Lesson 4: Tokens, Variables and data types in C++

4.1. Introduction

In our previous lesson, we learnt about object oriented programming languages. In our lesson today, we will discuss the tokens, variables and data types in C++

4.2. Lesson objectives

By the end of this lesson, you will be able to:

- Define a token
- Declare and use a variable
- Define and use various data types in C++
- Define and use constants

4.3. Lesson outline

This lesson is structured as follows:

4.1. Introduction
4.2. Lesson objectives
4.3. Lesson outline
4.4. Tokens and keywords in C++
4.5. Variables in C++
4.6. Data types in C++
4.7. Constants in C++
4.8. Revision questions
4.9. Summary
4.10. Suggested reading

4.4. Tokens and keywords in C++

A token is the smallest unit in a program. In c++ a token can be a character, white space, digit etc.

[a]. **Character set:** This consists of a set of alphabets, letters, white spaces and some special characters that are valid in C++ language.

[b]. Alphabets: These consists of alphabets in upper or lower case. Remember C++ is case sensitive. This means A and a are different.

- Uppercase: A B C ..... X Y Z
- Lowercase: a b c ..... x y z

[c]. **Digits:** These consist digits such as 0 1 2 3 4 5 6 8 9

[d].Special Characters: These can include the following:

,	<>	.	()	;	$	:	%	[]	#	?	‘	&	{}	“	^	!	/	*	|	-	~	+

[e]. **White space Characters:** White space refers to spaces, horizontal tabs, vertical tabs, and blank lines in your source code.

[f]. **Keywords:** These are the reserved words used in programming. Each keyword has fixed meaning that cannot be changed by user. C++ language is case ensitive and all keywords must be written in lowercase.

Auto	Double	int	Struct
Break	Else	long	Switch
Case	Enum	register	Typedef
Char	Extern	return	Union
Continue	For	signed	Void
Do	If	static	While
Default	Goto	sizeof	Volatile
Const	Float	short	unsigned

4.5. Variables in C++

A variable is a named memory location used to store data. A variable must be declared before it is used in a program. A variable name or identifier:

- Can consist of letters, numbers and underscore
- Can only begin with a letter
- Must be meaningful and easy to remember Must not contain a reserved keyword.

There are three types of variables in C++:

- Local variable –A variable declared and used within the function
- Global variable- A variable declared usually at the top of the program and can be used anywhere within the program
- Formal parameters- A variable declared as the function argument.

Examples in a program:

A program that uses local variables x,y and sum

```
#include<i
ostream>
using
namespace
std;  int
main()
{
  int  x, y, sum;                          //Declare local variables
  cout << "Enter value of x:" << endl;   //statement to prompt user
  for x   cin >> x;                                  //statement to
  read x value  cout << "Enter value of y:" << endl;    //Statement
  to prompt user for y valu e
```

```
    cin >> y;                                   //statement to
read y value      sum = x + y;
//statement to calculate sum    cout << "The sum is :" << sum
<< endl; //statement to display sum

       return 0;                              //returns a value to OS
```

A program that uses global variables x,y and sum.

```
#include
<iostrea
m> using
namespac
e std;
int x, y, sum;                              //Declare global
variables

int main() {       cout << "Enter value of x:" << endl;
//statement to prompt user for x        cin >> x;
//statement to read x value      cout << "Enter value of y:" <<
endl;    //Statement to prompt user for y value       cin >> y;
//statement to read y value      sum = x + y;
//statement to calculate sum
```

A program to use formal parameters x and y and function called total

```
#include
<iostrea
m> using
namespac
e std;
int total(int x, int y)             //declare formal
parameters in total function. {
     int sum;                          //declaration of local
variable sum       sum = x + y;
       return sum;
}
```

```
int main()
{                                          //execution begins
here        int a, b;                                  //declare
local variables a and b   cout << "Enter value of a:" << endl;
//statement to prompt user for a        cin >> a;
//statement to read a value      cout << "Enter value of b:" <<
endl;    //Statement to prompt user for b value       cin >> b;
//statement to read b value      cout << "The sum is :" <<
total(a,b) << endl; //statement to call total function
       return 0;                               //returns a value to OS
   }                                                                    //Execution
```

4.6. Data types in C++

A data type is a keyword that define the type of data that a variable or constant should have. It defines a named memory location. Every variable or constant declared must belong to a data type. The data types in C++ can be classified into two broad categories:

a) Primary (intrinsic) data type: These basic types include integers, floating point, characters .

b) Secondary data type: These include data types such as enumerated type, arrays, pointers and structures.

4.7. Constants in C++

A constant is a data storage location used by a program. Unlike a variable, the value stored in a constant can't be changed during program execution. The constants refer to fixed values that the program may not alter during its execution. C++ has two types of constants, each with its own specific uses:

a) Literal Constants

b) Symbolic Constants

Literal constants: A literal constant is a value that is typed directly into the **source code** wherever it is needed. Literal constants can be of any of the basic data types like:

[i]. An integer constant e.g. 0,1,2,3,4,5,6,7,8,9

[ii]. A floating constant, e.g. 0.3,0.004

[iii]. A character constant e.g. 'A'

[iv]. A string literal e.g. "Monday"

[v]. An enumeration constant e.g. int enum{Monday,Tuesday,Wednesday,Thursday,Friday,sunday}

Symbolic constant: This is a constant that is represented by a name (symbol) in the program. Like a literal constant, a symbolic constant can't change. Whenever the constant's value is needed in the program, then use its name as you would use a variable name. The actual value of the symbolic constant needs to be entered only once, when it is first defined.

Symbolic constants have two significant advantages over literal constants:

- Using symbolic constant, the code is clearer.
- When changing the value of the constant you do not have to change every occurrence in the code you only change it at the declaration area.

Defining symbolic constants

There are two simple ways in to define symbolic constants:

[i]. Using #define preprocessor.

Study the program below then compile and run.

```
#include<iostream>
using namespace std;
#define PI 3.142
void main()
{
	float r;
	cout << "Enter radius of the circle:" << endl;
	cin >> r;
	cout << "Ther area of the circle is: " << PI*r*r << endl;
}
```

[ii]. Using const keyword.

Study the program below that uses a constant , compile and run it.

```
#include<iostream>
using namespace std;
void main()
{
	float r;
	float const PI=3.142;
	cout << "Enter radius of the circle:" << endl;
	cin >> r;
	cout << "Ther area of the circle is: " << PI*r*r << endl;
}
```

4.7. Revision questions

a) Differentiate between symbolic constant and literal constants
b) Explain why symbolic constants are important
c) Describe any three rules followed when naming a variable.
d) State any five reserved keywords in C++

4.8. Summary

In this lesson, you have learnt how to work with variables and constants in C++. In addition, you have also learnt about various data types in C++.

4.9. Suggested reading

[1]. Object oriented programming with C++ by E Balagurusamy 3rd ed; publisher: Tata Mcraw Hill

[2]. Sams teach yourself c++ in 24 hours by Jesse Liberty and Rogers Cadenhead. [3]. Object oriented programming in c++ by Joyce Farrel [4]. Object-oriented programming with c++ by Sourav Sahay.