C Programming

Trainer: Ketan G Kore

Data Types, Variables & Constants

- Variable examples
  - int number = 10;
  - double basic_salary = 20000.0;
  - char letter = 'A';
  - int roll_number;
  - roll_number = 20;
  - double price = 200.0;
  - price = 300.0;
- Constant examples
  - -23, 1L, 34U, 3UL, 0x41, 0101,
  - 1.234f, 1.234567e+2, …
  - "SunBeam"
- Each variable is assigned some memory location.
- Size of data type of given variable or constant is found by sizeof() operator.

printf() and scanf()

- #include <stdio.h> -- function declaration

- printf()
  - Used to print values & string on terminal.
  - Various format specifiers %d, %c, %f, …
  - Formatting: %5d, %-7d, %08d, %8.2f, …

- scanf()
  - Used to input values from user.
  - Same format specifiers as of printf().
  - Do not use any char other than format specifiers in format string.
  - To skip a char from input use %*c.

Operators 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

Operators

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

- Types of operators
  - Unary Operators (+ , - , ++, --, &, *)
  - Binary Operators (+, -, *, +=, …)
  - Ternary Operators (? :)

Operators

- An expression in C is made up of operands , for eg a= 2 + 3 is a meaningful expression which involves 3 operands 2 , 3 , a and 2 operators i.e =,+ , Thus expression is sequence of Operators and operands
- Precedence of Operators:-Each Operator in C has a precedence associated with it , In a compound expression operator involved are of different precedence so operator with highest priority is evaluated first .
- Associativity : In compound expression when several operator are of same precedence operators are evaluated according to there associativity either left to right or right to left.

Classification of operators

- Unary Operators – Unary Operator operates on only one operand for example in the expression -3 – is a unary minus operator examples of unary operator are &,sizeof,!(logical negation),~(bitwise negation),++(increment),--(decrement) operator
- Binary Operators – Binary operator operates on 2 operands for example expression 2-3, - acts as a binary minus operator as it operates on 2 operands 2 and 3 for exampe *,/,<<(left shift),>>(Right shift),Logical And(&), Bitwise And(&)
- Ternary Operator – A ternary operator operates on three operands for example Condtional operator (?:) is the only ternary operator in C

Arithmetic operators

- Arithmetic operators work with all primitive data types i.e. int, float, char, double.
- Precedence of * & / is higher than + & -.

- % operator doesn't work with float and double type.
- % operator follows sign of numerator

- If two operands are of different types, the lower type is promoted temporarily for computation.

- char and short are promoted to int temporarily for computation.

- Char types are treated as integers (ASCII values) for calculation.

- If result exceed range of data type (overflow), then it rollback.

Short-hand operators

- Short-hand operators will change value in variable.
- +=, -=, …
  - num+=2;
  - num=+2;
  - num-=2;
  - num=-2;

- Pre-increment/decrement
  - x = ++a;
  - y = --b;

- Post-increment/decrement
  - x = a++;
  - y = b--;

Comma, Relational and logical operators

- Comma operator
  - evaluate to right-most value.
  - have lowest precedence.

- Relational and logical operators result in 0 or 1.
  - 0 – indicate false condition
  - 1 – indicate true condition
- Relational operators
  - <, >, <=, >=, ==, !=
- Logical operators
  - &&, ||, !

Logical operators

- Logical operators
  - &&, ||, !

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

  - Logical operators operate according to the truth table given above
  - Logical AND and Logical OR operator guarantee left to right evaluation
  - Logical NOT OperatorIt is used to reverse the logical state of its operand. If a condition is true, then Logical NOT operator will make it false.

Bit-wise operators

The C language provides six operators for bit manipulation they operate on the individual bits of the operands
. The Bitwise operators available in C are

- Bitwise AND &

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

Bitwise AND 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 0000 → O/P is 0

- 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 truth table shown above

Example : - 10 | 5

0000 1010 -> Binary of 10

0000 0101 -> Binary of 5

- - - - - - - - -

0000 1111 → O/P is 15

- 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 NOT ~

  Bitwise NOT operator results in one’s compliment of its operand

NOT "~"	
INPUT	OUTPUT
0	1
1	0

- Left shift << and Right shift >>
  - The bitshift operators take two arguments, and looks like:
    - x << n : shifts the value of x left by n bits
    - x >> n : shifts the value of x right by n bits

- Left shift operator : - num << n = num * 2 raise to n
  - **5 << 2 = 5 * 2 to the power 2**

    5 * 4 = 20
- Right Shift operator :- num >> n = num / 2 raise to n
  - **9 >> 1 = 9 / 2 to the power n**

    = 4

Thank you!