Introduction to C++

Introduction and History

Until 1980, C programming was widely popular, and slowly people started realizing the drawbacks of this language and at the same time, the engineers had come up with a new programming approach that was Object Oriented programming**. Bjarne Stroustrup developed C++**at AT& T Bell Laboratories in Murray Hill, New Jersey. He developed this language in early 1980's and was very much concerned in making C++ more efficient. The C++ programmers not only make use of the new features but also have the privilege of using the traditional C features.

In the early stages, this language was called 'C with classes' and later in 1983, it was named as 'C++' by Rick Mascitti. In the year 1997 when the ANSI/ISO standards committee standardized the language.

Characteristics of C++

C++ has certain characteristics over other programming languages. It provides huge function Library, that's why its popularity is increasing day by

- **Object-oriented programming**: The possibility to orient programming to objects allows the programmer to design applications from a point of view more like a communication between objects rather than on a structured sequence of code. In addition it allows a greater reusability of code in a more logical and productive way.
- **Portability**: we can carry program from place to other place and can run without much modification
- **Brevity**: The code size is short compare to other language
- **Modular programming**: The several source code compiled separately and then linked together
- **Speed** : Due to reduced size faster execution
- **Machine** independent : It will not depend on internal architecture of machine
- **Flexibility** : The modification can be added easily
- **Wide range** of library functions : By using library function we can reduce the line of code
- **System software development**: Using this language we can develop system software, like editor , OS.
- **C compatibility** : The code written in C can directly used in this

Token

The tokens are smallest individual unit of program. The C++ has the following tokens

- ❖ Key words
- ❖ Identifiers
- ❖ Constants
- ❖ Punctuators
- ❖ Operators
- ❖ Character set

Character set

The set of the character is known as character set and it can be grouped as

- o Letters
- o Digits
- o Special character

- White space

Letters: These are all alphabets from A to Z or a to z
Digits: These are numbers from 0 to 9
Special character: These are special symbols like

Symbol	Name	Symbol	Name
.	Period	,	Comma
!	Exclamation	$	Dollar
=	Equal	[	Left square bracket
<	Less than	&	Ampersand
;	Semicolon	-	Minus

White space: It is defined as a space, carriage return, tab and form feed. These are ignored by the compiler. But there are some exceptions

- Sting constant can not be spited
- #include<header file> must be written on a single line

Keywords
The key words are the identifiers or variables with pre assigned or reserved meaning. These names can not be used as variable name in the program
The some of C++ Keywords are

auto	do	for	if	return	typedef		break	double
int	float	struct		while	long	sizeof		switch
default	goto	union						

Identifiers
Identifier refers to **the name of the variables**, functions, arrays, classes etc. It is created by the user. While creating the variable we have to follow the following rule

- The variable name should start with alphabet and it may contain anything
- Variable name should not contain any special character except underscore
- Variable name can not start with digit
- The upper case and lower case are separate
- Reserved word can not be used as variable name
- Usually 32 bit is the maximum number of character

Constants
It refers the fixed value and is not changes during the execution of the program. This constant can again divided into different type based on the type of the value it will store and are integer, character, floating point, and string

a) **Integer constant**: These are the constant without any fractional part and it is categorized as
   1) **Decimal Constant**: These are the constant starts with numbers ranging from 0 to 9. These numbers will not start with zero.
   2) **Octal Constant**: These are the constant starts with numbers ranging from 0 to 7. These numbers will starts with zero.
   3) **Hexadecimal constant**: These are the constant starts with numbers ranging from 0 to 9 and A to F. These numbers will starts with 0X. Here we can use upper case x or lowercase x along with zero. We can also use upper case A to F or we can use lower case a to f. In generally case does no matter.
b) **Floating point constant**: These constant contain the fractional value it means it contain decimal point and exponent. Using this we can represent the smaller or bigger number also. The exponent value is represented by e or E with optional +or – symbol. The E specifies power of ten

c) **Character constant**: A single character is enclosed in side of a pair of single quotation mark is called character constant.
d) **String constant**: A group of character enclosed inside of a pair of double quotation mark is called string constant. By default the compiler adds the null character ('\0') to the end of the string.

Example:

```
123                 // integer constant
0123                // octal constant
0X123               // hexadecimal constant
12.3              //floating point constant
6.23E-6             // floating constant
'c'               // character   constant
"SMS"     // string constant
```

Escape sequences: The character constant starts with back slash is called as escape sequence These are denoted by backslash (\ ) and a character. E.g. : \t is an escape sequence because slash causes an escape from the normal way the character are interpreted.
The list of common escape sequence

Escape sequence	Character	Escape sequence	Character
\a	Beep	\b	Back space
\f	Form feed	\n	New line
\r	Return	\t	Tab
\\	Back slash	\'	Single quotation mark
\"	Double quotation mark	\x	Hexadecimal representation

Punctuators: These are the symbols used for some specific purpose some of them are

Symbol	use	Symbol	use
!	Not	;	Statement terminator
%	Along with format specifier	[]	Array subscript
#	Preprocessor directive	()	Function call
{}	Compound statement	\	Escape sequence

Operators: An operator **is a symbol operates on operand and constant**. In C++ we have mainly three types of operands and are classified based on the number of operand and they are unary, binary and ternary operators. The unary operator requires one operand, the binary operator requires 2 operand and ternary operator require 3 operands

In generally we have the following operators in C++

Arithmetic Operators: C++ provides all the basic arithmetic operators. To use this we require two operands. They are listed below

Consider the variable A=5 and B=10 then we get

Operator	Meaning	Example
+	Adds two operand	A+B will give 15
-	Subtract second operand from first	B-A will give 5
*	Multiply the both operand	A*B will give 50
/	Divide numerator by denominator	A/B will give 0 B/A will gives 2
%	Modulus operator is used to find the reminder of integer division	A%B will give 5

Relational operators: We often compare two quantities and depending on their relation take certain decisions. For example, we may compare price of two items or age of two persons and so on. These comparisons can be done with the help of relational operators. An expression such as a < b or 1 < 20 containing a relational operator is termed as a relational expression. The value of relational expression is either one or zero. It is one if the specified relation is true and zero if the relation is false.
For example,
10 < 20 is true
But 20 <10 is false.
C supports 6 relational operators.

Operator	**Meaning**	**Example**	**RESULT**
<	is less than	4.5 < -10	FALSE
< =	is less than or equal to	4.5 < = 10	TRUE
>	is greater than	5>2	TRUE
> =	is greater than or equal to	5>=5	TRUE
= =	is equal to	5==5	TRUE
! =	is not equal to	5!=5	FALSE

Relational expressions are used in decision statements such as, if and while to decide the course of action of a running program.
Logical Operators: C++ has following three logical operators.
Let us assume A=1 and B=0

Operator	Meaning	Example
&&	Logical AND operator. If both the operands are true (1) it gives true value otherwise false(0)	A&&B is false
||	Logical OR operator. If both the operands are false (0) it gives false value otherwise gives true(1)	A||B is true
!	Logical NOT operator. It is used to reverse the current state of the operand	!(A&&B) is true

Assignment operators: Assignment operators are used to assign the result of an expression at right side to a variable at left side. We know usual assignment operator, **i.e. '='.**
Syntax: Variable op=expression;
The compound assignment statement: If we have the same operand at right and left side we can use compound assignment statement
variable = variable op (expression); it can be written as **variable op = (expression);**
Example: x = x + (y + 1); can be written as **x + = y + 1;**
a=a+10 can be written as a+=10

c=c*a can be written as c*=a

Conditional Operators: It is a ternary operator. We use? and: to construct conditional expression Syntax : Variable= exp1? exp2:exp3;

Where exp1, exp2 & exp3 are expressions.

exp1 is evaluated first, if it is non-zero(true), then the expression exp2 is evaluated and assigned to variable . If exp1 is false then the expression exp2 is evaluated and assigned to variable.

For example,

```
a=10;
b=15;
x=(a>b)?a:b;
```

Here, x is assigned the value of b.

Increment & Decrement Operators: The operator ++ adds one to the operand and is called increment operator while the operator - - subtracts one from the operand and is called decrement operator. Both are unary operators and take the following form

```
++m;    or      m++;
--m;    or      m--;
```

++m; is equivalent to m=m+1; (or m+=1;)

--m; is equivalent to m=m-1; (or m-=1;)

We use ++ and -- statements in for and while loops extensively. While ++m and m++ mean the same thing when they form statements independently, they behave differently when they are used in expressions on the right-hand side of an assignment statement.

```
m=5;
y=++m;
```

In this case, the value of y and m would be 6 because first increment the value of m and then assigned to variable y. This statement is called pre increment.

Suppose if we rewrite the above statements as

```
m=5;
y=m++;
```

the value of y would be 5 and m would be 6 , because first the value of m is assigned to variable y, then the vale of m is incremented by one . This statement is called post increment.

For example,

```
m=5;
y= - -m;
```

In this case, the value of y and m would be 4 because first decremented the value of m and then assigned to variable y. This statement is called pre decremented.

Suppose if we rewrite the above statements as

```
m=5;
y=m--;
```

The value of y would be 5 and m would be 4, because first the value of m is assigned to variable y, then the vale of m is decremented by one. This statement is called post decremented.

A prefix operator first subtract 1 to the variable on left but postfix operators first assigns the value to the variable on left & then decremented the operand.

Bitwise Operators: C++ has special operators known as bitwise operators for manipulation of data at bit level. These operators are used for testing the bits, or shifting them right or left. Bitwise operators may not be applied for float or double. Bitwise operators are listed below:

Operator	Meaning
&	Bitwise AND
|	Bitwise OR
^	Bitwise Exclusive OR
<<	Left shift
>>	Right shift
~	One's complement

Bitwise AND, Bitwise OR and Bitwise Exclusive OR are called Logical Operators, left shift and right shift operators are called bitwise shift operators.

Operator-1 (Op-1)	Operator-2 (Op-2)	Value of the expression		
		Op-1 & Op-2	Op-1 | Op-2	Op-1 ^ Op-2
1	1	1	1	0
1	0	0	1	1
0	1	0	1	1
0	0	0	0	0

Examples:

a=13➔ a 0000 0000 0000 1101

b=25➔ b 0000 0000 0001 1001

Then

c=a&b 0000 0000 0000 1001

d=a|b 0000 0000 1101 1101

e=a^b 0000 0000 0001 1101

Bitwise shift operator:

If X 0100 1001 1100 1011

X<<1 1001 0011 1001 0110

X<<2 0100 1110 0101 1000

X>>1 0010 0100 1110 0101

X>>3 0000 1001 0011 1001

One's Complement Operator (~): If true then false, if false then true.

Example:

x 0100 1001 1100 1011

~x 1011 0110 0011 0100

Special Operators: The special operators in C++ are comma operator, sizeof operator, pointer operator (& and *) and member selection operator (. and ->). Comma operator is used to link the related expressions together.

For example,

x = (a=5, b=5, a +b)

sizeof operator is a compile time operator that returns the number of bytes occupied when used with an operand. The operand may be a variable, a constant or a data type qualifier. For example,

x = sizeof(long int);

Dot and arrow (. and ->) operator: These are member operators used to reference individual members of classes , structures , union

Pointer operator(* and &): Pointer operator * used to display the value of pointer variable and the pointer operator & is used access the address of actual variable

Cast operator is used to convert from one data type to other

Note: The –a is the example for unary operator used to change the sign of the vale

Precedence of the operator OR **Hierarchy of operator**

When an arithmetic expression involves 2 or more arithmetic operator, then the hierarchy or the order of precedence is important. It means if 2 arithmetic operator comes in same expression at that movement which operator should evaluate first

Associativity: In the expression if the 2 or more than two operator of same precedence appear at that movement how we can solve that expression that is from left to right or right to left that is known as associativity

The precedence of the operator along with associativity is given below

priority	Description	Represented By	Associativity
1	Parenthesis	( ) [ ]	Left to right
1	Structure Access	_ ->	Left to right
2	Unary	! - ++ -- - * &, sizeoff	Right to left
3	Multiply, Divide, Modulus	* / %	Left to right
4	Add, Subtract	+ -	Left to right
5	Shift Right, Left	>> <<	Left to right
6	Greater, Less than, etc. (relational)	> < =	Left to right
7	Equal, Not Equal	== !=	Left to right
8	Bitwise AND	&	Left to right
9	Bitwise Exclusive OR	^	Left to right
10	Bitwise OR	|	Left to right
11	Logical AND	&&	Left to right
12	Logical OR	||	Left to right
13	Conditional Expression	?:	Right to left
14	Assignment	= += -= etc.	Right to left
15	Comma	,	Left to right

Expressions: The expression is the valid combination of operator, variable and constant.

The following rules regarding the expression

- ✓ A signed or unsigned variable name or constant is an expression
- ✓ Two arithmetic operator should not occur in succession in an expression

E.g. A=b+7;

X=y+z;

Evaluate the following expression

1) 5*(4+(10-6)/2)+5
   =5*(4+4/2)+5 [inner bracket evaluate first]
   =5*(4+2)+5 [division will perform]
   =5*6+5 [evaluate bracket by multiplication]
   =30+5 [multiplication will perform]
   =35 [addition will perform and gives final result]
2) 8+4*6-8/3
   =8+24-8/3
   =8+24-2 [integer by integer gives integer value]
   =32-2
   =30
3) 21/3+6%2
   =7+6%2
   =7+0
   =7

Note: What will be the result of the following expressions if a=3 and b=5

i)a+=a*++b; ii)a*=2*b++;

Ans: i)a+=b++; =8 ii)a* = 2*b++; =30

What will be the result

1) if a =400,b=150 then a-b<100?180:400

Answer =400

ii) if a = 300, b=400 then a-b <100?180:400

Answer =180.

Write the mathematical expressions into C++ expressions:

1. $A = \pi r^2$
2. $s=(a^2+b^2+c^2)/4$
3. area=$\sqrt{s(s-a)(s-b)(s-c)}$
4. volume=πr^2h
5. volume=3/4πr^3h

Ans:

1. A=3.14 *r*r;
2. S=(a*a +b*b +c*c)/4;
3. Area = sqrt (s*(s-a)*(s-b)*(s-c));
4. Volume = 3.14*r*r*h ;
5. Volume=3.0/4*3.14*r*r*r*h ;

Type Conversion

Converting an expression from one data type to another data type. The type conversions are of two types and they are

1) Implicit conversion

2) Explicit conversion

Implicit conversion

If the operand of 2 different type appears in the single expression then the lower type variable is converted in to higher type variable automatically

Hierarchy of data type

Long double — higher order

Double

Float

Long

Int

Char — Lower order

```
int a;
char ch='A';
a=ch;
cout<<a;   //automatically convert from character type to integer
```

will gives output 65

i.e. it will print ASCII value of given character

Explicit conversion or Type Casting

It is the process of converting form one data type to another data type by forcibly

Syntax

Variable-name = (new-data-type) variable-name;

Example

```
int a=2,b=3;
float avg;
avg=(float)a+b/2;
```

A simple program OR structure of C++ program

The general structure is

Comment or documentation section
Linker section
Global variable declaration section
void main()
{
declaration section
executable statement section
}

user defined function

Consider the following program

```
1        #include<iostream.h>
2        void main(void)
3        {
4        cout<<"Hello welcome to C++\n";
5        }
```

Annotation

1. This line uses preprocessor directive #include to include the contents of the header file iostream.h in the program . The iostream.h is a standard C++ header file and contains definition for input and output.
2. This line define the function called main. A function may have zero or more parameters , these always appear after the function name between a pair of brackets. The word void appearing between the brackets indicates that main as no parameters. The function may also have the return type. This always appear before the function name. All C++ function have one main function and the execution of program will begin from this point
3. This brace makes the beginning of the body of main
4. This line is a statement . A statement is a calculation step which may produce a value. The end of the statement is always marked with a semicolon (;). The information given inside the double quote (" ") will be printed as it is . The last character in the string (\n) is a new line character . The cout is a output stream object used to print the value on the screen. The symbol << is an output operator which takes an output stream as its left operand and expression is its write operand
5. This brace is used to end the body of main

   The { and } are used to declare a set of statement together and the content present within side of this brace is called block.

Comments

The comments are non executable part of the program and it can be represented in two different types

a) Inline or single line comment represented as // (double slash) : This comment starts with double slash and ends with end of the line

b) Multi line comment represented as /* */ : This comment starts with slash and star and ends with star and slash the statement enclosed within side of this symbol will ignored by the compiler . Using this we can eliminate one line or more than one line.

The importance of iostream file

This is include directive in the program and is called as preprocess directive used to add content of iostream file to the program. This file contain the definition regarding the cin and cout and the operator << . The iostream.h header file should include at the beginning of all program. This is a main heading file which will process first before processing any other statement due to that we call it as preprocess directive.

We can include this file by using two different method

1) #include<iostream.h>
   In this method compiler will search for file iostream.h inside the include directives.
2) #include"iostream.h"
   In this method compiler will search for file iostream.h inside the current directory

Note:

1) Creating the source program, compiling and executing

C++ program can be created by using any text editor like vi , dos editor , or else we can also use C editor or C++ editors like turbo C++ or Borland C++ to develop the program. After typing the program save the program with file extension **.CPP**

The Turbo C++ and Borland C++ provide an integrated program development environment under MS DOS . They provide built in editor and menu bar which include menu items such as File ,Edit, Run and compile. We can create and save the source file under the File option , we can compile under compile option or shortcut is ALT+C, we can run the program under Run option or shortcut is ALT+R. We can also perform the compilation and running together by pressing CTRL +F9 key . After viewing the results we can come back to editor window by pressing ALT+F5

2) **Global variable**: If we declare a variable before the main function is called global variable and can be accessed throughout the program.

3) **Definition section** : here we can declare the symbolic constant by using #define symbol Ex: **#define pi 3.142**

4) **Declaration Section:** here we declare the variable which is required in the program, these variable can also called as local variable because it has scope only in that function module.

Library function : C++ provides a built in function that can be used to calculation. To use this function we have to use proper header file based on the type . Some of them are

1) **Mathematical function** : To use this function we have to include header file called **math.h** the some functions present under

this files are

Function	Meaning
fabs(x)	Absolute value of real number x
abs(x)	Absolute value of integer number
pow(x,y)	The x to the power y
sqrt(x)	Square root of given number
log(x)	Logarithm of x
log10(x)	Logarithm of the number x to the base 10
exp(x)	Exponential function of x
sin(x)	Sine of the angle x . where x measured in radians
cos(x)	Cosine of the angle x . where x measured in radians
tan(x)	Tangent of the angle x . where x measured in radians
asin(x)	Will used to find $\sin^{-1}(x)$ where x is measured in radian
acos(x)	Will used to find $\cos^{-1}(x)$ where x is measured in radian

2) **character function** : To use this function we should use header file **ctype.h.** The some of the functions present under this file is

Function	Meaning
isalpha(ch)	It returns true if ch is alphabet (upper case or lower case) otherwise return false
isdigit(ch)	It returns true if ch is digit(0 to 9) otherwise return false
isalnum(ch)	It returns true if ch is alphabet or a digit otherwise return false
islower(ch)	It returns true if ch is lower case else return false
isupper(ch)	It returns true if ch is upper case else return false
toupper(ch)	Converts ch from lower case to upper case
tolower(ch)	Converts ch from upper case to lower case

3) **String function**: Using these function we can perform string related operation to use this we have to include header file **string.h** The some of the functions are

Function	Meaning
strlen(str)	Gives the number of character in a given string str
strrev(str)	To convert the string to its reverse
strupr(str)	Converts the str to upper case
strlwr(str)	Converts the str to lower case
strcpy(str1,str2)	Copies the content of string to to string1
strcmp(str1,str2)	Compares the str1 and str2 and we get 3 possible output 0 if both string are equal 1 if first string is greater than second -1 if the first string is less than second This compression is case sensitive and comparison will takes place based on ASCII value
strcmpi(str1,str2)	Compares the str1 with str2 by ignoring case
strcat(str1,str2)	Content of str2 is added to the end of str1

4) **Console I/O function** : using this we can input and output the vale and is present under header file **stdio.h**

Function	Meaning
puts()	To print the string value on screen
gets()	To read one line of text from key board till new line character encounters
putchar()	To print the character on screen
getchar()	Input a single character from keyboard

5) **Standard library function** : If we use these function we have to include header file **stdlib.h**

Function	Meaning
itoa(n)	Used to convert number n to string
atoi(s)	Used to convert stings s to numerical value
random(n)	Used to generate random number between 0 to n-1

6) **Some commonly used function**: To use the following function we have to include header file called **conio.h**

Function	Meaning

clrscr()	This function is used to clear the screen content
getch()	This function is used to get the character value but enter character will not be displayed
getche()	This function is used to get the character value but enter character will be displayed