Introduction to Computing and Programming

Number System, Operators, Precedence and Associativity of Operators

Recap

- INPUT / OUTPUT IN C
- COMMENTS
- OPERATORS
- NUMBER SYSTEM IN C
- CONDITIONAL STATEMENTS

Contents

Number System: Represents a quantity through a set of Numbers or Data can be stored in the system

Decimal Notation

123

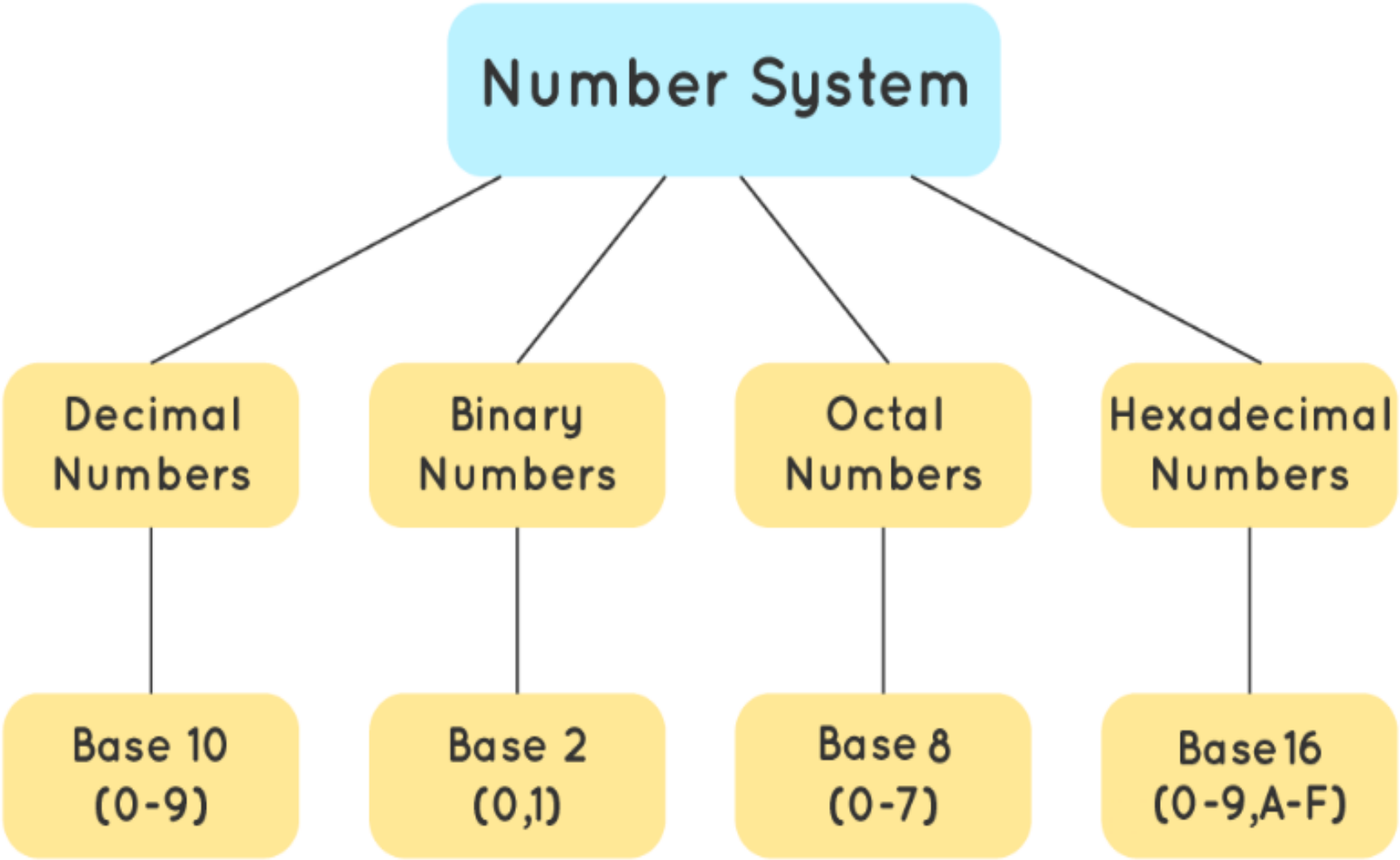
100	10	1	- Places
1	2	3	

$$123 = 1 \times 100 + 2 \times 10 + 3 \times 1$$

Decimal Number representation

- Examples are:
- 762

Types of Number System



Number System Cont..

From Decimal to Binary

- Examples are:
- $(15)_{10}$ to $(\ )_2$
- $(17.35)_{10}$ to $(\ )_2$

From Binary to Decimal

- Examples are:
- $(1111)_2$ to $(\ )_{10}$
- $(10001.1011)_2$ to $(\ )_{10}$

Number System Cont..: Try your own

From Decimal to Binary

- $(256)_{10}$ to $(\ ?)_2$
- $(198.29)_{10}$ to $(\ ?)_2$

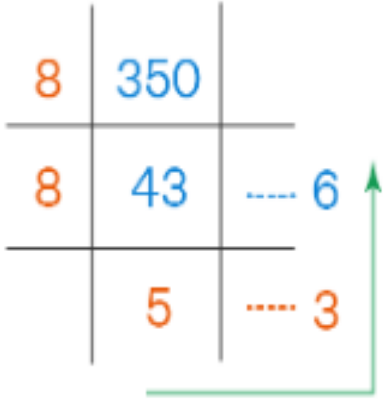
From Binary to Decimal

- $(100000000)_2$ to $(\ ?)_{10}$
- $(1011.011)_2$ to $(?\ )_{10}$

Number System Cont..

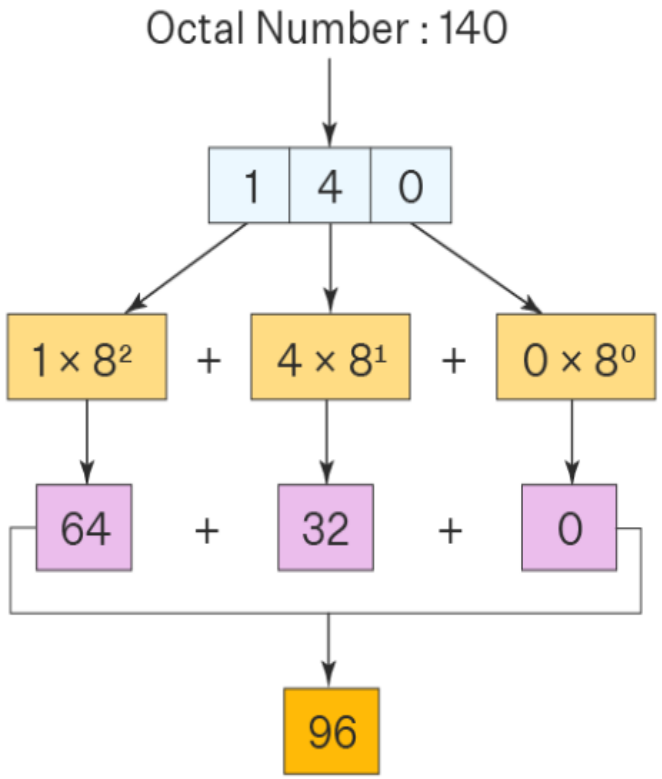
From Decimal to Octal

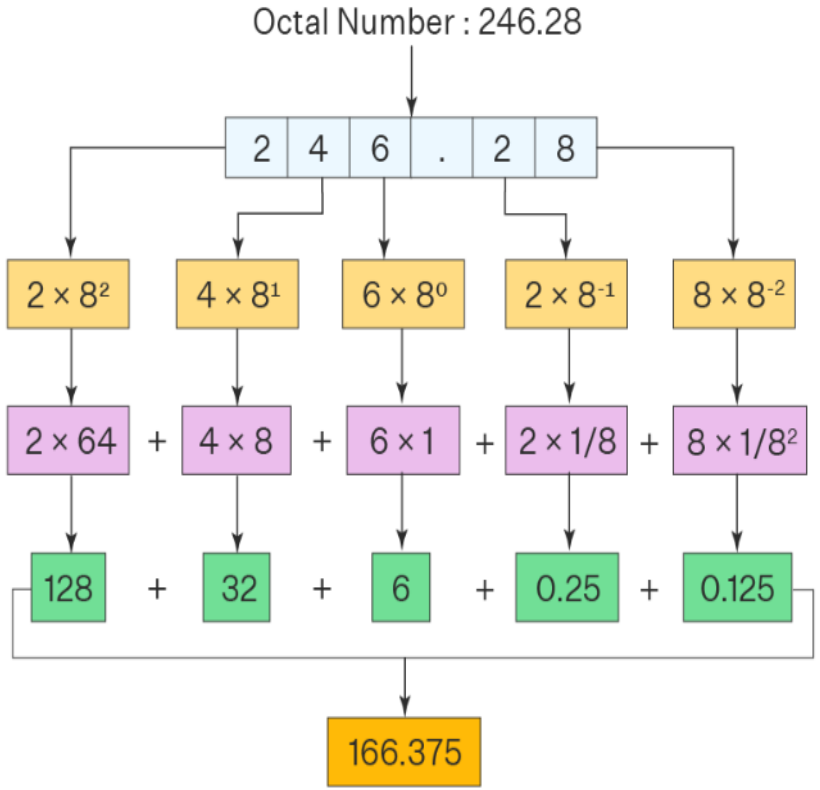
- $(350)_{10}$ to $(\ )_8$
- $(198.29)_{10}$ to $(?\ )_2$



From Octal to Decimal

- $(140)_8$ to $(\ )_{10}$
- $(246.28)_8$ to $(\ )_{10}$





Number System Cont..

From Decimal to Hexadecimal

- $(765)_{10}$ to $(\ )_{16}$
- $(765.245)_{10}$ to $(\ )_{16}$

Integer Part

16	765
16	47 - 13
16	2 - 15
	0 - 2

Fractional Part

0.245
× 16
3.920
× 16
14.720
× 16
11.520

$(2FD.3EB)_{16}$

From Hexadecimal to Decimal

- $(1A)_{16}$ to $(\ )_{10}$
- $(1BD.2F)_{16}$ to $(?\ )_{10}$

$(1\ A)_{16}$ → Hexadecimal

10

→ $10 \times 16^0 = 10$

→ $1 \times 16^1 = 16$

↓

$16 + 10 = 26$

$(1\ A)_{16} = (26)_{10}$

Number System Cont..

01001000 01001001

72 73

H I

Char to ASCII

```
#include <stdio.h>
int main()
{
char c = ‘a’;
int asciiValue = (int)c;
printf("%d\n", asciiValue);
return 0;
}
```

ASCII to Char

```
#include <stdio.h>
int main()
{
int c = 100;
char asciiValue = (char)c;
printf("%c\n", asciiValue);
return 0;
}
```

Number System Cont..

Representing Text

- The size of a file = number of bytes stored in the file

- 1 KB = 1024 bytes = 2^{10} bytes
- 1 MB = 1024 KB = 2^{20} bytes
- 1 GB = 1024 MB = 2^{30} bytes
- 1 TB = 1024 GB = 2^{40} bytes

Coding Question for Practice:

- **Write a C program to that takes input in integer but prints output in equivalent Octal & Hexadecimal.**

Hint: Use %o and %x specifier.

What are Operators?

Operators are symbols that tell the compiler to perform specific mathematical, logical, or relational operations.

They act on variables and values (operands).

Types of Operator

- Arithmetic Operators
- Relational Operators
- Logical Operators
- Bitwise Operators
- Assignment Operators
- Unary Operators
- Ternary (Conditional) Operator
- Miscellaneous Operators

Arithmetic Operators

- +: Addition
- -: Subtraction
- *: Multiplication
- /: Division
- %: Modulus

```
int main()
{
    int a = 10, b = 4, res;
    printf("a is %d and b is %d\n", a, b);
    res = a + b; // addition
    printf("a + b is %d\n", res);
    res = a - b; // subtraction
    printf("a - b is %d\n", res);
    res = a * b; // multiplication
    printf("a * b is %d\n", res);
    res = a / b; // division
    printf("a / b is %d\n", res);
    res = a % b; // modulus
    printf("a %% b is %d\n", res);
    return 0;
}
```

Relational Operators

==: Equal to

!=: Not equal to

>: Greater than

<: Less than

>=: Greater than or equal to

<=: Less than or equal to

```
int main()
{
    int a, b;
    printf("Enter the value of a:");
    scanf("%d",&a);
    printf("Enter the value of b:");
    scanf("%d",&b);
    if (a > b)
        printf("a is greater than b\n");
    if (a >= b)
        printf("a is greater than or equal to b\n");
    if (a < b)
        printf("a is lesser than b\n");
    if (a <= b)
        printf("a is lesser than or equal to b\n");
    if (a == b)
        printf("a is equal to b\n");
    if (a != b)
        printf("a is not equal to b\n");
    return 0;
}
```

Logical Operators

- &&: Logical AND
- ||: Logical OR
- !: Logical NOT

```
#include <stdio.h>
int main()
{
    int a = 10, b = 20;
    if (a > 0 && b > 0 || a != b) {
        printf("Both values are greater than 0 or they are
not equal \n");
    }
    else {
        printf("Both values are less than 0 or equal \n");
    }
    return 0;
}
```

Bitwise Operators

- &: Bitwise AND
- |: Bitwise OR
- ^: Bitwise XOR
- ~: Bitwise NOT
- <<: Left shift
- >>: Right shift

```
#include <stdio.h>
int main()
{
    short int a = 5, b = 9;
    printf("a = %d, b = %d\n", a, b);
    printf("a&b = %d\n", a & b);
    printf("a|b = %d\n", a | b);
    printf("a^b = %d\n", a ^ b);
    printf("~a = %d\n", a = ~a);
    printf("b<<1 = %d\n", b << 1);
    printf("b>>1 = %d\n", b >> 1);
    return 0;
}
```

Assignment Operators

- =: Assign
- +=: Add and assign
- -=: Subtract and assign
- *=: Multiply and assign
- /=: Divide and assign
- %=: Modulus and assign

```
int main()
{  int a;
   printf("Enter the value of a");
   scanf("%d", &a);
   a += 10;
   printf("Value of a is %d\n", a);
   a -= 10;
   printf("Value of a is %d\n", a);
   a *= 10;
   printf("Value of a is %d\n", a);
   a /= 10;
   printf("Value of a is %d\n", a);
   a %= 10;
   printf("Value of a is %d\n", a);
   return 0;
}
```

Unary Operators

- ++: Increment
- --: Decrement
- +: Unary plus
- -: Unary minus
- &: address of a variable
- sizeof: size of its operand in byte

```
int main()
{
    int a = 5;
    int b = 5;
    printf("Positive Integer = %d\n", a);
    printf("Negative Integer = %d\n", -a);
    printf("Pre-Incrementing a = %d\n", ++a);
    printf("Post-Incrementing b = %d\n", b++);
    printf("Pre-Decrementing a = %d\n", --a);
    printf("Post-Decrementing b = %d\n", b--);
    if (!(a > b))
        printf("b is greater than a\n");
    else
        printf("a is greater than b\n");
    printf("Address of a = %p\n", &a);
    printf("Size of int: %d\n", sizeof(short int));
    return 0;
}
```

Ternary (Conditional) Operator

Syntax: condition ? expression_if_true : expression_if_false

Acts as a shorthand for **if-else**

Example:

```
#include <stdio.h>
int main()
{
    int m = 5, n = 4;
    (m > n) ? printf("m is greater than n that is %d > %d",
                    m, n)
            : printf("n is greater than m that is %d > %d",
                    n, m);
    return 0;
}
```

Miscellaneous Operators

*: Pointer dereference

,: Comma operator

->: Structure pointer

Comma Operator

- Comma behaves as a separator in the case of function calls and definitions, variable declarations, and similar constructs.

```
#include <stdio.h>

int main(void) {

  int a = 1, b = 2, c = 3, x;
  x = a, b, c;
  printf("x = %d", x);
}
```

Operands vs. Operators

X+y: X and y are operand; + is operator

Unery operator: 1 operand ++

Binary operator: 2 operands +

Ternary operator: 3 operands ?:

Precedence of an operator

Precedence	Operator	Description	Associativity
1	**()**	Parentheses (function call)	Left-to-Right
	[]	Array Subscript (Square Brackets)	
	.	Dot Operator	
	->	Structure Pointer Operator	
	++ , —	Postfix increment, decrement	

2	**++ / —**	Prefix increment, decrement	Right-to-Left
	+ / –	Unary plus, minus	
	! , ~	Logical NOT, Bitwise complement	
	(type)	Cast Operator	
	*****	Dereference Operator	
	&	Addressof Operator	
	sizeof	Determine size in bytes	

3	***,/,%**	Multiplication, division, modulus	Left-to-Right
4	**+/-**	Addition, subtraction	Left-to-Right
5	**<< , >>**	Bitwise shift left, Bitwise shift right	Left-to-Right
6	**< , <=**	Relational less than, less than or equal to	Left-to-Right
	> , >=	Relational greater than, greater than or equal to	
7	**== , !=**	Relational is equal to, is not equal to	Left-to-Right
8	**&**	Bitwise AND	Left-to-Right
9	**^**	Bitwise exclusive OR	Left-to-Right
10	**|**	Bitwise inclusive OR	Left-to-Right
11	**&&**	Logical AND	Left-to-Right
12	**||**	Logical OR	Left-to-Right
13	**?:**	Ternary conditional	Right-to-Left

Precedence	Operator	Description	Associativity
14	=	Assignment	Right-to-Left
	+= , -=	Addition, subtraction assignment	
	*= , /=	Multiplication, division assignment	
	%= , &=	Modulus, bitwise AND assignment	
	^= , |=	Bitwise exclusive, inclusive OR assignment	
	<<=, >>=	Bitwise shift left, right assignment	
15	,	comma (expression separator)	Left-to-Right

Example of Expression

1. X=3+4*2=3+8=11

2. a= 3*4%5/2

=12%5/2

=2/2

=1

Example

```
#include <stdio.h>
int main(void) {
  int x = 10, y = 5;
  y = x++ + ++y;
  printf("x = %d y = %d", x, y);
  /* post incr ++ is has highest priority, so x becomes 11 but it'll increase only after the statement is evaluated, so it is not reflectred in the value of 'y' y = 10 + 5 x = x + 1 */
  return 0;
}
```

Let’s Solve

10*4>>2 || 3

5/10*5+5*2

10/(5<10 && 20<30)

5|10&12>>2

10/(5-5)

Upcoming lecture

- Type Conversion,
- Conditional Statements in C,
- Loops