C Programming

Trainer : Nisha Dingare

Email : nisha.dingare@sunbeaminfo.com

Width Specifiers :

- int num = 12;
- printf(“%4d”,num); output : _ _ 12

- int num = 12;
- printf(“%-4d”,num); output : 1 2 _ _

- float fval= 12.48;
- printf(“%6.2f”,fval); output : _ 1 2 . 4 8

Expressions and Operators :

- An expression is made up of operands and operators.
  Example : int a = 2 + 3;
  Here a,2,3 are operands. '+' '=' are operators.
- Precedence of operators :
  - Each operator in C has a precedence associated with it. In a compound expression where there are multiple operators involved, then the expression is solved as per their precedence.
- Associativity :
  - In a compound expression when several operators are of same precedence, then operators are evaluated according to there associativity either left to right or right to left.
- **Classification of operators**
- Unary Operators : Unary Operator operates only on one operand.
  Example: expression : -3 here – is a unary minus operator.
  types of unary operator are : &, sizeof , !(logical negation),~(bitwise negation),++(increment),--(decrement).
- Binary Operators : Binary operator operates on 2 operands
  Example: expression : 2 - 3, - acts as a binary minus operator as it operates on two operands 2 and 3.
  types of binary operators are * ,/ ,<<(left shift), >>(Right shift), Logical And(&), Bitwise And(&)
- Ternary Operator : A ternary operator operates on three operands
  Example : Conditional operator (? :) is the only ternary operator in C

Operators :

- Arithmetic Operators (+ , -, * , / , %)
- Assignment & shorthand Operators (= , += , -= , *= , /= , %= , &= , |= , ^= , ~= , <<= , >>=)
- Relational Operators (< , <= , > , >= , !=)
- Logical Operators (&&, ||, !)
- Conditional Operator (? :)
- Bitwise Operators (& , | , ^ , ~ , << , >>)
- Unary Operators (+ , -, ++, --, * , & (address of))
- Special Operator (, comma operator , sizeof() , [ ])

Operator Precedence and Associativity :

OPERATOR	TYPE	ASSOCIAVITY
( ) [ ] . ->		left-to-right
++ -- + - ! ~ (*type*) * & sizeof	Unary Operator	right-to-left
* / %	Arithmetic Operator	left-to-right
+ -	Arithmetic Operator	left-to-right
<< >>	Shift Operator	left-to-right
< <= > >=	Relational Operator	left-to-right
== !=	Relational Operator	left-to-right
&	Bitwise AND Operator	left-to-right
^	Bitwise EX-OR Operator	left-to-right
|	Bitwise OR Operator	left-to-right
&&	Logical AND Operator	left-to-right
||	Logical OR Operator	left-to-right
? :	Ternary Conditional Operator	right-to-left
= += -= *= /= %= &= ^= |= <<= >>=	Assignment Operator	right-to-left
,	Comma	left-to-right

Short-hand operators :

- Short-hand operator change the value of the variable :
  - num += 2; num = num+2; Adds 2 to the actual value of num.
  - num =+ 2; num = 2; Assigns positive 2 to the variable num.
  - num -= 2; num = num – 2; Subtracts 2 from the actual value of num.
  - num =- 2 num = -2; Assigns negative 2 to the variable num.

Relational and Logical operators :

- **Comma Operator :**
  - Has Lowest precedence.
- **Relational operators :** These operators are used to compare two operands. The result is either 0 indicating false or 1 indicating true.
  - Types : <, >, <=, >=, ==, !=
- **Logical operators :** These operators are used to combine two conditions or to negate a condition.
  - The result is either 0 indicating false or 1 indicating true.
  - Types : &&(logical and), ||(logical or), !(logical not).
  - Logical AND and Logical OR operator guarantee left to right evaluation.
  - Logical NOT operator is used to reverse the logical state of its operand. If a condition is true, then Logical NOT operator will make it false.
  - Logical operators operate according to the below truth table.

P	Q	P && Q	P || Q	!P
T	T	T	T	F
T	F	F	T	F
F	T	F	T	T
F	F	F	F	T

Control statements : Decision control : if – else :

- 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;
}
```

Conditional operator / ternary operator :

- Conditional operator is also known as ternary operator as it takes three operands.
- It is a decision making operator hence it is similar to if – else statements.
- It follows right to left associativity rule.
- Syntax :

  condition ? Expression1(if true) : expression2 (if false) ;

  - If the condition evaluates to true, expression1 is executed.
  - If the condition evaluates to false, expression2 is executed.

Thank You