# DAILY ASSESSMENT

|  |  |  |  |
| --- | --- | --- | --- |
| Date: | 22/06/2020 | Name: | Chesmi B R |
| Course: | **C++** | USN: | 4AL16EC100 |
| Topic: | **Module 1: Basic Concepts** | Semester & Section: | 8TH SEM & A Section |
| Github Repository: | chesmibr |  |  |

|  |
| --- |
| **FORENOON SESSION DETAILS**     Report: C++, as we all know is an extension to C language and was developed by Bjarne stroustrup at bell labs. C++ is an intermediate level language, as it comprises a confirmation of both high level and low level language features. C++ is a statically typed, free form, multiparadigm, compiled general-purpose language.  C++ is an Object Oriented Programming language but is not purely Object Oriented. Its features like Friend and Virtual, violate some of the very important OOPS features, rendering this language unworthy of being called completely Object Oriented. Its a middle level language. Benefits of C++ over C Language: The major difference being OOPS concept, C++ is an object oriented language whereas C language is a procedural language. Apart form this there are many other features of C++ which gives this language an upper hand on C laguage.  Following features of C++ makes it a stronger language than C,   1. There is Stronger Type Checking in C++. 2. All the OOPS features in C++ like Abstraction, Encapsulation, Inheritance etc makes it more worthy and useful for programmers. 3. C++ supports and allows user defined operators (i.e Operator Overloading) and function overloading is also supported in it. 4. Exception Handling is there in C++. 5. The Concept of Virtual functions and also Constructors and Destructors for Objects. 6. Inline Functions in C++ instead of Macros in C language. Inline functions make complete function body act like Macro, safely. 7. Variables can be declared anywhere in the program in C++, but must be declared before they are used.   Header files are included at the beginning just like in C program. Here iostream is a header file which provides us with input & output streams. Header files contained predeclared function libraries, which can be used by users for their ease.  Using namespace std, tells the compiler to use standard namespace. Namespace collects identifiers used for class, object and variables. NameSpace can be used by two ways in a program, either by the use of using statement at the beginning, like we did in above mentioned program or by using name of namespace as prefix before the identifier with scope resolution (::) operator.  Example: std::cout << "A";  main(), is the function which holds the executing part of program its return type is int.  cout <<, is used to print anything on screen, same as printf in C language. cin and cout are same as scanf and printf, only difference is that you do not need to mention format specifiers like, %d for int etc, in cout & cin. Comments in C++ Program: For single line comments, use // before mentioning comment  For multiple line comment, enclose the comment between /\* and \*/ Operators in C++ Operators are special type of functions, that takes one or more arguments and produces a new value. For example : addition (+), substraction (-), multiplication (\*) etc, are all operators. Operators are used to perform various operations on variables and constants.  Types of operators   1. Assignment Operator 2. Mathematical Operators 3. Relational Operators 4. Logical Operators 5. Bitwise Operators 6. Shift Operators 7. Unary Operators 8. Ternary Operator 9. Comma Operator  Assignment Operator (=) Operates '=' is used for assignment, it takes the right-hand side (called rvalue) and copy it into the left-hand side (called lvalue). Assignment operator is the only operator which can be overloaded but cannot be inherited. Mathematical Operators There are operators used to perform basic mathematical operations. Addition (+) , subtraction (-) , diversion (/) multiplication (\*) and modulus (%) are the basic mathematical operators. Modulus operator cannot be used with floating-point numbers.  C++ and C also use a shorthand notation to perform an operation and assignment at same type. Example,  int x=10;  x += 4 // will add 4 to 10, and hence assign 14 to X.  x -= 5 // will subtract 5 from 10 and assign 5 Relational Operators These operators establish a relationship between operands. The relational operators are : less than (<) , grater thatn (>) , less than or equal to (<=), greater than equal to (>=), equivalent (==) and not equivalent (!=).  You must notice that assignment operator is (=) and there is a relational operator, for equivalent (==). These two are different from each other, the assignment operator assigns the value to any variable, whereas equivalent operator is used to compare values, like in if-else conditions, Example  int x = 10; //assignment operator  x=5; // again assignment operator  if(x == 5) // here we have used equivalent relational operator, for comparison  {  cout <<"Successfully compared";  } Logical Operators The logical operators are AND (&&) and OR (||). They are used to combine two different expressions together.  If two statement are connected using AND operator, the validity of both statements will be considered, but if they are connected using OR operator, then either one of them must be valid. These operators are mostly used in loops (especially while loop) and in Decision making. Bitwise Operators There are used to change individual bits into a number. They work with only integral data types like char, int and long and not with floating point values.   * Bitwise AND operators & * Bitwise OR operator | * And bitwise XOR operator ^ * And, bitwise NOT operator ~   They can be used as shorthand notation too, & = , |= , ^= , ~= etc Shift Operators Shift Operators are used to shift Bits of any variable. It is of three types,   1. Left Shift Operator << 2. Right Shift Operator >> 3. Unsigned Right Shift Operator >>>  Unary Operators These are the operators which work on only one operand. There are many unary operators, but increment ++ and decrement -- operators are most used.  Other Unary Operators : address of &, dereference \*, new and delete, bitwise not ~, logical not !, unary minus - and unary plus +. Ternary Operator The ternary if-else ? : is an operator which has three operands.  int a = 10;  a > 5 ? cout << "true" : cout << "false" Comma Operator This is used to separate variable names and to separate expressions. In case of expressions, the value of last expression is produced and used. |

**DAILY ASSESSMENT**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date:** | **22/06/2020** | **Name:** | **Chesmi B R** |
| **Course:** | **C++** | **USN:** | **4AL16EC100** |
| **Topic:** | **Module 2: Conditionals and loops** | **Semester & Section:** | **8TH SEM & A Section** |
| **Github Repository:** | **chesmibr** |  |  |

|  |
| --- |
| **AFTERNOON SESSION DETAILS** |
|  |

|  |
| --- |
| **Report**- Decision making in C++ - if, else and else if Decision making is about deciding the order of execution of statements based on certain conditions or repeat a group of statements until certain specified conditions are met. C++ handles decision-making by supporting the following statements,   * if statement * switch statement * conditional operator statement * goto statement  Decision making with if statement The if statement may be implemented in different forms depending on the complexity of conditions to be tested. The different forms are,   1. Simple if statement 2. if....else statement 3. Nested if....else statement 4. else if statement  Simple if statement The general form of a simple if statement is,  if(expression)  {  statement-inside;  }  statement-outside;  If the expression is true, then 'statement-inside' will be executed, otherwise 'statement-inside' is skipped and only 'statement-outside' will be executed.  Example:  #include< iostream.h>  int main( )  {  int x,y;  x=15;  y=13;  if (x > y )  {  cout << "x is greater than y";  }  }  x is greater than y if...else statement The general form of a simple if...else statement is,  if(expression)  {  statement-block1;  }  else  {  statement-block2;  }  If the 'expression' is true or returns true, then the 'statement-block1' will get executed, else 'statement-block1' will be skipped and 'statement-block2' will be executed.  Example:  void main( )  {  int x,y;  x=15;  y=18;  if (x > y )  {  cout << "x is greater than y";  }  else  {  cout << "y is greater than x";  }  }  y is greater than x Nested if....else statement The general form of a nested if...else statement is,  if(expression)  {  if(expression1)  {  statement-block1;  }  else  {  statement-block2;  }  }  else  {  statement-block3;  }  if 'expression' is false or returns false, then the 'statement-block3' will be executed, otherwise execution will enter the if condition and check for 'expression 1'. Then if the 'expression 1' is true or returns true, then the 'statement-block1' will be executed otherwise 'statement-block2' will be executed.  Example:  void main( )  {  int a,b,c;  cout << "enter 3 number";  cin >> a >> b >> c;  if(a > b)  {  if( a > c)  {  cout << "a is greatest";  }  else  {  cout << "c is greatest";  }  }  else  {  if( b> c)  {  cout << "b is greatest";  }  else  {  cout << "c is greatest";  }  }  }  The above code will print different statements based on the values of a, b and c variables. else-if Ladder The general form of else-if ladder is,  if(expression 1)  {  statement-block1;  }  else if(expression 2)  {  statement-block2;  }  else if(expression 3 )  {  statement-block3;  }  else  default-statement;  The expression is tested from the top(of the ladder) downwards. As soon as the true condition is found, the statement associated with it is executed.  Example:  void main( )  {  int a;  cout << "enter a number";  cin >> a;  if( a%5==0 && a%8==0)  {  cout << "divisible by both 5 and 8";  }  else if( a%8==0 )  {  cout << "divisible by 8";  }  else if(a%5==0)  {  cout << "divisible by 5";  }  else  {  cout << "divisible by none";  }  }  If you enter value 40 for the variable a, then the output will be:  divisible by both 5 and 8 C++ Loops In any programming language, loops are used to execute a set of statements repeatedly until a particular condition is satisfied.  A sequence of statement is executed until a specified condition is true. This sequence of statement to be executed is kept inside the curly braces { } known as loop body. After every execution of loop body, condition is checked, and if it is found to be true the loop body is executed again. When condition check comes out to be false, the loop body will not be executed. There are 3 type of loops in C++ language  1. while loop 2. for loop 3. do-while loop  while loop while loop can be address as an entry control loop. It is completed in 3 steps.   * Variable initialization.(e.g int x=0;) * condition(e.g while( x<=10)) * Variable increment or decrement (x++ or x-- or x=x+2)   Syntax:  variable initialization;  while (condition)  {  statements;  variable increment or decrement;  } for loop for loop is used to execute a set of statement repeatedly until a particular condition is satisfied. we can say it an open ended loop. General format is,  for(initialization; condition; increment/decrement)  {  statement-block;  }  In for loop we have exactly two semicolons, one after initialization and second after condition. In this loop we can have more than one initialization or increment/decrement, separated using comma operator. for loop can have only one condition. Nested for loop We can also have nested for loop, i.e one for loop inside another for loop. Basic syntax is,  for(initialization; condition; increment/decrement)  {  for(initialization; condition; increment/decrement)  {  statement;  }  } do...while loop In some situations it is necessary to execute body of the loop before testing the condition. Such situations can be handled with the help of do-while loop. do statement evaluates the body of the loop first and at the end, the condition is checked using while statement. General format of do-while loop is,  do  {  // a couple of statements  }  while(condition); Jumping out of a loop Sometimes, while executing a loop, it becomes necessary to skip a part of the loop or to leave the loop as soon as certain condition becocmes true, that is jump out of loop. C language allows jumping from one statement to another within a loop as well as jumping out of the loop. 1) break statement When break statement is encountered inside a loop, the loop is immediately exited and the program continues with the statement immediately following the loop. 2) continue statement It causes the control to go directly to the test-condition and then continue the loop process. On encountering continue, cursor leave the current cycle of loop, and starts with the next cycle. |