**DAILY ASSESSMENT FORMAT**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date:** | **22/06/2020** | **Name:** | **Namratha S Hipparagi** |
| **Course:** | **C++** | **USN:** | **4AL16EC040** |
| **Topic:** | **Basic concepts**  **Conditions and loops** | **Semester & Section:** | **8 A** |
| **Github Repository:** | **namrathahipparagi\_1** |  |  |

|  |
| --- |
| **FORENOON SESSION DETAILS** |
| **Image of session** |
| **Report**  **MODULE 1**  C++ is a high-level programming language developed by Bjarne Stroustrup at Bell Labs. C++ adds object-oriented features to its predecessor, C. C++ is one of the most popular programming language for graphical applications, such as those that run in Windows and Macintosh environments. **New Line** The **cout**operator does not insert a line break at the end of the output. One way to print two lines is to use the **endl**manipulator, which will put in a line break.  #include <iostream> using namespace std;  int main() { cout << "Hello world!" << **endl**; cout << "I love programming!"; return 0; } **Multiple New Lines** Using a single **cout**statement with as many instances of **\n** as your program requires will print out multiple lines of text.  #include <iostream> using namespace std;  int main() { cout << " Hello **\n** world! **\n** I **\n** love **\n** programming!"; return 0; }  **Integer**, a built-in type, represents a whole number value. Define integer using the keyword **int**. C++ requires that you specify the **type**and the **identifier**for each variable defined.An **identifier**is a name for a variable, function, class, module, or any other user-defined item. An identifier starts with a letter (A-Z or a-z) or an underscore (\_), followed by additional letters, underscores, and digits (0 to 9). For example, define a variable called **myVariable**that can hold **integer**values as follows:**int** myVariable = 10;  a program to calculate and print the sum of two integers.  #include <iostream> using namespace std;  int main() { int a = 30; int b = 12; int sum = a + b;  **cout << sum;**  return 0; }  //Outputs 42 **Variables** Creating a variable reserves a memory location, or a space in memory for storing values. The compiler requires that you provide a data type for each variable you declare. **Arithmetic Operators** C++ supports these arithmetic operators. The addition operator adds its operands together.  Int x = 40 + 60; cout << x;  // Outputs **100** **Assignment Operators** The simple assignment operator (=) assigns the right side to the left side.  **For example:** int x = 10; **x += 4**; // equivalent to x = x + 4 **x -= 5**; // equivalent to x = x – 5  **MODULE 2**   **Decision Making** The **if** statement is used to execute some code if a condition is true.  **Syntax:if**(condition){ statements } The conditionspecifies which expression is to be evaluated. If the condition is true, the statements in the curly brackets are executed. The if Statement use relational operators to evaluate conditions.  **For example:**  **if (7 > 4)** { cout << "Yes";  }  // Outputs "Yes"  **The else Statement**  An if statement can be followed by an optional else statement, which executes when the condition is false.  **Syntax:**  **if** (condition) { //statements } **else** { //statements } The code above will test the condition: - If it evaluates to **true**, then the code inside the **if**statement will be executed. - If it evaluates to **false**, then the code inside the **else**statement will be executed.  **Loops**  A loop repeatedly executes a set of statements until a particular condition is satisfied. A while loop statement repeatedly executes a target statement as long as a given condition remains true.  **Syntax:**  **while** (condition) { statement(s); } The loop iterates while the condition is **true**. **The for loop** A forloop is a repetition control structure that allows you to efficiently write a loop that executes a specific number of times.  **Syntax:**  **for** ( **init**; **condition**; **increment** ) { statement(s); }  For example:**for** (int x = 1; x < 10; x++) { // some code } **The do-while Loop** Unlike for and while loops, which test the loop condition at the top of the loop, the do...while loop checks its condition at the bottom of the loop. A do...while loop is similar to a while loop. The one difference is that the do...while loop is guaranteed to execute at least one time.  **Syntax:**  **do** { statement(s); } **while** (condition); **The switch Statement** The switch statement tests a variable against a list of values, which are called cases, to determine whether it is equal to any of them.  switch (expression) { case value1: statement(s); **break;** **case** value2: statement(s); **break;** ... **case** valueN: statement(s); **break;** } |