C Programming

Trainer : Nisha Dingare
Email : nisha.dingare@sunbeaminfo.com

Unary operators :

- **Increment operator (++) :**
  - ++ is the increment operator that increments the value of the variable by 1.
  - It is a unary operator and its operand must be a variable. Any expression or constant is not allowed.
  - This operator has 2 types :
    - Pre-increment :
      - In this, the value of the variable is incremented first and then the modified value is used.
      - Syntax : ++num;
    - Post-increment :
      - In this, the value of the variable is first used and then its modified.
      - Syntax : num++;
- **Decrement operator (--) :**
  - -- is the decrement operator that decrements the value of the variable by 1.
  - Other properties are similar to the increment operator.
  - It has 2 types :
    - Pre-decrement :
      - In this, the value of the variable is decremented first and then the modified value is used.
      - Syntax : --num;
    - Post-decrement :
      - In this, the value of the variable is first used and then its modified.
      - Syntax : num--;

Bitwise operator AND (&) :

- Bitwise operators can be used on integral types. They operate on binary bits corresponding to that number.
- Bitwise AND & :

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

- Bitwise AND operator on the individual bits of the operand according to the above truth table
- Example : - 10 & 5
- 0000 1010 -> Binary of 10
- 0000 0101 -> Binary of 5
- ----------
- 0000 0000 -> O/P is 0

Bitwise operator OR (|) :

- Bitwise OR | :

x	y	x | y
1	1	1
1	0	1
0	1	1
0	0	0

- Bitwise OR operators on the individual bits of the operand according to the above truth table.
- Example : -10 | 5
- 0000 1010 -> Binary of 10
- 0000 0101 -> Binary of 5
- ----------
- 0000 1111 -> O/P is 15

Bitwise operator XOR (^) :

- Bitwise XOR ^ :

Input		Output
A	B	A xor B
0	0	0
0	1	1
1	0	1
1	1	0

- Bitwise XOR operators on the individual bits of the operand according to the truth table shown above
- Example : -10 | 5
- 0000 1010 -> Binary of 10
- 0000 0101 -> Binary of 5
- ----------
- 0000 1111 -> O/P is 15

Bitwise operator Not (~) :

- Bitwise NOT ~ :
  - Bitwise NOT operator results in one’s compliment of its operand.
  - Each bit is inverted i.e. 0 to 1 and 1 to 0.

NOT "~"	
INPUT	OUTPUT
0	1
1	0

Shift operators :

- **Left shift operator :**

  x << n : shifts the value of x left by n bits

  13 << 1

  13<<1 = 26

Formula : num << n = num * 2 raise to n

13 << 1 = 13 * 2^1

= 13 * 2 = 26

- **Right shift operator :**

  x >> n : shifts the value of x right by n bits

  13 >> 1

  13 >> 1 = 6

Formula : num >> n = num / 2 raise to n

13 >> 1 = 13 / 2 ^ 1

= 13 / 2 = 6

Points to remember :

- If precedence of two operators in an expression is same, their associativity is considered to decide their binding with operands.
- Data type conversions and ranges should be considered while doing arithmetic operations.
- sizeof() is compile time operator. Expressions within sizeof() are not executed at runtime.
- Relational and logical operators always result in 0 or 1.

- In logical AND, if first condition is false, second condition is not evaluated. Result is false.
- In logical OR, if first condition is true, second condition is not evaluated. Result is true.
- Increment/Decrement operators in arithmetic expressions are compiler dependent.

Control Statements :

- Decision or Selection
  - if-else
  - switch-case
- iteration (loop)
  - for
  - while
  - do-while
- Jump
  - break
  - continue
  - go to
  - return

If statement :

- Conditional statements in C are used to make a decision based on certain conditions. They are implemented using if.. else keywords and conditional operators.
- Syntax :

```
if( condition )
{
    block of statements;
}
```

If the condition evaluates to true, the block of statements is executed, else the block is skipped and the execution continues from the next statement after the block.

- Most of the problems require one set of action to be performed if particular condition is true and another set of action to be performed if condition is false. We can use if.. else block in that case.

```
If ( condition )
{
block of statements;
}
else
{
block of statements;
}
```

Nested if block :

- Nested if Statement :

  If the body of if statement contains another if statement then we say that if are nested and the statement is known as nested if statement

- If(condition)

```
{
    statement
    ---------------
    if statement
                ------------------
 }
```

- Nested if-else statement :

  In nested if-else statement the if body or else body of an if-else statement contains another if statement or if else statement

Decision control : Switch case :

- Switch statements is an alternate to if-else ladder.
- When there are multiple conditions it becomes difficult to use if- else. In such situations switch provide easy and organized way to select from multiple options.
- switch (expression)

```
{
   case const-expr1:
      statement(s);
      break;
   case const-expr2:
      statement(s);
      break;
.......
  default:
      statement(s);
      break;
}
```

Switch case :

1. Default clause is optional.
2. Case has be followed by integer or character constant or expression evaluating to integer.
3. It is necessary to use break statement as a last statement of each case.
4. Break is one of jump statement. It helps to move execution control forcefully out of switch/loop.
5. if we do not include break statement as a last statement in switch execution, the control is given to all next cases even though they are not satisfied.
6. Duplicate case is not allowed
7. Sequence of cases do not matter.

typedef :

- typedef is a keyword in c programming.
- It Is used to give alias (another name) to already existing data types.
- It improves readability of the code.
- Helps to simplify complicated declarations.

- Syntax : typedef existing data_type alias;
  - typedef unsigned int unit;

- typedef unsigned int size_t; // predefined in c library.

enum :

- Enum or enumeration in c program is a user defined data type.
- It contains integer values and it is used to give meaningful names to the values.
- It makes the program easy to understand and maintain.
- Enum constants are replaced with integer values.
- Internally enum is an integer, so size of enum = size of integer.
- Enum constant values start from 0 by default if we do not provide any value.
- Programmer may choose to modify any field to any +ve or –ve number.
- Enum values can be duplicated.

Enum syntax :

- Syntax:
- enum<tag> {[<fields>] } ;
- To find size of enum, we use sizeof(enum_name)
- Variation 1 : enum created with nick name
  - e.g. typedef enum depts{TCT=10,ADMIN,PL=40} NickDEPT;
- Variation 2 : normal enum creation
  - e.g. Enum colors{RED=1,BLUE,GREEN=30};
- Variation 3 : Anonymous Enum
  - e.g: typedef enum{JAN=1,FEB,MAR,APR,MAY} MONTH;

Thank You