DAILY ASSESSMENT FORMAT

Date:	**23-06-2020**	**Name:**	**PRINCIA MELITA DSOUZA**
Course:	**sololearn**	**USN:**	**4AL17EC075**
Topic:	**C++**	**Semester & Section:**	**6TH B**
Github Repository:	**MELITA-1999**		

FORENOON SESSION DETAILS

Image of session

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

C++ Data Types

All variables use data-type during declaration to restrict the type of data to be stored. Therefore, we can say that data types are used to tell the variables the type of data it can store. Whenever a variable is defined in C++, the compiler allocates some memory for that variable based on the data-type with which it is declared. Every data type requires a different amount of memory.

Data types in C++ is mainly divided into three types:

1. **Primitive Data Types**: These data types are built-in or predefined data types and can be used directly by the user to declare variables. example: int, char , float, bool etc. Primitive data types available in C++ are:
   - Integer
   - Character
   - Boolean
   - Floating Point

- Double Floating Point
- Valueless or Void
- Wide Character

2. **Derived Data Types**: The data-types that are derived from the primitive or built-in datatypes are referred to as Derived Data Types. These can be of four types namely:
   - Function
   - Array
   - Pointer
   - Reference
3. **Abstract or user-defined data types**: These data types are defined by user itself. Likedefining a class in C++ or a structure. C++ provides the following user-defined datatypes:
   - Class
   - Structure
   - Union
   - Enumeration
   - Typedef defined DataType

This article discusses primitive data types available in C++.

- Integer: Keyword used for integer data types is int. Integers typically requires 4 bytes of memory space and ranges from -2147483648 to 2147483647.
- Character: Character data type is used for storing characters. Keyword used for character data type is char. Characters typically requires 1 byte of memory space and ranges from -128 to 127 or 0 to 255.
- Boolean: Boolean data type is used for storing boolean or logical values. A boolean variable can store either *true* or *false*. Keyword used for boolean data type is bool.
- Floating Point: Floating Point data type is used for storing single precision floating point values or decimal values. Keyword used for floating point data type is float. Float variables typically requires 4 byte of memory space.
- Double Floating Point: Double Floating Point data type is used for storing double precision floating point values or decimal values. Keyword used for double floating point data type is double. Double variables typically requires 8 byte of memory space.
- void: Void means without any value. void datatype represents a valueless entity. Void data type is used for those function which does not returns a value.
- Wide Character: Wide character data type is also a character data type but this data type has size greater than the normal 8-bit datatype. Represented by wchar_t. It is generally 2 or 4 bytes long.

Datatype Modifiers

As the name implies, datatype modifiers are used with the built-in data types to modify the length of data that a particular data type can hold.

Arrays in C++

An array is a collection of elements of the same type placed in contiguous memory locations that can be individually referenced by using an index to a unique identifier. Five values of type int can be declared as an array without having to declare five different variables (each with its own identifier).C++ provides a data structure, the array, which stores a fixed-size sequential collection of elements of the same type. An array is used to store a collection of data, but it is often more useful to think of an array as a

collection of variables of the same type.Instead of declaring individual variables, such as number0, number1, ..., and number99, you declare one array variable such as numbers and use numbers[0], numbers[1], and ..., numbers[99] to represent individual variables. A specific element in an array is accessed by an index.All arrays consist of contiguous memory locations. The lowest address corresponds to the first element and the highest address to the last element.

Declaring Arrays

To declare an array in C++, the programmer specifies the type of the elements and the number of elements required by an array as follows −

type arrayName [arraySize];

This is called a single-dimension array. The **arraySize** must be an integer constant greater than zero and **type** can be any valid C++ data type. For example, to declare a 10-element array called balance of type double, use this statement −

double balance[10];

Initializing Arrays

You can initialize C++ array elements either one by one or using a single statement as follows −

double balance[5] = {1000.0, 2.0, 3.4, 17.0, 50.0};

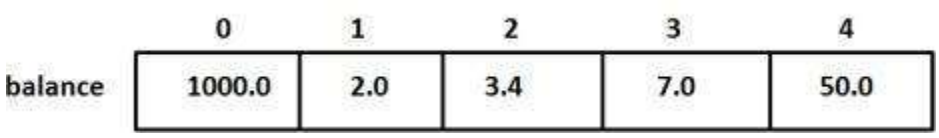
The number of values between braces { } can not be larger than the number of elements that we declare for the array between square brackets [ ]. Following is an example to assign a single element of the array −If you omit the size of the array, an array just big enough to hold the initialization is created. Therefore, if you write −

double balance[] = {1000.0, 2.0, 3.4, 17.0, 50.0};

You will create exactly the same array as you did in the previous example.

balance[4] = 50.0;

The above statement assigns element number 5^{th} in the array a value of 50.0. Array with 4^{th} index will be 5^{th}, i.e., last element because all arrays have 0 as the index of their first element which is also called base index. Following is the pictorial representaion of the same array we discussed above −



An element is accessed by indexing the array name. This is done by placing the index of the element within square brackets after the name of the array. For example −

double salary = balance[9];

The above statement will take 10^{th} element from the array and assign the value to salary variable. Following is an example, which will use all the above-mentioned three concepts viz. declaration, assignment and accessing arrays −

Pointers in C++

Pointer is a variable in **C++** that holds the address of another variable. They have data type just like variables, for example an integer type **pointer** can hold the address of an integer variable and an character type **pointer** can hold the address of char variable.

C++ pointers are easy and fun to learn. Some C++ tasks are performed more easily with pointers, and other C++ tasks, such as dynamic memory allocation, cannot be performed without them.

As you know every variable is a memory location and every memory location has its address defined which can be accessed using ampersand (&) operator which denotes an address in memory. Consider the following which will print the address of the variables defined −

```
#include <iostream>

using namespace std;
int main () {
   int  var1;
   char var2[10];

   cout << "Address of var1 variable: ";
   cout << &var1 << endl;

   cout << "Address of var2 variable: ";
   cout << &var2 << endl;

   return 0;
}
```

What are Pointers?

A **pointer** is a variable whose value is the address of another variable. Like any variable or constant, you must declare a pointer before you can work with it. The general form of a pointer variable declaration is –

```
type *var-name;
```

Date:	**23-06-2020**	**Name:**	**PRINCIA**
Course:	**sololearn**	**USN:**	**4AL17EC075**
Topic:	**C++**	**Semester & Section:**	**6th B**

AFTERNOON SESSION DETAILS

Image of session

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

A function is a group of statements that together perform a task. Every C++ program has at least one function, which is **main()**, and all the most trivial programs can define additional functions.You can divide up your code into separate functions. How you divide up your code among different functions is up to you, but logically the division usually is such that each function performs a specific task.A function **declaration** tells the compiler about a function's name, return type, and parameters. A function **definition** provides the actual body of the function.The C++ standard library provides numerous built-in functions that your program can call. For example, function **strcat()** to concatenate two strings, function **memcpy()** to copy one memory location to another location and many more functions.A function is known with various names like a method or a sub-routine or a procedure etc.

Defining a Function

The general form of a C++ function definition is as follows −

```
return_type function_name( parameter list ) {
   body of the function
}
```

A C++ function definition consists of a function header and a function body. Here are all the parts of a function −

- **Return Type** − A function may return a value. The **return_type** is the data type of the value the function returns. Some functions perform the desired operations without returning a value. In this case, the return_type is the keyword **void**.
- **Function Name** − This is the actual name of the function. The function name and the parameter list together constitute the function signature.
- **Parameters** − A parameter is like a placeholder. 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. Parameters are optional; that is, a function may contain no parameters.
- **Function Body** − The function body contains a collection of statements that define what the function does.

Function Declarations

A function **declaration** tells the compiler about a function name and how to call the function. The actual body of the function can be defined separately.

A function declaration has the following parts −

```
return_type function_name( parameter list );
```

For the above defined function max(), following is the function declaration −

```
int max(int num1, int num2);
```

Parameter names are not important in function declaration only their type is required, so following is also valid declaration −

int max(int, int);

Function declaration is required when you define a function in one source file and you call that function in another file. In such case, you should declare the function at the top of the file calling the function.

Calling a Function

While creating a C++ function, you give a definition of what the function has to do. To use a function, you will have to call or invoke that function.

When a program calls a function, program control is transferred to the called function. A called function performs defined task and when it's return statement is executed or when its function-ending closing brace is reached, it returns program control back to the main program.

To call a function, you simply need to pass the required parameters along with function name, and if function returns a value, then you can store returned value. For example −

```
#include <iostream>
using namespace std;

// function declaration
int max(int num1, int num2);

int main () {
   // local variable declaration:
   int a = 100;
   int b = 200;
   int ret;

   // calling a function to get max value.
   ret = max(a, b);
   cout << "Max value is : " << ret << endl;

   return 0;
}

// function returning the max between two numbers
int max(int num1, int num2) {
   // local variable declaration
   int result;

   if (num1 > num2)
      result = num1;
   else
      result = num2;

   return result;
}
```