LESSON THREE

1Character set:

C++ has the letters and digits, as show below:

Uppercase:	A, B, C, . . ., Z
Lowercase:	a, b, c, . . ., z
Digits:	0, 1, 2, . . .,9

Special Characters: All characters other than listed treated as special characters for example:

+	-	*	/	^
(	[	{	}	]
)	<	=	>	, (Comma)
" (Double Conations)	. (Dot)	: (Colon)	; (Semicolon)	␣ (Blank Space)

In C++ language, upper case and lower case letters are distinct and hence there are 52 letters in all. For example **bag** is different from **Bag** which is different from **BAG**.

2Identifiers:

An **identifier** is a name given to some program entity, such as variable, constant, array, function, structure, or class. An identifier is a sequence of alphanumeric (alphabetic and numeric) characters, the first of which must be a letter, and can't contain spaces. The length of an identifier is machine dependent. C++ allows identifiers of up to **127 characters**.

A **variable** should not begin with a digit. C++ does not set a maximum length for an identifier. Some examples of valid identifiers are as follows:

My_name	(7 char.)
i	(1 char.)
B	(1 char.)

Examples of invalid identifiers are: 3ab

a()test ros sal

LESSON THREE

3Keywords:

The keywords are also identifiers but cannot be user defined, since they are reserved words. All the keywords should be in lower case letters. Reserved words cannot be used as variable names or constant. The following words are reserved for use as keywords:

Some of C++ Language Reserved Words:				
break	case	char	cin	cout
delete	double	else	enum	false
float	for	goto	if	int
long	main	private	public	short
sizeof	switch	true	union	void

4 Constants:

There are three types of constants: **string constants**, **numeric constants**, and **character constants**.

1. String Constants: A string constants are a sequence of alphanumeric characters enclosed in double quotation marks whose maximum length is 255 characters. In the following are examples of valid string constants: ("The result=", "RS 2000.00", "This is test program"). The invalid string constants are like: (Race, "My name, 'this').

2. Numeric Constants: Numeric constants are positive or negative numbers. There are four types of numeric constants: integer, floating point, hexadecimal, and octal.

Integer	Integer Short integer (short) Long integer (long)
Float	Single precision (float) Double precision (double) Long double
Hexa	Short hexadecimal Long hexadecimal
Unsigned	Unsigned char Unsigned integer Unsigned short integer Unsigned long integer
Octal	Short octal Long octal

LESSON THREE

(a) Integer constants: Do not contain decimal points: int x,y; shortint x,y; longint x,y;

- Integer data: size (16 or 32) fill in -2^{15} to $2^{15}-1$ for 16 bit and -2^{31} to $2^{31}-1$ for 32 bit.
- Short integer: fill in -2^{15} to $2^{15}-1$.
- Long integer: fill in -2^{31} to $2^{31}-1$.
- Unsigned: fill in (0 to 65635) for 16 bit and (0 to 4,294, 967, 295) for 32 bit.

(b) Floating point constants: Positive or negative numbers are represented in exponential form. The floating point constant consists of an optionally (signed) integer or fixed point number (the mantissa) followed by the letter E and e and an optionally signed integer (the exponent). Ex. (9010e10, 77.11E-11).

- Float 4 bytes.
- Double 8 bytes.
- Long double 12 or 16.

(c) Hexadecimal constants: Hexadecimal numbers are integer numbers of base 16 and their digits are 0 to 9 and A to F.

(d) Octal constants: Octal numbers are numbers of base 8 and their digits are 0 to 7.

3. Character Constants: A character represented within single quotes denotes a character constant, for example ‘A’, ‘a’, ‘:’, ‘?’, etc…

Its maximum size is 8 bit long, signed, and unsigned char are three distinct types.

Char x; char x,y,z;

LESSON THREE

The backslash (\) is used to denote non graphic characters and other special characters for a specific operations such as:

Special Escape Code:	
Escape Code	**Description**
\n	New line. Position the screen cursor to the beginning of the next line.
\t	Horizontal TAB (six spaces). Move the screen cursor to the next tab stop.
\r	Carriage return. Position the cursor to the beginning of the current line, do not advance to the next line.
\a	Alert. Produces the sound of the system bell.
\b	Back space
\\	Backslash. Prints a backslash character.
\f	Form feed
\v	Vertical tab
\”	Double quote. Prints a (“) character.
\o	Null character
\?	question mark
\ooo	Octal value
\xhhh	Hexadecimal value

LESSON THREE

5.C++ operators:

C++ operators	Arithmetic operators	
	Assignment operators	
	Comparison and logical operators	Relational,equality,logical
	Bit wise logical operators	
	Special operators	Unary, ternary, comma Scope, new&delete, other

1. **Arithmetic operators:** These operators require two variables to be evaluated:

   + addition - subtraction ***** multiplication

   / division **%** modula (remainder of an integer division)

The division result are:

Integer / integer = integer ► 39/7=5 Integer
/ float = float ► 39/7.0 =5.57
float / integer = float ► 39.0/7 =5.57
float / float = float ► 39.0/7.0=5.57
while 39%5=7, since 39=7*5+4
Arithmetic operators as per precedence: ( ) for grouping the variables.

- Unary for negative number.

* / multiplication & division.

+ - addition and subtraction.

Example:

X+y*X-Z, where X=5, Y=6, and Z=8. 5 + (6*5)-8 →

(5+30)-8 → 35-8 → 27

2. **Assignment Operators:** The operatonal assignment operator has the form:

Variable = variable operator expression;

Ex: x=x+5; y=y*10;

The operational assignment operator can be written in the following form:

Variable operator = expression

Ex: x+=5; y*=10;

It is used to assign back to a variable, a modified value of the present holding:

=	Assign right hand side (RHS) value to the left hand side (LHS).
+=	Value of LHS var. will be added to the value of RHS and assign it back to the var. in LHS.
-=	Value of RHS var. will be subtracted to the value of LHS and assign it back to the var. in LHS.
*=	Value of LHS var. will be multiplied to the value of RHS and assign it back to the var. in LHS.
/=	Value of LHS var. will be divided to the value of RHS and assign it back to the var. in LHS.
%=	The remainder will be stored back to the LHS after integer division is carried out between the LHS var. and the RHS var.
>>=	Right shift and assign to the LHS.
<<=	Left shift and assign to the LHS.
&=	Bitwise AND operation and assign to LHS
|=	Bitwise OR operation and assign to LHS
~=	Bitwise complement operation and assign to LHS

LESSON THREE

This is a valid statements:

A=b=c+4; C=3*(d=12.0/x);

Exercise:

Rewrite the equivalent statements for the following examples, and find it results.
Assume: X=2 , Y=3 , Z=4 , V=12 , C=8.

Example	Equivalent Statement	Result
X += 5	X = X + 5	X ← 7
Y -= 8	Y = Y - 8	Y ← -5
Z *= 5	Z = Z * 5	Z ←
V /= 4		V ←
C %= 3		C ←

3. **Comparison and logical operators:** It has three types relational operators, equality operators, and logical operators.

(a) Relational operators: < less than, > greater than, <= less than or equal, >= greater than or equal, an expression that use relational operators return the value of one if the relational is TRUE ZERO otherwise.

Ex: 3 > 4 → **false,** 6 <=2 →**false,** 10>-32 → **true,** (23*7)>=(-67+89) → **true**

(b) Equality operators: == equal to , **!=** not equal to

Ex: a=4, b=6, c=8. A==b→**false,** (a*b)!=c→**true,**'s'=='y' →**false**.

(c) Logical operators: The logical expression is constructed from relational expressions by the use of the logical operators **not**(!), **and**(&&), **or**(||).

AND (&&) Table:		
A	**B**	**A && B**
T	T	T
T	F	F
F	T	F
F	F	F

AND (&&) Table:		
A	**B**	**A && B**
1	1	1
1	0	0
0	1	0
0	0	0

LESSON THREE

OR (| |) Table:		
A	B	A | | B
T	T	T
T	F	T
F	T	T
F	F	F

NOT (!) Table:	
A	!A
T	F
F	T

OR (| |) Table:		
A	B	A | | B
1	1	1
1	0	1
0	1	1
0	0	0

NOT (!) Table:	
A	!A
1	0
0	1

Examples:

Example 1:

a=4, b=5, c=6

(a<b)&&(b<c)	(a<b) | | (b>c)	!(a<b) | | (c>b)	a<b) | | (b>c)&&(a>b) | | (a>c)
T && T	T | | T	!(T) | | T	T | | F && F | | F
T	T	F | | T	T | | F | | F
		T	T | | F
			T

Example 2:

Assume: X=0, Y=1, Z=1.

Find the following expression:

M = ++X || ++Y && ++Z

M = ++X || ++Y && ++Z

= 1 || (2 && 2)

= T || (T && T)

= T || T

= T

= 1