Section 2 → Lesson 1.

→ DSA in C++.

Hello world code.

```
#include<stdio.h>
#include<iostream> //header section
using namespace std; //namespace

int main()          //body
{
    cout << "Hello World!" ;//;
                           statement
    return 0;
}
```

⇒ OUTPUT :-
Hello World!

// → comment
cout → print statement.
endl → print end of line
↳ It moves the cursor to the beginning of a new line.

cin → Take input

Page No.
Date

Program 2 → To show the working of endl.

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

int main()
{   // Without endl function
    cout << "Hello world ! ";
    cout << "My name is MAK";

    // with endl function.
        endl
    cout << "Hello World ! " << endl;
    cout << "My name is MAK";
    return 0;
}
```

⇒ OUTPUT :-

```
Hello World ! My name is MAK
Hello World !
My name is MAK
```

* Variable :-
↳ use to store set of bits & data.
↳ A variable is a named container for a set of bits or type of data.

Syntax :-
datatype variable-name = value;
or datatype variable-name;

* Datatype :-

int	→ integer
float	→ decimal floating no.
double	
char	→ characters.
string	→ set of characters.
bool	→ True & False.

* Constant value :-
↳ the value of the variable cannot be change in that particular code.

Syntax;
const <datatype> variable-name;

Eg) const int a = 15;

const< datatype > variable-name;
↳ Invalid Syntax.

* Prog 3 => Datatype & variable.

```
#include <iostream>
using namespace std;
int main(){
    int a = 10;
    float b = 10.11;
    double c = 10.1111;
    char d = 'A';
    string e = "Hello World!";
    bool f = true;

    cout << "The variable are :- " << endl
    << a << endl << b << endl << c
    << endl << d << endl << e <<
    endl << f << endl;

    return 0;
}
```

=> Output :-

The variable are :-

10

10.11

10.1111

A

Hello World!

1

Page No.
Date

⇒ Prog 4 :- Constant Datatype
↳ It does not change the value of a variable.

```
#include<iostream>
using namespace std;
int main()
{
    int a = 10;
    cout << a << endl;
    a = 25;                    //Variable Value changed.
    cout << a << endl;

    const int b = 10;
    cout << b << endl;

    //b = 25;                  ⇒ Errors
    // cout << b << endl;
```

⇒ Output

```
10
25
10
```

» Prog 5 » cin function.

```
#include<iostream.h>
using namespace std;
int main(){
    int a;
    cout<<"Enter the value of
            a :- ";
    cin>>a;
    cout<<a;

    return 0;
}
```

» output :-

Enter the value of a :- 45
45

---

* Arithmetic operators :-

a = b + c;	+	→ add
a = b - c;	-	→ sub
a = b * c;	*	→ multiply
a = b / c;	/	→ division
	%	→ modulas
		↳ returns remainder.

Page No.
Date

* Prog 6 :- arithmetic operators.

```
#include <iostream>
using namespace std;
int main() {
    int a = 100, b = 200;
    cout << a+b << endl;
    cout << a-b << endl;
    cout << a*b << endl;
    cout << a/b << endl;
    cout << a%b << endl;

    return 0;
}
```

OUTPUTS :—

```
120
80
2000
5
0
```

Page No.
Date

* Prog 6 :- Assignment operators.

=	assign
+ =	add & assign
- =	subtract & assign
* =	multiply & assign
/ =	divide & assign
% =	modular & assign
& =	bitwise and assignment operator
| =	" or " "
^ =	" not " "
>> =	" right " "
<< =	" left " "

* Prog 7 :- Assignment operators.

```
#include <iostream>
using namespace std;
int main()
{
  int a;
  a = 10;          // assign the value to the a
  cout << a << endl;
  a += 5;          // add & assign
  cout << a << endl;
  a -= 5;          // subtract & assign
  cout << a << endl;
  a *= 5;          // multiply & assign
  cout << a << endl;
  a /= 5;          // divide & assign
  cout << a << endl;
```

a *= 5;
a %= 5; // modular & assign
cout << a << endl;

return 0;
}

#include <iostream>
using namespace std;
int main()
{
int b = 5;
int c = 3;

b &= c; // Bitwise AND assignment operator
cout << b << endl;
b = 5;
b |= c; // Bitwise OR assignment operators.
cout << b << endl;
b = 5;
b ^= c; // Bitwise XOR assignment operators.
cout << b << endl;
b = 5;
b << 2 c; // Bitwise Left Shift Assignment operator
cout << b << endl;
b = 5;
b >>= c; // Bitwise Right Shift assignment operators.
cout << b << endl;

return 0;
}

Page No
Date

* Comparison operators.

==	→	Equal to
!=	→	Not equal to
<	→	Less than
>	→	Greater than
<=	→	Less than or equal to
>=	→	Greater than or equal to

Prog 8 :— Comparison operators.

```
#include <iostream>
using namespace std;
int main()
{
    int a = 100, b = 20;

    cout << (a == b) << endl;
    cout << (a != b) << endl;
    cout << (a < b) << endl;
    cout << (a > b) << endl;
    cout << (a <= b) << endl;
    cout << (a >= b) << endl;

    return 0;
}
```

* Logical operators.

&& → Logical AND
↳ returns True if both the statements are true.

|| → Logical OR
↳ returns True if one of statements is true.

! → Logical not
↳ returns True if the result is false.

Prog 9 :- Logical operators.

```
#include <iostream>
using namespace std;
int main()
{
    int a = 100;
    cout << (a > 10 && a < 100) << endl;
    cout << (a > 10 || a < 30) << endl;
    cout << !(a > 10 && a < 100) << endl;

    return 0;
}
```

Logical Table of logical operators.

1) Logical AND ⇒ &&

A	B	A && B
0	0	0
0	1	0
1	0	0
1	1	1

2) Logical OR ⇒ ||

A	B	A && B
0	0	0
0	1	1
1	0	1
1	1	1

3) Logical NOT ⇒ !

A	!A
0	1
1	0

* Bitwise operators :-

'&'	→	Bitwise	AND operator
|	→	"	OR "
^	→	"	XOR "
~	→	"	Complement "
<<	→	"	shift left "
>>	→	"	shift right

* Bitwise AND operators. → &

a	b	a & b
0	0	0
0	1	0
1	0	0
1	1	1

Eg)

12 = 01100
25 = 11001 { in binary }

```
  01100
& 11001
-------
  01000   → 8 in decimal.
-------
```

* Bitwise OR operator.

a	b	a | b
0	0	0
0	1	1
1	0	1
1	1	1

Eg)

```
  12 = 01100
| 25 = 11001
-------------
  29 = 11101
```

=> 29 in decimal

* Bitwise XOR operator

a	b	a ^ b
0	0	0
0	1	1
1	0	1
1	1	0

Eg

```
  12 = 01100
^ 25 = 11001
-------------
  21 = 10101
```

=> 21 in decimal

* Bitwise complement operator.

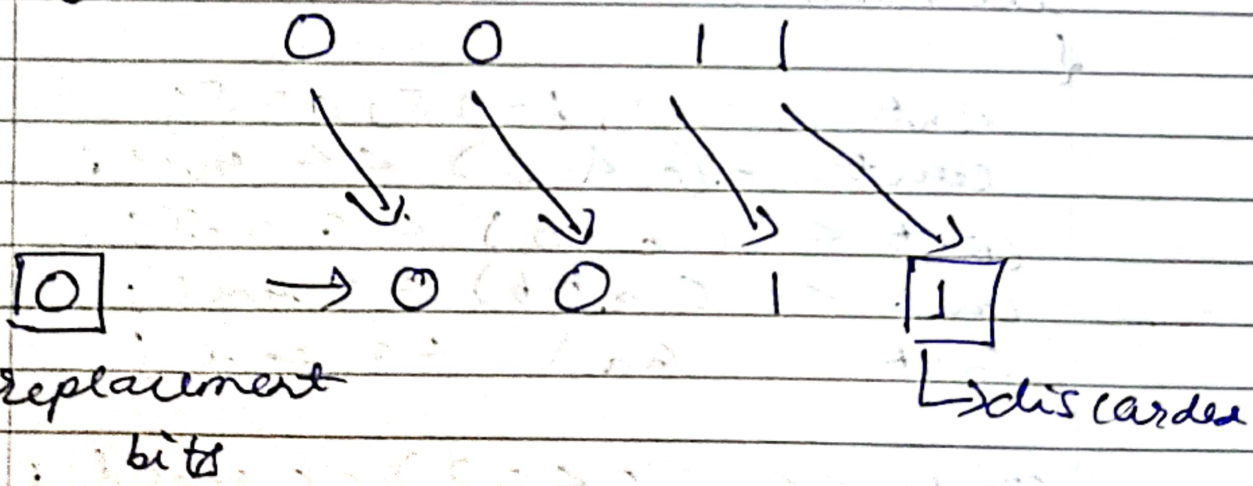
a	~a
0	1
1	0

* Eg) 12 = 0 1 1 0 0
~12 = ~ 1 0 0 1 1 → 19 in decimal.

* Bitwise Right shift operator :-

↳ It shifts all the bits towards the right by a certain no. of specified bits.

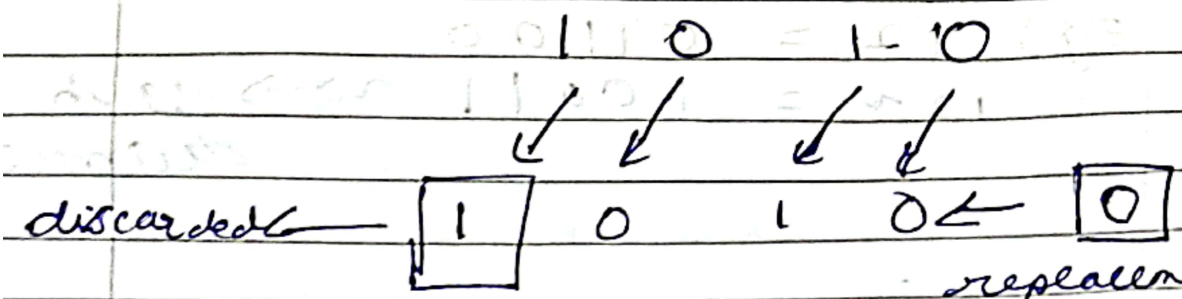
Eg)



Page No
Date

* Bitwise left shift operators:

↳ It shifts all the bits towards the left by a certain no of specified bits



Program 10 → bitwise operators

```
#include <iostream>
using namespace std;
int main()
{
    int a=12, b=25, c=5;
    cout<<(a & b) <<endl;
    cout<<(a | b) <<endl;
    cout<<(a ^ b) <<endl;
    cout<<(~a) <<endl;

cout<<(c<<2)<<endl;
cout<<(c>>1)<<endl;

return 0;
}
```