→ Variables :- variables are like vessels in which we store our data/input during coading

ex.-

```
Int x=2;
Char b=h;
```

here, x & b are variable of type int and char respectively.

→ Comments :-

Comments are sentences / lines written by us in programme but we dont want that it is run by compiler.

Comments generally contains information about programme or given line of coad.

Single line Comments :-

Made by putting // before that line.

Int

Multi line comment :-

If we have to write multiple line as a comment & it is not effective to pull // before each line, For that we can write that long comment in

/* */

It will also ignored by compiler.

→ ## Data type :-

Data type denotes type of data a variable can hold.

It is of three type :-

i) Built-in
ii) User defined
iii) Derived data type

Built-in :-

int
float
char
double
bool

User defined :-

Struct
Union
Enum

Derived :-

Array
function
pointer

NOTE :- Local variable will get more preferance that global variable.

<< are called insertion operator.

and >> is called extraction operator

3. WAP to input two integers and print sum of it.

```
#include<iostream>
using namespace std;

int main()
{
    int x, y;
    cout<<"write two integer values\n";
    cin >> x >> y;
    cout<<"sum of "<<x<<"and"<<y<<"is"
        <<x+y;

    return 0;
}
```

⇒ Header files :-

Header files are standard library which contains details about some specific functions we do in programms, so, to do that

specific programme, you have to include header file that contains details about that function.

ex. – <iostream>
<stdlib>
<cmath>

ex.

There are two types of header file

i) System defined header file:-

Those header file which comes with compiler and for whom we don't have to define for them we don't have to do.

To use them you don't have to define them, just include them in coad.

ex. –

<iostream>
<stdlib>

ex.

ii) User defined headerfile :-

Those header files which don't come with compiler and we have define them to use, are User define headerfile

Syntex :-

```
# include " this.h"
```

Also, if you only write this compiler will show error, so, you have to add a file name 'this.h' in the same directly to run the programme.

To know more about headerfiles & stuff like this you can visit

CPP refrence website.

alt + shift to make copy
alt * multiple cursors in VS

Q. WAP to demonstrate the use of comparison operators :-

```
#include<iostream>
Using namespace std;

int main()
{
   int a=4
   int b=7

   cout<<" value of a=b is"<<(a==b)
   Cout<<"value of a>b is"<<(a>b)<<endl;
   Cout<<" value of a<b is"<<(a<b)<<endl;
   Cout<<" value of a<=b is"<<(a<=b)<<
   cout<<" value of a!=b is"<<(a!=b)<<
   cout<<" value of a>=b is"<<(a>=b)<<

   return 0;
}
```

In this program by putting commands/codes like a==b & a>=b is paranthesis it will give result as 1 or 0

~~is a==b is give result as 1~~
~~& if a<=b i~~
if condition is true return 1
if condition is false return 0

We can also use logical operator like this for example

```
cout<<"Result of AND operator is:"<<
        ((a==b)&&(a>b))<<endl;
```

If we run this code with appropriate structure it will print result as 0 or 1.

if ((a==b)&&(a>b)) both true then 1
((a=b)&&(a>b)) both not true then 0

NOTE:- Comparison & logical operators gives results / returns boolean (zero or one).

→ Global variables & local variables:-

Global variables are those variables whose scope/reigon is all over code (i.e. for all functions). It is defined outside of any function like main(), user defined function etc.

Local variables are those variables whose scope is only inside the function it is defined. It is defined inside any function like we define any variable in main() function.

If we have a local variable & a global variable with same name and a function have to choose one, function will give priority to local variable, but if you want to use global variable inside function where, there is local variable with same name you have to use scope resolution

operator (::) before variable.

ex.

```
#include<iostream>
Using namespace std;
int c=45;
int main()
{
  int a,b,c;

  cout<<" enter value of a:"<<endl;
  cin>>a;

  cout<<" enter value of b:"<<endl;
  cin>>b;

   c=a+b;
  cout<<" The sum is :"<< c <<endl;
  cout<<" The value of global c is :"<<
                          ::c <<endl;

  return 0;
}
```

If we take a literal 34.4 in C++ it is by default a double, but if we want it to recognised it as float we write it as

34.4f

It is now a floating no.

If we write

34.4l

It is a long double no.

You can think, it is not necessary to do so, in general coding can write

```
float x = 34.4;
long double z = 34.4;
```

but it is useful when we do typecasting & function overloading.

To demonstrate this difference of of suffix used we can see a programme

```
#include<iostream>
using namespace std;

int main()
{

  cout<<"size of 34.4 is:"<< sizeof(34.4)<<endl;
  cout<<"size of 34.4f is:"<<sizeof(34.4f)<<endl;
  cout<<"size of 34.4l is:"<<sizeof(34.4l)<<endl;

  return 0;
}
```

→ Reference Variable :-

We create reference variables to call an existing variable with diffe. name.

For example we have a variable

```
int x = 10;
```

and now want that here is we want to call x with another name

Say 'y', to do so, we write

int & y = x;

By this read 'y' is reference variable of 'x', if we call 'y' actully we will call 'x'. Also here, we have not created a new variable, we have just give 'x' another name 'y' and after now 'x' will also be known as 'y'.

→ Constants :-

In c++ if you have variable say

int a = 35;

and later in programme you have changed

a = 45;

then value of 'a' changes as you wish.

But what if you want that your values doesn't changes for that you have to use keyword const.

ex-

```
#include<iostream>
using namespace std;

int main()
{
    const int x=45;

    cout<<"value of x:"<<x<<endl;

    x=49;
    cout<<"value of new x:"<<x;

    return 0;
}
```

In this programme you will get Error that you can't change the value of read only variable, as of

here, x is constant so you can't change its value.

→ Manipulators :-

Manipulators helps to give specific appearance to the output.

'endl' keyword we uses is also a manipulator.

'setw()' is also a manipulator which give output as right aligned.

'setw' comes in <iomanip> library.

```
#include<iostream>
#include<iomanip>

Using namespace std;

int main()
```

```
{
    int a=10, b=132, c=4321;

    cout<<"Value of a" <<a <<endl;
    cout<<"Value of b" << b<<endl;
    cout<<"Value of c" << c <<endl;

    cout<<"value of a" << setw(4) <<a<<endl;
    cout<<" value of b"<<setw(4)<<b <<endl;
    cout<<" value of c"<<setw(4)<<c <<endl;

    return 0;
}
```

→ Operator precedence :-

Operator precedence, ~~of~~ means operator priority, when there are more than two operator in a expression which will used first & estimated by operator priority order.

Operator whose priority is higher will be used first and whose

+ + a

a * +

priority is lower will be used later.

You can go to cppreference website to know operator precedence, but must learn mainly used operator's presedence.

If priority of two or more operator is same, then see we see associativity (means direction of work)

ex -

a * b + c - d + f

⇒ ((((a * b) + c) - d) + f)

here, priority of (*) is higher than (+ & -) so it will initialize first.

But priority of (+ & -) is same so, we see associativity, for (+ & -) associativity is left-to-right so, it initialize in that flow.

a+b
a-b
a*b
a*b
++
==
a+b
++a
-b
a+
b

The concept of operator precedence not seems useful now but it is, it will be used ~~very~~ various times in making complicated programms.

→ Control structure in C++

Control structure give flow, logic & structure to a C++ programme.

It is 3 types -

- → Sequence structure
- → selection structure
- → loop structure

} Basic control structure

→ Sequence structure :-

It gives a programme sequence, in which ~~programme~~ flow programm operate.