**DAILY ASSESSMENT FORMAT**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date:** | **22-06-2020** | **Name:** | **M V Ramya** |
| **Course:** | **C++ programming** | **USN:** | **4AL17EC045** |
| **Topic:** | **Module 1:Basic concepts** | **Semester & Section:** | **6th A** |
| **Github Repository:** | **MV-Ramya-045** |  |  |

|  |
| --- |
| **FORENOON SESSION DETAILS** |
| **Image of the session**      **Report –** C++ OOPs Concepts: The major purpose of C++ programming is to introduce the concept of object orientation to the C programming language.Object Oriented Programming is a paradigm that provides many concepts such as **inheritance, data binding, polymorphism etc.**The programming paradigm where everything is represented as an object is known as truly object-oriented programming language. **Smalltalk** is considered as the first truly object-oriented programming language. C++ Basic Input/Output: C++ I/O operation is using the stream concept. Stream is the sequence of bytes or flow of data. It makes the performance fast.If bytes flow from main memory to device like printer, display screen, or a network connection, etc, this is called as **output operation.**If bytes flow from device like printer, display screen, or a network connection, etc to main memory, this is called as **input operation.** I/O Library Header Files: Let us see the common header files used in C++ programming are:  <iostream>It is used to define the **cout, cin and cerr** objects, which correspond to standard output stream, standard input stream and standard error stream, respectively.  <fstream>It is used to declare services for user-controlled file processing. Standard output stream (cout): The **cout** is a predefined object of **ostream** class. It is connected with the standard output device, which is usually a display screen. The cout is used in conjunction with stream insertion operator (<<) to display the output on a console  Let's see the simple example of standard output stream (cout):   1. #include <iostream> 2. using namespace std; 3. int main( ) { 4. char ary[] = "Welcome to C++ tutorial"; 5. cout << "Value of ary is: " << ary << endl; 6. }  C++ Variable: A variable is a name of memory location. It is used to store data. Its value can be changed and it can be reused many times.It is a way to represent memory location through symbol so that it can be easily identified.  Let's see the syntax to declare a variable:  type variable\_list;  example:  int x=5,b=10;  //declaring 2 variable of integer type   1. float f=30.8; 2. char c='A';  C++ Identifiers: C++ identifiers in a program are used to refer to the name of the variables, functions, arrays, or other user-defined data types created by the programmer. They are the basic requirement of any language. Every language has its own rules for naming the identifiers.  Example:  #include <iostream>  using namespace std;  int main()  {      int a;      int A;      cout<<"Enter the values of 'a' and 'A'";      cin>>a;      cin>>A;      cout<<"\nThe values that you have entered are : "<<a<<" , "<<A;      return 0;  }  **Comments**  **Comments** are explanatory statements that you can include in the C++ code to explain what the code is doing.  The compiler ignores everything that appears in the comment, so none of that information shows in the result.   A comment beginning with **two slashes (//)**is called a single-line comment. The slashes tell the compiler to ignore everything that follows, until the end of the line.   **For example:**  #include <iostream> using namespace std;  int main() { **// prints "Hello world"** cout << "Hello world!"; return 0; }  **Multi-Line Comments**  Comments that require multiple lines begin with **/\*** and end with **\*/** You can place them on the same line or insert one or more lines between them.  /\* This is a comment \*/  /\* C++ comments can span multiple lines \*/  **Using Comments**  Comments can be written anywhere, and can be repeated any number of times throughout the code.  Within a comment marked with /\* and \*/, // characters have no special meaning, and vice versa. This allows you to "nest" one comment type within the other.    /\* Comment out printing of Hello world!  cout << "Hello world!"; // prints Hello world!  \*/  **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.  C++ offer a rich assortment of built-in as well as user defined **data types**.  **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;  **Arithmetic Operators**  C++ supports these arithmetic operators.    **Operator Precedence**  Operator **precedence**determines the grouping of terms in an expression, which affects how an expression is evaluated. Certain operators take higher precedence over others; for example, the multiplication operator has higher precedence over the addition operator. **For example:**  int x = 5+2\*2; cout << x; // Outputs **9**  The program above evaluates 2\*2 first, and then adds the result to 5.  As in mathematics, using **parentheses**alters operator precedence.  int x = (5 + 2) \*2; cout << x;  // Outputs 14  **Assignment Operators**  The simple **assignment**operator (=) assigns the right side to the left side.   C++ provides shorthand operators that have the capability of performing an operation and an assignment at the same time.  **For example:**int x = 10; **x += 4**; // equivalent to x = x + 4 **x -= 5**; // equivalent to x = x - 5  The same shorthand syntax applies to the multiplication, division, and modulus operators.x \*= 3; // equivalent to x = x \* 3 x /= 2; // equivalent to x = x / 2 x %= 4; // equivalent to x = x % 4  **Increment Operator**  The **increment**operator is used to increase an integer's value by one, and is a commonly used C++ operator. **x++;** //equivalent to x = x + 1 |
|  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Date:** | **22-06-2020** | **Name:** | **M V Ramya** | |
| **Course:** | **C++ programmimg** | **USN:** | **4AL17EC045** | |
| **Topic:** | **Module 2:conditions and loops** | **Semester & Section:** | **6th A** | |
|  |  |  |  | |
|  |  |  |  | |
|  |  |  |  | |
| **AFTERNOON SESSION DETAILS** | | | |
| **Image of the session**     IF Statement: The C++ if statement tests the condition. It is executed if condition is true.  Syntax:   1. if(condition){ 2. //code to be executed 3. }   Example :   1. #include <iostream> 2. using namespace std; 4. int main () { 5. int num = 10; 6. if (num % 2 == 0) 7. { 8. cout<<"It is even number"; 9. } 10. return 0; 11. }  IF-else Statement: The C++ if-else statement also tests the condition. It executes if block if condition is true otherwise else block is executed.  Syntax:   1. if(condition){ 2. //code if condition is true 3. }else{ 4. //code if condition is false 5. } 6. Example: 7. #include <iostream> 8. using namespace std; 9. int main () { 10. int num = 11; 11. if (num % 2 == 0) 12. { 13. cout<<"It is even number"; 14. } 15. else 16. { 17. cout<<"It is odd number"; 18. } 19. return 0; 20. }  While loop: In C++, while loop is used to iterate a part of the program several times. If the number of iteration is not fixed, it is recommended to use while loop than for loop.  Syntax:   1. while(condition){ 2. //code to be executed 3. } 4. Example: 6. #include <iostream> 7. using namespace std; 8. int main() { 9. int i=1; 10. while(i<=10) 11. { 12. cout<<i <<"\n"; 13. i++; 14. } 15. }  Do-While Loop: The C++ do-while loop is used to iterate a part of the program several times. If the number of iteration is not fixed and you must have to execute the loop at least once, it is recommended to use do-while loop.The C++ do-while loop is executed at least once because condition is checked after loop body.  Syntax:  do{  //code to be executed  }while(condition);  Example:   1. #include <iostream> 2. using namespace std; 3. int main() { 4. int i = 1; 5. do{ 6. cout<<i<<"\n"; 7. i++; 8. } while (i <= 10) ; 9. }  For Loop: The C++ for loop is used to iterate a part of the program several times. If the number of iteration is fixed, it is recommended to use for loop than while or do-while loops.The C++ for loop is same as C/C#. We can initialize variable, check condition and increment/decrement value.  Syntax:   1. for(initialization; condition; incr/decr){ 2. //code to be executed 3. } 4. Example: 5. #include <iostream> 6. using namespace std; 7. int main() { 8. for(int i=1;i<=10;i++){ 9. cout<<i <<"\n"; 10. } 11. } | | | |
|  | | | |