**ASSESSMENT 31**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date:** | 22-06-2020 | **Name:** | Sheela Golasangi |
| **Course:** | **C++** | **USN:** | 4AL16EC068 |
| **Topic:** | * Basic Concepts * Conditionals and Loops | **Semester & Section:** | VIII  ‘B’ |
| **Github Repository:** | Sheela-Course |  |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **MODULE 1**  **BASIC CONCEPTS**  **About C++, Tools, Commends, Variables, Basic Arithmetic, Assignment And Increment Operators.**        **C++**  C++  is a [general-purpose programming language](https://en.wikipedia.org/wiki/General-purpose_programming_language) created by [BjarneStroustrup](https://en.wikipedia.org/wiki/Bjarne_Stroustrup" \o "Bjarne Stroustrup) as an extension of the [C programming language](https://en.wikipedia.org/wiki/C_(programming_language)), or "C with [Classes](https://en.wikipedia.org/wiki/Class_(programming))". The language has expanded significantly over time, and modern C++ now has [object-oriented](https://en.wikipedia.org/wiki/Object-oriented_programming), [generic](https://en.wikipedia.org/wiki/Generic_programming), and [functional](https://en.wikipedia.org/wiki/Functional_programming) features in addition to facilities for [low-level](https://en.wikipedia.org/wiki/Low-level_programming_language) [memory](https://en.wikipedia.org/wiki/Memory_(computing)) manipulation. It is almost always implemented as a [compiled language](https://en.wikipedia.org/wiki/Compiled_language), and many vendors provide [C++ compilers](https://en.wikipedia.org/wiki/List_of_compilers#C.2B.2B_compilers), including the [Free Software Foundation](https://en.wikipedia.org/wiki/Free_Software_Foundation), [LLVM](https://en.wikipedia.org/wiki/LLVM), [Microsoft](https://en.wikipedia.org/wiki/Microsoft), [Intel](https://en.wikipedia.org/wiki/Intel), [Oracle](https://en.wikipedia.org/wiki/Oracle_Developer_Studio), and [IBM](https://en.wikipedia.org/wiki/IBM), so it is available on many platforms. Basic C++ Commands The basic commands are as follows. 1. #define This C++ Command can be used to substitute a particular value throughout the file in which it is located. This helps the compiler to go through the entire file and replaces the name of that macro which was created with the value which was defined. The replacement of this string stops at the end of the line.   2. #error This Command helps in displaying the error message which is defined and indicates the compiler to stop when this command is encountered. When the compiler comes across this command then the compiler displays the line number and message which is mentioned. This helps the programmer in debugging easily.   3. #include This includes Command helps in including the mentioned files in the code and making use of those. It can be used in two ways:   * #include <filename> * #include “filename”   When angular brackets are used when a file needs to search or include through the standard library directories. When there are quotes specified then it indicates that this filename needs to be searched in the current directory.   4. #pragma By making use of this C++ command the programmer has the capability to let the compiler know of what different things are to be done. #pragma is usually implementation specific and its use can vary from one compiler to another. There are different options which can be used and one can be to trace the program execution.  **Variable**  Variables are used in C++ where you will need to store any type of values within a program and whose value can be changed during the program execution. These variables can be declared in various ways each having different memory requirements and storing capability. Variables are the name of memory locations that are allocated by compilers, and the allocation is done based on the data type used for declaring the variable. Variable Definition in C++ A variable definition means that the programmer writes some instructions to tell the compiler to create the storage in a memory location. The syntax for defining variables is:  Syntax:  data\_typevariable\_name;  data\_typevariable\_name,variable\_name,variable\_name;  **MODULE 2**  **CONDITIONALS AND LOOPS**  **If, else, while loop, do while, for loop, switch, logic operators,**     C++ if Statement The syntax of the if statement is:  if (condition) {  // body of if statement  }  The if statement evaluates the condition inside the parentheses ( ).   * If the condition evaluates to true, the code inside the body of if is executed. * If the condition evaluates to false, the code inside the body of if is skipped.   **Note:** The code inside { } is the body of the if statement. C++ if...else The if statement can have an optional else clause. Its syntax is:  if (condition) {  // block of code if condition is true  }  else {  // block of code if condition is false  }  The if..else statement evaluates the condition inside the parenthesis. C++ while and do...while LoopIn this tutorial, we will learn the use of while and do...while loops in C++ programming with the help of some examples. In computer programming, loops are used to repeat a block of code.  For example, let's say we want to show a message 100 times. Then instead of writing the print statement 100 times, we can use a loop.  That was just a simple example; we can achieve much more efficiency and sophistication in our programs by making effective use of loops.  There are **3** types of loops in C++.   1. for loop 2. while loop 3. do...while loop  C++ for loop The syntax of for-loop is:  for (initialization; condition; update) {  // body of-loop  }  Here,   * initialization - initializes variables and is executed only once * condition - if true, the body of for loop is executed if false, the for loop is terminated * update - updates the value of initialized variables and again checks the condition  C++ while Loop The syntax of the while loop is:  while (condition) {  // body of the loop  }  Here,   * A while loop evaluates the condition * If the condition evaluates to true, the code inside the while loop is executed. * The condition is evaluated again. * This process continues until the condition is false. * When the condition evaluates to false, the loop terminates.    C++ do...while Loop The do...while loop is a variant of the while loop with one important difference: the body of do...while loop is executed once before the condition is checked.  Its syntax is:  do {  // body of loop;  }  while (condition);  Here,   * The body of the loop is executed at first. Then the condition is evaluated. * If the condition evaluates to true, the body of the loop inside the do statement is executed again. * The condition is evaluated once again. * If the condition evaluates to true, the body of the loop inside the do statement is executed again. * This process continues until the condition evaluates to false. Then the loop stops.  C++ switch statement A **switch** statement allows a variable to be tested for equality against a list of values. Each value is called a case, and the variable being switched on is checked for each case. Syntax The syntax for a **switch** statement in C++ is as follows −  switch(expression) {  case constant-expression :  statement(s);  break; //optional  case constant-expression :  statement(s);  break; //optional  // you can have any number of case statements.  default : //Optional  statement(s);  }  The following rules apply to a switch statement −   * The **expression** used in a **switch** statement must have an integral or enumerated type, or be of a class type in which the class has a single conversion function to an integral or enumerated type. * You can have any number of case statements within a switch. Each case is followed by the value to be compared to and a colon. * The **constant-expression** for a case must be the same data type as the variable in the switch, and it must be a constant or a literal. * When the variable being switched on is equal to a case, the statements following that case will execute until a **break** statement is reached. * When a break statement is reached, the switch terminates, and the flow of control jumps to the next line following the switch statement. * Not every case needs to contain a break. If no break appears, the flow of control will fall through to subsequent cases until a break is reached. * A **switch** statement can have an optional **default** case, which must appear at the end of the switch. The default case can be used for performing a task when none of the cases is true. No break is needed in the default case.  Arithmetic Operators There are following arithmetic operators supported by C++ language −  Assume variable A holds 10 and variable B holds 20, then −   |  |  |  | | --- | --- | --- | | **Operator** | **Description** | **Example** | | + | Adds two operands | A + B will give 30 | | - | Subtracts second operand from the first | A - B will give -10 | | \* | Multiplies both operands | A \* B will give 200 | | / | Divides numerator by de-numerator | B / A will give 2 | | % | Modulus Operator and remainder of after an integer division | B % A will give 0 | | ++ | [Increment operator](https://www.tutorialspoint.com/cplusplus/cpp_increment_decrement_operators.htm), increases integer value by one | A++ will give 11 | | -- | [Decrement operator](https://www.tutorialspoint.com/cplusplus/cpp_increment_decrement_operators.htm), decreases integer value by one | A-- will give 9 |  Relational Operators There are following relational operators supported by C++ language  Assume variable A holds 10 and variable B holds 20, then −   |  |  |  | | --- | --- | --- | | **Operator** | **Description** | **Example** | | == | Checks if the values of two operands are equal or not, if yes then condition becomes true. | (A == B) is not true. | | != | Checks if the values of two operands are equal or not, if values are not equal then condition becomes true. | (A != B) is true. | | > | Checks if the value of left operand is greater than the value of right operand, if yes then condition becomes true. | (A > B) is not true. | | < | Checks if the value of left operand is less than the value of right operand, if yes then condition becomes true. | (A < B) is true. | | >= | Checks if the value of left operand is greater than or equal to the value of right operand, if yes then condition becomes true. | (A >= B) is not true. | | <= | Checks if the value of left operand is less than or equal to the value of right operand, if yes then condition becomes true. | (A <= B) is true. |  Logical Operators There are following logical operators supported by C++ language.  Assume variable A holds 1 and variable B holds 0, then −   |  |  |  | | --- | --- | --- | | **Operator** | **Description** | **Example** | | && | Called Logical AND operator. If both the operands are non-zero, then condition becomes true. | (A && B) is false. | | || | Called Logical OR Operator. If any of the two operands is non-zero, then condition becomes true. | (A || B) is true. | | ! | Called Logical NOT Operator. Use to reverses the logical state of its operand. If a condition is true, then Logical NOT operator will make false. |  | |