* Data Structures and Algorithms *

(C++)

Basics to Advance.

LinkedIn Profile: subrat-kumar-singh-597973207.

written By: Subrat Kumar Singh.

Contact No: 9350857818.

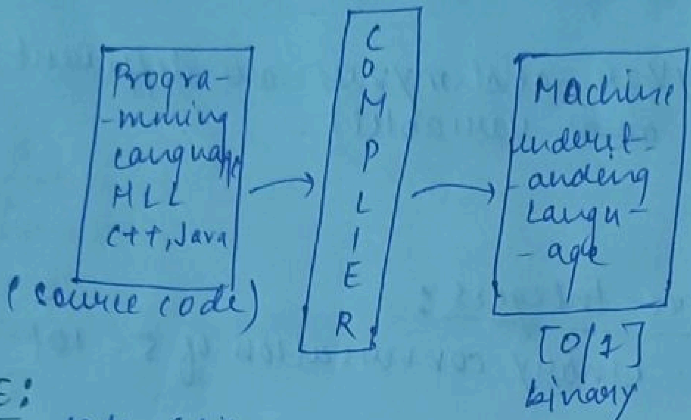
gmail: subratsingh2001@gmail.com

28/01/2023 **Programming language:**

why we need it

* Because to instruct the computer the carry out real-life problems in the code form.

* Programming language have their own complier which converts HLL to machine Level Language i.e 0/1 binary.



Programming language HLL C++, Java
(source code)
COMPILER
Machine understanding Language
[0/1] binary

* **IDE:**

code Editor where we can write our source code.
ex: vs-code, sublime, code-blocks.

* **our-first code:** Namaste Bharat

ex:
```
int main() {
std::cout << "Namaste Bharat";
}
```
code inside the { } brackets belongs to main()

Note: The execution of every program begun from main function. compiler always look for main function.

* std::cout << "Namaste Bharat";

output

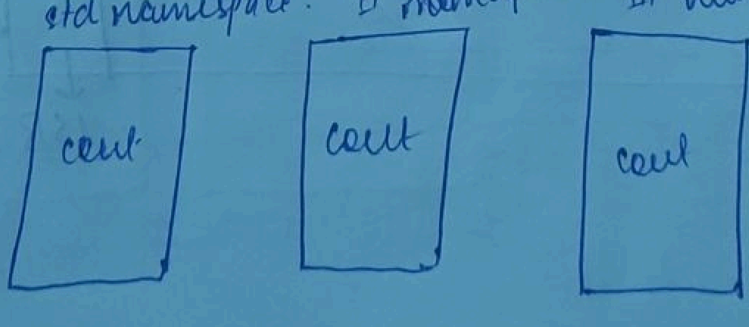
std → standard namespace

cout → print

<< → left shift operator for outputing.

* **Namespace:** It is area/region where particular keyword have different meaning or particular keyword scope exist

ex: std namespace. II nd namespace III rd namespace.



* variable naming convention:

(*) Name can contain letter, digits and underscore

(*) Name must begin with _ or letter.
↓ underscore

(*) There should no whitespace or special character like !, #, %.

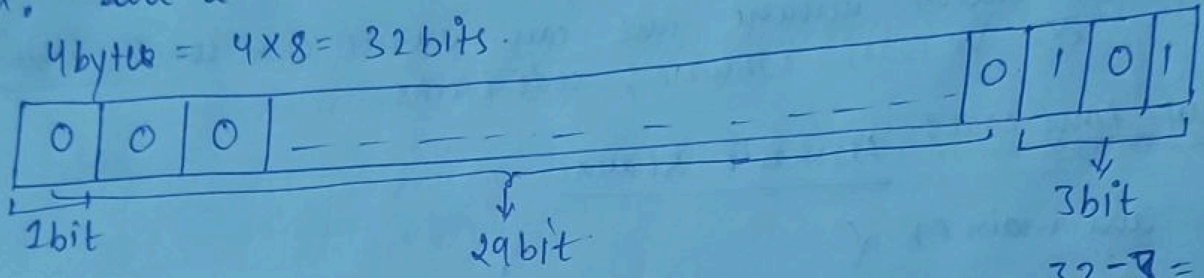
(*) Case sensitive (myVar and myvar are different ~~[illegible]~~ variables)

* How data is stored?

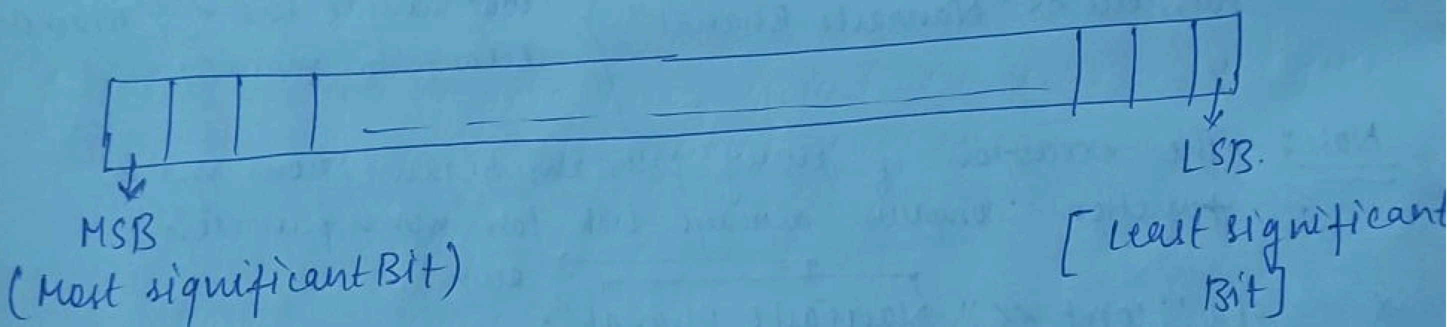
Positive vs Negative Integers:

ex: int a=5   binary conversation of 5 = 101

4bytes = 4×8 = 32 bits.

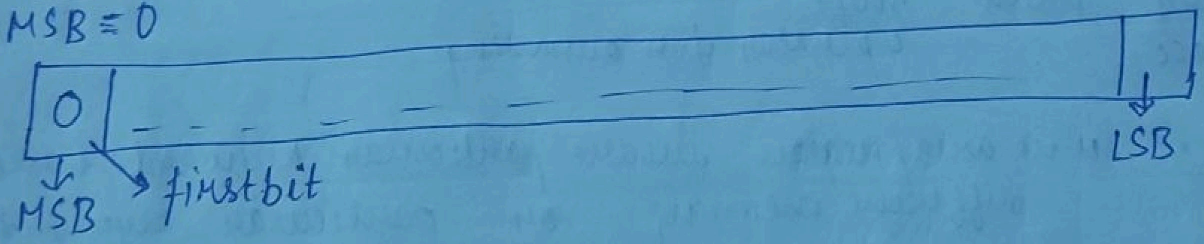


32-3 =

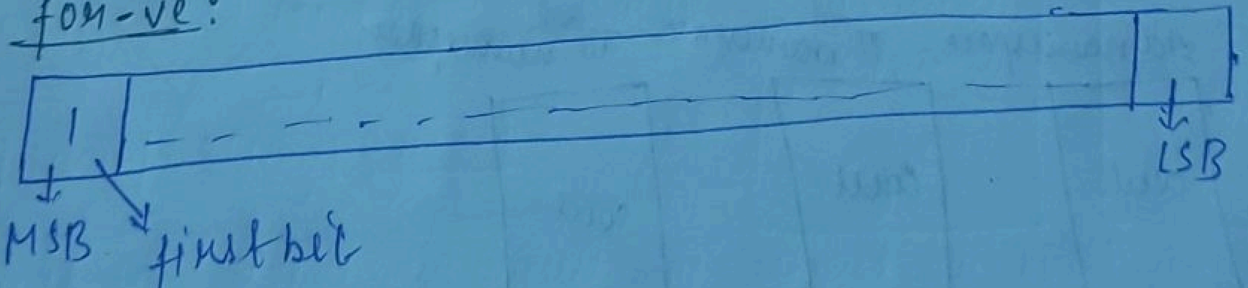


* For +ve:

MSB = 0



* for -ve:



* How negative no are stored in memory?

For -ve no are stored with the help of 2's compliment. For 2's compliment we need to find 1's compliment-

ex: int a=-5

① ignore -ve sign ② find the binary equivalent.

③ 2's complement

* Decimal

1+1=2

where 2 in binary = 10

$$\begin{array}{r} 1 \\ +1 \\ \hline 10 \end{array}$$

ex: int a=-5

① ignore -ve sign

② find the binary equivalent $\overset{5}{}$

5 = 0 0 0 - - - - - - - 0 1 0 1

③ 2's complement → 1's complement
→ then +1

5 = 0 00 - - - - - - 0 1 0 1

1's compliment

= $\boxed{1}$ 1 1 - - - - 1 0 1 0

↳ MSB = 1 = -ve number

$$\begin{array}{r} 1\ 1\ 1\ -\ -\ -\ -\ -\ -\ -\ 1010 \\ +\ \ 1 \\ \hline 1\ 1\ 1\ -\ -\ -\ -\ -\ 1011 \end{array}$$

int a=-5 is stored in memory having binary equivalent

1	1	1	- - -	1	0	1	1

* How to read the ~~binary equivalent~~ 2's compliment.

1 1 1 - - - - - - 1 0 1 1

find 1's compliment

0 0 0 - - - - - 0 1 0 0

then add 1

0	0	0	- - - - - -	0	1	0	1

= int a = -5

* #include <iostream>
  ↓
  input/output headerfile

cout << endl; }
cout << |n; } to print in next line.

' ' : to display single character

" " : to display the string combination of characters.

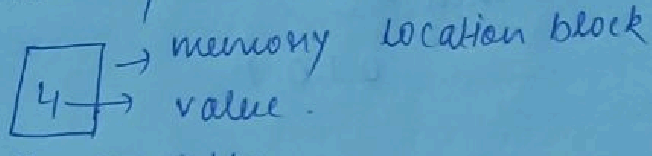
* Taking Input in C++:

cin keyword used to take input from user.

ex: cin >> username;
cin ↓ input
>> → right shift operator.

* Variables: Variables are the data containers or named memory location.

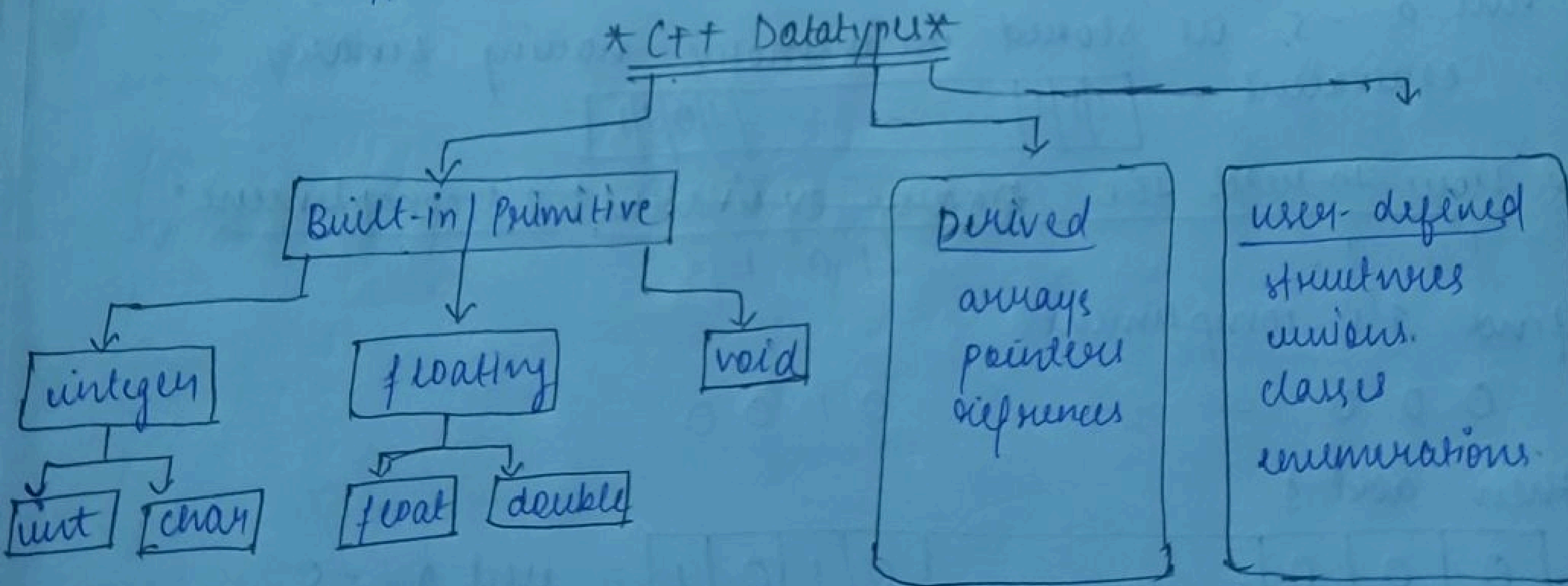
ex:



a = variable

* Datatype: type of data being stored. i.e could be integer, character, ~~[illegible]~~ decimal value etc.

ex: int ~~va~~age = 14;
int ↓ datatype
age ↓ variable
14 → value.



* Built-in: datatypes which pre-defined or built in before said primitive/ built in

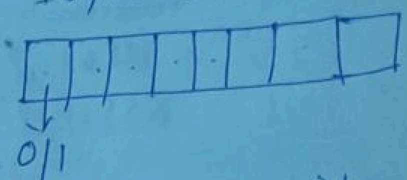
* **Derived:** Datatypes which derived with the help of inbuilt or primitive datatypes. ex: arrays etc.

* **user-defined:** Datatypes which are defined by user to create your own datatypes.
ex: class, structures etc.

* **Size of Datatypes:**

int: 4bytes / 2bytes depends on system
void: null
char = 1 byte.
short = 2byte
long = 4byte.
long long = 8byte.
float = 4byte
double = 8 bytes

1 byte = 8 bits

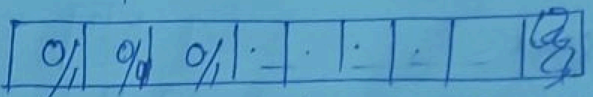


Each block consists of either 0/1 ∴ Total no. of combination is 2^8

The maximum value it can take 2^8-1

Taking example of char

char = 1 byte = 8 bits =



each block consists of 2 possible value i.e 0/1.

Total possible combination is $2^8 = 256$

~~Total~~ Max value it can take = $2^8-1 = 255$

[Here -1 because first ~~that~~ block/bit is reserved to determine whether number is +ve or -ve. if it is 0 then the number is +ve else it is 1 then it is -ve]

In general 2^n-1 → Max values and 2^n → Total combination

where n = no. of bits.

To find the range:

Total possible outcomes = $2^8 = 256$

Max value = 256 - 1 = 255

∴ $\frac{256}{2} = 128$ To be continued later

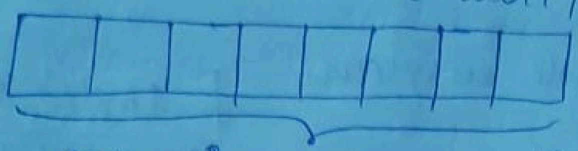
* **ASCII Table**: The ascii value or decimal table for

A : 65 a = 97

Z : 90 z = 122

char ch = 'a' ⟶ $\boxed{97}$ ch

97 is decimal value convert form of binary



256 = 2^8 = Total combination

255 = 2^8-1 : Total possible values.

int age = 14

8bits ←	8bits	8bits	8bits.
1byte	1byte	1byte	1byte

4X8 = 32 bits = 2^{32} → Total combination

$2^{32}-1$ → Max values it can take.

* **Boolean Datatypes**: 1byte size space . ,,

True/False value.

True → 1

0	0	0	0	0	0	0	1

False → 0

0	0	0	0	0	0	0	0

The filling the binary blocks take place from right to left.

* float: 4byte = 4X8 = 32 bits

double: 8byte = 8X8 = 64 bits.

1	01
2	10
3	11
4	100
5	101
6	110
7	111
8	1000
9	1001
10	1010
15	1111

Max 2 digit Binary No : 11 = 3 = 2^2-1

Max 3 " " " : 111 = 7 = 2^3-1

Max 4 " " " : 1111 = 15 = 2^4-1

$\boxed{=2^n-1}$

2X2X2X2

* Interesting Problem:

1byte	1byte	1byte	1byte

Datatype: → type of data i.e int, char etc.
will tell → will tell how much space will it take in memory.

* Signed vs Unsigned data:

signed : can take +ve, 0, -ve values.
unsigned : can take +ve value.

ex: consider xyz → 6byte = 6×8 = 48 bit
Total combination = 2^{48}
Max value it can take = $2^{48}-1$

To find Range

unsigned : $0 \longrightarrow 2^{48}-1$
signed : $\frac{2^{48}}{2} = 2^{47}$ [we need to divide it into two equal parts]
: $-2^{47} \rightarrow 2^{47}-1$
$-2^{47} \rightarrow 0 \rightarrow 2^{47}-1$

ex: short = 2bytes
= 8×2 = 16bits.
T.C = 2^{16}
Max possible value = $2^{16}-1$

find range

unsigned : $0 - 2^{16}-1$
signed : $-2^{15} \longrightarrow 0 \rightarrow 2^{15}-1$

ex: char = 1byte = 8 bits
T.C = 2^{8} , Max possible value = $2^{8}-1$

find range

unsigned : $0 - 2^{8}-1$
signed : $-2^{7} - 2^{7}-1$

Note
General Formula
signed:
$(-2^{n-1} \rightarrow 2^{n-1}-1)$
unsigned:
$(0 \rightarrow 2^{n}-1)$
where n is no. of bits.

* **Typecasting:** conversion of one datatype to another datatype

- → implicit Type casting
- → explicit Type casting

* **Implicit Type casting:** when one datatype is converted to another datatype automatically said to be implicit type casting.

ex:
```
char ch= '97;
cout << ch;        output : a
                   a = 97 (ASCII value)
```

* **Explicit Type casting:** when we convert one datatype to another said explicit type casting.

ex:
```
double d = 8.9;          output : 8
int x = (int) d;
cout << x;  ↳ we instruct to convert one
              datatype to another
double x = 4.5
int y = (int) x + 4;     output = 4+4
cout << y;                      = 8
```

* **Operators:**

① **Arithmetic:** %, +, -, /, *

② **Relational:** >, <, >=, <=, != ; ==,

③ **Assignment:** int a=5

④ **Logical:** &&, ||, ! → not operator

&& ↓ AND operation

|| ↓ OR operator

* $\frac{int}{int}$ = int, $\frac{float}{int}$ = float ; $\frac{float}{double}$ = double, $\frac{double}{int}$ = double.

$\frac{double}{float}$ = double, $\frac{int}{float}$ = float.

* char ch = 234567.

as 234567255