Rajneesh Kumar Notes
02 January 2024 11:43 PM

Namespace

std namespace include (cout, cin, endl) and it define inside iostream

std	abc	xyz
cout	cout	cout

#include<iostream> → to perform task like input and output (also called preprocessor directive)

using namespace std; → use std namespace for input/output operation to get rid of std::cout/cin/endl

int main(){ → int means return type, and main is function name. Basically it's entry point of program.

If program contain 1,00,000 line of code, compiler find where is int main()

return 0;

}

Note: /n for line break and typically faster than other method like endl. The reason for this is that \n is a simple escape sequence that insert a newline character, which is a low-level operation that directly moves the cursor to the beginning of the next line in the output. On the other hand, endl not only adds a newline character but also flushes the output buffer(cache or memory location where data held until output device/file is ready)

cout → character output

cin → character input

; → Termination of C++ Statement

return 0 → Successful execution of program.

Note: void main return nothing

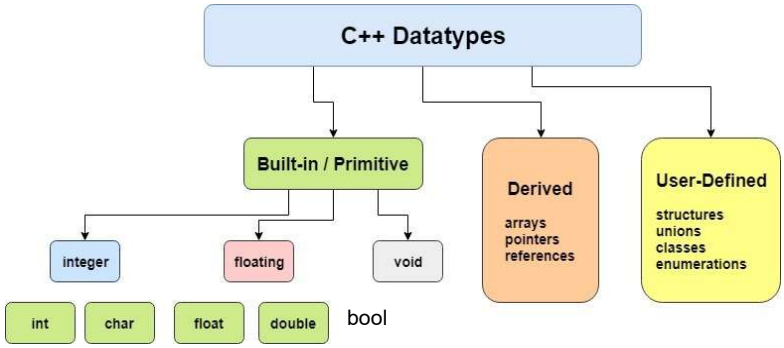
Datatype and Variable

→ Name given to memory location is called variable.

→ Value stored in variable can be change during program execution, operation done on variable effect that memory location.

☐ int name;

datatype variable name



Unsigned (only +ve)

unsigned(+ve and -ve both 1 byte = 8bit and bit could be 0(False) & 1 (True)

Type	Typical Bit Width	Typical Range
char	1byte	-127 to 127 or 0 to 255
unsigned char	1byte	0 to 255
signed char	1byte	-127 to 127
int	4bytes	-2147483648 to 2147483647
unsigned int	4bytes	0 to 4294967295
signed int	4bytes	-2147483648 to 2147483647
short int	2bytes	-32768 to 32767
unsigned short int	Range	0 to 65,535
signed short int	Range	-32768 to 32767
long int	4bytes	-2,147,483,648 to 2,147,483,647
signed long int	4bytes	same as long int
unsigned long int	4bytes	0 to 4,294,967,295
float	4bytes	+/- 3.4e +/- 38 (~7 digits)
double	8bytes	+/- 1.7e +/- 308 (~15 digits)
long double	8bytes	+/- 1.7e +/- 308 (~15 digits)
wchar_t	2 or 4 bytes	1 wide character

Note: Size of datatype are machine architecture dependent (32 bit or 64 bit). e.g. How many bit of our machine CPU.

Also Datatypes tell us 2 major thing

→ type of data

→ Size of data

How to calculate range (Signed +/-)

1. Remind size of datatype (use sizeof() operator)
2. Convert into bit (e.g. Size of int is 4, so 4*8=32 bits)
    1. Size of char is 1byte, so 1*8=8 bits (2^{7} to 2^{7-1})
3. Signed -> 2^{31} to 2^{31-1} = - 2,147,483,648 to 2,147,483,647

31 because there is 0 between positive range and negative range, so took 1 bit.

How to calculate range (Unsigned)

1. Remind size of datatype (use sizeof() operator)
2. Signed -> 2^{32} = 0 to 4,294,967,295

How data is stored in Memory ?

int a = 8; ⟶ Binary = 1000 (4 bit)

00000000 00000000 00000000 00001000

How Negative Number are stored in Memory ?

The first bit ("left most bit" or "Most Significant bit") represent the number is +ve or -ve.

→ If it starts with 0, then the number is +ve.

→ Else -ve.

If we store a negative no., then a series of steps to be followed:

→ Ignore the negative sign

→ Convert into binary representation

→ Take 2's complement and store

- 1's complement (flip the bit 0 - 1)
- 2's complement (Add +1 in last bit)

→ For e.g.

- A = -5 (ignore negative sign) -> A = 5
- 00000000 00000000 00000000 00000101
- 1's comp. 11111111 11111111 11111111 11111010
- 2's comp. and store +1
- 11111111 11111111 11111111 11111011

How to display the -ve number?

Simple take 2's complement

→ 11111111 11111111 11111111 11111011

→ 1's comp. 00000000 00000000 00000000 00000100

→ 2's comp. +1

→ 00000000 00000000 00000000 00000101

Operators

→ Arithmetic Operator	[+, -, *, /, %]	int/int → int float/int → float double/int → double
→ Relational	[<, <=, >, >=, ==, !=]	
→ Assignment	[=, +=, -=, *=, /=, %=]	
→ Logical	[&&, ||, !]	
→ Bitwise	[&, |, <<, >>, ~, ^]	
Special	[sizeof, ?:]	