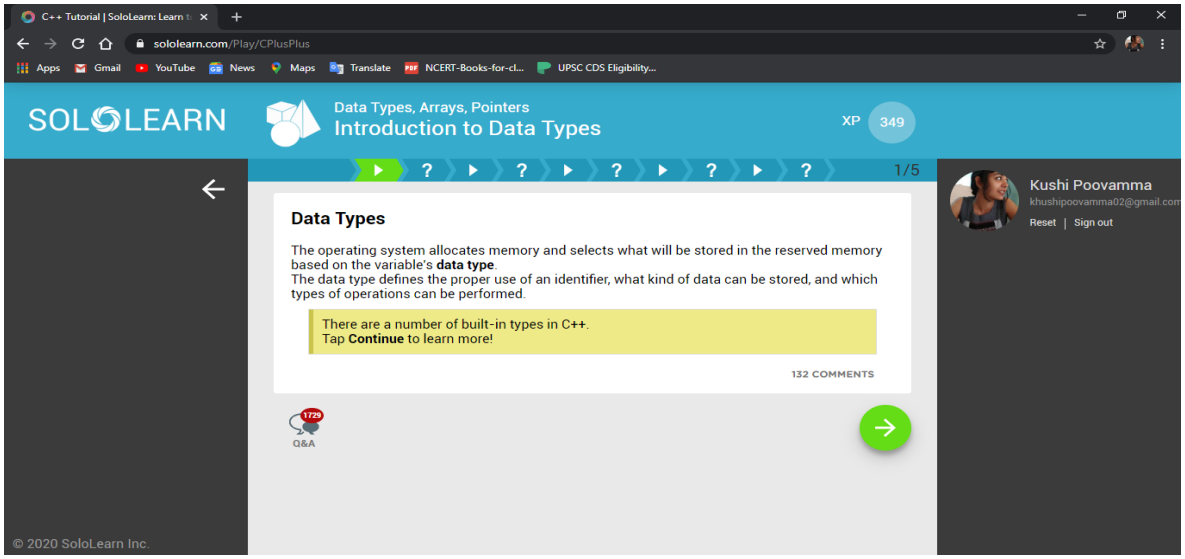
DAILY ASSESSMENT

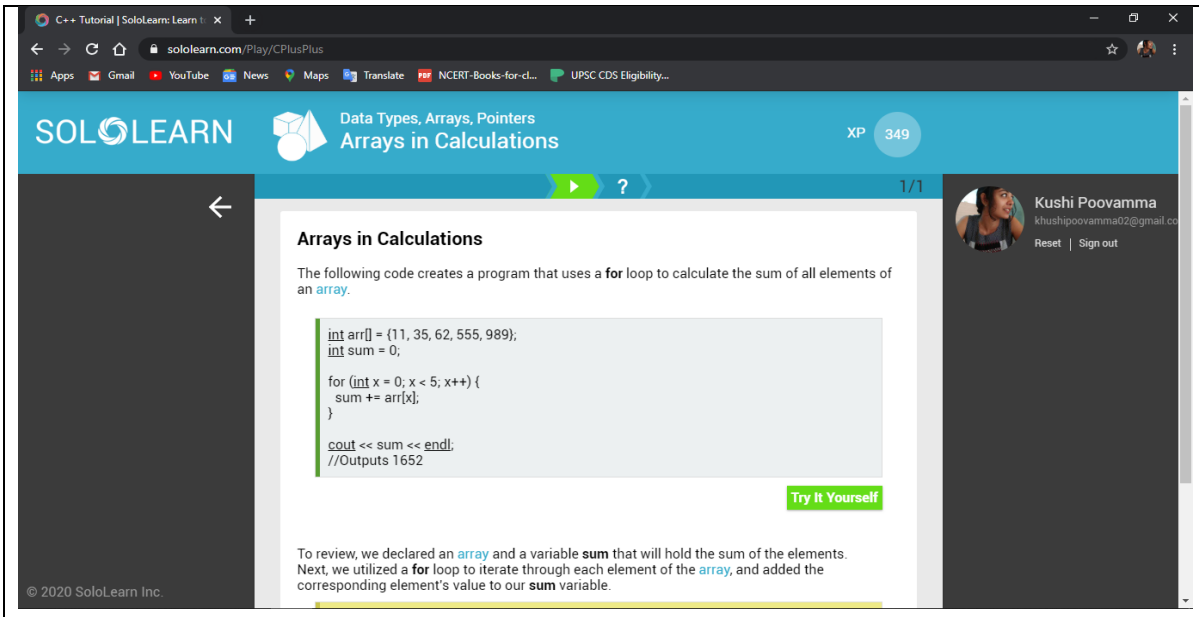
Date:	**23-06-2020**	**Name:**	**K B Kushi**
Course:	**C++ Programming**	**USN:**	**4AL17EC107**
Topic:	**Data Types,Arrays,Pointer Conditionals and Loops**	**Semester & Section:**	**6th & B**
GitHub Repository:	**https://www.github.com/alvas-education-foundation/KUSHI-COURSES.git**		

FORENOON SESSION DETAILS

Image of session

Report:





1. Data Types
The operating system allocates memory and selects what will be stored in the reserved memory based on the variable's data type.
The data type defines the proper use of an identifier, what kind of data can be stored, and which types of operations can be performed.

2.Numeric Data Types
Numeric data types include:
Integers (whole numbers), such as -7, 42.
Floating point numbers, such as 3.14, -42.67.
3. Strings & Characters
A string is composed of numbers, characters, or symbols. String literals are placed in double quotation marks; some examples are "Hello", "My name is David", and similar.
Characters are single letters or symbols, and must be enclosed between single quotes, like 'a', 'b', etc.
4. Booleans
The Boolean data type returns just two possible values: true (1) and false (0).
5.Integers
The integer type holds non-fractional numbers, which can be positive or negative. Examples of integers would include 42, -42, and similar numbers.
Floating Point Numbers
A floating point type variable can hold a real number, such as 420.0, -3.33, or 0.03325.
The words floating point refer to the fact that a varying number of digits can appear before and after the decimal point. You could say that the decimal has the ability to "float".
There are three different floating point data types: float, double, and long double.
In most modern architectures, a float is 4 bytes, a double is 8, and a long double can be equivalent to a double (8 bytes), or 16 bytes.

6. Strings

A string is an ordered sequence of characters, enclosed in double quotation marks.

It is part of the Standard Library.

You need to include the <string> library to use the string data type. Alternatively, you can use a library that includes the string library.

7. Characters

A char variable holds a 1-byte integer. However, instead of interpreting the value of the char as an integer, the value of a char variable is typically interpreted as an ASCII character.

A character is enclosed between single quotes (such as 'a', 'b', etc).

8. Booleans

Boolean variables only have two possible values: true (1) and false (0).

To declare a boolean variable, we use the keyword bool.

9. Variable Naming Rules

Use the following rules when naming variables:

- All variable names must begin with a letter of the alphabet or an underscore(_).

- After the initial letter, variable names can contain additional letters, as well as numbers. Blank spaces or special characters are not allowed in variable names.

Case-Sensitivity

C++ is case-sensitive, which means that an identifier written in uppercase is not equivalent to another one with the same name in lowercase.

For example, *myvariable* is not the same as *MYVARIABLE* and not the same as *MyVariable*.

These are three different variables.

Variable Naming Rules

C++ keyword (reserved word) cannot be used as variable names.

For example, int, float, double, cout cannot be used as a variable name.

10.Pointers

Every variable is a memory location, which has its address defined.

That address can be accessed using the ampersand (&) operator (also called the address-of operator), which denotes an address in memory.

11.Contiditonal statement

Conditional statements, also known as selection statements, are used to make decisions based on a given condition. If the condition evaluates to True, a set of statements is executed, otherwise another set of statements is executed.

The if Statement: The if statement selects and executes the statement(s) based on a given condition. If the condition evaluates to True then a given set of statement(s) is executed. However, if the condition evaluates to False, then the given set of statements is skipped and the program control passes to the statement following the if statement. The syntax of the if statement is

```
if (condition)
{
statement 1;
statement 2;
if (condition)   // if part
```

{

Statement1;

statement2;

}

else ***// else*** **part**

statement3;

Here, the if-else statement comprises two parts, namely, if and else. If the condition is True the if part is executed. However, if the condition is False, the else part is executed.

-
-
-

if(x>y)

cout<<"X is *greater";*

else

cout<<"y is greater";

Nested if-else Statement: A nested if-else statement contains one or more if-else statements. The if else can be nested in three different ways, which are discussed here.

• The if - else statement is nested within the if part.

The syntax is

if (conditionl)

{

statementl;

if(condition2)

statement2;

else

statement3;

}

else

statement4;

• The if-else statement is nested within the else part.

The syntax is

```
if(conditionl)
statement1;
else
{
statement4;
if (condition2)
statement2;
else
statement3;
}
statement5;
```

• The if-else statement is nested within both the if and the else parts.

The syntax is

```
if(conditionl)
{
statementl;
if (condition2)
statement2;
else
statement3;
}
else
{
statement4;
if (condition3)
statement5;
else
statement6;
```

}

statement7;

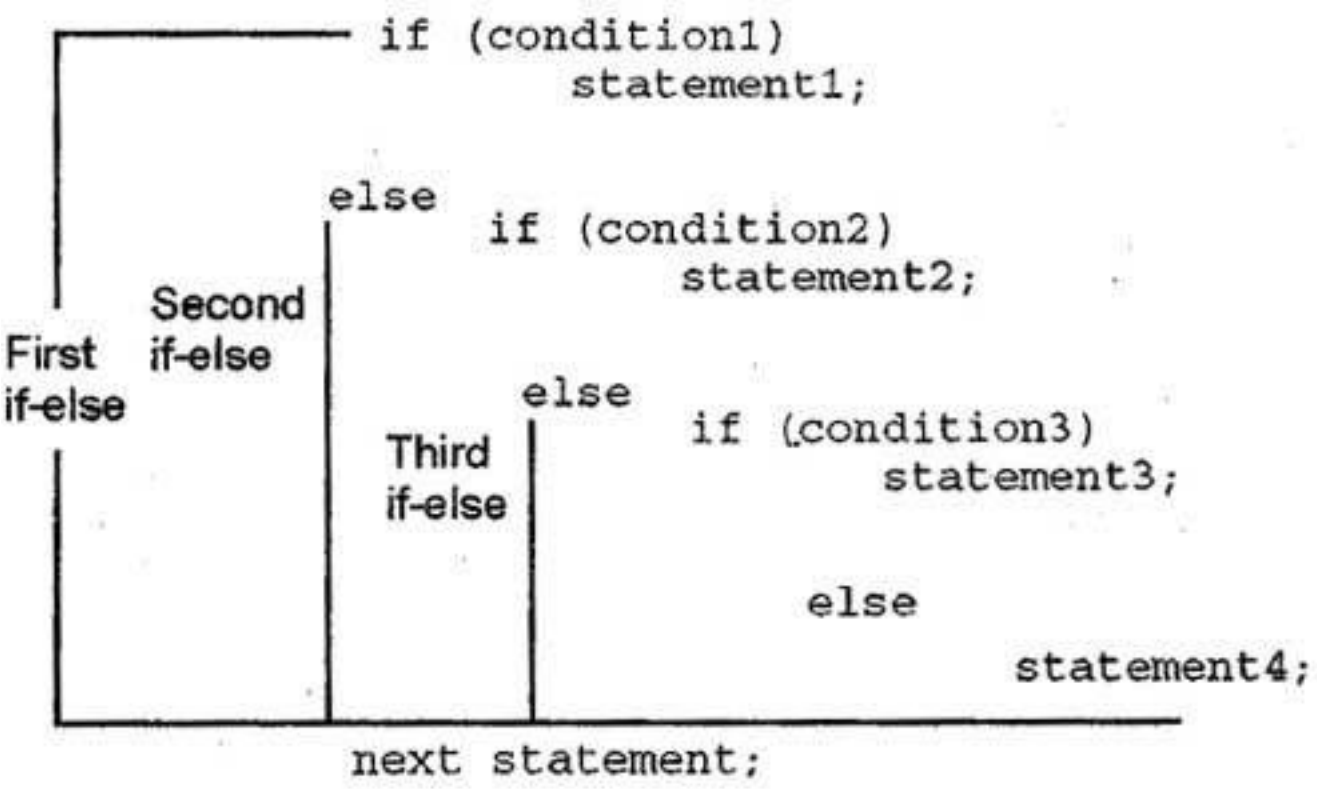
To understand the concept of nested if-else, consider this example.

Example : A code segment to determine the largest of three numbers

```
if (a>b)
{
if (a>c)
cout<<"a is largest";
}
else          //nested if-else statement within else
{
if (b>c)
cout<<"b is largest";
else
cout<<"c is largest";
}
```

The if-else-if ladder, also known as the if-else-if *staircase,* has an if-else statement within the outermost else statement. The inner else statement can further have other if-else statements.

The syntax of the if -e1se- if ladder is



Conditional Operator as an Alternative: The conditional operator '? :' selects one of the two values or expressions based on a given condition. Due to this decision-making nature of the conditional operator, it is sometimes used as an alternative to if-else statements. Note that the conditional operator selects one of the two values or expressions and not the statements as in the case of an if-else statement. In addition, it cannot select more than one value at a time, whereas if-else statement can select and execute more than one statement at a time. For example, consider this statement.

max = (x>y ? x : y)

This statement assigns maximum of x and y to max

The switch Statement: The switch statement selects a set of statements from the available sets of statements. The switch statement tests the value of an expression in a sequence and compares it with the list of integers or character constants. When a match is found, all the statements associated with that constant are executed.

The syntax of the switch statement

switch(expression)

{

case <constant1>: statement1;

[break;]

case <constant2>: statement2;

[break;]

case <constant3>: statement3;

[default: statement4;]

[break;]

}

Statement5;

The C++ keywords case and default provide the list of alternatives. Note that it is not necessary for every case label to specify a unique set of statements. The same set of statements can be shared by multiple case labels.

The keyword default specifies the set of statements to be executed in case no match is found. Note that there can be multiple case labels but there can be only one default label. The break statements in the switch block are optional. However, it is used in the switch block to prevent a fall through. Fall through is a situation that causes the execution of the remaining cases even after a match has been found. In order to prevent this, break statements are used at the end of statements specified by each case and default. This causes the control to immediately break out of the switch block and execute the next statement.

To understand the concept of switch statement, consider this code segment.

Example : A code segme!1t to demonstrate the use of switch statement

```
cin>>x;
int x;
switch(x)
{
case l: cout<<"Option1 is selected";
break;
case 2: cout<<"Option2 is selected";
break;
case 3: cout<<"Option3 is selected";
break;
case 4: cout<<"Option4 is selected;
break;
default: cout<<"Invalid option!";
}
```