Introduction to C++ Notes Week 1

Hello everyone welcome to the weekly lecture notes

Topics to be covered:

1. Introduction to C++
2. Variables
3. Data Types
4. Identifiers
5. C++ Output
6. C++ Input
7. C++ Operators
8. C++ Operator Precedence and Associativity

Introduction to C++

1. C++ is a middle-level programming language developed by Bjarne Stroustrup in 1979 at Bell Labs.
2. C++ runs on a variety of platforms, such as Windows, Mac OS, and the various versions of UNIX.C++ is closer to the compiler and faster than C language.
3. C++ is both a procedural and an object-oriented programming language. It supports OOP features such as polymorphism, encapsulation, and inheritance. As C++ is an object-oriented programming language so it gives a clear structure to programs and allows code to be reused, lowering development costs.

Variables

1. A variable is the title of a reserved region allocated in memory. In other words, it may be referred to as the name of a memory location.
2. Syntax for Declaring a Variable:

   `<data_type> <variable_name>`

   Example:

```
int x = 10; // here int is the data type and x is variable
int y = 50;
```

3. Changing values of variables: A variable's value can also be changed in the program.

   Example :

```
int x = 10;
x = 20; // now the value changes to 20
int z = 30;
z = 40; // now the value changes to 40
```

4. **NOTE:** Assigning value to variable while declaring is optional.

Variables Naming Rules

1. Variables can start from an alphabet or underscore _ or $.
2. Special characters except _ and $ are not allowed.
3. Some particular keywords are not allowed.
4. Commas or blanks are not allowed.

Data Types in C++

1. Data types specify the different sizes and values that can be stored in the variable. Hence, by assigning different data types, we can store integers, decimals, or characters in these variables.
2. The different types are :
   1. **C++ int:** int keyword is used to indicate integers.
   2. **C++ char:** This data type is used to store a single character.
   3. **C++ bool:** A Boolean data type is declared with bool keyword.
   4. **C++ float :** float is used to store floating-point numbers (decimals and exponentials).
   5. **C++ double:** double is used to store floating-point numbers (decimals and exponentials).
   6. **C++ void:** The void keyword means "nothing" or "no value".

Identifiers in C++

1. It is a name given to a package, class, interface, method, or variable. All identifiers must have different names.
2. Rules for identifiers in C++ are :
   1. All identifiers should begin with a letter (A to Z or a to z), $ and _ and must be unique.
   2. After the first character/letter, identifiers can have any combination of characters.
   3. A keyword cannot be used as an identifier.
   4. The identifiers are case sensitive.
   5. Whitespaces are not permitted.

3. Examples of Identifiers in C++ are:

   `abc, _abc, __xyz`

Output in C++

1. In C++, all lines that start with a pound or hashtag `#` sign are called directives and are processed by a preprocessor which is a program invoked by the compiler.
2. `#include<iostream>` tells the preprocessor to include the iostream header file(file that has some already pre-written code which we are importing to avoid reinventing the wheel) in the program and iostream is the header file which contains all the basic functions of the program like input/output etc.
3. `Using namespace std` is used to define which input/output form is going to be used. (Explained in the forthcoming lectures).
4. `int main():` This line is used to declare a function having the name "main" whose return type is integer. Every C++ program's execution begins with the main() function, no matter where the function is located in the program.
5. Example :

```
#include <iostream>
using namespace std;
int main()
{
    // prints hello world in C++
    cout << "Hello World in C++";
}
```

Output:

```
Hello World in C++
```

NOTE :

To print a new line in C++, we use `endl` command or `'\n'` command.

Taking Input in C++

1. This can be done using `cin`. `cin` is a predefined object that reads data from the user with the extraction operator (`>>`) (like `scanf` is used in case of C language).
2. Example :

```
#include <iostream>
using namespace std;
int main() {
  int x, y;
  cout << "Enter two numbers: ";
  cin >> x >> y;
  sum = x + y;
  // prints sum
  cout << sum <<endl;
  return 0;
}
```

C++ Operators

1. Operators are the symbols that are used to perform pre-defined operations on variables and values (commonly referred to as operands).
2. Operators in C++ can be classified into 6 types:
    1. Arithmetic Operators
    2. Relational Operators
    3. Logical Operators
    4. Assignment Operators
    5. Bitwise Operators
    6. Misc. Operator
3. **Arithmetic Operators:** They are used in mathematical expressions in a program. They function in the same way as they do in algebra.

    Example :

```
#include<iostream>
using namespace std;
int main() {
   int x = 5;
   int y = 10;
   int sum = x+y; // using addition operator
   int diff = x-y; // using subtraction operator
   int div = y/x; // using division operator
   int product = x*y;
   cout << sum << endl;
   cout << diff << endl;
   cout << product << endl;
   cout << div << endl;
}
```

4. **C++ Relational Operators:**

Operator	Description	Example
it is ==	Is Equal To	4 == 5 returns false
!=	Not Equal To	1!= 5 returns true
>	Greater Than	1 > 5 returns false
<	Less Than	2 < 5 returns true
≥	Greater Than or Equal To	4 >= 5 returns false
<=	Less Than or Equal To	1 <= 5 returns true

5. **C++ Logical Operators :** Logical operators are used for decision making. This class of operators is used to check whether an expression is true or false. Some of the commonly used logical operators are mentioned in the table below.

Operator	Example	Meaning
&& (Logical AND)	expression1 && expression2	true only if both expression1 and expression2 are true
| (Logical OR)	expression1 | expression2	true if either expression1 or expression2 is true
! (Logical NOT)	!expression	true if expression is false and vice versa

6. **C++ assignment operators:**
It assigns the value of its right-hand operand to a variable, a property, or an indexer element given by its left-hand operand.

Operator	Example	Equivalent to
it is =	p = q;	p = q;
+=	p += q;	p = p + q;
-=	p -= q;	p = p - q;
*=	p *= q;	p = p * q;
/=	p /= q;	p = p / q;
%=	p %= q;	p = p % q;

7. **Bitwise Operators :**

Operator	Description
~	Bitwise Complement
<<	Left Shift
>>	Right Shift
>>>	Unsigned Right Shift
&	Bitwise AND
^	Bitwise exclusive OR

8. **Miscellaneous Operators:**

S.No	Operator & Description
1	`sizeof` This operator is used to compute the size of a variable. For example, `sizeof(p)`, where 'p' is integer, and will return 4.
2	`Condition ? Expression1 : Expression2` Conditional operator Condition is true then it returns the value of Expression1 otherwise it returns the value of Expression2.
3	`,` Comma operator is a binary operator. The value of the entire comma expression is actually the value of the last expression of the comma-separated list.
4	`. (dot)` and `-> (arrow)` Member operators are for reference individual members of classes, structures, and unions. The dot operator is used for the actual object. The arrow operator is used with a pointer to an object.
6	`&` Address-of operator & returns the memory address of a variable. For example &p; will give the actual memory address of the variable p.

C++ Operator Precedence and Associativity

1. Operator precedence determines the order/sequence of evaluation of operators in an expression (analogous to BODMAS concept in math).
2. Associativity specifies the order in which operators are evaluated by the compiler, which can be left to right or right to left.

Category	Operators	Associativity
Postfix	++, - -	Left to right
Unary	+, -, !, ~, ++, - -	Right to left
Multiplicative	*, /, %	Left to right
Additive	+, -	Left to right
Shift	<<, >>	Left to right
Relational	< <=, > >=	Left to right
Equality	!= , ==	Left to right
Bitwise AND	&	Left to right
Bitwise XOR	^	Left to right
Bitwise OR	|	Left to right
Logical AND	&&	Left to right
Logical OR	|	Left to right
Conditional	?:	Right to left
Assignment	+= , -=, >> = , << = , ^=, | =	Right to left
