**DAILY ASSESSMENT FORMAT**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Date:** | **23/06/2020** | **Name:** | **Abhishek Vasudev Mahendrakar** | |
| **Course:** | **C++ Tutorial by SOLOLEARN** | **USN:** | **4AL17EC003** | |
| **Topic:** | 1. **Data types, Array, Pointers.** 2. **Functions** | **Semester & Section:** | **6th-‘A’** | |
| **Github Repository:** | **ECEAbhishekVMahendrakar** | **E-mail:** | **abhi2244mahendrakar@gmail.com** | |
| **FORENOON SESSION DETAILS** | | | |
| **Image of session** | | | |
| **Report – Report can be typed or hand written for up to two pages.**   1. **Data types, Array, Pointers:** 2. **Introduction to 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. * **Integers** (whole numbers), such as -7, 42. * **Floating point**numbers, such as 3.14, -42.67. * 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. * The **Boolean** data type returns just two possible values: **true**(1) and **false**(0).  1. **int, float, double:**  * The **integer**type holds non-fractional numbers, which can be positive or negative. Examples of integers would include 42, -42, and similar numbers. * Use the **int**keyword to define the integer data type. * Several of the basic types, including integers, can be modified using one or more of these type **modifiers**: **signed**: A signed integer can hold both negative and positive numbers. **unsigned**: An unsigned integer can hold only positive values.   **short**: Half of the default size. **long**: Twice the default size.   * 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**.  1. **string, char, bool:**  * 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. * 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. * Boolean variables only have two possible values: **true**(1) and **false**(0). * To declare a boolean variable, we use the keyword **bool**.  1. **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. * C++ is **case-sensitive**, which means that an identifier written in uppercase is not equivalent to another one with the same name in lowercase. * C++ keyword (reserved word) cannot be used as variable names.  1. **Arrays:**  * An **array**is used to store a collection of data, but it may be useful to think of an array as a collection of variables that are all of the **same type**. * Instead of declaring multiple variables and storing individual values, you can declare a single array to store all the values. * When declaring an array, specify its element types, as well as the number of elements it will hold. * If you omit the size of the array, an array just big enough to hold the initialization is created. * To access array elements, index the array name by placing the element's index in square brackets following the array name. * Index numbers may also be used to assign a new value to an element.  1. **Using Arrays in Loops:**  * It's occasionally necessary to iterate over the elements of an array, assigning the elements values based on certain calculations.  1. **Array in Calculations:**  * The following code creates a program that uses a **for**loop to calculate the sum of all elements of an array.   **int arr[] = {11, 35, 62, 555, 989}; int sum = 0;   for (int x = 0; x < 5; x++) { sum += arr[x]; }  cout << sum << endl; //Outputs 1652**   1. **Multi- Dimensional Arrays:**  * A **multi-dimensional** array holds one or more arrays. Declare a multidimensional array as follows.   type name[size1][size2]...[sizeN];   * Arrays can contain an unlimited number of dimensions.  1. **Introduction to 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**. * A **pointer**is a variable, with the address of another variable as its value. * All pointers share the same data type - a long **hexadecimal**number that represents a memory address. * The **asterisk**sign is used to declare a pointer (the same asterisk that you use for multiplication), however, in this statement the asterisk is being used to designate a variable as a pointer.  1. **More on Pointers:**  * The **dereference**operator (\*) is basically an **alias**for the variable the pointer points to.  1. **Dynamic memory:**  * In a C++ program, memory is divided into two parts: **The stack**: All of your local variables take up memory from the stack. **The heap**: Unused program memory that can be used when the program runs to **dynamically**allocate the memory. * The allocated address can be stored in a **pointer**, which can then be dereferenced to access the variable. * The **delete**operator frees up the memory allocated for the variable, but does not delete the pointer itself, as the pointer is stored on the stack.  1. **The sizeof() Operator:**      * The **sizeof**operator can be used to get a variable or data type's size, in bytes. * The C++ **sizeof**operator is also used to determine the size of an **array**.  1. **Functions:** 2. **Introduction to functions:**  * A **function**is a group of statements that perform a particular task. * A function's**return type** is declared before its name. In the example above, the return type is **int**, which indicates that the function returns an integer value. * **return-type**: Data type of the value returned by the function. * **return-type**: Data type of the value returned by the function. * **parameters**: When a function is invoked, you pass a value to the parameter. This value is referred to as actual parameter or argument. The parameter list refers to the type, order, and number of the parameters of a function. * **body of the function**: A collection of statements defining what the function does. * You must declare a function prior to calling it. * A function **declaration**, or **function prototype**, tells the compiler about a function name and how to call the function. The actual body of the function can be defined separately.  1. **Function parameters:**  * Once parameters have been defined, you can pass the corresponding arguments when the function is called. * You can pass different arguments to the same function.  1. **Function with multiple parameters:**  * You can define as many parameters as you want for your functions, by separating them with **commas**.  1. **The rand() function:**  * Being able to generate **random**numbers is helpful in a number of situations, including when creating games, statistical modeling programs, and similar end products. * In the C++ standard library, you can access a pseudo random number generator function that's called **rand()**. When used, we are required to include the header **<cstdlib>**. * A **for**loop can be used to generate multiple random numbers. * Use the **modulo**(%) operator to generate random numbers within a specific range. * A solution to generate truly random numbers, is to use the current time as a seed value for the srand() function.  1. **Default Arguments:**  * When defining a function, you can specify a **default**value for each of the last parameters. If the corresponding argument is missing when you call a function, it uses the **default**value.  1. **Function Overloading:**  * Function **overloading**allows to create multiple functions with the**same name**, so long as they have different parameters. * When overloading functions, the definition of the function must differ from each other by the types and/or the number of arguments in the argument list. * You **can not**overload function declarations that differ only by **return**type.  1. **Recursion:**  * A **recursive function** in C++ is a function that calls itself.  1. **Passive Arrays to Functions:**  * An **array**can also be passed to a function as an argument. * The parameter should be defined as an array using square brackets, when declaring the function.  1. **Pass by Refference with Pointers:**  * There are two ways to pass arguments to a function as the function is being called.  **By value:** This method copies the argument's actual value into the function's formal parameter. Here, we can make changes to the parameter within the function without having any effect on the argument.  **By reference:** This method copies the argument's reference into the formal parameter. Within the function, the reference is used to access the actual argument used in the call. This means that any change made to the parameter affects the argument. * By default, arguments in C++ are passed **by value**. * **Pass-by-reference** copies an argument's address into the formal parameter. Inside the function, the address is used to access the actual argument used in the call. This means that changes made to the parameter affect the argument. | | | |