- **First stage (Letters)**

A to Z

a to z

0 to 9

({;"/\?<>+=*&%!

- **Second stage (Identifier)**

1.Constant

2.Variable

3.Keywords

- **Third stage (Instruction)**

1.Data Type Declaration Instruction

2.Input/Output Instruction

3.Arithmetic Instruction

4.Control Instruction

- **Fourth stage**

Program

#Identifire

1.Constant

-Any information is constant

-Data=Information=Constant

- **Types of Constants**

-There are two types of constants are existing.

1.Primary constants

Integer

Real

Character

2.Secondary constants

Arry

String

Pointer

Union

Structure

Enumerator

- **Primary constant**

1.Integar: ex. 1,2,55, 88, -33, -4737,0

2.Real: ex. 21.4,32.46, -0.4728,2.0

3.Character: Any symbol in single ' '

Examples: 'a','z','3','%'

Not an example: '-2','Siddharth' (because there are more than one symbol)

- **Secondary constant**

-Those consents which are made using Primary constants.

……

2.Variable

- Variables are the name of memory locations where we store data.

- **Rules for naming the Variables**

1.Variable name is any combination of alphabet, digit and underscore.

2.A valid variable name cannot start with digit.

……

3.Keywords

-Keywords are predefined words or reserved words.

-There are 32 Keywords in c language.

-We cannot use these words as variable name.

auto	double	goto	signed	unsigned
break	default	if	sizeof	void
case	enum	int	static	volatile
char	else	long	struct	while
continue	extern	register	switch	
const	for	return	typedef	
do	float	short	union	

#Instructions

-Program statements are called instructions.

-Instruction is commands.

- **Types of Instructions**

1.Data Type Declaration Instruction

2.Input Output Instruction

3.Arithmetic Instruction

4.Control Instruction

- **Data Type**

-Some Keyword showing bellow are called Data Type.

1.int

2.char

3.float

4.double

5.void

- **Primitive Data Type**

-Those words which are "Keywords" and "Data Type" both, are called "Primitive Data Type".

-Chack out the Table showing below (Underlined keywords are primitive data type).

auto	double	goto	signed	unsigned
break	default	if	sizeof	void
case	enum	int	static	volatile
char	else	long	struct	while
continue	extern	register	switch	
const	for	return	typedef	
do	float	short	union	

1.Data Type Declaration Statement/Instruction

-Those statements which starts from Primitive Data Type are called "Data Type Declaration Statement".

-Statement must has only one Data Type which must be at first position.

Examples:

```
int a,b=5;
float k;
char ch,m;
double d1;
```

-These statements are starting from primitive data type, so this are "Data Type Declaration Statement" or "Data Type Declaration Instruction".

.....

2.Input Output Instruction

Standard Input/Output Device

-Keyboard is standard input device.

-Monitor is standard output device.

- **printf();**

-printf() is not "Keyword".

-printf() is predefined "function".

-It is use for print any text or print value of expression or value of variable.

Example:

```
main()
{
printf("Siddharth Patel"); // If you are not using turbo C
}
```

Or

```
main()
{
clrscr();
printf("I am going to be scientist in future");
getch();
}
```

Extra Information:

-When you will run the code, you can see the text (text which is between " ") as output in console.

-clrscr() and getch() both are also predefined functions.

-clrscr() use for clear the script, which was run before.

-getch() use for hold the screen until we give it any character. Basically getch() function is made to get the character.

……

```
main()
{
printf(Siddharth\nPatel");
}
```

-Here \n is Escape Sequence.

- **Escape Sequences**

\n

\t

\b

\\

\"

\r

```
main()
{
int a=4, b=5;
printf("Sum of %d and %d is %d",a,b,a+b);  //If we want to print Sum of 4 and 5 is 9
}
```

-Here %d is Format Specifier.

- **Format Specifier**

%d	int
%f	float
%c	char
%lf	double

……

- **scanf();**

-scanf() is not a "Keyword".

-scanf() is a predefined function.

-It is use for take data from keyboard and store it into variable.

scanf("Format Specifier", Variable Address);

Example: Make a software in which if you enter a number then you will get square of that number as output.

```
main()
{
int X;
scanf("%d",&X);  //&X means address of variable X
printf("Square of %d is %d",X,X*X);
}
```

Example: Make a software in which if you enter two numbers then you will get multiplication of that numbers as output.

```
main()
{
int X,Y;
printf("Enter Two Numbers for Multiplication\n");
scanf("%d%d", &X, &Y);
printf("Multiplication of %d and %d Is %d",X,Y,X*Y);
}
```

- **How to use gotoxy(); function in code::blocks ?**

-Like in turbo C++ you used to use this function gotoxy(X,Y);.

-But this function will not work in Modern software.

```
SetConsoleCursorPosition(HANDLE,COORD);
```

-This function use as gotoxy(); function in Modern software.

Example: How to use SetConsoleCursorPosition(HANDLE,COORD); function.

```
#include<windows.h>
main()
{
COORD c;
c.X=20;
c.Y=8;
SetConsoleCursorPosition(GetStdHandle(STD_OUTPUT_HANDLE),c);
printf("Hello");
getch();
}
```

```
#include<windows.h>
void gotoxy(int x,int y)
{
COORD c;
c.X=x;
c.Y=y;
SetConsoleCursorPosition(GetStdHandle(STD_OUTPUT_HANDLE),c);
}
/*
-Here I define new function named gotoxy(); with same application as SetConsoleCursorPosition(); function.
*/
main()
{
gotoxy(20,8);
printf("Hello");
getch();
}
```

……

```
#include<windows.h>
void gotoxy(int x,int y) {
COORD c;
c.X=x;
c.Y=y;
SetConsoleCursorPosition(GetStdHandle(STD_OUTPUT_HANDLE),c);
}
main() {
for(int j=1; j<=10; j++)
{
```

```
gotoxy(20,j);
printf("Hello");
}
}
```

Output is 10 times Hellow in a Colum

……

- **Operator**

3+4

-Here 3 and 4 are Operands/Data.

- "+" is Operator.

3.Arithmetic Instruction

-An instruction which is used to manipulate data using operators, is known as Arithmetic instruction.

- **Operator Group**

1.Unary Operators	+, -, ++, --, sizeof()
2.Arithmetic Operators	*, /, %, +, -
3.Bitwise Operators	&, |, ^, ~, >>, <<
4.Relational Operators	<, >, <=, >=, ==, !=
5.Logical Operators	!, &&, ||
6.Conditional Operator	? :
7.Assignment Operator	+= ,-= , *=, /=, %=, &=, |=, ^=

……

1.Unary Operators

-Those operators which needs only one operende to do its work is called unary operators.

+, - ,++, --, sizeof()

Extra:

2.Binary Operators

-These operators which needs two operands to do its work is called Binary operators.

3.Ternary Operators

-These operators which needs three operands to do its work is called Ternary operators.

……

(1) '+' Positive Operator

-Denotes negativity of the number.

Example: +2, +67, +8.483

……

(2) '-' Negative Operator

-Denotes negativity of the number.

Example: -6, -49, -9.73

……

(3) '++' Increment Operator

Example:

```
main()
{
int x=3
x++; //x=x+1
printf("%d",x);
}
```

Output is 4

Or

```
main()
{
int x=3
++x; //x=x+1
```

```
printf("%d",x);
}
```

Output is 4

x++ : post increment

++x : pre increment

-Bothe has same work function which is increment by one (x=x+1), But differs in priority.

-Post increment has least priorit (bellow Assignment Operator)

……

(4) '--' Decrement Operator

Example:

```
main()
{
int x=3
x--; //x=x-1
printf("%d",x);
}
```

Output is 2

Or

```
main()
{
int x=3
--x; //x=x-1
printf("%d",x);
```

```
}
```

Output is 2

……

Advance Example:

```
int x=3,y;
main()
{
y=x++;
printf("%d %d",x,y);
}
```

Output is '4 3'

```
int x=3,y;
main()
{
y=++x;
printf("%d %d",x,y);
}
```

Output is '4 4'

-In both of the case output is different because of operator priority.

-In "y=x++" 'x++' is post operator, so it has least priority. And '=' (assigned operator) has more priority.

-So fist 'y' will assign 'x' (y=x), Then 'x' will be increased by one (x++).

-in "y=++x" '++x' is pre operator, so it will have most priority. First, x will be increased by one (++x) then y will assign value of '++x' (y=++x).

(5) 'sizeof()' Operator

-sizeof(data type)

-sizeof(variable)

-sizeof(constant)

-sizeof() is an Operator which gives size of Data type, Variable and Constant in bites.

-By default, size of Data type is shown below (In 32 bits software).

int	4 bites
char	1 bite
float	4 bites
double	8 bites

Example:

```
int x1,y1,z1,w1,x;
char y;
float z;
double w;
main()
{
   x1=sizeof(x);
   y1=sizeof(y);
   z1=sizeof(z);
   w1=sizeof(w);
   printf("%d %d %d %d",x1,y1,z1,w1);
}
```

Output is 4 1 4 8

2.Arithmetic Operators

-There are 5 Operators in Arithmetic Operators.

* / % Greater priority

+ - Lessor priority

- **Associativity Rule L to R:**

- *, / and % has same priority. But if they are in same instruction then we will start solving from left to right.

……

(1) Multiplication Operator '*'

-Multiplication Operator multiplies the Operands.

Example:

```
main()
{
int x=3,y=4;
printf("%d",x*y);
}
```

Output is 12

……

(2) Division Operator '/'

-Division Operator divides the Operands.

Example:

```
main()
{
int x=10,y=5;
printf("%d",x/y);
}
```

Output is 2

……

(3) Plus Operator '+'

-Plus Operator plus the Operands.

Examples:

```
main()
{
int x=10,y=5,X;
X=x+y;
printf("%d",X);
}
```

Output is 15

……

(4) Minus Operator '-'

-Minus Operator Minus the Operands.

Examples:

```
main()
{
int x=10,y=5,X;
X=x-y;
printf("%d",X);
}
```

Output is 5

(5) Modules Oprator '%'

-Modules Operator gives reminder when we divide those numbers.

Example:17%5

Answer will be reminder of 17/5 which is '2'

```
int x=17,y=5;
main()
{
   printf("%d",x%y);
}
```

Output is 2

-If left operand is smaller than right operand, then answer will be left operand (smaller operand).

Example:5%17

```
int x=5,y=17;
main()
{
   printf("%d",x%y);
}
```

Output is 5

-Output will only be negative when left operator is negative.

Example:

```
 5%2    1
-5%2   -1
 5%-2   1
-5%-2  -1
```

- **Application of % Operator**

-If we want last digit/digits as output then.

Do 123456789%10 for last one digit

Do 123456789%100 for last two digits

Do 123456789%1000 for last three digits

Do 123456789%10000 for last four digits

Example:

```
int x=123456789,y=10;
main()
{
   printf("%d",x%y);
}
```

Output is 9

……

3.Bitwise Operators

-There are 6 Operators in Bitwise Operators.

Bitwise AND	&
Bitwise OR	|
Bitwise XOR	^
Bitwise NOT	~
Right Shift	>>
Left Shift	<<

-Bitwise Oprator only works on Binary numbers

(1) Bitwise AND Operator '&'

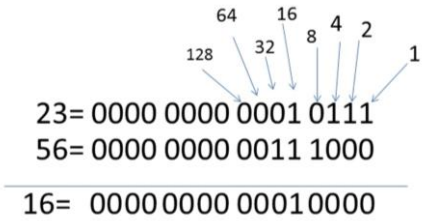
A	B	A&B
0	0	0
0	1	0
1	0	0
1	1	1

Example:

```
main()
{
int x=23,y=56;
printf("%d",x&y);
}
```

Output is 16

- **How Bitwise '&' Operator works?**



……

(2) Bitwise OR Operator '|'

A	B	A|B
0	0	1
0	1	1
1	0	1
1	1	0

Example:

```
main()
{
int x=23,y=56;
printf("%d",x|y);
}
```

Output is 63

- **How Bitwise '|' Operator works?**

```
23= 0000 0000 0001 0111
56= 0000 0000 0011 1000
-----------------------
63=  0000 0000 0011 1111
```

……

(3) Bitwise NOT Operator '~'

-Sign of Bitwise NOT Operator is '~'.

-Bitwise NOT Operator is also Unary Operator.

Example:1

```
main()
{   int x;
    x=~5;
    printf("%d",x);
}
```

Output is -6

- **How Bitwise '~' Operator works?**

-First writ number 5 in binary (This example is in reference of 32 bits software

like code::blocks)

5 = 00000000 00000000 00000000 000101 let b1

~5 = 11111111 11111111 11111111 111010 let b2

-NOT Operator converts 1 into 0 and 0 into 1

-Now we just have to identify the number b2

……

- **Some Properties by Which we can calculate value of b2**

1.If Most Significant Bit (MSB) is 1 then number is negative and MSB is 0 then number is positive. (MSB is first bit)

-So b2 is negative number.

2.If we take 2's Compliment of any positive number then it will be converted into negative of that number and vice versa is true.

e.g.

b1 = 20

If 2's Compliment of b1 is b2 then

b2 = -20

And vice versa is true (If we take 2's Compliment of b2

then new number will be b1.)

- **What is 2's Complement?**

-We will get 2's Complement by adding 1 into 1's complement.

Let b2 is -x

Then 2's Complement of b2 will be x.

1's Complement of b2 is 00000000 00000000 00000000 000101

2's Complement of b2 is...

```
   00000000 00000000 00000000 000101
                                    +1
  ___________________________________
   00000000 00000000 00000000 000110 = 6 in Decimal
```

So, 2's Complement of b2 is 6, and vice versa is true so b2 is -6

So ~5 is -6

.....

Example:2

```
main()
{   int x;
   x=~12;
   printf("%d",x);
}
```

Output is -13

12 = 00000000 00000000 00000000 0001100 let b1

~12 = 11111111 11111111 11111111 1110011 let b2

Let b2 is -x

Then 2's Complement of b2 will be x.

1's Complement of b2 00000000 00000000 00000000 0001100

2's Complement of b2 is...

00000000 00000000 00000000 001100

+1

00000000 00000000 00000000 001101 = 13 in Decimal

So, 2's Complement of b2 is 13, and vice versa is true so b2 is -13

So ~12 is -13

.....

(4) XOR Operator '^'

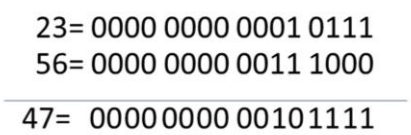
A	B	A^B
0	0	0
0	1	1
1	0	1
1	1	0

Example:

```
main()
{
int x=23,y=56;
printf("%d",x&y);
}
```

Output is 47

- **How Bitwise '^' Operator works?**



.....

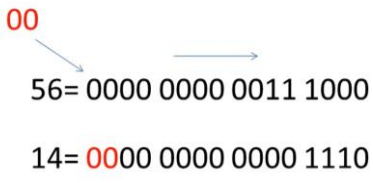
(5) Right Shift Operator '>>'

Example:

```
main()
{
int x=56,y=2;
printf("%d",x>>y);
}
```

Output is 14

- **How Bitwise '>>' Operator works?**



.....

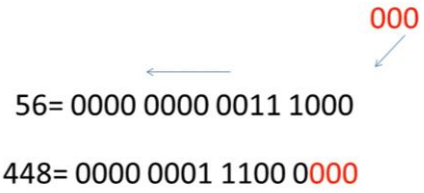
(6) Left Shift Operator '<<'

Example:

```
main()
{
int x=56,y=3;
printf("%d",x<<y);
}
```

Output is 448

- **How Bitwise '<<' Operator works?**



.....

4.Relational Operators

-There are 6 Operators in Relational Operators.

Less then	<
Grater then	>
Less than equal to	<=
Grater then equal to	>=
Equal to	==
Not equal to	!=

< > <= >= Greater priority

== != Lessor priority

-Relational Operator always yields result either 0 or 1.

-Every Non-zero Value is True and Zero is False.

-True is 1 and False is 0.

Example:

```
main()
{
   int x;
   x=3<4;
   printf("%d",x);
}
```

Output is 0 (False)

……

```
main()
{
   int x;
   x=3!=4;
   printf("%d",x);
}
```

Output is 1 (True)

……

```
main()
{
   int x;
   x=5>4>3;
   printf("%d",x);
}
```

Output is 0 (False)

-Because of Associativity Rule (L to R) first '5>4' calculation will take place.Because 5>4

Statement is True so output will be 1.

-Now Instruction will be 'x=1>3'. Which is False statement.

……

5.Logical Operators

-There are 3 Operators in Logical Operators.

NOT ! Greater Priority

AND &&

OR || Lesser Priority

(1) NOT Operator '!'

-NOT Operator is also Unary Operator.

-Priority level is same as of Unary Operators.

-It inverts the Truth value of statement.

!T = F

!F = T

Example

```
main()
{
   int x;
   x=!5<4;
   printf("%d",x);
}
```

Output is 1 (True)

(2) AND Operator '&&'

-AND Operator is also Binary Operator.

-It takes two Statements as Operands.

Statement1	Statement2	Statement1&&Statement2
F	-	F
T	F	F
T	T	T

Example:

```
main()
{
   int x;
   x=5>4&&4>5;
   printf("%d",x);
}
```

Output is 0 (False)

-Because Statement1 is F and Statement2 is T.

……

(3) OR Operator '||'

-OR Operator is also Binary Operator.

-It takes two Statements as Operands.

Statement1	Statement2	Statement1&&Statement2
F	F	F
F	F	T
T	-	T

Example:

```
main()
{
```

```
int x;
   x=5>4||4>5;
   printf("%d",x);
}
```

Output is 1 (True)

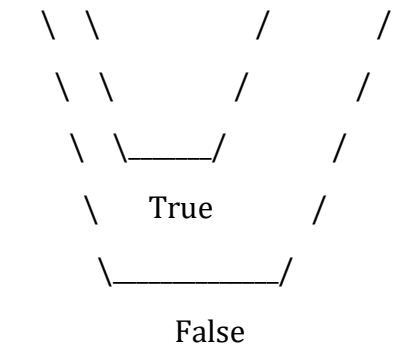
-Because Statement1 is T.

……

6.Conditional Operator '? :'

-Conditional Operator is only one Operator which is Ternary Operator.

```
expression1?expression2:expression3;
```



-Conditional Operator is exactly like if() else().

```
 if(expression1);
 {
  expression2
 }
 else
 {
  expression3
 }
```

Example: If we want to print bigger value from a and b.

```
int a=5,b=10,x;
main()
{
  x=a>b?a:b;
  printf("%d",x);

}
```

Output is 10

Or

```
int a=5,b=10,x;
main()
{
if(a>b)
{
   x=a;
   printf("%d",x);
}
else
{
   x=b;
   printf("%d",x);
}
}
```

Output is 10

Example: Make a software in which if you insert two numbers then it will show Grater number as output.

```
main()
{   int a,b,x;
   printf("Enter Two Numbers\n");
   scanf("%d %d",&a,&b);
   a>b?printf("Bigger Number is %d",a):printf("Grater Number is %d",b);
}
```

Or

```
main()
{   int a,b,x;
   printf("Enter Two Numbers\n");
   scanf("%d %d",&a,&b);
   printf("Grater Number is %d",a>b?a:b);
}
```

……

7.Compound Assignment Operator

-There are 8 Operators in Compound Assignment Operator.

+=, -=, *=, /=, %=, &=, |=, ^=

x+=a x=x+a

x-=a x=x-a

x/=a x=x/a

x*=a x=x*a

x%=a x=x%a

x&=a x=x&a

x|=a x=x|a

x^=a x=x^a

-But there are few differences between statements like 'x+=a' and 'x=x+a' . ('x+=a' is taken as just example, it is true for all Operators)

1.Better performance if you use 'x+=a' instead of using 'x=x+a'

2.Priority dispends on question, as example

```
main()
{   int x=5;
    x*=3+4;
    printf("%d",x);
}
```

Output is 35

```
main()
{   int x=5;
    x*=3+4;
    printf("%d",x);
}
```

Output is 19