Unit - 2

Operators :— A operator is an symbol that which mathematical and logical computation are to perform In 'C' language operators can be classified as.

(1) Arithmatic operators (+ , - , * , / , %)
(2) Relational operators (< , <= , > , >= , !=)
(3) Logical operators (&& , || , !)
(4) Bitwise operators (~ , << , >> , & , ^ , |)
(5) Assignment operators (=)
(6) Increament, decreament operators (++ , --)
(7) Conditional operator (? :)
(8) other operators (comma(,) sizeof() etc.

— Arithmatic operators —

Operator	Name	Example
+	Addition	5+2=7
-	Subtraction	5-2=3
*	multiplication	5×2=10
/	division	5/2 = 2
%	modulus to find out remainder.	5%2= 1

Note :— Arithmatic operation between.

int - int → int
float - float → float
int - float → float

Relational operators :— Relational operator is used to compare two operators, the result is either true (1) or false (0)

Operator	Name	Example
<	less than	5<5.5 gives 1
<=	less than or equal to	5.5<=5.5 gives 1
>	Greater than	5.5>6 gives 0
>=	Greater than or equal to	5>=5.5 gives 0
!=	not equal to	5!=5 gives 0
==	equal to	5== gives 1

Logical operators :— logical operator operate upon two logical expression to build a complex expression. The result is either true (1) or false (0).

Operator	Name	Example
&&	logical AND	(5>4&&5<7) Gives 1
||	logical OR	(5>1||5<6) Gives 1
!	logical NOT	!5 gives 0

Logical AND —

Condition 1	Condition 2	Condition 2 && condition 2
False	False	False
False	True	False
True	false	false
True	True	True

- Logical OR —

Condition 1	Condition 2	Condition 1 // condition 2
false	false	false
false	True	True
True	false	True
True	True	True.

- Logical NOT — !

A	A!
True	false
false	True

- Bitwise operator :— Bitwise operator operate on bit and binary (0,1)

Operator	Name
~	Bitwise not
<<	Bitwise left shifting
>>	Bitwise Right shifting
&	Bitwise AND
^	Bitwise XOR
|	Bitwise OR

(i) Bitwise NOT (1's complement) - (convert 1 into 0 & 0 into 1)

e.g- Suppose a = 11

Convert into Binary

a = 1011

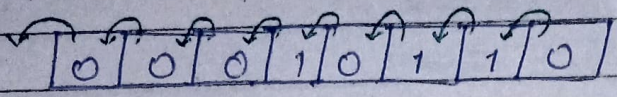
~a = 0100

(2) Bitwise shifting -

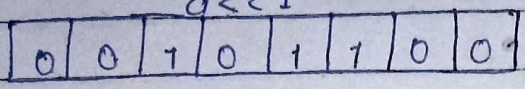
Syntax:- <operand><shift operator><no. of bits shifted>

(i) Bitwise left shifting (<<)

eg. a = 22



a<<1

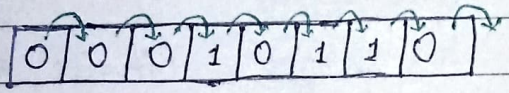


a = 44

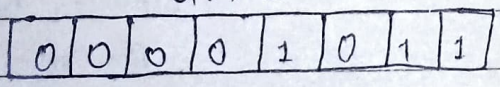
In bitwise left shifting value is multipeied by 2^n (n → no. of Bits shift)

(ii) Bitwise Right shifting (>>)

E.g. a = 22



a>>1



a = 11

In right shifting value is devided by 2^n where n → no. of Bits shifted.

③ Bitwise AND (&)

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

Bitwise OR (|)

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

④ Bitwise XOR (^)

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

⑤ Assignment operator (=)

This operator is used to Assign a value to a variable.

Syntax: [Variable = value of Expression;]

E.g. (i) $x = 5$

(ii) $y = x + 2$

Operator —

=	Assignment
+=	Assign Sum
-=	Assign difference
*=	Assign Product
/=	Assign Quotient
%=	Assign Remainder
~=	Assign bitwise not
<<=	Assign bitwise left shifting
>>=	Assign bitwise Right shifting
&=	Assign bitwise AND
^=	Assign bitwise XOR
|=	Assign bitwise OR

⑥ Increament (++) and decreament (--) → operator Both are Unary operator which works on one operand.

++ → It increase the value of the operand by 1.
-- → It decrease the value of the operand by 1.

E.g. x=5

x++ → It is a post Increament.
++x → It is a pre Increament.

In both the cases the value of x will Increase by 1.
But when we use in an operation then both works differently.

E.g. y = ++x

Suppose value of x is 5 In preincreament first the value of x will Increase by 1 then It will assign to y.

y=6
x=6

E.g. y = x++

In post Increament, first the value of x will assign to y without Increment, after assigning the value of x will Increase by 1.

y=5
x=6

⑦ Conditional Operator (? :)

Syntax :- <condition> ? <True value> : <false value>

E.g c = a>b ? a : b;

Suppose a=5, b=4

c=a
c=5.

(8) other operators:-

(i) Sizeof() → This operator provide the size of the given operand or data Type in byte (memory size).

Syntax: Sizeof (operand or data Type)

Eg. int x

(i) Sizeof (x); // provide 2 byte.
(ii) Sizeof (int); // provide 2 byte.
(iii) Sizeof (float); // provide 4 byte.

(ii) Comma (,) operator →

If we want to write multiple statement in a single row, we use comma operator.

E.g: temp = x;
x = y;
y = temp;

temp = x , x = y , y = temp ;

— Precedence and Associativity of operators —

(i) Precedence → means priority. when more than one operator used in an expression, then it is the relatives priorities of operators with respect to each other that will determine the order of operator, will evaluate in expression. high priority operators will ~~execute~~ evaluate first then low priority operator will evaluate.

for example :- x = 6 + 3 * 5;

In this * has high precedence / priority than +, so * will evaluate first then +

x = 6 + 15
x = 21.

(ii) Associativity → when operators have equal precedence (priority) then we use Associativity.

Associativity defines the direction from left to right or right to left.

E.g. 6*3/2%3

In this multiplication division modulus have same precedence, so we will use Associativity from left to right.

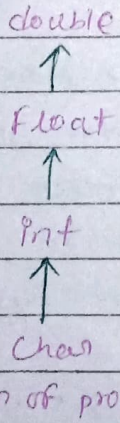
$x = 18/2\%3$

$x = 9\%3$

$x = 0$

Data Type conversion & Type casting → data Type conversion and Type casting means changing a variable of one data Type into another data Type.

① Data Type conversion or Implicit Type conversion → when C-compiler automatically convert one data Type into another data Type to make them uniform data Type then it is called implicit Type conversion.



Order of promotion

② Type casting or Explicit Type conversion → forcefully conversion from one data Type into another data Type is known as explicit Type conversion or Type casting it is done by the programmer.

Syntax: (data Type) variable conversion;

E.g. int 8.9 gives 8
float 5 gives 5.0

Nested if → when one condition depend on another condition and so on we use nested if in this the first condition become true then only the second condition will tested.

One of the syntax can be :-

```
if (condition 1)
{
  If (condition 2)
Statement 1;
else
   Statement 2;
}
  { if (condition 3)
   Statement 3;
   else
      Statement 4;
   }
```

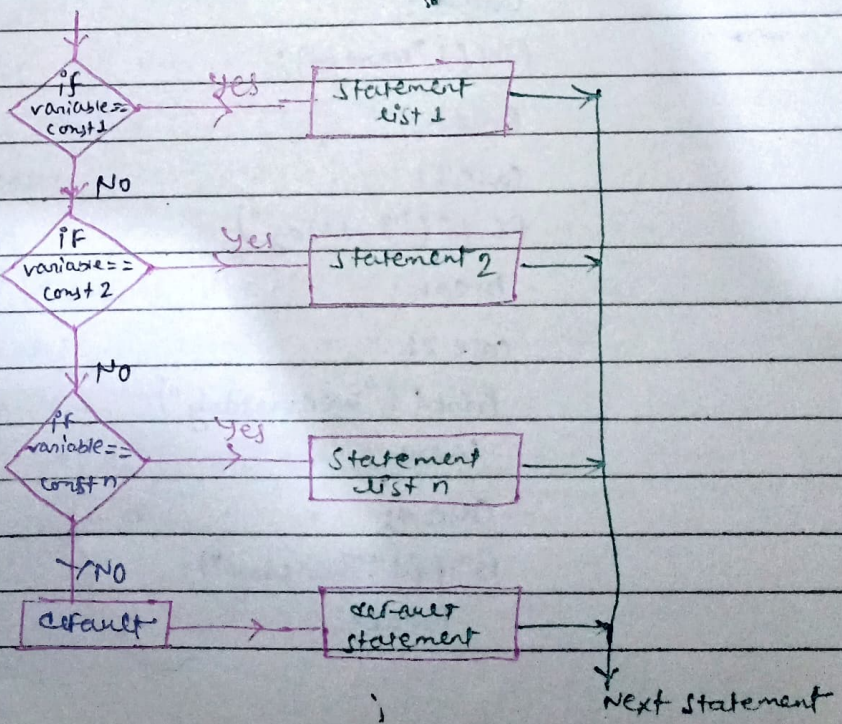
Q. WAP to find greatest among three numbers using nested if.

```
#include <stdio.h>
#include <conio.h>
void main()
{ int x,y,z;
printf("Enter three numbers");
Scanf("%d %d %d", &x, &y, &z);
if (x>y)
{
  if (x>z)
Printf("Greatest"=%d", x);
else
  Printf("Greatest =%d", z);
  }
    else
      { if x(y>z)
Printf("Greatest = %d", y);
else
    Printf("Greatest =%d", z);
  }
    getch();
  }
```

Switch case → when we want to solve multiple option Type problem and we need to choose one option at a time then we use Switch case.

Syntax:

```
Switch (variable expression)
{
case constant 1 :
Statement list 1 ;
break ;
case constant 2 :
Statement list 2;
 break ;
- - - - - - - - -
case constant n :
Statement list n ;
default :
default statement ;
}
```



Working —

A switch statement test the value of variable or expression and compare it with different case constant once a case is matched the statement associated with that case executed. and rest of case will not be tested. if non of the case is matched then default statement will execute (default is optional).

Q. Write a program to input a number and print the corresponding week.day.

Sol.

```
#Include<stdio.h>
# Include<conio.h>
  void main()
{  int choice;
Printf("Enter your choice");
Scanf("%od", &choice);
 switch (choice)
  {
    case 1:
    Printf("Monday");
    Break;
    case2:
    Printf("Tuesday")
     break;
     case 3:
     Printf("wednesday")
      break;
      Case 4:
      Printf("Thursday");
       Break;
       case 5:
       Printf("Friday");
        break;
        Case 6:
        Printf("Saturday");
         break;
         case 7:
         Printf("Sunday");
          Break;
          default:
           Printf(Invalid choice);
            getch();
}
```

Que Write a Program to print the category of the student using switch case

Code	Category
1	General
2	Backward
3	Schedule
4	Schedul Tribe

```
# Include <stdio.h>
# include <conio.h>
  void main ( )
  { int code;
    Printf ("Enter code");
    Scanf ("%d", & code);
     Switch (code)
    { case 1:
     Printf (" General");
      Breake;
      Case 2:
     Printf (" Backward");
      Break;
     Case 3:
     Printf ("Schedule");
      Break;
     Case 4:
     Printf ("Schedule Tribe");
      Break;
      default:
        Printf ("Invalid code");
      } getch ( );
        }
```

Que. WAP to print the category of the student using switch case.

Code	Category
g or G	General
b or B	Backward
s or S	Schedule
t or T	Schedule Tribe

```
#include<stdio.h>
#include<conio.h>
void main()
{ char code;
  Printf("Enter code");
  Scanf("%c",&code);
   Switch (code).
{ case 'g':
   Case 'G':
  Printf("general");
    Break;
  Case'b':
  case'B':
  Printf("Backward");
   Break;
  Case 's':
  Case 'S':
  Printf("Schedule");
    Break;
  Case 't':
  Case'T':
  Printf("Schedule Tribe");
    break;
  default:
    Printf("Invalid code");
}
  getch();
}
```

Que. write a program to find the value of y for the given value of N using switch case.

N = 1	$y = ax \% n$
N = 2	$y = ax^2 + b^2$
N = 3	$y = -bx$
N = 4	$y = a + b/x$

```
#include <stdio.h>
#include <conio.h>
void main()
{ int N, a, x, y, b;
  Printf("Enter a, x, b");
  Scanf("%d %d %d", &a, &x, &b);
  Printf("Enter choice");
  Scanf("%d", &N);
   Switch (N)
   { case 1:
     y = a * x % b;
    Printf("value of y = %d", y);
     break;
    Case 2:
    y = a * x * x + b * b;
    Printf("value of y = %d", y);
     break;
    Case 3:
    y = -b * x
    Printf("value of y = %d", y);
     breake;
    Case 4:
    y = a + b/x;
    Printf("value of y = %d", y);
     break;
    default:
    Printf("Invalid Entry");
   }
    getch();
}
```

Que- Write a program to simulate a calculator using switch case (+,-,*,/)

```
#include <stdio.h>
#include<conio.h>
Void main()
{  char choice;
   int a,b,r;
Printf("Enter your choice +,-,*,/");
 Scanf("%c",& choice);
Printf ("Enter two numbers");
 Scanf ("%d %d", & a, & b);
 Switch (choice)
 {
  case '+':
  r = a+b
  Printf ("After Addition = %d", r);
     break;
  case '-':
   if (a>b)
      r = a-b;
   else
      r = b-a;
  Printf ("After subtraction value = %d", r);
     break;
  case '*':
  r = a*b;
 Printf ("After multiplication values = %d", r);
     break;
  case '/'
  if (a>b)
  r = a/b;
  else
     r = b/a;
 Printf ("After division value = %d", r)
   break;
 default:
   printf ("Invalide code");
    } getch();
```

Switch Case —

WAP to print vowel.

#—

#—

```
void main()
{ char ch;
Printf("Enter a character");
Scanf("%c",&ch)
  Switch (ch)
    {
      case 'a':
      case 'e':
      case 'i':
      case 'o':
      case 'u':
printf("lower case vowel");
      break;
      getch();
      }.
```

```
      case 'A':
      case 'E':
      case 'I':
      case 'O':
      case 'U':
      default;
Printf("upper case vowel");
      Break;
      getch();
      }
```