# Introduction to C++ Language

* C++ language is one of the world's most popular language.
* It is high-level language that means it is machine independent language.
* C++ language supports object oriented programming structure so it is called "Object Oriented Programming language.
* It is case sensitive language.
* It is developed by Bjarne Stroustrup in 1979 at AT and T's bell laboratory (USA).
* This language is first choice in programming competition because of its speed and easir less complex syntax.
* Features of C++ language -

i) Simple
ii) Portable
iii) Powerful
iv) Machine language
v) Structure oriented
vi) High level programming lang
vii) High speed
viii) High efficiency
ix) Flexible

* Application of C++ language -

i) Operating System development
ii) Web browser development (Not commonly)
iii) Game development.
iv) Database system
v) Network drivers
vi) Interpreters

* Difference b/w C and C++

| C | C++ |
|---|---|
| 1) C follows procedural style programming. | 1) It follows both procedural and object oriented programming. |
| 2) Data is less secured in 'C'. | 2) In C++, we can use modifiers for class member to make it secure. |
| 3) It follows top-down approach. | 3) It follows bottom-up approach. |
| 4) C doesn't support reference variable | 4) C++ support reference variable. |
| 5) Scanf() and printf() mainly used for ~~out~~ input/output. | 5) cin and cout are used to perform input/output operations. |
| 6) It doesn't provide feature of namespace. | 7) It provides features of namespace. |

## * First Program (first.cpp)

```
#include <iostream>        → library
using namespace std;
int main()    // main() is where program execution begins.
{
    cout << "Hello World";  // Prints Hello world.
    return 0;
}
```

## * Addition Program -

```
#include <iostream>
using namespace std;
int main()
{
    int a, b, c;
    cout << "Enter two no.";
    cin >> a >> b;
    c = a+b;
    cout << "Addition is " << c;

    return 0;
}
```

* How to read String -

```
#include <iostream>
using namespace std;
int main()
{
    string name;
    cout << "May I know your name";
    cin >> name;
    cout << "Welcome Mr." << name;
    return 0;
}
```

→ If we give "Anil kumar" as a input then it reads only "Anil".

→ To read all word, we use getline (cin, name)

# Data types and Variables

* Any information is called as data.
  Eg:- name, age, marks etc.

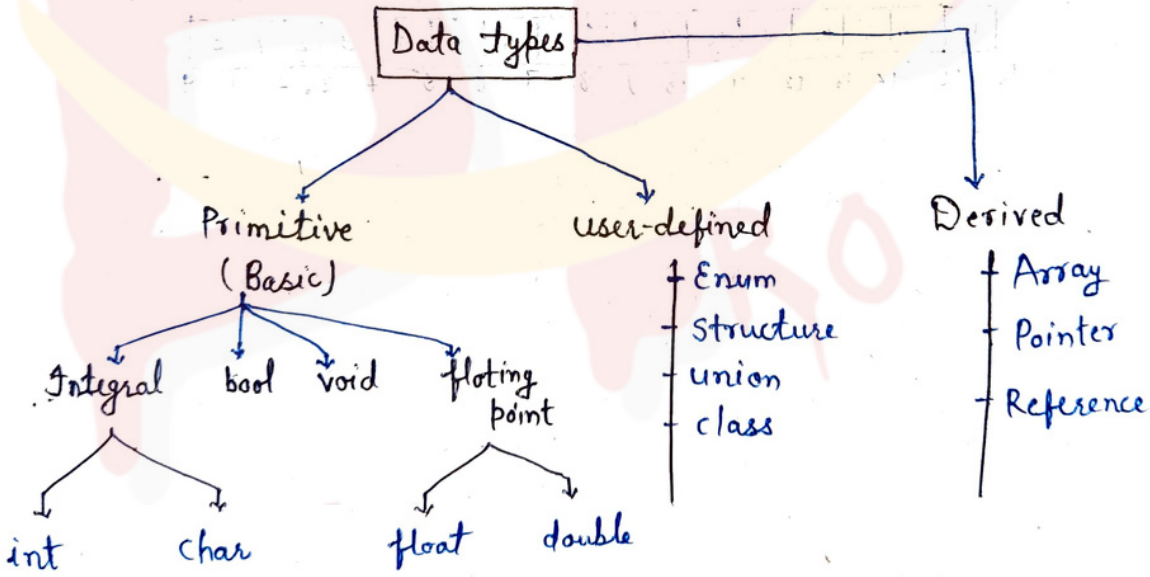
* Data types -

  Data type is used to specify the type of data that a variable can hold.

→ Data stored only after the variable is declared.

→ Data can be both sign and unsigned.

* Types of data types -



| Data type | Size | Range |
|---|---|---|
| int | 2 or 4 bytes | -32768 to 32767 (for 2 bytes) |
| float | 4 bytes | $-3.4 \times 10^{-38}$ to $3.4 \times 10^{38}$ |
| double | 8 bytes | $-1.7 \times 10^{-308}$ to $1.7 \times 10^{308}$ |
| char | 1 byte | -128 to 127 |
| bool | undefined | true/false |

* How integer gets range -32768 to 32767 (2 bytes)

2 bytes = 16 bits


MSB (Most Significant Bit)

LSB (least Significant bit)

This bit is reserved for sign bit

if bit is -

1 → Negative

0 → Pos.

$2^{15} = 32768$

0 to 32767 (It is start from 0)

add

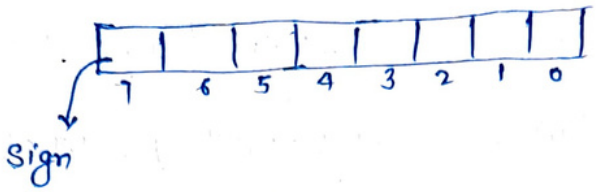
-32767 to -0 (Neg) | 0 to 32767 (Pos) (It takes pos and neg both values)

It is not a number

**-32768 to 32767**

* How character gets range -128 to 127 -

1 byte = 8 bits




$2^7 = 128$

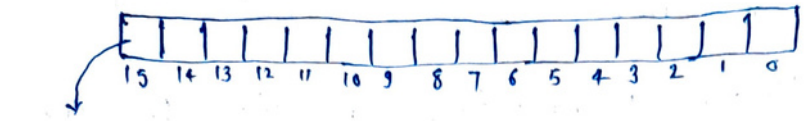
-127 to -0 (Neg) | 0 to 127 (Pos)

**-128 to 127**

* ASCII code for character -

| | | |
|---|---|---|
| A = 65 | a = 97 | 0 = 48 |
| B = 66 | b = 98 | 1 = 49 |
| C = 67 | c = 99 | 2 = 50 |
| ⋮ | ⋮ | ⋮ |
| Z = 90 | z = 122 | 9 = 57 |

* How 10 and -10 are represented as bits in memory-




At it is

0 → +ve

1 → -ve

10 → 


→ Negative ~~are~~ numbers are stored in 2's complement form-

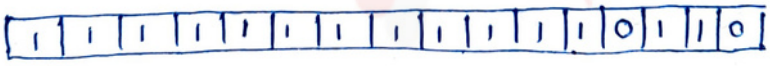
10 → 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0

1's → 1 1 1 1 1 1 1 1 1 1 1 1 0 1 0 1

2's → + 1

1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 0

This is '1' means it is a negative number

-10 → 

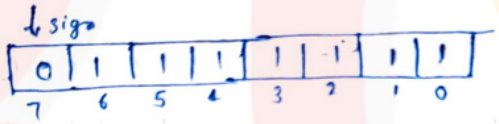
* Overflow -

char x = 127

( As we know, a character can store 127 as a max^m value because it ranges from -128 to 127 )

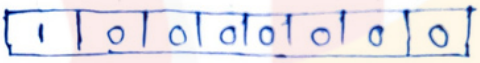
→ If we add one more value which is out of its capacity then it again start from the beginning. Means it behave like cyclic. This is called overflow.)

Eg:- x




→ 127

\+ 1




→ -128

→ Similarly, if we go beyond from '-128' then it will go to the 127.

* Variables -

A variable is a named containner that stores a value.

→ It is used to hold data that can be manipulated and used in a program.

Eg:- int roll = 10;

(Here "roll" is a variable name)

→ int roll ; // Declaration of variable

→ roll = 10; // Initialisation of variable

→ int roll = 10 ; // declaration and initialisation

* Every character type variable should be enclosed in ''.

char group = 'A';

* float price = 127.55f // declaration and initialisation of float.

→ If 'f' is not written at the end then it will be taken as double.

→ The decimal no. should be stored in float data type.

* Types of Variable -

i) Local variable -

When we create variable in the body of a function, called local variable or private variable.

→ The local variables can not be used outside the function.

ii) Global variable -

When we create variables in global declaration section, called global variable or public variable.

→ A public variable can be used by any function in the program, because it lives till the execution of the whole program.

* How to take variable name -

Use meaningful names while declaring variables for better understanding purpose.

* Some rules for declarations is as follows -

→ int x1; ✓ → int roll_no; ✓

→ int 1x; ✗ → int rollNo; ✓

→ int rollno; ✓ → int RollNo; ✓

→ int roll no; ✗

# Operator and Expressions

* Operator -

An operator is a symbol or keyword that performs an operation or a set of operations on one or more operands.

* Types of operators -

1) Arithmetic Operators -

These are used for performing mathematical operation.

| Operator | meaning | example |
|---|---|---|
| + | Addition | A + B |
| - | Subtraction | A - B |
| * | Multiplication | A * B |
| / | Division | A/B |
| ^ | Power | A^3 |
| %. | Reminder | A %. B |

2) Assignment Operators -

An assignment operators is used for assign a value to a variable. The most common assignment operator is '='.

Eg:- assign value 5 to the variable x :

x = 5

* The statement " C = A + B " means that add the value stored in variable A and B then assign/store the value in variable C.

* +=, -=, /=, *=, %= these operators are known as compound operator and it is also called shorthand notation.

Eg:- x = x + 10; can be replaced by

→ x += 10

3) Relational Operators -

These operators which are used to compare or check the relation b/w two or more quantities.

| Operator | Meaning | Example |
|---|---|---|
| < | less than | A < B |
| <= | less than or equal to | A <= B |
| == | Equal to | A == B |
| != | Not equal to | A != B |
| > | greater than | A > B |
| >= | greater than or equal to | A >= B |

4) Logical Operators -

These are used to combine multiple conditions.

→ AND (&&) -

Eg:- A < B && B < C

(Result is true if both A<B and B<C are true else Result will become false.)

→ OR (||) -

Eg:- A < B || B < C

(Result is true if either A<B or B<C are true else false.)

→ NOT (!) -

Eg:- !(A > B)

(Result is true if A > B is false else false.)

5) Increment and Decrement operator - (Unary)

- Increment operators are the unary operators used to increment or add 1 to the operand value.

* The increment operator is denoted by the "++"

- Decrement operator is the unary operator, which is used to decrease the original value of the operand by 1.

* It is represented as the "--".

* ++, -- can be used as -

  → ++pre : Pre increment

  → post++ : Post-increment

  → --pre : Pre-decrement

  → post-- : post decrement

* In pre-increment/decrement first the value is incremented/decremented and then utilized.

* In post-increment/decrement first the value is utilized and then incremented/decremented.

Eg:- 1) int i, j = 2;

i = ++j; // i = 3, j = 3

2) int i, j = 2;

i = j++; // i = 2, j = 3

6) Bitwise Operators -

These operators works on bits of data only and it ~~works~~ use only integer type of data.

- → & (AND)
- → | (OR)
- → ~ (NOT)
- → ^ (XOR)
- → << (Left shift)
- → >> (Right shift)
- → >>> (Unsigned right shift)

*

| bit1 | bit2 | bit1 & bit2 |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

Bitwise AND(&)

*

| bit1 | bit2 | bit1 \| bit2 |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

Bitwise OR(|)

*

| bit1 | bit2 | bit1 ^ bit2 |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

Bitwise XOR(^)

*

| bit | ~bit |
|---|---|
| 0 | 1 |
| 1 | 0 |

Bitwise NOT (~)

* AND(&) -

Eg:- int x = 10, y = 6, z;

x → 0 0 0 0 1 0 1 0

y → 0 0 0 0 0 1 1 0

---

z = x & y → 0 0 0 0 0 0 1 0 → 2

* OR(|) -

x → 0 0 0 0 1 0 1 0

y → 0 0 0 0 0 1 1 0

---

z = x | y → 0 0 0 0 1 1 1 0 → 14

* NOT(~) -

int x = 5;

x → 0 0 0 0 0 1 0 1

∴ ~x → 1 1 1 1 1 0 1 0 → This is a negative value, to know its value, we have to convert it into 2's complement

1's → 0 0 0 0 0 1 0 1

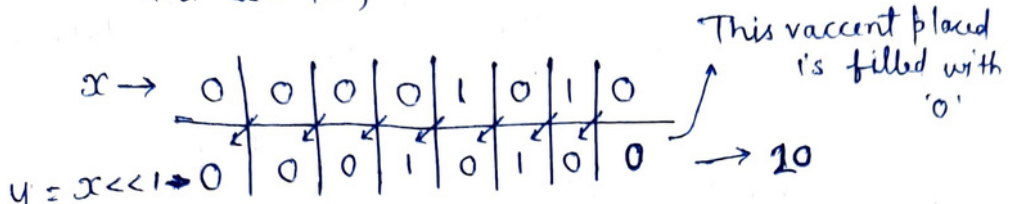
+ 1

---

2's → 0 0 0 0 0 1 1 0 → -6

* Left shift (<<) -

int x = 10; (n)



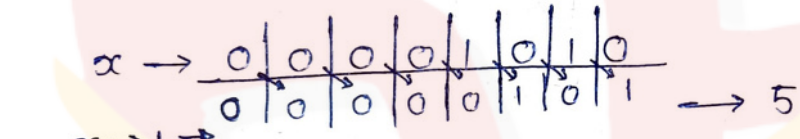

( Bits are shift by 1 space to left)

$\therefore y = x^{n} <<1^{k}$

$\therefore y = 20 \quad (n * 2^k)$ → Formula

$= 10 \times 2^1 = 20$

* Right shift (>>)




( Bits are shifted by 1 space to right)

$\therefore z = x >> 1$

$z = 5 \quad \left( \frac{n}{2^k} = \frac{10}{2^1} = 5 \right)$

* Enum (Enumrated) -

It is a userdefined data types. By using it, we can define our own data types.

Eg:- Enum day {Mon, tue, wed, thur, fri, sat, sun};
(0, 1, 2, 3, 4, 5, 6)

↓
data type

```
int main()
{
    day d;
    d = mon;
    d = fir;
    d = 0;   X
}
```

→ In this 'd' is a variable name of ~~dat~~ day data type.

→ And we can assign only that name or value which are declared in enum.

* Type definition (typedef)-

The typedef is a keyword using which a programmer can create a new data type name for an already existing data type name.

→ The purpose of 'typedef' is to redefine or rename the name of an existing data type.

→ By using it, we can make program more readable.

Syntax:

typedef data type new name

typedef int marks;

(Now we can use 'marks' instead of 'int'.)

Example:

```
int main()
{
  int m1, m2, m3, r1, r2, r3;
}
```

→

```
typedef int marks;
typedef int roll;
int main()
{
  marks m1, m2, m3;
  roll r1, r2, r3;
}
```