Programming languages

Friday, 25 August 2023 9:08 PM

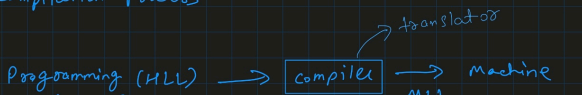
→ Why do we need programming languages?

→ to communicate with machines effectively

→ programming language has a fixed set of rules or has a standardized formate

→ every language has it's own compiler / interpreter.

→ Compilation Process



→ Where to code

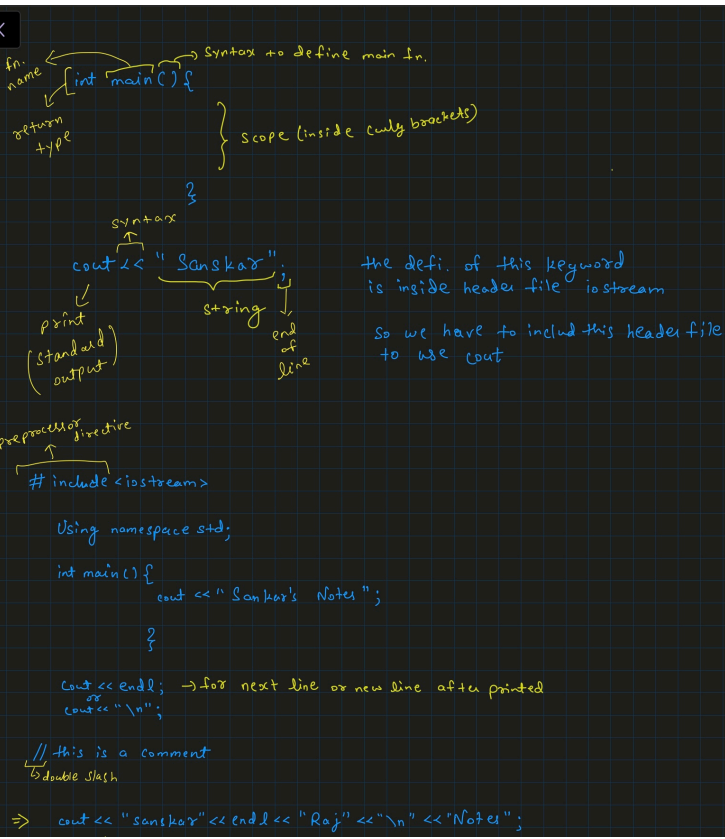
→ code editor / IDE
→ VS code
→ sublime
→ code blocks
→ X code
→ Atom
→ codehelp website has an online compiler

→ Writing down first code in cpp

code starts → int main() → fn.

function is a block of code which takes input(s) and gives output

→ it is used for code reusability


fn. name
Syntax to define main fn.
int main() {
return type
scope (inside curly brackets)
}
syntax
cout << " Sanskar ";
string
print (standard output)
end of line
the defi. of this keyword is inside header file iostream
So we have to includ this header file to use cout
preprocessor directive
include <iostream>
Using namespace std;
int main() {
cout << " Sankar's Notes ";
}
Cout << endl; → for next line or new line after printed
or
cout << "\n";
// this is a comment
double slash
⇒ cout << "sanskar" << endl << "Raj" << "\n" << "Notes";

⇒ cout << "sanskar" << endl << "Raj" << "\n" << "Notes";

Output

» sanskar
» Raj
» Notes

H.W
return 0 at the end of code

① #include <iostream>
② Using namespace std; → to use only defin. inside namespace

We can create our own name space → portion of code

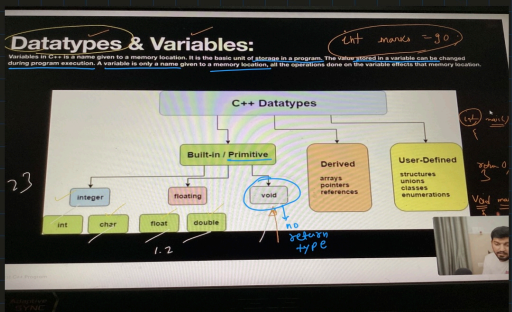
② int main()
{
cout << "Sanskar"; → End of line
↳ string
}

H.W
What if we write the line again after semi colon

cout → character output
cin → character input

→ Datatypes & Variables:

Variables in C++ is a name given to a memory location



A variable is a name given/assigned to a memory location

the type of data stored in the variable is told by datatype of variable

int → 4 byte
char → 1 byte

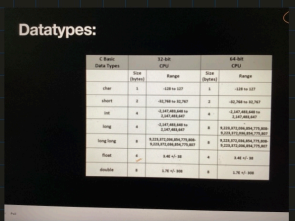
'a'	4
ch	int
1 byte	4 byte

Eg: int x = 9

9
x

Declaration
↳ int num;

initialisation
↳ int num = 50;



→ size of the integer is dependent on machine it can be of 2 byte or 4 byte.

→ we can check it by "sizeof(variable_name)"

→ check on system

int datatype sizeof(4); → 4 byte
char " sizeof('a'); → 1 byte
float " sizeof(2.143); → 4 byte
long datatype sizeof(23); → 8 byte

if we don't assign any value like
int num;
cout << num; → it will print garbage value

→ Variable Naming Convention

→ should begin with Alphabet
→ should not have any spaces
→ digits may be used after alphabet
→ No special character except underscore (_)
→ No keywords should be used

How data is stored?

→ Numbers are stored in binary digits in memory
int → 4 byte → 32 bits

7 → 00000000000 00111
(32 bits)

→ Every character is mapped with an integer value
'a' → 97
So, char ch = 'a';
↳ 97 → int → 4byte → 32 bits
0000 01011

Note: this 97 is in char data type of size 1 bit

→ bool ⇒ true or false
'1' '0'
1byte → 8bits
Since bool required only 1 bit ∴ all the remaining 7 bits got wasted.

→ binary equivalent of +ve no.

10 → 1010
7 → 0111
15 → 1111

→ 1's compliment

00011 → 11100
10101 → 01010
11111100 → 00000011
10111001 → 01000110
11100111 → 00011000

→ 2's compliment

- (A) find 1's complement
- (B) add +1

i/p 1011

(A) 0100

(B) + 1

0101

i/p 1000

(A) 0111

(B) + 1

1000

→ binary eq. of −ve no.

- (A) ignore −ve sign
- (B) find binary equivalent
- (C) take 2's compliment

→ find a = −5

(A) 5

(B) 00000000 00000000 00000000 00000101

(C) 11111111 11111111 11111111 11111010

\+ 1

11111111 11111111 11111111 11111011

→ char → 1 byte → 8 bit

index →	0	1	2	3	4	5	6	7
	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1

↓ 2 choices

total combination → $2^8 = 256$

range → $2^8 - 1$

total comb^n. → 2^n , n = no. of bits

range → $2^n - 1$ (n bits − 1 because index starts from 0)

Ex → 71 byte → 71 × 8 bits

→ 568 bits

→ signed int can store both +ve and -ve int but unsigned can only store +ve.

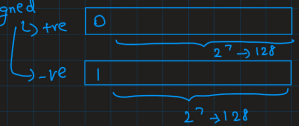
until this time we were talking about unsigned

→ In signed we represent sign with the left most bit, 1 for -ve and 0 for +ve.

H.W
Check range of signed and unsigned int

H.W
-100, -125 Mapped by which character

for signed



+ve ⇒ 0 → 128 - 1
0 → 127

-ve ⇒ -1 → -128
↳ not taking zero because already included in +ve

-128 → 127

→ Operators

Arithmetic	→ +, *, -, /, %	int/int → int
Relational	→ >, <, >=, <=, !=, ==	float/int → float
Assignment	→ =, +=, -=, *=, /=	double/int → double
Logical	→ &&, ||, !	
Bitwise	→	

&& → saari cond". true then true other wise false
|| → any one cond". true than true

→ cin ⇒ to take input from user

```
int marks;
cin>>marks; ⇒ now we can take an integer as input
```