

1.2 write your first c++ program

Made By :-

Aditya Jain



Why do we need Programming Lang?

A prominent purpose of programming lang is to provide instructions to a computer.

Is C++ is a high-level lang?

C++ is regarded as a middle-level lang, as it comprises a combination of both high-level & low-level lang features. It is a superset of C & that virtually any legal C prog is a legal C++ program.

* The code is portable & the syntax is human-readable. for these reason, C & C++ are HLL.

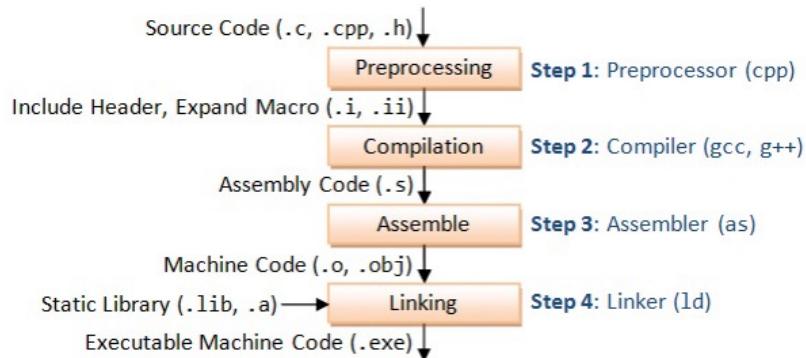
Why use C++?

→ C++ is one of the world's most popular prog. lang. that can be found in today's OS, GUI & embedded systems.

→ It is object-oriented prog. lang. which give us a clear structure to program & allows code to be re-used, lowering development cost.

* The main difference b/w C & C++ is that C++ supports classes & objects, while C does not.

How Compiler compile the Code :-



Where to Start the code :-

Code starts from :-

`int main ()`

→ function
Only it is run when called

a block of code
put together
perform a task

return-type
integer

function name

int main ()
{
}
Syntax

Block
of
code

↳ means all the code inside that belongs to
the main function

first C++ Program

Header file library :- tell us work with input & output objects, such as cout.

+ Header file add functionality to C++ prog.

means that
we can use
names for objects
& variables from
the standard library.

#include <iostream>

using namespace std;

int main ()

{

cout << "Aditya Jain"

return 0;

)

pronounce as }

See-out is an object

with the insertion operator

<< to output / print text

cannot use without header

file

String

Sequence of
characters

end of
line

return 0 in main function means
that the program executed successfully

Use-case	return 0	return 1
In the main function	return 0 in the main function means that the program executed successfully.	return 1 in the main function means that the program does not execute successfully and there is some error.
In user-defined function	return 0 means that the user-defined function is returning false.	return 1 means that the user-defined function is returning true.

Without using namespace std ; cout work ?

Yes , cout work without using Standard namespace . But we have to write std :: before cout .

std :: cout << "Aditya Jain " ;

Variables :- A container that is used to store the data values .

→ value stored in a variable can be changed during program execution



→ A variable is only a name given to a memory location , all the operations done on the variable effects that memory location

How to declare single variable

Syntax :- type variable_name ;

for ex

int age = 15 ;

int

age

variable name

assignment operator

15

value

data type

Explain in
data types

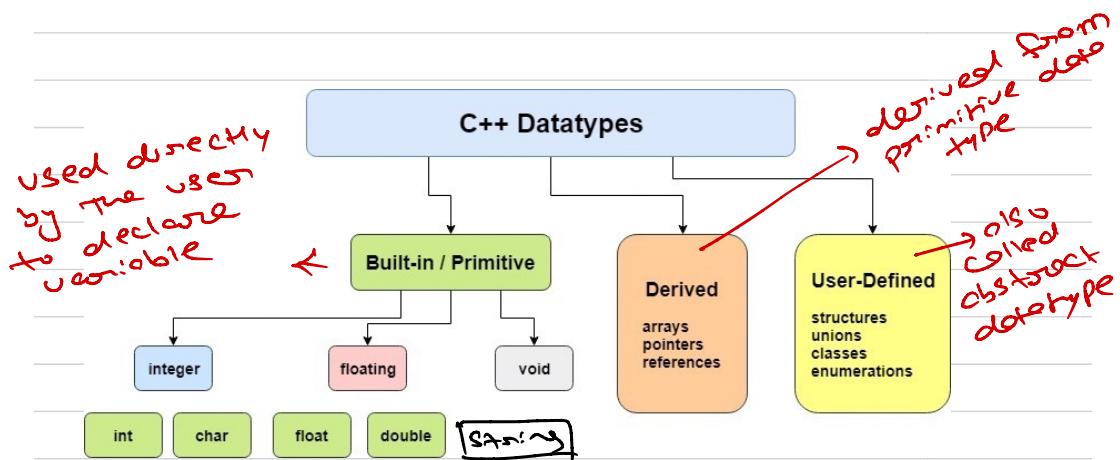
RAM
Reserved memory
for variable

How to declare multiple variable

Syntax type variable-name1 , variable-name2;

- * A variable name can consist of alphabets (both upper & lower case), number, & the underscore '_' character.
However, the name must not start with a number or special characters.

Data types → specifies the size & type of information the variable will store



```
int myNum = 5;           // Integer (whole number)
float myFloatNum = 5.99; // Floating point number
double myDoubleNum = 9.98; // Floating point number
char myLetter = 'D';     // Character
bool myBoolean = true;   // Boolean
string myText = "Hello"; // String
```

Basic Data types

Data Type	Size	Description
boolean	1 byte	Stores true or false values
char	1 byte	Stores a single character/letter/number, or ASCII values
int	2 or 4 bytes	Stores whole numbers, without decimals
float	4 bytes	Stores fractional numbers, containing one or more decimals. Sufficient for storing 6-7 decimal digits
double	8 bytes	Stores fractional numbers, containing one or more decimals. Sufficient for storing 15 decimal digits

Data type with their size & range

Data Type	Size (In Bytes)	Range
short int	2	-32,768 to 32,767
unsigned short int	2	0 to 65,535
unsigned int	4	0 to 4,294,967,295
int	4	-2,147,483,648 to 2,147,483,647
long int	4	-2,147,483,648 to 2,147,483,647
unsigned long int	4	0 to 4,294,967,295
long long int	8	-(2^63) to (2^63)-1
unsigned long long int	8	0 to 18,446,744,073,709,551,615
signed char	1	-128 to 127
unsigned char	1	0 to 255
float	4	
double	8	
long double	12	
wchar_t	2 to 4	1 wide character

How data is stored?

Signed ($-ve \rightarrow +ve$)
unsigned (+ve)

Data are stored in the memory in the form of binary.

Unsigned Data values \rightarrow It can hold positive number & zero. $0 \rightarrow +ve$

for ex int a = 5 ;

5
a

In actual, values are not stored like this in memory.

In memory, it is stored the binary of 5.

Binary :- numbers & values are represented by only two digits \rightarrow 0 and 1.

Binary is base-2, meaning that it only uses two digits or bits.

In the above ex int a = 5 ; where a is an integer data type which contains the 4 bytes. i.e. 32 bits.

int \rightarrow 4 byte \rightarrow 32 bit



Another Ex :-

Char letter = 'A' ;

Each character is mapped with ascii value.

for Ex A $\xrightarrow[\text{with}]{\text{mapped}}$ 65

Binary of 65 $\Rightarrow 1000001$

char \rightarrow 1 byte \rightarrow 8bit \rightarrow $0 - 255$,
range



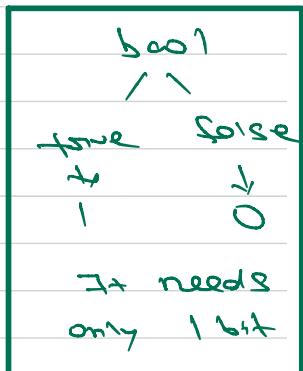
bool flag = true ;

1 byte \rightarrow 8 bits

why bool datatype contain 8 bits in memory?

Smallest space addressable in memory is 1 byte. that's

why bool contain 1 byte memory space



Signed Data values → It can hold
negative, positive & zero -ve ↔ 0 ↔ +ve

→ UR values are stored same as the
values? NO

There are three steps to stored the
negative values :-

- ① Ignore the -ve sign
- ② find binary equivalent
- ③ takes 2's compliment

It is a mathematical operation to reversibly
convert a +ve binary number into a
-ve binary number with equivalent -ve value.

How to make 2's compliment :-

{ 1's compliment
adding 1 } ↘

inverting or flipping all bits —
changing every $0 \rightarrow 1$ & every
 $1 \rightarrow 0$.

int a = -5;

-5
a

① Ignore the -ve sign

② binary of a in 32 bit

0 0000000 0000000 0000000 00000101

③ Convert 2's complement

③.1 1's complement

1111111 1111111 1111111 1111010

③.2 Adding 1

1111111 1111111 1111111 1111010
+ 1

1111111 1111111 1111111 1111011



binary of -5 in memory

* binary converted into -ve decimal number with the process

→ 2's complement

→ binary to decimal conversion

→ adding negative sign

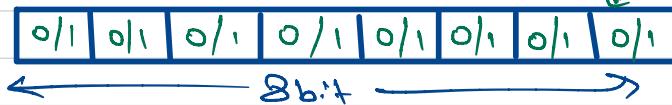
formula $\rightarrow 0 \rightarrow 2^n - 1$ where $n = \text{bits}$

Ranges of data types \rightarrow unsigned data

1) unsigned char \rightarrow 1 byte \rightarrow 8 bits

$$\text{Total Combination} = 2^8 \Rightarrow 256$$

why use 2^8 ?



Possibility of each bit contain value is 2 that is 0 or 1

$$\text{Range} \Rightarrow 0 - 255$$

001

$$0 - \boxed{2^8 - 1}$$

why -1 ?

because the range calculated from 0

2) unsigned int \rightarrow 4 byte \rightarrow 32 bits

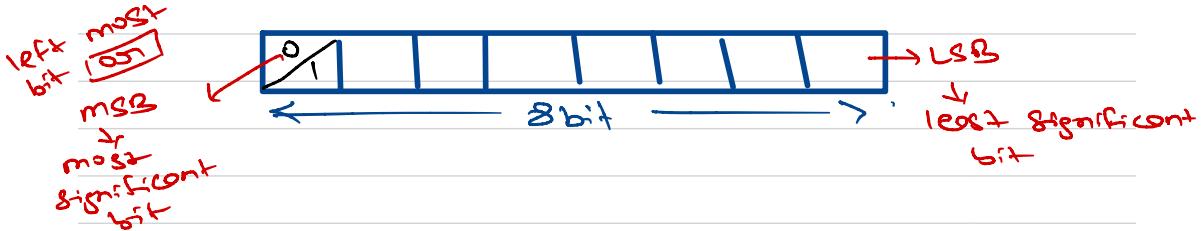
$$\text{Total combination} = 2^{32}$$

$$\text{Range} \Rightarrow 0 - 4294967295$$

$$0 - \overset{001}{2^{32}} - 1$$

we -o we
Ranges of Data types \rightarrow Signed data

formula :- $-2^{n-1} \rightarrow 2^{n-1} - 1$



MSB works like the **Sign bit**.
MSB bit tell us whether the number is
-ve or +ve.

In Signed data we does not consider
the MSB.

therefore, we have only 7 bits.

? char \rightarrow 1 byte \rightarrow 8 bits

+ve = $\Rightarrow 2^7$

The diagram shows a horizontal row of 7 boxes. The first box contains a '0'. A bracket underneath the boxes is labeled '7 bit'.

-ve = $\Rightarrow 2^7$

The diagram shows a horizontal row of 7 boxes. The first box contains a '1'. A bracket underneath the boxes is labeled '7 bit'.

range $\sim -2^7 \rightarrow 2^7 - 1 \Rightarrow -128 \rightarrow 127$

2) Int \rightarrow 4 byte \rightarrow 32 bit

the range $\Rightarrow 0 \rightarrow 2^{n-1} - 1$

$$0 \rightarrow 2^{32-1} - 1$$
$$0 \rightarrow 2^{31} - 1$$

the range $\Rightarrow -2^{n-1} \rightarrow -1$

$$-2^{32-1} \rightarrow -1$$
$$-2^{31} \rightarrow -1$$

combined range $\Rightarrow -2^{31} \rightarrow 2^{31} - 1$

* How to check the datatype size in C++ program?

Using `sizeof()` operator we can check the size of data.

Syntax `sizeof (datatype);`

Operators

operator is a symbol that operates on a value to perform specific mathematical & logical computations

The operators that perform operations on single operand to produce a new value

Operators in C++		
	Operator	Type
Unary operator	<code>++</code> , <code>--</code>	Unary operator
Binary operator	<code>+</code> , <code>-</code> , <code>*</code> , <code>/</code> , <code>%</code> <code><</code> , <code><=</code> , <code>></code> , <code>>=</code> , <code>==</code> , <code>!=</code> <code>&&</code> , <code> </code> , <code>!</code> <code>&</code> , <code> </code> , <code><<</code> , <code>>></code> , <code>~</code> , <code>^</code>	Arithmetic operator Relational operator Logical operator Bitwise operator
Ternary operator	<code>=</code> , <code>+=</code> , <code>-=</code> , <code>*=</code> , <code>/=</code> , <code>%=</code> <code>:</code>	Assignment operator Ternary or conditional operator

Arithmetic Operator:- used to perform mathematical operations

Operator	Name	Description	Example
<code>+</code>	Addition	Adds together two values	<code>x + y</code>
<code>-</code>	Subtraction	Subtracts one value from another	<code>x - y</code>
<code>*</code>	Multiplication	Multiplies two values	<code>x * y</code>
<code>/</code>	Division	Divides one value by another	<code>x / y</code>
<code>%</code>	Modulus	Returns the division remainder	<code>x % y</code>
<code>++</code>	Increment	Increases the value of a variable by 1	<code>++x</code>
<code>--</code>	Decrement	Decreases the value of a variable by 1	<code>--x</code>

2) Assignment operators :- used to assign the values to the variable

Operator	Example	Same As
=	x = 5	x = 5
+=	x += 3	x = x + 3
-=	x -= 3	x = x - 3
*=	x *= 3	x = x * 3
/=	x /= 3	x = x / 3
%=	x %= 3	x = x % 3
&=	x &= 3	x = x & 3
=	x = 3	x = x 3
^=	x ^= 3	x = x ^ 3
>>=	x >>= 3	x = x >> 3
<<=	x <<= 3	x = x << 3

3) Comparison operators :- used to compare the values of variables

Operator	Name	Example
==	Equal to	x == y
!=	Not equal	x != y
>	Greater than	x > y
<	Less than	x < y
>=	Greater than or equal to	x >= y
<=	Less than or equal to	x <= y

4) Logical operators :- is used to determine the logic between values of variables

Operator	Name	Description	Example
&&	Logical and	Returns true if both statements are true	$x < 5 \&\& x < 10$
	Logical or	Returns true if one of the statements is true	$x < 5 x < 4$
!	Logical not	Reverse the result, returns false if the result is true	$!(x < 5 \&\& x < 10)$