DAILY ASSESSMENT REPORT

Date:	23 June 2020	Name:	Gagan M K
Course:	C Plus Plus	USN:	4AL17EC032
Topic:	• **Data Types, Arrays, Pointers** • **Functions**	Semester & Section:	6th sem & 'A' sec
GitHub Repository:	Alvas-education-foundation/Gagan-Git		

FORENOON SESSION DETAILS
Image of session

Report – Report can be typed or hand written for up to two pages.

Data Types, Arrays, Pointers:

- 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.
- 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.
- In C++, single quotation marks indicate a character; double quotes create a string literal. While 'a' is a single a character literal, "a" is a string literal.
- The Boolean data type returns just two possible values: true (1) and false (0).
- Conditional expressions are an example of Boolean data type.
- The integer type holds non-fractional numbers, which can be positive or negative. Examples of integers would include 42, -42, and similar numbers.
- 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".
- For example, myvariable is not the same as MYVARIABLE and not the same as MyVariable.
- These are three different variables.
- C++ keyword (reserved word) cannot be used as variable names.
- For example, int, float, double, cout cannot be used as a variable name.
- 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.
- A multi-dimensional array holds one or more arrays. Declare a multidimensional array as follows.
        type name[size1][size2]...[sizeN];

	Column 1	Column 2	Column 3	Column 4
Row 1	x[0][0]	x[0][1]	x[0][2]	x[0][3]
Row 2	x[1][0]	x[1][1]	x[1][2]	x[1][3]
Row 3	x[2][0]	x[2][1]	x[2][2]	x[2][3]

- 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.
- In C++, pointers help make certain tasks easier to perform. Other tasks, such as dynamic memory allocation, cannot be performed without using pointers.
- All pointers share the same data type - a long hexadecimal number that represents a memory address
- A pointer is a variable, and like any other variable, it must be declared before you can work with it.
- 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.
- There are two operators for pointers:
- Address-of operator (&): returns the memory address of its operand.
- Contents-of (or dereference) operator (*): returns the value of the variable located at the address specified by its operand.
- To be successful as a C++ programmer, it's essential to have a good understanding of how dynamic memory works.
- For local variables on the stack, managing memory is carried out automatically.
- On the heap, it's necessary to manually handle the dynamically allocated memory, and use the delete operator to free up the memory when it's no longer needed.

Category	Type	Minimum Size
boolean	bool	1 byte
character	char	1 byte
integer	short	2 bytes
	int	2 bytes
	long	4 bytes
	long long	8 bytes
floating point	float	4 bytes
	double	8 bytes
	long double	8 bytes

Functions:

- A function is a group of statements that perform a particular task.
- You may define your own functions in C++.
- 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.
- Occasionally, a function will perform the desired operations without returning a value. Such functions are defined with the keyword void.
- For a function to use arguments, it must declare formal parameters, which are variables that accept the argument's values.
- Once parameters have been defined, you can pass the corresponding arguments when the function is called.
- You can define as many parameters as you want for your functions, by separating them with commas.
- Function overloading allows to create multiple functions with the same name, so long as they have different parameters.
- A recursive function in C++ is a function that calls itself.
- To demonstrate recursion, let's create a program to calculate a number's factorial.
- In mathematics, the term factorial refers to the product of all positive integers that are less than or equal to a specific non-negative integer (n). The factorial of n is denoted as n!
- 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.
- 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.
- Example Program:

```cpp
void myFunc(int x) {
  x = 100;
}

int main() {
  int var = 20;
  myFunc(var);
  cout << var;
}
// Outputs 20
```