SoloLearn Inc."

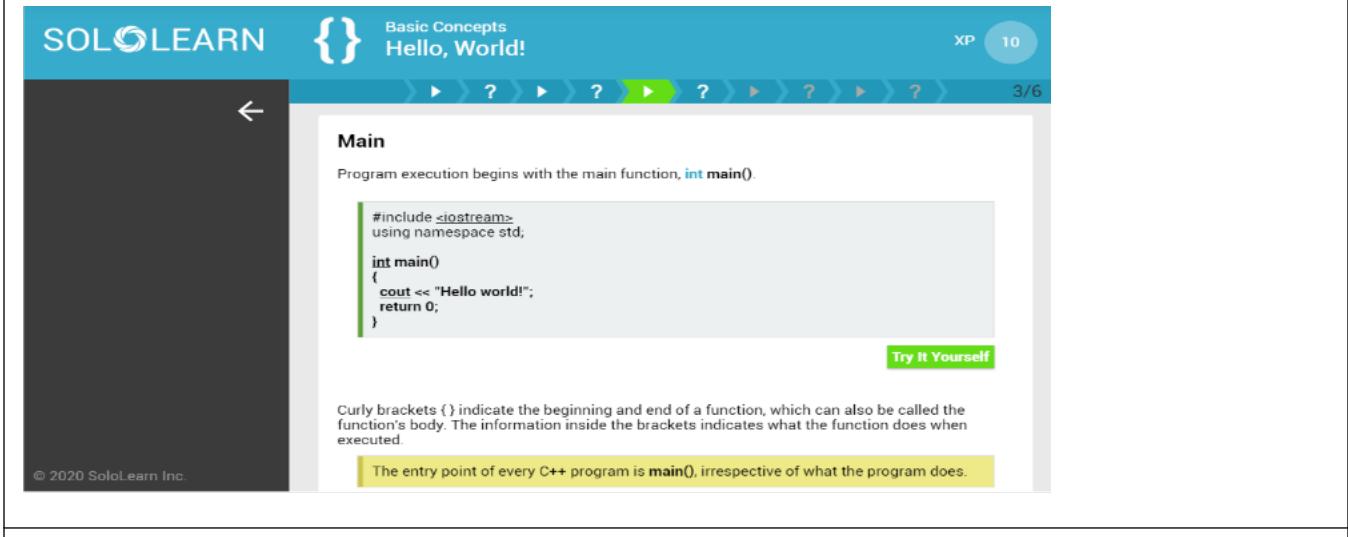


The screenshot shows the second page of a C++ Hello, World! tutorial on SoloLearn. The top navigation bar includes the SoloLearn logo, course name "Basic Concepts", and title "Hello, World!". A progress bar indicates 2/6 steps completed. The main content area is titled "Your First C++ Program". It explains that the C++ compiler ignores blank lines and that whitespace is also ignored. Below this is a code editor window containing the following C++ code:

```
#include <iostream>
using namespace std;

int main()
{
    cout << "Hello world!";
    return 0;
}
```

A green "Try It Yourself" button is located at the bottom right of the code editor.



The screenshot shows the third page of the same C++ Hello, World! tutorial. The top navigation bar remains the same. The main content area is titled "Main". It states that program execution begins with the `int main()` function. Below this is another code editor window with the same C++ code as the previous page.

```
#include <iostream>
using namespace std;

int main()
{
    cout << "Hello world!";
    return 0;
}
```

A green "Try It Yourself" button is located at the bottom right of the code editor. A note below the code explains that curly brackets {} indicate the beginning and end of a function body. A yellow callout box at the bottom states: "The entry point of every C++ program is `main()`, irrespective of what the program does."

## C++ Programming

Monday: 22/06/2020

### Module 1: Basic Concepts

#### > What is C++?

C++ is a general-purpose programming language. C++ is used to create complex programs. Anything from art applications, music players & even video games!

#### > Hello World!

\* You just C++ is a collection of commands or statements.

#### \* The C++ compiler ignores blank lines.

In general, blank lines help to improve the code readability & effectiveness.

#### > Main

Program execution begins with the main function, int main().

The next line, cout < "Hello World!" results in the display of "Hello World!" to the screen.

#### > Statements

A block is a set of logically connected statements surrounded by opening & closing curly braces.

#### > Return

The last instruction in the program is the return statement. The last command, terminate the main() function & cause it to return the value to the calling process. A non-zero value signals abnormal termination.

#### > Getting the tools

You can run, save & share your C++ code on our Code playground without installing any additional software.

#### > Printing @ Text

##### \* You just C++ Program

You can add multiple insertion operators after cout.

##### \* Newline

The endl operator does not insert a line break at the end of the output. One way to force this is to use the shift key, which will put in a new line.

##### \* Newline

The new line character \n can be used as an alternative to endl.

The backslash(\) is called an escape character, indicates a "special character".

\* Two neutral characters placed together with in a blank line.

\* Multiple line lines

### > Comments

Comments are explanatory statements that you can include in the code to explain what the code is doing.

A comment beginning with `//` is called a single-line comment. The compiler will ignore everything that follows, until the end of the line.

### \* Multi-line Comments

Comments that require multiple lines begin with `/*` and end with `*/`.

### \* Using Comments

Comments can be written anywhere, for example repeated every 10 lines throughout the code.

### > Variables

Integers, built-in type equivalents or variables with user-defined values. If you integer using keyword `int`.

A variable can be used to assign a value & can be used to perform operations.

### > Working with variables

- Declaring variables.
- User Input.
- Accepting User Input.

### > More on variables

A variable's value may be changed at any time necessary throughout the program.

### > Arithmetic Operators

Operator	Symbol	Form
Addition	+	$x + y$
Subtraction	-	$x - y$
Multiplication	*	$x * y$
Division	/	$x / y$
Modulus	%	$x \% y$

### \* Operator precedence

Operator precedence determines the grouping of terms in an expression, which affects how calculations are evaluated.

### > Assignment and Increment Operators

The simple assignment operator (`=`) assigns the right side to the left side.

### \* Increment Operator

The increment operator is used to increase an integer's value by one, for a commonly used C++ operator.

The increment operator has two forms for programming:

• Post-increment: increments the value of the variable after the expression.

• Pre-increment: increments the value of the variable before the expression.

\* Decrement Operator: `(--)` it decreases the value by one.

## SOLOLEARN

COURSES

CODE PLAYGROUND

DISCUSS

TOP LEARNERS

BLOG

MY CODES(76)

SHARE

```
C++  
1 #include <iostream>  
2 using namespace std;  
3  
4 int main()  
5 {  
6     cout << "Hello world! \n";  
7     cout << "I love programming!";  
8     return 0;  
9 }
```

Dark Light C++

Output

Hello world!  
I love programming!

^ 0 Hello\_world Public

SAVE

SAVE AS

RESET

▶ RUN

GET IT ON  
Google play

Download on the  
App Store

The screenshot shows the SoloLearn platform's code playground interface. At the top, there are navigation links: COURSES, CODE PLAYGROUND (which is highlighted in green), DISCUSS, TOP LEARNERS, BLOG, and MY CODES(77). Below this, there are tabs for C++, Dark mode, Light mode, and Output. The Output panel displays the number '10'. The main area contains a C++ code editor with the following code:

```
1 #include <iostream>
2 using namespace std;
3
4 int main()
5 {
6     int myVariable = 10;
7     cout << myVariable;
8     return 0;
9 }
```

Below the code editor, there are buttons for SAVE, SAVE AS, RESET, and RUN (highlighted in green). To the left of the code editor, there is a user profile icon and the text 'variable Public'. At the bottom right, there are download links for Google Play and the App Store.

Date:	22-06-2020	Name:	Rajeshwari Gadagi
Course:	C++ Programming	USN:	4AL17EC076
Topic:	Module 2	Semester & Section:	6th SEM & 'B' SEC
Github Repository:	Rajeshwari-gadagi		

AFTERNOON SESSION DETAILS	
Image of session	

The if Statement

Conditionals and Loops

XP 75

1/5

Decision Making

The if statement is used to execute some code if a condition is true.

Syntax:

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

The **condition** specifies which expression is to be evaluated. If the condition is true, the statements in the curly brackets are executed.

If the condition is **false**, the statements are simply ignored, and the program continues to run after the if statement's body.

363 COMMENTS

Q&A

© 2020 SoloLearn Inc.

The if Statement

Conditionals and Loops

XP 75

2/5

The if Statement

Use **relational operators** to evaluate conditions.

For example:

```
if (7 > 4) {  
    cout << "Yes";  
}  
  
// Outputs "Yes"
```

Try It Yourself

The if statement evaluates the condition (7>4), finds it to be **true**, and then executes the **cout** statement. If we change the greater operator to a less than operator (7<4), the statement will not be executed and nothing will be printed out.

A condition specified in an if statement does not require a semicolon.

© 2020 SoloLearn Inc.

The if Statement

Conditionals and Loops

XP 75

3/5

Relational Operators

Additional relational operators:

Operator	Description	Example
<code>&gt;=</code>	Greater than or equal to	<code>7 &gt;= 4</code> True
<code>&lt;=</code>	Less than or equal to	<code>7 &lt;= 4</code> False
<code>==</code>	Equal to	<code>7 == 4</code> False
<code>!=</code>	Not equal to	<code>7 != 4</code> True

Example:

```
if (10 == 10) {  
    cout << "Yes";  
}  
  
// Outputs "Yes"
```

© 2020 SoloLearn Inc.

## C++ programming

Day: Monday, 22/06/2020

### Module 1: Conditionals & Loops

#### > The if statement

- This statement is used to execute some code if a condition is true.

Format:

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

- \* Using relational operators to evaluate conditions

- \* Relational Operators

Operator	Description
>	Greater than or equal to
<	Less than or equal to
==	Equal to
!=	Not Equal to

- \* Relational Operator

The not equal operator evaluate the operands determine whether what they are equal. If the operands are not equal, the condition is evaluated as true.

- \* Relational Operator

You can use relational operators to compare variables in their statement.

#### > The else Statement:

And if statement can be followed by an optional else statement, which executes when the condition is false.

Format:

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

```
else
```

```
{ statements }
```

```
}
```

- \* The else statement

#### > Nested if statements

if statements within non-opening statement.

- \* Nested if else statements

C++ provides the possibility of nesting an unlimited number of if / else statements.

- \* The do loop

The do loop statement, a single statement, can be enclosed without enclosing it in curly braces.

#### > Loops

A loop repeatedly executes a set of statements until a predefined condition is satisfied.

do while loop statement executes a target statement as long as a given condition remains true.

else while loop

- > using a while loop
  - Using Statement in Declaration
    - The statement in declaration separator can be used to change values in the loop.
- > try for loop
  - A for loop is a repetition control structure that allows a program to repeatedly execute a loop with a certain number of times.
    - Structure: for (init; condition; increment) { Statement(s); }
- > the do-while loop
  - do { Statement(s); } while (condition);
  - while (c), do...while
    - If the condition evaluated to false, the statements in the do would still run once.
- > the switch statement
  - Multiple conditions
    - Determines which is used to test a variable for equality against multiple values.
- > logical operator
  - The logical operator to combine conditions statements. Returns true or false.

Operator	Name of the operator	form
	AND Operator	if (x & y)
	OR Operator	if (x    y)
!	NOT Operator	if (!x)

### logical NOT

The logical NOT (!) operator works with first a single operator, returning the opposite value. Thus, if a condition is true, the not operator makes it false, & vice versa.

**SOLOLEARN**

COURSES CODE PLAYGROUND DISCUSS TOP LEARNERS BLOG MY CODES(79)

**C++** Dark Light C++ Output SHARE ↗

```
1 #include <iostream>
2 using namespace std;
3
4 int main()
5 {
6     int mark = 90;
7
8     if (mark < 50) {
9         cout << "You failed." << endl;
10    cout << "Sorry" << endl;
11 }
12 else {
13     cout << "Congratulations!" << endl;
14     cout << "You passed." << endl;
15     cout << "You are awesome!" << endl;
16 }
17
18 return 0;
19 }
```

▲ 0 Else Public 🔍

SAVE SAVE AS RESET ⚡ RUN GET IT ON Google play Download on the App Store

Congratulations!
You passed.
You are awesome!

**SOLOLEARN**

COURSES CODE PLAYGROUND DISCUSS TOP LEARNERS BLOG MY CODES(80)

**C++** Dark Light C++ Output SHARE ↗

```
1 #include <iostream>
2 using namespace std;
3
4 int main()
5 {
6     int mark = 100;
7
8     if (mark >= 50) {
9         cout << "You passed." << endl;
10        if (mark == 100) {
11            cout << "Perfect!" << endl;
12        }
13    }
14 else {
15     cout << "You failed." << endl;
16 }
17
18 return 0;
19 }
```

▲ 0 Nested Public 🔍

SAVE SAVE AS RESET ⚡ RUN GET IT ON Google play Download on the App Store

You passed.
Perfect!

