

## UNIT -III INTRODUCTION TO C++

## CHAPTER

## 9

*Introduction to C++***1. History of C++**

- C++ was developed by **Bjarne Stroustrup** at **AT & T Bell Laboratory** during **1979**
- it was referred "**New C**" and "**C with Classes**".
- In **1983**, the name was changed as C++ by **Rick Mascitti**.

**2. Benefits of learning C++**

- C++ is called as **Hybrid Language**.
- C++ is for multi-device, multi-platform app development.
- C++ is an **object-oriented programming language**.
- It includes classes, inheritance, polymorphism, data abstraction and encapsulation.
- C++ has a rich function library.
- C++ allows exception handling, inheritance and function overloading which are not possible in C.
- C++ is a powerful, efficient and fast language.
- It finds a wide range of applications.

**3. List the languages which are influenced by C++**

- C# (C-Sharp), D, Java and newer versions of C

**4. What are the Character set in C++?**

- Character set is a set of characters which are allowed to write a C++ program. They are

Alphabets	A .... Z, a ..... z
Numeric	0 .... 9
Special Characters	+ - * / ~ ! @ # \$ % ^ & [ ] ( ) { } = > < _ \   ? . , : ' " ;
White space	Blank space, Horizontal tab (→), Carriage return (↵), Newline, Form feed
Other characters	C++ can process any of the 256 ASCII characters as data.

**5. What are Lexical units or Lexical elements or Tokens. in C++?**

- The smallest **individual unit** in a program is known as a **Lexical unit or Lexical elements or Token**. They are Keywords • Identifiers • Literals • Operators • Punctuators

**6. Define Keywords.**

- Keywords are the **reserved words**.
  - It conveys **specific meaning** to the C++ compiler.
  - They are the **essential elements** to construct C++ programs.
  - Most of the keywords are **common** to **C, C++** and **Java**.
  - All the keywords must be in lower case in C++.
  - **keywords** cannot be used as an identifier name.
- Ex. if ,for ,this, cout , cin, while etc....

**7. What are Identifiers?**

- Identifiers are the **user-defined names** given to **variables, functions, arrays, classes** etc.,
- These are the fundamental building blocks of a program.
- Every language has specific **rules** for naming the identifiers.

**8. What are the Rules for naming an identifier?**

- Only **alphabets, digits** and **underscore(\_)** are permitted.
- Other special characters ,space are not allowed.
- The first character of an identifier must be an **alphabet** or an **underscore (\_)**.
- C++ is **case sensitive**
- **keywords** cannot be used as an identifier name.

**9. Define Literals /Constant? What is meant by literals?**

- Literals are called as **Constants**.
- Literals are **data items** whose values do not change during the execution of a program.

C++ has several kinds of literals:

- Numeric Constants:
- Boolean Literals
- Character constant
- String constant.

**10. Numeric Constants: What are numeric constant and its types.**

The numeric constants are **only numeric values**, They are,

1. Integer Constants (or) Fixed point constants.
2. Real constants (or) Floating point constants.

### 11.How many types of integer literals(or) Fixed point constants available in C++? or Explain the types of integer literals in C++.

- Integers are **whole** numbers **without** a **decimal point**.
- It may be **signed** (negative) or **unsigned**.
- Commas** and **blank spaces** are not allowed.

Three types : (i) **Decimal** (ii) **Octal** (iii) **Hexadecimal**

#### Decimal

- Consists of digits (0 .... 9) Ex. 28 , -28 , 28.11

#### Octal

- Consists of digits (0.. 7) begins with **0** is considered as an Octal constant. Ex.022 , - 022

#### Hexadecimal

- Consists of digits (0...9 ,A..F)
- begins with **0x** or **0X** is considered as a Hexadecimal constant. Ex. 0x28 , 0X3AC

### 13. What are Real constants (or) Floating point constants

- It is a numeric constant having a **fractional component**.
- It may be **signed** (negative) or **unsigned** with **decimal point**
- It may be written in **fractional** or in **exponent** form.
- Exponent form consists of two parts:  
(1) Mantissa and (2) Exponent.
- The mantissa followed by a letter **E** or **e** and the exponent.

### 14. How to write 58000000 . 00 in Exponent form.?

It may be written as  $0.58 \times 10^8$  or  $0.58E8$ .

**Other Example:**

5.864 E-1 -  $5.864 \times 10^{-1}$  - 58.64

5864 E-2 -  $5864 \times 10^{-2}$  - 58.64

0.5864 E-2 -  $0.5864 \times 10^{-2}$  - 58.64

### 15.What is Boolean Literals?

- Used to represent Boolean values(**True** (1) or **false**(0)).

### 16.Differentiate between Character constant and String constant?

#### Character constant

- It must contain **one character**
- must be enclosed within a **single quote**.  
Ex. 'p' , '6' , '+'
- Each single character constant has an equivalent **ASCII** value. For 'A' is 65.

#### String Literals

- Set of characters** are called **String literals**.
- It enclosed within **double quotes** (" ").
- String should be terminated with a **'\0' (NULL)** character..
- Size of "welcome" is not 7 but 8 including **\0**.  
Ex. "S" , "Elango" , "123" , "ELANGO\0"

### 17.What is the significance of null (\0) character in a string?

- String should be **terminated** with a **'\0' (NULL)** character Ex. "ELANGO\0"

### 18.What is Escape sequences (or) Non-graphic Characters (or ) non- Printable Characters ?

- Non-printable** characters cannot be typed directly from a keyboard .
- These **non-printable** characters can be represented by using **escape sequences**.
- An escape sequence is represented by a **backslash** ( \ ) followed by one or two characters.

Escape sequence	Non-graphical character
\a	Audible or alert bell
\b	Backspace
\f	Form feed
\n	Newline or linefeed
\r	Carriage return
\t	Horizontal tab
\v	Vertical tab
\\	Backslash
\'	Single quote
\"	Double quote
\?	Question Mark
\On	Octal number
\xHn	Hexadecimal number
\0	Null

Ex.\a , \n , \t etc...

### 19.How to represent a long and unsigned constant ?

- The suffix **L** or **l** used to represent **long** constant  
Ex.25L
- The suffix **U** or **u** used to represent **unsigned** constant. Ex. 25U

#### Evaluate Yourself

- What is meant by literals? How many types of integer literals available in C++? **Ref.9 & 11**
- What kind of constants are following?  
i) 26 ii) 015 iii) 0xF iv) 014.9
- What is character constant in C++? **Ref.16**
- How are non graphic characters represented in C++? **Ref.18**
- Write the following real constants into exponent form: i) 32.179 ii) 8.124 iii) 0.00007  
Ans : i)32.179E3 ii)8.124E3 iii)0.7E-4
- Write the following real constants into fractional form: i) 0.23E4 ii) 0.517E-3 iii) 0.5E-5 iv)5E-5  
Ans: i)2300 ii) 0.000517 iii) 0.000005 iv) 0.00005
- What is the significance of null (\0) character in a string? **Ref.17.**

**20.What are Operators and Operands**

- The **symbols** which are used to do some mathematical or logical operations are called as **"Operators"**.
- The **values** that the operators act upon are called as **"Operands"**.

A + B : + operator A, B - Operands

**21.How are the operators classified based on operand requirements?**

- Unary Operators** - Require only **one** operand
- Binary Operators** - Require **two** operands
- Ternary Operators** - Require three operands

**22.How are the C++ Operators are classified based on their function?**

- (1) Arithmetic Operators
- (2) Relational Operators
- (3) Logical Operators
- (4) Bitwise Operators
- (5) Assignment Operators
- (6) Conditional Operator
- (7) Other Operators

**23.Define Arithmetic Operators**

- Arithmetic operators are **binary** operators because it require **two** operands.
- It performs operations like addition, subtraction, multiplication, division etc.,

Ex .+ (Addition) , - (subtraction) \*( Multiplication )  
/ (division – Quotient of the division)  
% (Modulus – Remainder of the division)

**24.What does the modulus operator % do?**

- It gives the remainder of the division  
Ex let **x=5; x%2;** the result is **1**

**25. What will be the result of 8.5 % 2?**

- Ans: Error** occurs because Modulus operator % **cannot** be used on **floating** point data.

**26.Write short note on Increment and Decrement Operators.**

- Increment ( ++ ) Operator and Decrement ( -- ) Operator are **unary** operators
- The Increment ( ++ ) Operator **adds 1** to its operand
- The Decrement ( -- ) Operator **subtracts 1** to its operand.
  - x++ is the same as x = x+1;  
It adds 1 to the present value of x
  - ++ x is the same as x = x+1;  
It adds 1 to the present value of x
  - x-- is the same as to x = x-1;  
It subtracts 1 from the present value of x
  - x is the same as to x = x-1;  
It subtracts 1 from the present value of x

**Example:**

let n=2	
Prefix	Postfix
<b>a=++n</b> First increment the value of <b>n</b> by <b>1</b> and then <b>a</b> gets the value. value of <b>a=3</b> , <b>n=3</b>	<b>a=n++</b> First <b>a</b> gets the value of <b>n</b> , then increment it by <b>1</b> value of <b>a=2</b> , <b>n=3</b>
<b>a=--n</b> *First decrements the value of <b>n</b> by <b>1</b> *then <b>a</b> gets the value *value of <b>a=1</b> , <b>n=1</b>	<b>a=n--</b> *First <b>a</b> gets the value of <b>n</b> , *then decrement it by <b>1</b> *value of <b>a=2</b> , <b>n=1</b>

**27.** Assume that **R** starts with value **35**. What will be the value of **S** from the following expression? **S=(R--)+(++R)**

**Result: S = 70**

**28.** What will be the value of **j = -- k + 2k**. if **k** is **20** initially? **Result j = 57**

**29.** What will be the value of **p = p \* ++j** where **j** is **22** and **p = 3** initially? **Result is 69**

**30.Write short note on Relational Operators**

- Relational operators are **binary** operators.
- Relational operators are used to **compare** numeric values
- A **relational expression** is consists of any two operands with a relational operator Ex. **10 > 5**
- The result will be a **Boolean value**(1(TRUE)or 0(FALSE) ) They are **> , < , <= , >= , == , !=**

**31.Write short note on Logical Operators**

- Logical operators **combine** the results of one or more **conditions**.
- C++ provides **three** logical operators.
  - AND(&&), OR(||)** are binary operators
  - NOT (!)** is an unary operator

**AND(&&)** – The result is **1 (TRUE)** when **all** condition are **1 (TRUE)**

**OR(||)** - The result is **1 (TRUE)** when **atleast** one condition is **1 (TRUE)**

**NOT (!)** – it inverts , The result is **1 (TRUE)** when condition is **False** and vice versa.

Ex. if a = 5, b = 6, c = 7;

Expression	Result
(a<b) && (b<c)	1 (True)
(a>b) && (b<c)	0 (False)
(a<b)    (b>c)	1 (True)
!(a>b)	1 (True)

**32. Write short note on Bitwise Operators**

- Bitwise operators perform bit-by-bit operation.
- In C++, there are three kinds of bitwise operators, which are:
  - Logical bitwise operators
  - Bitwise shift operators
  - One's complement operators

**33.Logical bitwise operators:**

- **& Bitwise AND (Binary AND)**




The result is 1 (TRUE) when all operands are 1 (TRUE)

- **| Bitwise OR (Binary OR)**

The result is 1 (TRUE) when atleast one operand is 1 (TRUE)

- **^ Bitwise Exclusive OR (Binary XOR)**

If **both** are **same**, it will return **0 (False)**.

Operator	Operation	Result																											
<<	a << 3	<table><tr><td>a</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>1</td><td>1</td><td>1</td></tr><tr><td colspan="9"></td></tr><tr><td>a &lt;&lt; 3</td><td>0</td><td>1</td><td>1</td><td>1</td><td>1</td><td>0</td><td>0</td><td>0</td></tr></table> <p><math>(a \ll 3) = 01111000_2 = 120_{10}</math></p>	a	0	0	0	0	1	1	1	1										a << 3	0	1	1	1	1	0	0	0
		a	0	0	0	0	1	1	1	1																			
																													
a << 3	0	1	1	1	1	0	0	0																					
>>	a >> 2	<table><tr><td>a</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>1</td><td>1</td><td>1</td></tr><tr><td>a &gt;&gt; 2</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>1</td><td>0</td><td>0</td></tr></table> <p><math>(a \gg 2) = 00000110_2 = 3_{10}</math></p>	a	0	0	0	0	1	1	1	1	a >> 2	0	0	0	0	1	1	0	0									
		a	0	0	0	0	1	1	1	1																			
		a >> 2	0	0	0	0	1	1	0	0																			

**34. Write the Truth Table for bitwise operators**

A	B	A & B	A   B	A ^ B
1	1	1	1	0
1	0	0	1	1
0	1	0	1	1
0	0	0	0	0

**35.If a = 65, b=15 find the value of &, | and ^**

In binary 65 = 1000001 in 8 bits : 0100 0001

15 = 1111 in 8-bits = 00001111

Operator	Operation	Result								
&	a & b	a	0	1	0	0	0	0	0	1
		b	0	0	0	0	1	1	1	1
		a & b	0	0	0	0	0	0	0	1
		(a&b) = 0000 0001 <sub>2</sub> = 1 <sub>10</sub>								
	a   b	a	0	1	0	0	0	0	0	1
		b	0	0	0	0	1	1	1	1
		a   b	0	1	0	0	1	1	1	1
		(a b) = 01001111 <sub>2</sub> = 79 <sub>10</sub>								
^	a ^ b	a	0	1	0	0	0	0	0	1
		b	0	0	0	0	1	1	1	1
		a ^ b	0	1	0	0	1	1	1	0
		(a^b) = 0100 1110 <sub>2</sub> = 78 <sub>10</sub>								

**36. The Bitwise shift operators:**

There are two bitwise shift operators in C++,

1.Shift left (<<)      2. Shift right (>>).

**Shift left (<<) :**

- The value of the **left operand** is moved to **left** by the number of bits **specified** by the **right** operand.
- **Right operand** should be an **unsigned** (positive)integer. Ex **a<<3;**

**Shift right (>>) :**

- The value of the **left operand** is moved to **right** by the number of bits **specified** by the **right** operand.
- Right operand should be an **unsigned** (positive)integer. Ex. **a>>3;**

**37.If a =15;find a<<3 and a>>2**

Equivalent binary value of a is 0000 1111

**38. The Bitwise one's complement operator:**

- The bitwise One's complement operator **~ (Tilde)**
- It **inverts**, that is, all 1's become 0 and all 0's become
- It is an **unary** operator.

**39.If a =15; find ~a.**

Equivalent binary values of a is 0000 1111

Operator	Operation	Result								
~	(~a)	a	0	0	0	0	1	1	1	1
		(~a)	1	1	1	1	0	0	0	0
		(~a) = 1111 0000 <sub>2</sub> = -16 <sub>10</sub>								

**40. Define Assignment Operator.**

- It is used to **assign** a value to a variable.
- = is commonly used as the assignment operator.
- This operator **copies** the **value** at the **right** side to the **left** side **variable**.
- It is also a **binary** operator.

Syntax : **Variable = Value \ constant \ expression ;**

Ex. **a = 10;**

**41.What are Shorthand assignment operators?**

- +=, -=, \*=, /=, % =

**42. Define Conditional Operator:**

- This is a **Ternary** Operator.
  - **?:** is a **conditional** Operator.
  - In C++, there is **only one conditional** operator is used.
  - This operator is used as an alternate to **if ... else**
- Syntax : **( Condition ) ? Statement 1 : Statement 2 ;**
- If condition is TRUE Statement 1 is executed.
  - If condition is FALSE Statement 2 is executed

**43. Define comma operator.**

- **Comma ( , )** is an operator in C++ used to **together** several expressions.
- The group of expression **separated** by comma.
- It is evaluated from **left to right**.

**44.What is sizeof operator or Define compile time operator**


- This is called as compile time operator.
- It returns the **size of a variable** in bytes.
- Ex. `int x = 10; cout << sizeof(a);`

Output : 2 bytes

**45.What is Association or Define order of precedence.**

- Operators are executed in the order of precedence.
- The operands and the operators are grouped in a specific **logical way** for evaluation.
- This **logical grouping** is called as an Association.

**46.Order of precedence of C++ Operators**

() []	Operators within parenthesis are performed first	Higher
++, --	Postfix increment / decrement	
++, --	Prefix increment / decrement	
*, /, %	Multiplication, Division, Modulus	
+, -	Addition, Subtraction	
<, <=, >, >=	Less than, Less than or equal to, Greater than, Greater than or equal to	
==, !=	Equal to, Not equal to	
&&	Logical AND	
	Logical OR	
?:	Conditional Operator	
=	Simple Assignment	
+=, -=, *=, /=	Shorthand operators	Lower
,	Comma operator	

**47.What is Punctuators ?**

- Punctuators are **symbols**, which are used as **delimiters**, while constructing a C++ program.
- They are also called as **"Separators"**.
- { }** -curly braces indicate the **start** and the **end** of a block of code. Ex. `{int a; a=10;}`
- ( )** Parenthesis indicate **function calls** and **function parameter**. Ex `add(); main()`
- [ ]** Square brackets indicates **arrays** ex.`int a[5];`
- ,** comma is used as **separator** ex, `int a,b,c;`
- ;** every executable statement in C++ should **terminate** with a semicolon ex. `int a;`
- :** colon is used to **label** a statement. Ex. `private:`
- //** single line comment
- /\* \*/** Multi line comment

**Evaluate Yourself....**

- 1.What is use of operators? 20.
  2. What are binary operators? Give examples arithmetic binary operators.23.
  3. What does the modulus operator % do? 24.
  4. What will be the result of `8.5 % 2`?
  8. Give that `i = 8, j = 10, k = 8`, What will be result of the following expressions?  
(i) `i < k` (ii) `i < j` (iii) `i > = k` (iv) `i = = j` (v) `j ! = k`
- Result: i) 0 ii)0 iii)1 iv)0 v)1**
9. Write an expression involving a logical operator to test, if marks are 75 and grade is 'A'.

- If `(marks > 75) && ( marks == 75)`  
`cout<<"Grade is A";`

**10.Define order of precedence**

- The order of precedence is used to determine how an expression involving more than one operator is evaluated.

**11. What will be the order of evaluation for the following expressions?**

- (i) `i + 3 >= j - 9`                      (ii) `a +10 < p - 3 + 2 q`

*1/0 Operators*

**48. What is the use of a header file or Library file?**

- A header file comprises of all standard declarations and definitions for **predefined functions**
- One can include the header file in the program by using a **preprocessor** directive (**#**) .
- It instructs the compiler to do the required job.

Syntax : **# include <header file>**

Ex **#include<iostream.h>**

**49.What are the pre processor operators?**

- **#** and **##** are pre processor operators
- The symbol **#** is a directive for the pre-processor.
- These are processed before the compilation process begins.

Ex. **#include<iostream>**

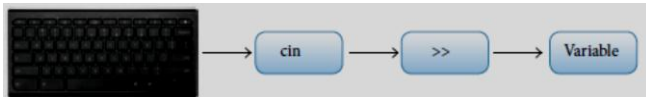


**50. Define Input operator or define Input statement .**

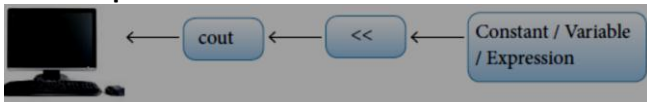
- The operator >> is called as “Stream extraction” or “get from” operator.
- >> operator to perform input operation.
- It **extracts** the **value** through the **keyboard** and assigns it to the **variable** on its right.
- It is a **binary** operator.
- The **first** operand is the pre-defined identifier **cin** that identifies keyboard as the input device.
- The **second** operand must be a **variable**.

Syntax : **cin>>variable;** ex. **cin>>a;**

**cin>>var1>>var2>>var3;** ex. **cin>>a>>b;**

**51. Define Output Operator: or output statement?**

- << is called the “Stream insertion” or “put to” operator.
- << operator to perform output operation.
- It is used to **send** the string or the values of the **variables** on its right to the **object** on its left.
- << is a **binary** operator.
- The **first** operand is the pre-defined identifier **cout** that identifies **monitor** as the standard output object.
- The **second** operand may be a **constant**, **variable** or an **expression**



Syntax : **cout<<variable;** Ex. **cout<<a;**

**For yourself:**

- To display the contents of the variable:  
Syntax : **cout<<variable;** Ex. **cout<<a;**
- To display the Message only  
Syntax : **cout<<"Message";** Ex. **cout<<"INDIA";**
- To display the Escape Sequence only  
Syntax : **cout<<"\escape seq.";** Ex. **cout<<"\n";**
- To display the Message with Escape Sequence  
Syntax : **cout<<"Message esc.seq.";**  
Ex. **cout<<"INDIA\n";**
- To display the result of expression  
**cout<<a+b;**
- To display the constant or data.  
**cout<<7;** or **cout<<2+3;**
- To display Message escape sequence variable  
**cout<<"Message"<<"\n"<<variable;**  
ex. **cout<< " The Result is.."<< '\t' << c;**

**52. What is the use of namespace std; or Define namespace std;**

- It tells the **compiler** to use standard namespace.
- Namespace **collects identifiers** used for class ,object and variables.
- Namespace provide a method of **preventing name conflicts** in large projects.
- It is introduced by the **ANSI C++ standards committee**.

**53. What are the importance of main() function?**

- Every C++ program **must have** a main function.
- The main() function is the **starting** point where all C++ programs **begin** their execution.
- The **executable** statements should be **inside** the **main()** function.

*Execute C++ program*

**54. How to creating and executing a C++ program****(1) Creating Source code**

- **Typing and editing** the valid C++ code as per the rules followed by the C++ Compiler.

**(2) Saving source code with extension .cpp**

- After typing, the source code should be saved with the **extension .cpp**

**(3) Compilation**

- In compilation, compiler links the **library files** with the **source code** and verifies each and every line of code.
- If any mistake or error is found, it will inform you to make corrections.
- If there are no errors, it **translates** the **source code** into machine readable **object** file with an **extension .obj**

**(4) execution**

- In this stage, the **object** file becomes an **executable** file with **extension .exe**.
- An executable file, can run your application without the help of any compiler or IDE.

**55. What is IDE in C++?**

- **Integrated Development Environment (IDE)** makes it easy to create, compile and execute a C++ program.

**List some Familiar C++ Compilers with IDE**

Compiler	Availability
Dev C++	Open source
Geany	Open source
Code::blocks	Open source
Code Lite	Open source
Net Beans	Open source
Digital Mars	Open source
Sky IDE	Open source
Eclipse	Open source

**56.Explain how to work with Dev C++?**

- **Double click** Dev C++ icon to open IDE
- To create a source file,
- Select **File → New → Source file** or Press **Ctrl + N**.
- After save, Click **Execute → Compile and Run** or press **F11** key.
- If your program contains any error, it displays the errors under compile log.
- If your program is **without any error**, the display will appear as follows.
- After successful compilation, output will appear in **output console**.

**57.Define Syntax error**

- Syntax error occur when grammatical rules of C++ are violated.
- Ex. `cout<<"INDIA"` - it will throw an error because this statement does not end with a semicolon.

**58.Define semantic error or logical error .**

- It may be happened by wrong use of variable / operator /order of execution etc.
- Here program is grammatically correct but it contains some logical error.

**59.Define run time error.**

- A run time error occurs during the execution of a program.
- It occurs because of some illegal operation that takes place.

**Evaluate Yourself....**

- 1.What is meant by a token? Name the token available in C++. Ref.5
2. What are keywords? Can keywords be used as identifiers?ref.6
3. The following constants are of which type?  
(i) 39 (ii) 032 (iii) 0XCAFE (iv) 04.14
5. Assume n=10; what will be result of `n>>2`;
6. Match the following

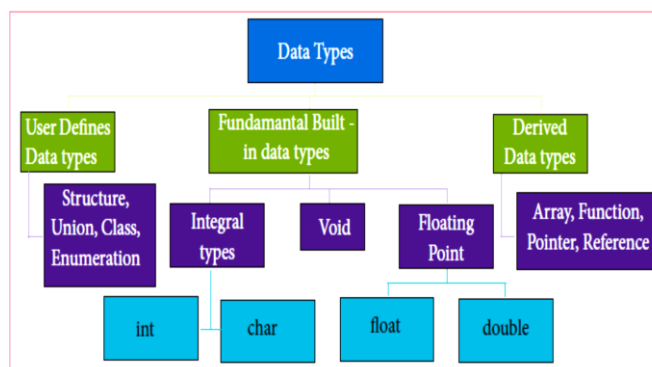
A	B
(a) Modulus	(1) Tokens
(b) Separators	(2) Remainder of a division
(c) Stream extraction	(3) Punctuators
(d) Lexical Units	(4) get from

- 1.Describe the differences between keywords and identifiers? Ref. 6 and 7
2. Is C++ case sensitive? What is meant by the term "case sensitive"?
  - C++ is **case sensitive** as it treats upper and lower-case characters differently.
3. Differentiate "=" and "==".
4. Assume a=10, b=15; What will be the value of `a^b`?
5. What is the difference between "Run time error" and "Syntax error"?
6. What are the differences between "Logical error" and "Syntax error"?
7. What is the use of a header file?
8. Why is main function special?
9. Write two advantages of using include compiler directive.
10. Write the following in real constants.  
(i) 15.223 (ii) 211.05 (iii) 0.00025

*Data Types, Variables and Expressions***60.What are the categories of Data types?**

In C++, the data types are classified as three main categories

- (1) Fundamental data types
- (2) User-defined data types and
- (3) Derived data types

**61.Write about Fundamental data types.**

- Fundamental (atomic) data types are **predefined** data types available with C++.
- There are five fundamental data types in C++: **char, int, float, double** and **void**.

**int data type:**

- Integers are whole **numbers without** any fraction.
- Integers can be **positive** or **negative**.
- Integer data type **accepts** and **returns only integer numbers**.
- .Ex. `int a = 5; int x = -4;`  
`int a = 4.5` (it will accept only 4)

**char data type:**

- Character data type **accepts** and **returns** all valid **ASCII characters**.
- Character data type is an integer type.
- All the characters are represented in memory by their associated **ASCII Codes**.

Ex. 'A' represent 65

**float data type:**

- Float data type accepts **floating** point values with **6** digits of precision.
- It includes **integer** portion, a **decimal** point, **fractional** portion and an **exponent**.

Ex. float a = 3.14;

**There are two advantages of using float data types.**

- (1) They can represent values between the integers.
- (2) They can represent a much greater **range** of values.

**Disadvantage**

- Floating point operations are **slower** than integer operations.

**double data type**

- double data type accepts **double precision** floating point numbers with **14** digits of precision.
- This type occupies **double the space than float type**

Ex. double a = 3.456744;

**void data type**

- It is used as a **return** type for functions that do not return any value.
- The void data type specifies **an empty** set of **values**. It is used as a return type for functions that do not **return any value**.
- Ex. void main(), void add()

**What are the advantages and disadvantages of float data type? Ref. above answer****Evaluate Yourself....**

1. What do you mean by fundamental data types? **Ref. 61**

2. The data type char is used to represent characters. then why is it often termed as an integer type?

- Character data type **accepts** and **returns** all valid **ASCII characters**.
- All the characters are represented in memory by their associated **ASCII Codes**.
- ASCII** codes are an integer type.

3. What is the advantage of floating point numbers over integers? **Ref. 61**

4. The data type double is another floating point type. Then why is it treated as a distinct data type?

- Because more fractions accommodated in double than in float type

5. What is the use of void data type? **Ref. 61**

**62. What are the Memory representation of Fundamental Data types in C++?**

- C++ compiler allocates specific memory space for each and every **data**.
- Every data is stored inside the computer memory in the form of **binary digits**

Data type	Space in memory		Range of value
	in terms of bytes	in terms of bits	
char	1 byte	8 bits	-127 to 128
int	2 bytes	16 bits	-32,768 to 32,767
float	4 bytes	32 bits	$3.4 \times 10^{-38}$ to $3.4 \times 10^{38} - 1$
double	8 bytes	64 bits	$1.7 \times 10^{-308}$ to $1.7 \times 10^{308} - 1$

**63. Write a short note on modifiers Or qualifiers.**

- Modifiers can be used to **modify** (expand or reduce) the **memory allocation** of any **fundamental** data type.
- They are also called as **Qualifiers**

There are four modifiers used in C++. They are:

- (1) signed
- (2) unsigned
- (3) long
- (4) short

**Integer type**

Data type		Space in memory		Range of value
		in terms of bytes	in terms of bits	
short	short is a short name for short int	2 bytes	16 bits	-32,768 to 32,767
unsigned short	an integer number without minus sign.	2 bytes	16 bits	0 to 65535
signed short	An integer number with minus sign	4 bytes	32 bits	-32,768 to 32,767
Both short and signed short are similar				
int	An integer may or may not be signed	2 bytes	16 bits	-32,768 to 32,767
unsigned int	An integer without any sign (minus symbol)	2 bytes	16 bits	0 to 65535
signed int	An integer with sign	2 bytes	16 bits	-32,768 to 32,767
Both short and int are similar				
long	long is short name for long int	4 bytes	32 bits	-2147483648 to 2147483647
unsigned long	A double spaced integer without any sign	4 bytes	32 bits	0 to 4,294,967,295
signed long	A double spaced integer with sign	4 bytes	32 bits	-2147483648 to 2147483647

integer type accepts only **2** bytes of data

long accepts **2** bytes



**char type**

Data type		Space in memory		Range of value
		in terms of bytes	in terms of bits	
char	Signed ASCII character	1 byte	8 bits	-128 to 127
unsigned char	ASCII character without sign	1 byte	8 bits	0 to 255
signed char	ASCII character with sign	1 byte	8 bits	-128 to 127

**Floating point type**

Data type		Space in memory		Range of value
		in terms of bytes	in terms of bits	
float	signed fractional value	4 bytes	32 bits	$3.4 \times 10^{-38}$ to $3.4 \times 10^{38}-1$
double	signed more precision fractional value	8 bytes	64 bits	$1.7 \times 10^{-308}$ to $1.7 \times 10^{308}-1$
long double	signed more precision fractional value	10 bytes	80 bits	$3.4 \times 10^{-4932}$ to $1.1 \times 10^{4932}-1$

**Memory allocation by Turbo C++ and Dev C++**

Data type	Memory size in bytes	
	Turbo C++	Dev C++
short	2	2
unsigned short	2	2
signed short	2	2
int	2	4
unsigned int	2	4
signed int	2	4
long	4	4

unsigned long	4	4
signed long	4	4
char	1	1
unsigned char	1	1
signed char	1	1
float	4	4
double	8	8
long double	10	12

**64.What is Number Suffixes in C++?**

- There are different suffixes for integer and floating point numbers.
- Suffix can be used to assign the same value as a different type.

Ex.To store 45 in **long int** suffix letter L : **45L**

Unsigned **int** : **45U** for float : **3.14F**

*Variables***65.Define variables.**

- Variables are **user-defined names**.
- Variables are **memory locations** in which the **values** are **stored**.
- Variables are also **identifiers**.
- These are called as **symbolic variables** because these are named locations.
- There are two values associated with a symbolic variable;
- They are R-value and L-value.
  - R-value is **data stored** in a memory location
  - L-value is the **memory address**.
- The **memory addresses** are in the form of **Hexadecimal** values
- Ex.Let **int a = 5** ; Here **Variable name** is **a** ;R value is **5** ; L value is **0x12b**(for example)

**66.How to Declare of Variables in C++?**

- Every variable should be declared before they are actually used in a program.
- Declaration is a process to instruct the compiler to **allocate memory** as per the **data type** along with the **variable name**.

Syntax : **Data type space variable name ;**

Ex. **int a;**

Declaration of more than one variable:

Syntax : **Data type space var1,var2,var3 ;**

Ex. **int a,b,c;** (comma is used to separating the variables).

**67. What is meant by Junk" or "Garbage"?**

- If you declare a variable **without any initial value**, variable will be occupied with some **unknown value**.
- These unknown values are called as "**Junk**" or "**Garbage**" values.

**68.How to initialize the variable in C++?**

- Assigning an **initial** value to a variable during its declaration is called as "**Initialization**".  
Suntax: **Data type space variable name = value;**  
Ex. **int a = 10;**
- Variables that are of the same type can be initialized in a single statement.  
**int x1 = -1, x2 = 1;**

**69.What is Dynamic Initialization?**

- A variable can be initialized during the execution of a program. It is known as "**Dynamic initialization**".  
For example, **int c = a+b;**

**70. Define Access modifier or qualifier or const qualifier.**

- **const** is the keyword used to declare a constant.
- The **const** qualifier specifies that the value of a **variable** will not change during the run time of a program.
- Any attempt to alter the value, an error message will be displayed as **"Cannot modify the const object"** in Turbo compiler .
- **"assignment of read only memory num"** in Dev C++.

**Evaluate Yourself....**

1. What are modifiers? What is the use of modifiers? **63.**
2. What is wrong with the following C++ statement:  
long float x; **there is no long float**
3. What is variable? Why a variable called symbolic variable? **65.**
4. What do you mean by dynamic initialization of a variable? Give an example. **69**
5. What is wrong with the following statement?  
**const int x;**
  - Error because **const** modifier should be used during initialization. Here value is not initialized.

**71. What is Reference variable?**

- A reference provides an **alias** for a previously defined variable.
  - Declaration of a reference consists of **base type** and an **&** (ampersand) symbol;
  - Reference variable name is assigned the value of a previously declared variable.
- Syntax:

**Data type & reference\_variable = original\_variable;**

Ex. int a=10; **int &b=a;**

- The output of **a** is **10** and the output of **b** is **10**

**72. What are Manipulators?**

- Manipulators are used to **format the output** of any C++ program.
- Manipulators are functions specifically designed to use with the **insertion (<<) and extraction (>>) operators**.
- Commonly used manipulators are: **endl, setw, setfill, setprecision and setf**.
- **endl** is a member of **iostream** header file.
- **setw, setfill, setprecision and setf** are members of **iomanip** header file.

**73. Explain some formatting output or Manipulators****1) endl (End the Line)**

- **endl** – Inserts a **new line** and **clean** the buffer
- **'\n'** - Inserts **only a new line**.

**2) setw ( )**

**setw** manipulator sets the **width of the field** assigned for the output.

Syntax:

**setw(number of characters)**

**3) setfill ( )**

- It is usually used after **setw**.

**cout << "\n H.R.A : " << setw(10) << setfill (0) << hra;**  
from above example ,

- **setw** creates a field to show the presented value,
- **setfill** is used to fill **un-occupied spaces** with **0**
- if **hra=1200**, The output will be, **0000001200**.

**4) setprecision ( )**

- This is used to display numbers with fractions in specific number of digits.

Syntax:

**setprecision (number of digits);**

Ex.

- float **hra = 1200.123;**
- **cout << setprecision (5) << hra;**
- The output will be **1200.1**(including fraction)
- **setprecision** can also be used to **set the number of decimal places to be displayed**.
- In order to do this task, you will have to set an **ios flag** within **setf()** manipulator.
- There are two types of flags i) **fixed** ii) **scientific**

Example

**cout.setf(ios::fixed);**

**cout << setprecision(2)<<0.1;**

- In the above statements, **ios flag** is set to **fixed** type;
- It prints the floating point number in fixed notation.  
So, **the output will be, 0.10**

**cout.setf(ios::scientific);**

**cout << setprecision(2) << 0.1;**

- In the above statements, **ios flag** is set to **scientific** type;
- it prints the floating point number in scientific notation. So, **the output will be, 1.00e-001**

**74. Explain the types of Expression in C++?**

There are seven types of expressions, and they are:

- (i) Constant Expression
- (ii) Integer Expression
- (iii) Floating Expression
- (iv) Relational Expression
- (v) Logical Expression
- (vi) Bitwise Expression
- (vii) Pointer Expression

**(i) Constant Expression**

- Consist only **constant** value Ex. **int a=10;**

**(ii) Integer Expression**

- Consist of **integer** and **character** values\ variables \ expression
- Produce **integer** results

Ex. **avg=sum/5;**

**(iii) Floating Expression**

- Consist of **floating** point values\ variables \ expression.
- Produce **floating** point results

Ex. `float a= 3.14*r*r;`

**(iv) Relational Expression**

- Consist of values\ variables with **relational** operators.
- Produce either True(1) or False(0).ex. `a>b`

**(v) Logical Expression**

- Consist of values\ variables with **Logical** operators.
- Produce either True(1) or False(0).
- Ex. `(a>b) && (a>c)`

**(vi) Bitwise Expression**

- Consist of values\ variables with **Bitwise** operators

**(vii) Pointer Expression**

- A Pointer is a variable that holds a memory address.
- Pointer declaration statements:

Syntax : `Data type space *variable;` ex. `int *a;`

**75.What is Type Conversion? Give examples**

The process of **converting** one fundamental type into another is called as **"Type Conversion"**.

C++ provides two types of conversions.

- (1) Implicit type conversion
- (2) Explicit type conversion.

**76.Define Implicit type conversion**

- An Implicit type conversion is performed by the **compiler** automatically.
- It is also called as **"Automatic conversion"**.
- Compiler converts smaller type into wider type which is called **Type Promotion**

Ex.`int a=6; float b=3.14; cout << a+b;`

The output is **9.14**(float is wider than int)

**77.Define Explicit type conversion ( What is Type casting?)**

- Explicit conversion is performed by the **programmer** in the program itself.
- Explicit conversion is converting of variables or expressions from one data type to another **specific** data type.
- It is called as **"type casting"**.

Syntax. : **(type-name) expression;**

Ex. `float varf=78.685; cout << (int) varf;`

output is **78**

**78. What is type promotion? Ref. 76.****Evaluate Yourself..**

1. What is meant by type conversion? **77**
2. How implicit conversion different from explicit conversion? **76 , 77**
3. What is difference between endl and \n? **73.**
4. What is the use of references? **71**

**5. What is the use of setprecision ( ) ? 73**

1. What is the difference between a keyword and an identifier?
2. What are literals in C++? How many types of literals are allowed in C++?
3. How many ways are there in C++ to represent an integer constant?
4. **What is the different between 'a' and "a" in C++?**
  - Character enclosed in **single quotes** are **character constant**
  - The **size** of 'a' is **1** character.
  - Character enclosed in **double quotes** are **string constant**
  - **\0 (nul)** used to terminate the string
  - The **size** of "a" is **2** character.
5. Who was developer of C++? Ref 1
6. What was the original name given to C++? Who gave the name "C++"?
7. What is meant by token? Name the token available in C++.
8. What are keywords? Can keywords be used as identifiers?
9. What is an identifier ? what are the rules to form an identifiers?
10. What are literals ?how many types of integer literals are available in c++?
11. How are integer constants represented in C++? Explain with example
12. What kind of constant are the following?  
14 , 011 , 4.324, 'a' , "rmk" , 0X2A
13. How are nongraphic characters represented in c++?
14. What kind of program element (token) are the following? 28 , a , for , // , +
15. Which escape sequence represent the newline & null character?
16. Which character is automatically added to string in C++?
17. What header file must you include with your source file to use *cout* and *cin*?
18. What are the main data types of C++?
19. Explain fundamental data types.
20. Why is char often treated as integer data type?
21. **Consider the following two C++ statements. Are they equivalent.. `char g=65; char g='A';` Yes both are equivalent ASCII code value of 'A' is 65**
22. **What is the different between 25L and 25**

25L	-	long integer constant
25	-	integer constant.