UNIT-2
Introduction to C++

C++ CHARACTER SET

Character set is asset of valid characters that a language can recognize . A character can represents any letter, digit, or any other sign . Following are some of the C++ character set.

LETTERS	A to Z and a to z
DIGITS	0 -9
SPECIAL SYMBOLS	+ -* ^ \ [] {} = != < > . ‘ ‘ ; : & #
WHITE SPACE	Blankl space , horizontal tab (- >), carriage return , Newline, Form feed.
OTHER CHARACTERS	256 ASCII characters as data or as literals.

TOKENS:

The smallest lexical unit in a program is known as token. A token can be any keyword,Identifier,Literals, Puncutators, Operators.

KEYWORDS :

These are the reserved words used by the compiler. Following are some of the Keywords.

auto	continue	float	new	signed	volatile
short	long	class	struct	else	inline
delete	friend	private	typedef	void	template
catch	friend	sizeof	union	register	goto

IDENTIFIERS:

An arbitrary name consisting of letters and digits to identify a particular word.C++ is case sensitive as nit treats upper and lower case letters differently. The first character must be a letter . the underscore counts as a letter

Pen time580 s2e2r3 _dos _HJI3_JK

LITERALS:

The data items which never change their value throughout the program run. There are several kind of literals:

- Integer constant
- Character constant
- Floating constant
- String constant.

Integer constant :

Integer constant are whole numbers without any fractional part. An integer constant must have at least one digit and must not contain any decimal point. It may contain either + or _. A number with no sign is assumed as positive.

e.g 15, 1300, -58795.

Character Constant:

A character constant is single character which is enclosed within single quotation marks.

e.g ‘ A’

Floating constant:

Numbers which are having the fractional part are referred as floating numbers or real constants. it may be a positive or negative number. A number with no sign is assumed to be a positive number.

e.g 2.0, 17.5, -0.00256

String Literals:

It is a sequence of letters surrounded by double quotes. E.g "abc".

PUNCTUATORS:

The following characters are used as punctuators which are also know as separators in C++

[ ] { } ( ) , ; : * ……. = #

Puntuator	Name	Function
[]	Brackets	These indicates single and multidimensional array subscripts
()	Parenthesis	These indicate function calls and function parameters.
[ ]	Braces	Indicate the start and end of compound statements.
;	Semicolon	This is a statement terminator.
:	Colon	It indicates a labeled statement
*	Asterisk	It is used as a pointer declaration
…	Ellipsis	These are udes in the formal argument lists of unction prototype to indicate a variable number of arguments.
=	Equal to	It is used as an assigning operator.
#	Pound sign	This is used as preprocessor directives.

OPERATORS:

These are those lexical units that trigger some computation when applied to variables and other objects in an expression. Following are some operators used in C++

Unary operators: Those which require only one operand to trigger. e.g. & , + , ++ , -- ! .

Binary operators: these require two operands to operate upon. Following are some of the Binary **operators.**

Arithmatic operators :

+	Addition
_	substraction
A*	Multiplication
/	Division
%	Remainder.

Logical Operators :

&& - logical AND || - Logical OR

Relational Operator:

< less than
a> Greater than
<= Less than equal to.
>= greater than equal to.
== equal to.
!= not equal to.

Conditional operator: ? (question) : (colon)

Assignment Operator:

= assignment operator
*= Assign Product.
/= Assign quotient
%= assign Remainder
&= Assign bitwise AND
^= Assign bitwise XOR.
|=Assign bitwise OR

Conditional operator (?)

The conditional operator evaluates an expression returning a value if that expression is true and a different one if the expression is evaluated as false. Its format is:

condition ? result1 : result2
e.g 7==5 ? 4 : 3 // returns 3, since 7 is not equal to 5.

Comma operator (,)

The comma operator (,) is used to separate two or more expressions that are included where only one expression is expected. When the set of expressions has to be evaluated for a value, only the rightmost expression is considered.

For example, the following code:

```
a = (b =3 , b +2 );
```

Would first assign the value 3 to b, and then assign b+2 to variable a. So, at the end, variable a would contain the value 5 while variable b would contain value 3.

Explicit type casting operator

Type casting operators allow you to convert a datum of a given type to another. There are several ways to do this in C++. The simplest one, which has been inherited from the C language, is to precede the expression to be converted by the new type enclosed between parentheses ( ) :

```
int i;
float f =3014;
i = ( int ) f;
```

The previous code converts the float number 3.14 to an integer value (3), the remainder is lost. Here, the typecasting operator was (int). Another way to do the same thing in C++ is using the functional notation: preceding the expression to be converted by the type and enclosing the expression between parentheses:

```
i  = int (f );
```

Both ways of type casting are valid in C++.

sizeof()

This operator accepts one parameter, which can be either a type or a variable itself and returns the size in bytes of that type or object:

```
a= sizeof (char);
```

This will assign the value 1 to a because char is a one-byte long type.
The value returned by sizeof is a constant, so it is always determined before program execution.

Input Output (I/O) In C++

The cout Object:

The cout object sends to the standard output device. cout sends all out put to the screen i.e monitor.

The syntax of **cout** is as follows:

```
cout<< data;.
```

e.g

```
cout<< a ;          ( here a can be any variable)
```

The cin operator :

The cin operator is used to get input from the keyboard. When a program reaches the line with cin, the user at the keyboard can enter values directly into variables.

The syntax of **cin** is as follows:

```
cin>> variablename
```

e.g

```
cin>> ch;      ( here ch can be any variable)
```

- ***Basic structure of a C++ program:***

Following is the structure of a C++ program tht prints a string on the screen:

```
#include<iostream.h>
 void main ()
{
cout<<" Study material for Class XI";
}
```

The program produces following output:

Study material for Class XI

The above program includes the basic elements that every C++ program has. Let us check it line by line

#include<iostream.h> : This line includes the preprocessor directive include which includes the header file iostream in the program.

void main () :this line is the start of compilation for this program. Every C++ programs compilation starts with the **main (). void** is the keyword used when the function has no return values.

{ : this is the start of the compound block of main ().

cout<<" Study material for class XI";: this statement prints the sequence of string **" Study material for class XI"** into this output stream i..e on monitor.

Every statement in the block will be terminated by a semicolon (;) which specifies compiler the end of statement.

COMMENTS in a C++ program.:

Comments are the line that compiler ignores to compile or execute. There are two types of comments in C++.

1. **Single line comment:** This type of comment deactivates only that line where comment is applied. Single line comments are applied with the help of " //" .

   e.g // cout<<tomorrow is holiday

   the above line is proceeding with // so compiler wont access this line.

2. **Multi line Comment :** This Type of comment deactivates group of lines when applied. This type of comments are applied with the help of the operators "/*" and "*/ ". These comment mark with /* and end up with */. This means every thing that falls between /*and */ is considered even though it is spread across many lines.

```
e.g   #include<iostream.h>
      int main ()
{
  cout<< "  hello world";
 /* this is the program to print hello world
      For demonstration of comments */
}
```

In the above program the statements between **/*** and ***/** will be ignored by the compiler.

CASCADING OF OPERATOR:

When shift operators (**<<** and **>>**) are used more than one time in a single statement then it is called as cascading of operators.

e.g cout<< roll<< age<< endl;

DATATYPES IN C++:

A datatype is just an interpretation applied to a string of bytes. Data in C++ are of two types:

1.Simple /Fundamental datatypes .

2.Structures/Derived datatypes.

Simple /Fundamental data types:

When programming, we store the variables in our computer's memory, but the computer has to know what kind of data we want to store in them, since it is not going to occupy the same amount of memory to store a simple number than to store a single letter or a large number, and they are not going to be interpreted the same way.

The memory in our computers is organized in bytes. A byte is the minimum amount of memory that we can manage in C++. A byte can store a relatively small amount of data: one single character or a small integer (generally an integer between 0 and 255). In addition, the computer can manipulate more complex data types that come from grouping several bytes, such as long numbers or non-integer numbers.

Next you have a summary of the basic fundamental data types in C++, as well as the range of values that can be represented with each one:

Name	Description	Size	Range
char	Character or small integer.	1byte	signed: -128 to 127 unsigned: 0 to 255
short int (short)	Short Integer.	2bytes	signed: -32768 to 32767 unsigned: 0 to 65535
int	Integer.	4bytes	signed: -2147483648 to 2147483647 unsigned: 0 to 4294967295
long int (long)	Long integer.	4bytes	signed: -2147483648 to 2147483647 unsigned: 0 to 4294967295
float	Floating point number.	4bytes	+/- 3.4e +/- 38 (~7 digits)
double	Double precision floating point number.	8bytes	+/- 1.7e +/- 308 (~15 digits)
long double	Long double precision floating point number.	8bytes	+/- 1.7e +/- 308 (~15 digits)

Derived Data Types:

The datatypes which are extracted / derived from fundamental data types are called derived datatypes. These datatypes can be derived by using the declaration operator or punctuators for e.g Arrays, function, Pointer, Class , Structure, union etc.

Class : A class represents a group of similar objects. To represent class in C++ it offers a user defined datatypes called CLASS .Once a Class has been defined in C++, Object belonging to that class can easily be created. A Class bears the same relationship to an object that a type does to a variable.

Syntax of CLASS:

```
Class  class_name
{
Private:
        Data members 1
```

```
                "
        Data members n
        Member functions  1
                "
        Member functions  n
Public:
        Data members 1
                "
        Data members n
        Member functions  1
                "
        Member functions  n
};//end of class
```

Class name object of Class; // creating an object of class. Private and Public are the access specifiers to the class.

STRUCTURE:

A Structure is a collection of variables of different data types referenced under one name .It also may have same data types. The access to structure variables is by default global i.e they can be accessed publicly throughout the program.
Syntax of structure.

```
 struct  structure_name
{
   Structure variable 1;
Structure variable n;
};  // end of structure
```

Structure_name structure object // creating object variable of structure.

```
 e.g
struct  student
{
    int roll;
    float marks ;
   };
```

Student s;

Access to structure variables
Structure variable can be accessed by their objects only as shown below

structure object_name. variable

e.g

student . roll

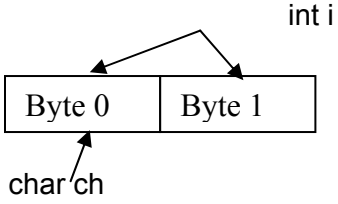
here student is the structure and roll is the member of the structure.

UNION :

A memory location shared between two different variables of different datatypes at different times is know as Union. Defining union is similar as defining the structure.

Syntax of Union :

```
   union  show
{
        int I;
        char ch;
};
```



Union show obj;

References:

A reference is an alternative name for an object. A reference variable provides an alias for a previously defined variable. A reference declaration consists of base type , an & (ampersand), a reference variable name equated to a variable name .the general syntax form of declaring a reference variable is as follows.

Type & ref_variable = variable_name;

Where is type is any valid C++ datatype, ref_variable is the name of reference variable that will point to variable denoted by variable_name.

e.g int a= 10;
int &b= a;

then the values of **a is 10** and the value of **b is also 10;**

Constant :

The keyword const can be added to the declaration of an object to make that object constant rather than a variable. Thus the value named constant can not be altered during the program run.

Syntax:-

const type name=value;

Example:-

const int **uage=50;** // it declares a constant named as **uage** of type integer that holds value 50.

Preprocessor Directives:

#include is the preprocessor directive used in C++ programs. This statement tells the compiler to include the specified file into the program. This line is compiled by the processor before the compilation of the program.

e.g #include<iostream.h>

the above line include the header file ***iostream*** into the program for the smooth running of the program.

Compilation and Linking

Compilation refers to the processing of source code files (.c, .cc, or .cpp) and the creation of an 'object' file. This step doesn't create anything the user can actually run. Instead, the compiler merely produces the machine language instructions that correspond to the source code file that was compiled. For instance, if you compile (but don't link) three separate files, you will have three object files created as output, each with the name <filename>.o or

<filename>.obj (the extension will depend on your compiler). Each of these files contains a translation of your source code file into a machine language file -- but you can't run them yet! You need to turn them into executables your operating system can use. That's where the linker comes in.

Linking refers to the creation of a single executable file from multiple object files. In this step, it is common that the linker will complain about undefined functions (commonly, main itself). During compilation, if the compiler could not find the definition for a particular function, it would just assume that the function was defined in another file. If this isn't the case, there's no way the compiler would know -- it doesn't look at the contents of more than one file at a time. The linker, on the other hand, may look at multiple files and try to find references for the functions that weren't mentioned.

ERRORS:

There are many types of error that are encountered during the program run. following are some of them:

1. **Compiler error**.: The errors encountered during the compilation process are called Compiler error. Compiler error are of two types

- Syntax error.
- Semantic error.

Syntax Error: Syntax error is the one which appears when we commit any grammatical mistakes. These are the common error and can be easily corrected. These are produced when we translate the source code from high level language to machine language.

e.g **cot<<endl;** This line will produce a syntax error as there is a grammatical mistake in the word **cout**

Semantic error: These errors appear when the statement written has no meaning.

e.g **a + b =c** ; this will result a semantically error as an expression should come on the right hand side of and assignment statement.

2. **Linker Errors.** Errors appear during linking process e.g if the word **main** written as **mian** . The program will compile correctly but when link it the linking window will display errors instead of success.

3. **Run Time error:** An abnormal program termination during execution is known as Run time Error.

e.g. If we are writing a statement X = (A + B) /C ;

the above statement is grammatically correct and also produces correct result. But what happen if we gave value 0 to the variable c, this statement will attempt a division by 0 which will result in illegal program termination. Error will not be found until the program will be executed because of that it is termed as run time error.

3. Logical Error.: A logical error is simply an incorrect translation of either the problem statement or the algorithm.

e.g : root1 = -b + sqrt(b * b -4*a*c) / (2 *a)

the above statement is syntactically correct but will not produce the correct answer because the division have a higher priority than the addition, so in the above statement division is performed first, then addition is performed but in actual practice to do addition performed then divide the resultant value by (2* a).

Manipulators :

Manipulators are the operators used with the insertion operator << to modify or manipulate the way data is displayed. There are two types of manipulators **endl** and **setw.**

1. **The endl manipulator :** The endl manipulator outputs new line . It takes the compiler to end the line of display.

```
cout << " Kendriya Vidyalaya Sangathan"<<endl;
cout<< " Human Resource and Development";
```

The output of the above code will be

Kendriya Vidyalaya Sangathan
Human Resource and development

2. **The Setw Manipulator** : The **setw** manipulator causes the number (or string) that follows it in the stream to be printed within a field **n** characters wide where n is the arguments to **setw (n).**

Increment and Decrement Operators in C++:

The increase operator (++) and the decrease operator (--) increase or reduce by one the value stored in a variable. They are equivalent to +=1 and to -=1, respectively. Thus:

C++

C +=1;

C=C+1;

are all equivalent in its functionality: the three of them increase by one the value of C.

A characteristic of this operator is that it can be used both as a prefix and as a suffix. That means that it can be written either before the variable identifier (++a) or after it (a++). Although in simple expressions like a++ or ++both have exactly the same meaning, in other expressions in which the result of the increase or decrease operation is evaluated as a value in an outer expression they may have an important difference in their meaning:

In the case that the increase operator is used as a prefix (++a) the value is increased **before** the result of the expression is evaluated and therefore the increased value is considered in the outer expression;

Example 1

```
B=3;

A =++B;          // (here A contains 4, B contains 4).
```

In case that it is used as a suffix (a++) the value stored in **a** is increased **after** being evaluated and therefore the value stored before the increase operation is evaluated in the outer expression.

Example 2

B=3;

A=B++; // (here a contains 3, B contains 4).

In Example 1, B is increased before its value is copied to A. While in Example 2, the value of B is copied to A and then B is increased.

Practice Session:

1. What is the name of the function that should be present in all c++ program?

Ans:main()

2. What are C++ comments?
Ans: comments are internal documentation of a program which helps the program for many purposes.

3. What is indentation of a program?
Ans: It is the systematic way of writing the program which makes it very clear and readable.

4. What is #include directives?
Ans :it instructs the compiler to include the contents of the file enclosed within the brackets into the source file.

5. What is role of main() in c++ program?
Ans:This is the first line that a C++ compiler executes. Program starts and end in this function.

6. What is a header file?
Ans:Header file provide the declaration and prototypes for various token in a program.

7. What is the purpose of comments and indentation?
Ans: the Main purpose of comments and indentation is to make program more readable and understandable.

8. What are console input /output functions?
Ans: Console I/O functions are cout and cin.

9. Write an appropriate statement for each of the following:
    1. Write the values for a&b in one unseperated by blanks and value of after two blanks lines.
    Ans: cout<<a<<b<<endl<<endl<<c;

    2. Read the values for a,b and c.
    Ans: cin>>a>>b>>c;

3. Write the values for a and b in one line, followed by value of c after two balnk lines.
Ans: cout<a<<b<<'\n\n'<<c;

10.What type of errors occurs while programming?
Ans: There are three types of errors generally occur are:
1.Syntax error
2.Semantic error
3.Type error.

11. How '/' operator is different from '%' operator?
Ans: '/' operator is used to find the quotient whereas % operator is used to find the remainder.
12. Which type of operator is used to compare the values of operands?
Ans: Relational operators.
13. How will you alter the order of evaluation of operator?
Ans: We can use parentheses to alter the order of evaluation of an equation.
14. What is the unary operator? Write 2 unary operator .
Ans : The operator which needs only one operand is called as unary operator .The '++' (increment) and '_ _'(decrement) operators.
15. What is output operator and input operator?
Ans: The output operator ("<<") is used to direct a value to standard output. The input operator (">>") is used to read a value from standard input.
16. What will be the output of following code:

```
void main()
{
int j=5;
cout<<++j<<j++<<j;  // in cascading processing starts from right to left
}
```

Ans. 7 5 5

17. What will be the output of following code:

```
void main()
{
int j=5;
cout<<++j + j++ +j++;  // values will be: 6 6 7 (From left to right)
}
```

Ans. 19
18. What will be the output of following code:

```
void main()
{
Int j=5, k;
k= a++ +a+ ++a;
cout<<k;
}
```

Ans. 18 (Because in evaluation of expression first of all prefix are evaluated, then it's value is assigned to all occurrences of variable)