Ashish Ranjan

LOK NAYAK JAI PRAKASH INSTITUTE OF TECNOLOGY,CHAPRA

12/6/2023

C++

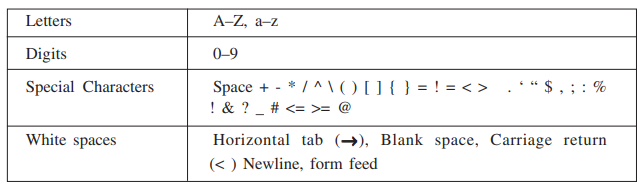


***`1.Introduction to c++***

* 1. **C++ character set**

Character set is a set of valid characters that a language can recognize. A character represents any letter, digit or any other special character. The C++ programming language also has some character set. Now let us learn C++ language character set.

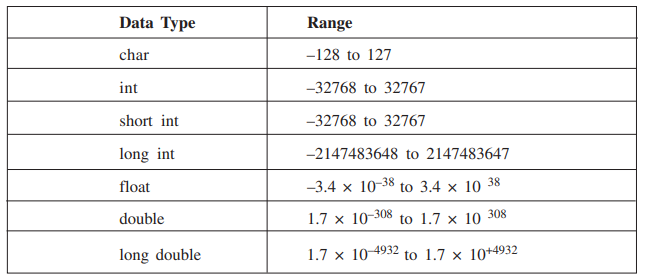
Table 12.1: C++ character set

****

* 1. **Basic data types**

Now you will learn about basic data types used in C++ language. Every program specifies a set of operations to be done on some data in a particular sequence. However, the data can be of many types such as a number, a character, boolean value etc. C++ supports a large number of data types. The built in or basic data types supported by C++ are integer, floating point and character type. A brief discussion on these types is given below:

Table 12.2: Data Types with Range

****

**2.1 Integer type (int)**

An integer is an integral whole number without a decimal point. These numbers are used for counting. For example 26, 373, –1729 are valid integers. Normally an integer can hold numbers from –32768 to 32767. However, if the need be, a long integer (long int) can also be used to hold integers from –2, 147, 483, 648 to 2, 147, 483, 648.

**2.2 Floating point type (float)**

A floating point number has a decimal point. Even if it has an integral value, it must include a decimal point at the end. These numbers are used for measuring quantities.

Examples of valid floating point numbers are: 27.4, -927., 40.03 A float type data can be used to hold numbers from 3.4\*10–38 to 3.4\*10+38 with six or seven digits of precision. However, for more precision a double precision type (double) can be used to hold numbers from 1.7\*10–308 to 1.7\*10+308 with about 15 digits of precision.

**2.3 Character Type (Char)**

It is a non numeric data type consisting of single alphanumeric character. Examples of valid character types are : ‘A’, ‘9’, ‘P’, ‘8’, ‘&’. It may be noted here that 9 and ‘9’ are of different data types. The former is of type int and later of type char.

* 1. **Tokens**

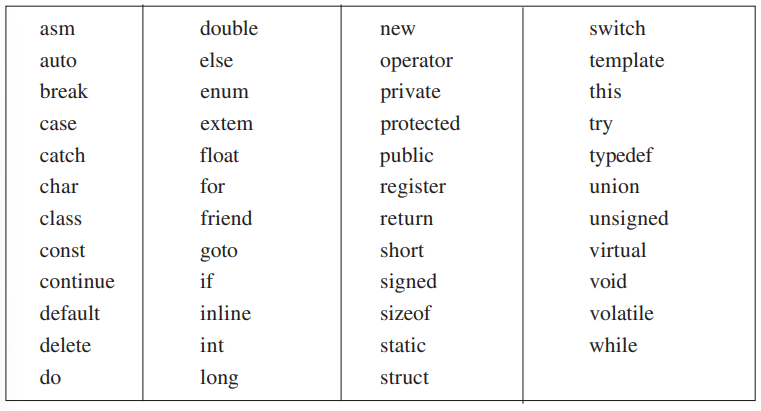
A token is a group of characters that logically belong together. The programmer can write a program by using tokens. C++ uses the following types of tokens.

* Keywords
* Identifiers
* Literals
* Punctuators
* Operators

**3.1 Keywords**

There are some reserved words in C++ which have predefined meaning to complier called keywords. Some commonly used keywords are given below:

Table 12.3: List of keywords



**3.2 Identifiers**

Symbolic names can be used in C++ for various data items used by a programmer in his/her program. For example, if you want to store a value 50 in a memory location, you can choose any symbolic name (say MARKS) and use it as given below:

**MARKS = 50**

The symbol ‘=’ is an assignment operator. The significance of the above statement is that ‘MARKS’ is a symbolic name for a memory location where the value 50 is being stored. A symbolic name is generally known as an identifier. The identifier is a sequence of characters taken from C++ character set. The rules for the formation of an identifier are:

1. An identifier can consist of alphabets, digits and/or underscores.
2. It must not start with a digit.
3. C++ is case sensitive, i.e., upper case and lower case letters are considered different from each other. It may be noted that TOTAL and total are two different identifier names.
4. It should not be a reserved word (keywords).

**3.3 Literals**

Literals(often referred to as constants) are data items that never change their value during the execution of the program. The following types of literals are available in C++.

1. integer-constants
2. character-constants
3. floating-constants
4. string-literals

**Integer constants**

Integer constants are whole numbers without any fractional part. It may contain either + or – sign, but decimal point or commas does not appear in any integer constant. C++ allows three types of integer constants.

1. decimal (Base 10)
2. octal (Base 8)
3. hexadecimal (Base 16)

**Decimal integer constants**

It consists of sequence of digits and should not begin with 0 (zero). For example 124, - 179, + 108.

**Octal integer constants**

It consists of sequence of digits starting with 0 (zero). For example, 014, 012.

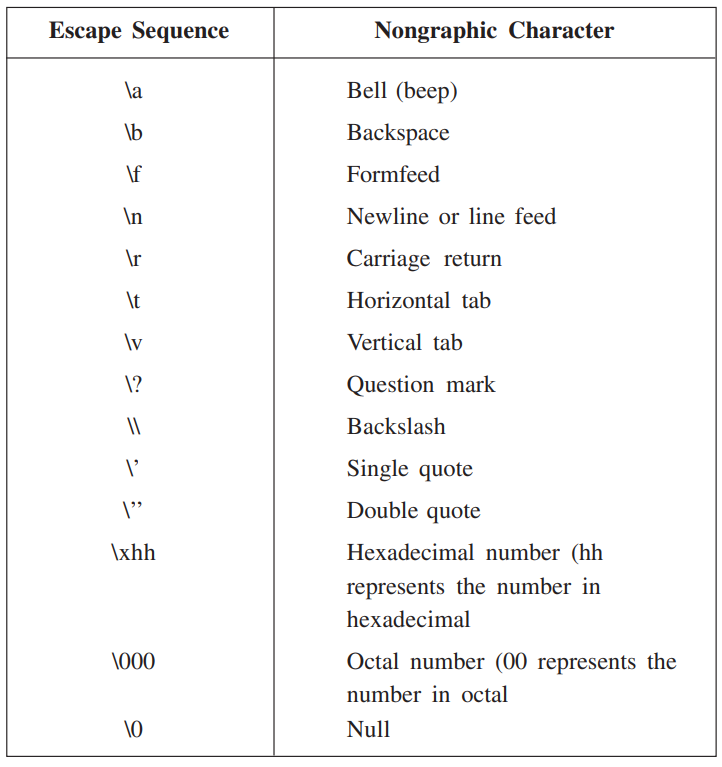
**Hexadecimal integer constant**

It consists of sequence of digits preceded by ox or OX. For example OXD, OXC. The suffix l or L and u or U attached to any constant forces it to be represented as a long and unsigned respectively.

**Character constants**

A character constant in C++ must contain one or more characters and must be enclosed in single quotation marks. For example ‘A’, ‘9’, etc. C++ allows nongraphic characters which cannot be typed directly from keyboard, e.g., backspace, tab, carriage return etc. These characters can be represented by using an escape sequence. An escape sequence represents a single character. The following table gives a listing of common escape sequences.

Table 3.4: list of escape sequence



**Floating constants**

Floating constants are also called real constants. These numbers have fractional parts. They may be written in fractional form or exponent form. A real constant in fractional form consists of signed or unsigned digits including a decimal point between digits. For example 3.0, -17.0, -0.627 etc.

A real constant in exponent form has two parts: a mantissa and an exponent. The mantissa is either an integer or a real constant followed by letter E or e and the exponent which must be an integer. For example 2E03, 1.23E07.

**String Literals**

A sequence of character enclosed within double quotes is called a string literal. String literal is by default (automatically) added with a special character ‘\O’ which denotes the end of the string. Therefore the size of the string is increased by one character. For example “COMPUTER” will be represented as “COMPUTER\O” in the memory and its size is 9 characters.

**3.4 punctuators**

The following characters are used as punctuators in C++.

*1)Brackets [ ]*

* opening and closing brackets indicate single andmultidimensional array subscript.

*2)Parentheses ( )*

* opening and closing brackets indicate functions calls, function

parameters for grouping expressions etc.

*3)Braces { }*

* opening and closing braces indicate the start and end of a

compound statement.

*3)Comma ,*

* it is used as a separator in a function argument list.

5)Semicolon ;

* it is used as a statement terminator.

*6)Colon :*

* it indicates a labelled statement or conditional operator

*7)Asterisk \**

* it is used in pointer declaration or as multiplication operator.

*8)Equal sign =*

* it is used as an assignment operator.

*9)Pound sign #*

* it is used as pre-processor directive.

**3.5 Operators**

Operators are special symbols used for specific purposes. C++ includes many operators.

* Arithmetical operators
* Relational operators
* Logical operators
* Unary operators
* Assignment operators
* Conditional operators
* Comma operator

**Arithmetical operators**

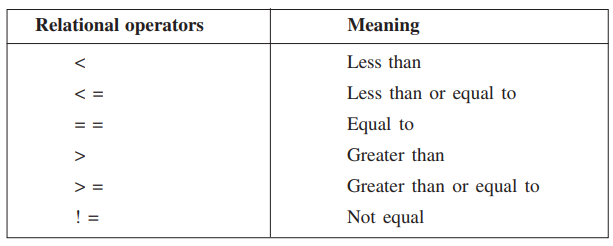
An operator that performs an arithmetic (numeric) operation +, -, \*, / , or %. For these operations always two or more than two operands are required. Therefore these operators are called binary operators. The following table shows the arithmetic operators.

**Note** : There is no operator which gives exponent in C++.

**Relational operators**

The relational operators are used to test the relation between two values. All relational operators are binary operators and therefore require two operands. A relational expression returns zero when the relation is false and a non-zero when it is true. The following table shows the relational operators.

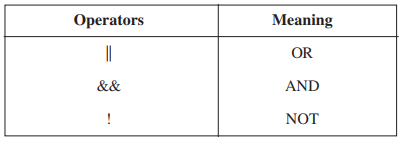
Table 1.5: list of relational operators



**Logical operators**

The logical operators are used to combine one or more relational expression. The table 12.6 shows the logical operators.

Table 1.6: list of logical operators



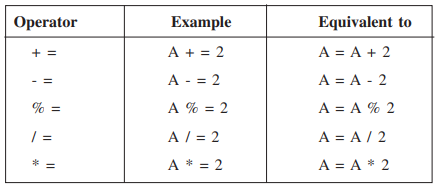
The NOT operator is called the unary operator because it requires only one operand.

**Unary operators**

C++ provides two unary operators for which only one variable is required.

**Assignment operator**

The assignment operator ‘=’ stores the value of the expression on the right hand side of the equal sign to the operand on the left hand side.



**Conditional operator**

The conditional operator ?: is called ternary operator as it requires three operands. The format of the conditional operator is: Conditional\_expression? expression1 : expression2;

If the value of conditional\_expression is true then the expression1 is evaluated, otherwise expression2 is evaluated.

**The comma operator**

The comma operator gives left to right evaluation of expressions. It enables to put more than one expression separated by comma on a single line.

* 1. **The size of operator**

As we know that different types of variables, constant, etc., require different amounts of memory to store them. The size of operator can be used to find how many bytes are required for an object to store in memory.

**Example 8**

size of (char) returns 1

size of (int) returns

size of (float) returns 4

If k is an integer variable, the sizeof (k) returns 2.

The size of operator determines the amount of memory required for an object at compile time rather than at run time.

* 1. **The order of precedence**

The order in which the arithmetic operators ( +, -, \*, / , %) are used in a given expression is called the order of precedence. The following table 12.7 shows the order of precedence.

Table 12.8: Precedence of Arithmetic operators

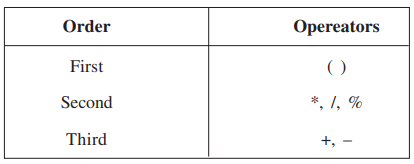
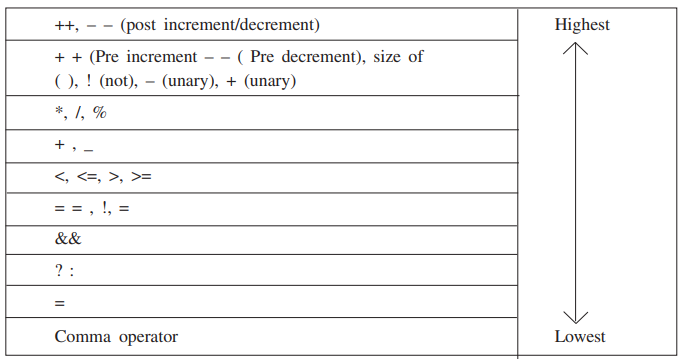
****

Table 12.9: Precedence of Operator



* 1. **Type consersion**

The process in which one pre-defined type of expression is converted into another type is called conversion. There are two types of conversion in C++.

* Implicit conversion
* Explicit conversion
  1. **Constants**
  2. **Variable**
  3. **Input and output(I/O)**
  4. **Structure of C**