SNS COLLEGE OF ENGINEERING

Kurumbapalayam (Po), Coimbatore – 641 107

An Autonomous Institution

Accredited by NBA – AICTE and Accredited by NAAC – UGC with 'A' Grade
Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

COURSE NAME : 23ITT101- PROBLEM SOLVING AND C PROGRAMMING

I YEAR /I SEMESTER

Unit 2- C-Programming Basics

Topic 5: Operators and expressions

Brain Storming

1. What are operators?
2. How mathematical and logical operations are performed in C?

C Operators

- An operator is simply a symbol that is used to perform operations.
- There are following types of operators to perform different types of operations in C language.

  - ✓ Arithmetic Operators
  - ✓ Relational Operators
  - ✓ Shift Operators
  - ✓ Logical Operators
  - ✓ Bitwise Operators
  - ✓ Ternary or Conditional Operators
  - ✓ Assignment Operator
  - ✓ Misc Operator

Operators and expressions

```
#include <stdio.h>
 void main()
{
int a = 20;
int b = 10;
 int c = 15;
int d = 5;
 int e;
 e = a + b * c / d;
printf("e : %d\n" , e );
}
```

Result?

```
#include <stdio.h>

Void main(){

   int a = 20;
   int b = 10;
   int c = 15;
   int d = 5;
   int e;
   e = (a + b) * c / d;
   printf("e:  %d\n",  e);
}
```

Result?

Arithmetic Operators

Operator	Description	Example
+	Adds two operands.	A + B = 30
−	Subtracts second operand from the first.	A − B = -10
*	Multiplies both operands.	A * B = 200
/	Divides numerator by de-numerator.	B / A = 2
%	Modulus Operator and remainder of after an integer division.	B % A = 0
++	Increment operator increases the integer value by one.	A++ = 11
--	Decrement operator decreases the integer value by one.	A-- = 9

Relational Operators

Operator	Description	Example
==	Checks if the values of two operands are equal or not. If yes, then the condition becomes true.	(A == B) is not true.
!=	Checks if the values of two operands are equal or not. If the values are not equal, then the condition becomes true.	(A != B) is true.
>	Checks if the value of left operand is greater than the value of right operand. If yes, then the condition becomes true.	(A > B) is not true.
<	Checks if the value of left operand is less than the value of right operand. If yes, then the condition becomes true.	(A < B) is true.
>=	Checks if the value of left operand is greater than or equal to the value of right operand. If yes, then the condition becomes true.	(A >= B) is not true.
<=	Checks if the value of left operand is less than or equal to the value of right operand. If yes, then the condition becomes true.	(A <= B) is true.

Logical Operators

Operator	Description	Example
&&	Called Logical AND operator. If both the operands are non-zero, then the condition becomes true.	(A && B) is false.
||	Called Logical OR Operator. If any of the two operands is non-zero, then the condition becomes true.	(A || B) is true.
!	Called Logical NOT Operator. It is used to reverse the logical state of its operand. If a condition is true, then Logical NOT operator will make it false.	!(A && B) is true.

Bitwise Operators

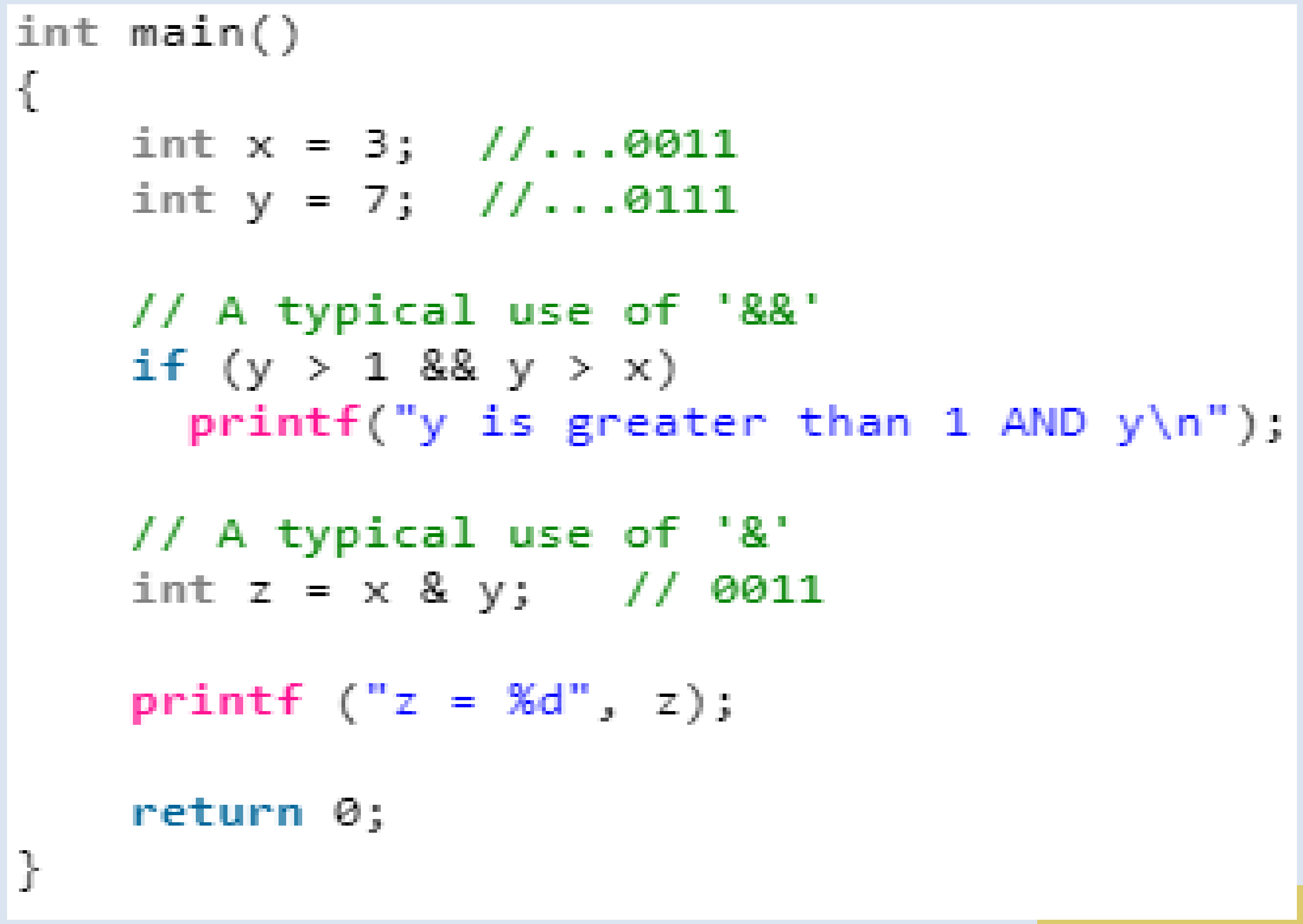
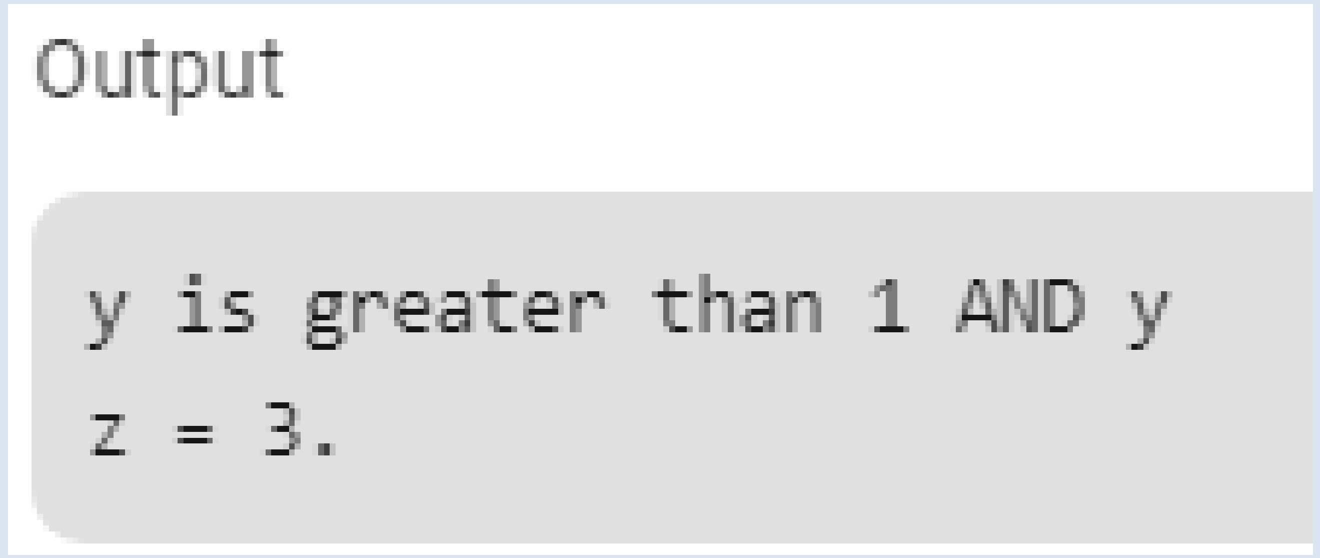
p	q	p & q	p | q	p ^ q
0	0	0	0	0
0	1	0	1	1
1	1	1	1	0
1	0	0	1	1

Conti…

Operator	Description	Example
&	Binary AND Operator copies a bit to the result if it exists in both operands.	(A & B) = 12, i.e., 0000 1100
|	Binary OR Operator copies a bit if it exists in either operand.	(A | B) = 61, i.e., 0011 1101
^	Binary XOR Operator copies the bit if it is set in one operand but not both.	(A ^ B) = 49, i.e., 0011 0001
~	Binary One's Complement Operator is unary and has the effect of 'flipping' bits.	(~A) = ~(60), i.e,. -0111101
<<	Binary Left Shift Operator. The left operands value is moved left by the number of bits specified by the right operand.	A << 2 = 240 i.e., 1111 0000
>>	Binary Right Shift Operator. The left operands value is moved right by the number of bits specified by the right operand.	A >> 2 = 15 i.e., 0000 1111

Bitwise Vs logical AND operators

A Bitwise AND operator is represented as '&' and a logical operator is represented as '&&'.

LOGICAL	Bitwise
The logical AND operator '&&' expects its operands to be boolean expressions (either 1 or 0) and returns a boolean value. 	The bitwise and operator '&' works on Integral (short, int, unsigned, char, bool, unsigned char, long) values and return Integral value. 

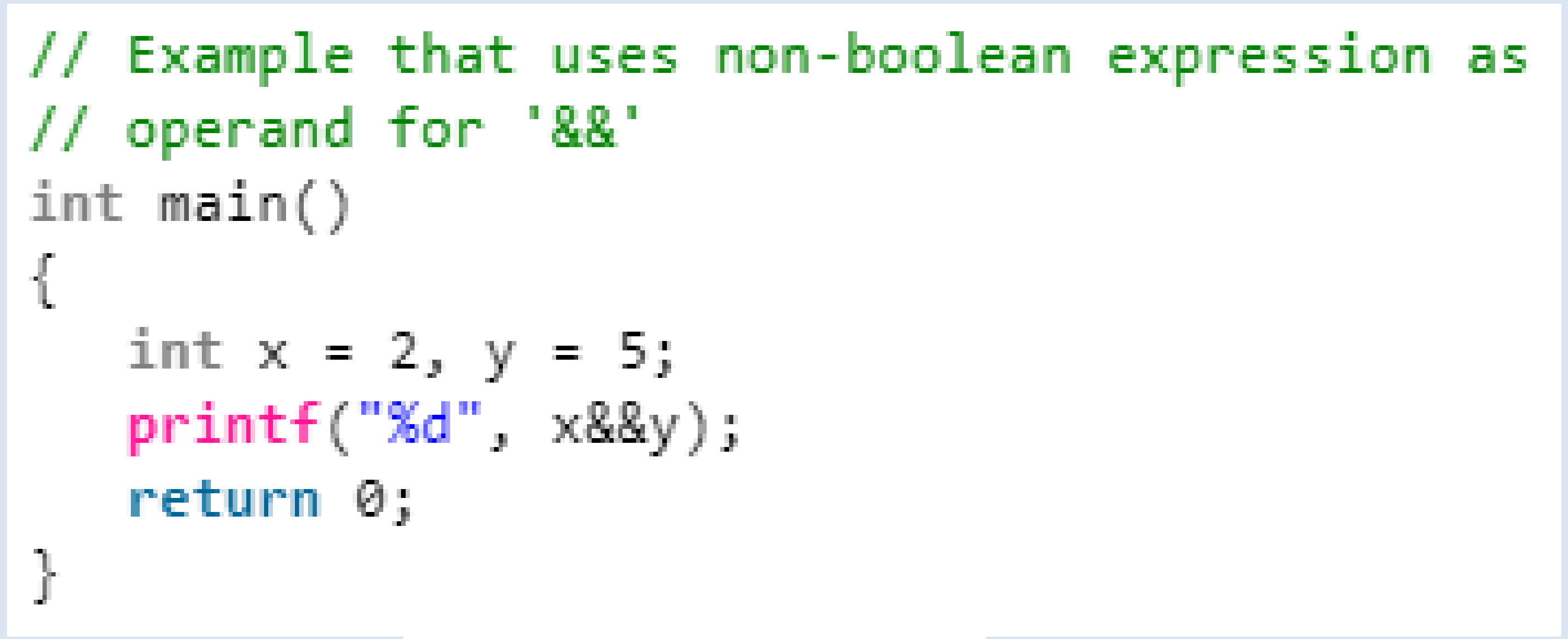
```c
int main()
{
    int x = 3;  //...0011
    int y = 7;  //...0111

    // A typical use of '&&'
    if (y > 1 && y > x)
      printf("y is greater than 1 AND y\n");

    // A typical use of '&'
    int z = x & y;    // 0011

    printf ("z = %d", z);

    return 0;
}
```

Output

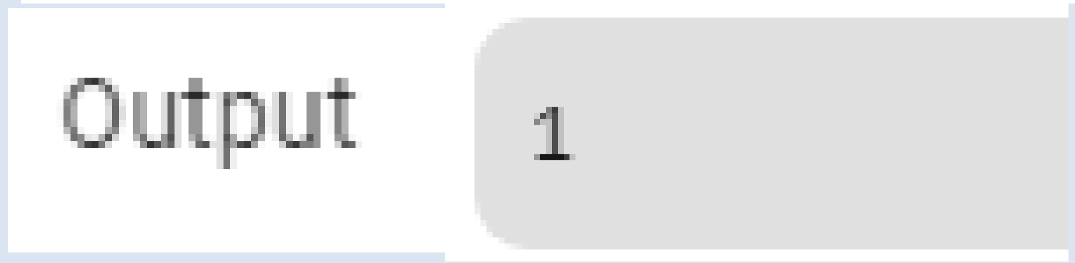
```
y is greater than 1 AND y
z = 3.
```

Bitwise Vs logical AND operators

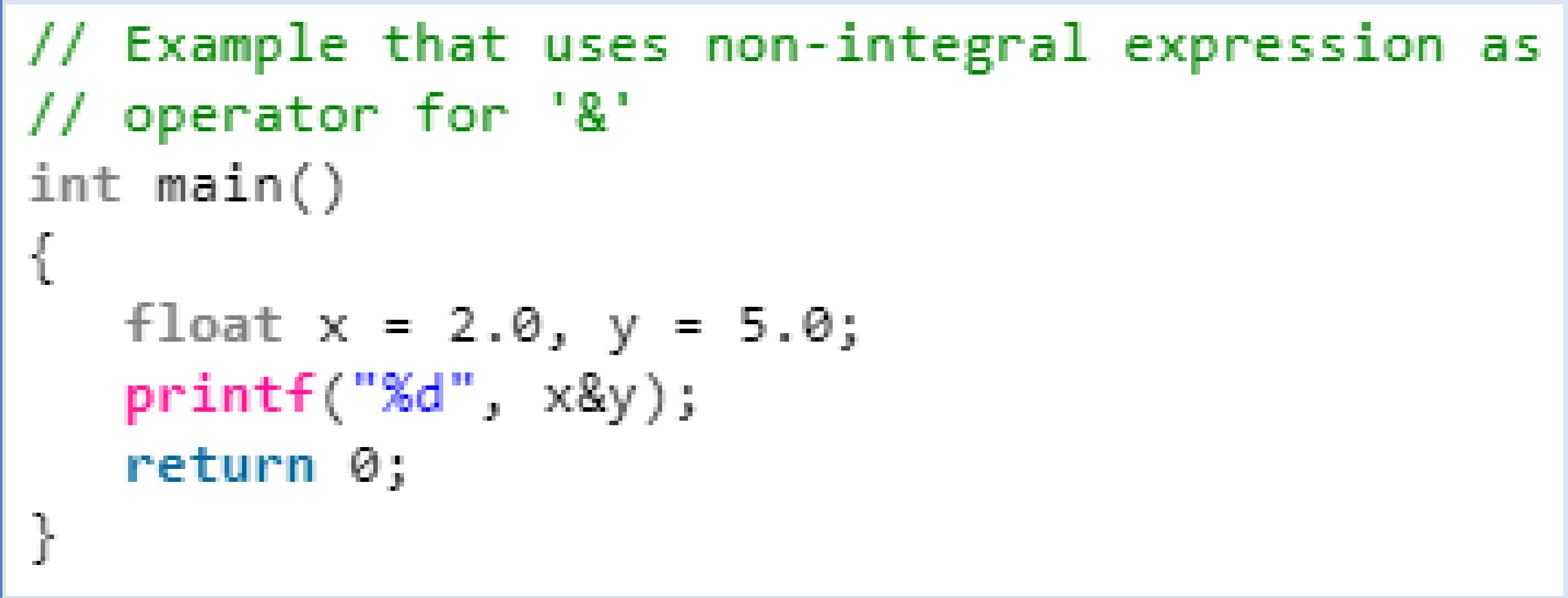
LOGICAL	Bitwise
If an integral value is used as an operand for ‘&&’ which is supposed to work on boolean values,(A zero is considered as false and non-zero is considered as true.)	It is compiler error to use non-integral expression as operand for bitwise &.



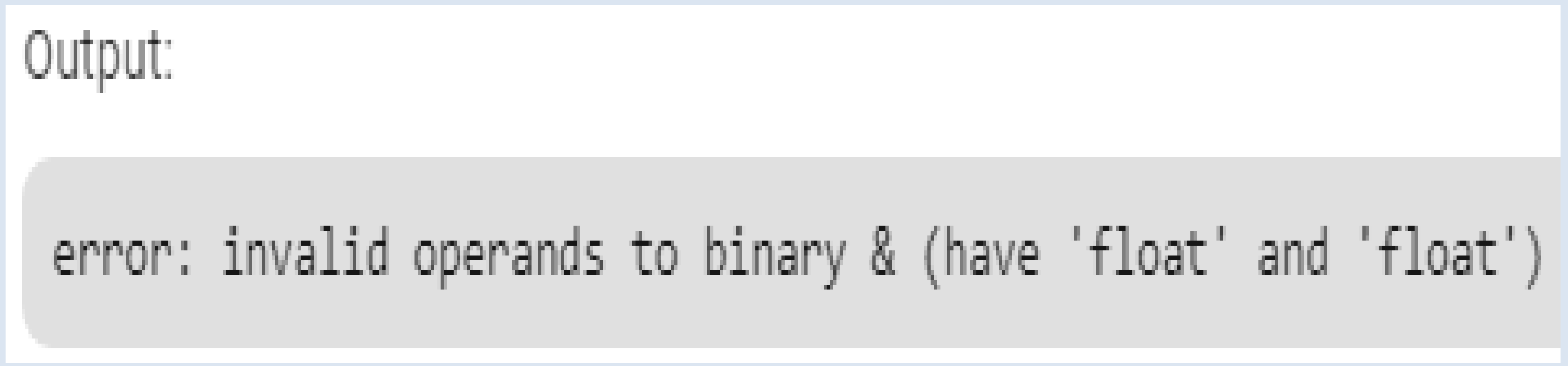
```
// Example that uses non-boolean expression as
// operand for '&&'
int main()
{
   int x = 2, y = 5;
   printf("%d", x&&y);
   return 0;
}
```



```
Output    1
```

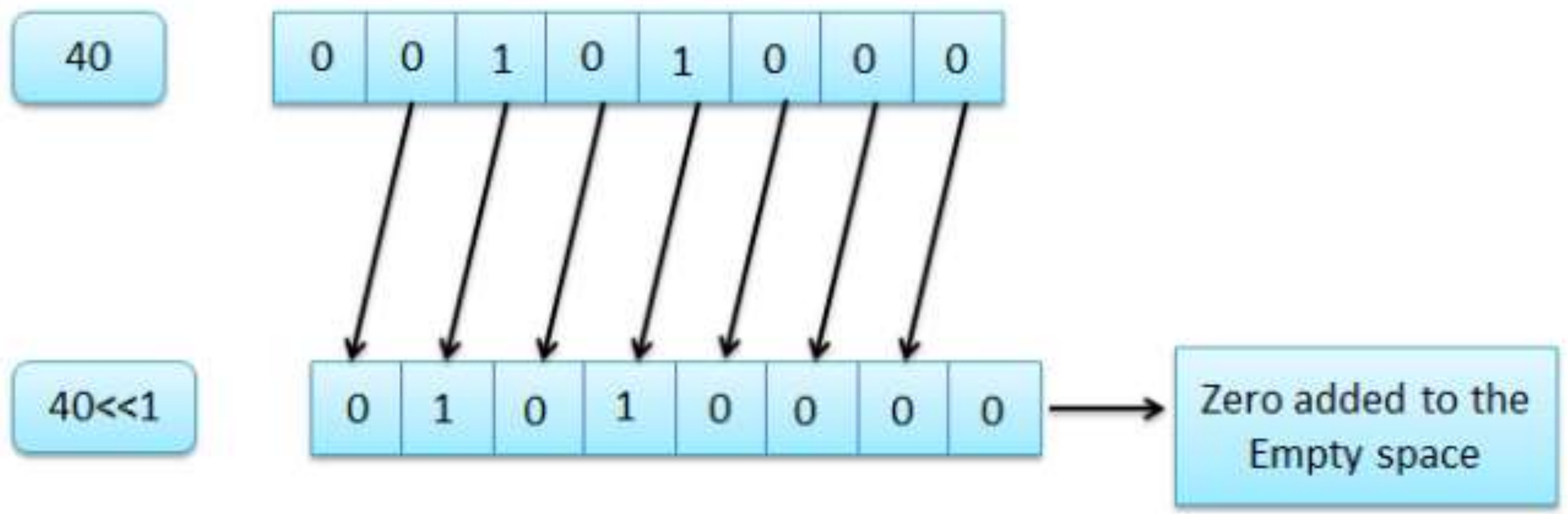


```
// Example that uses non-integral expression as
// operator for '&'
int main()
{
   float x = 2.0, y = 5.0;
   printf("%d", x&y);
   return 0;
}
```

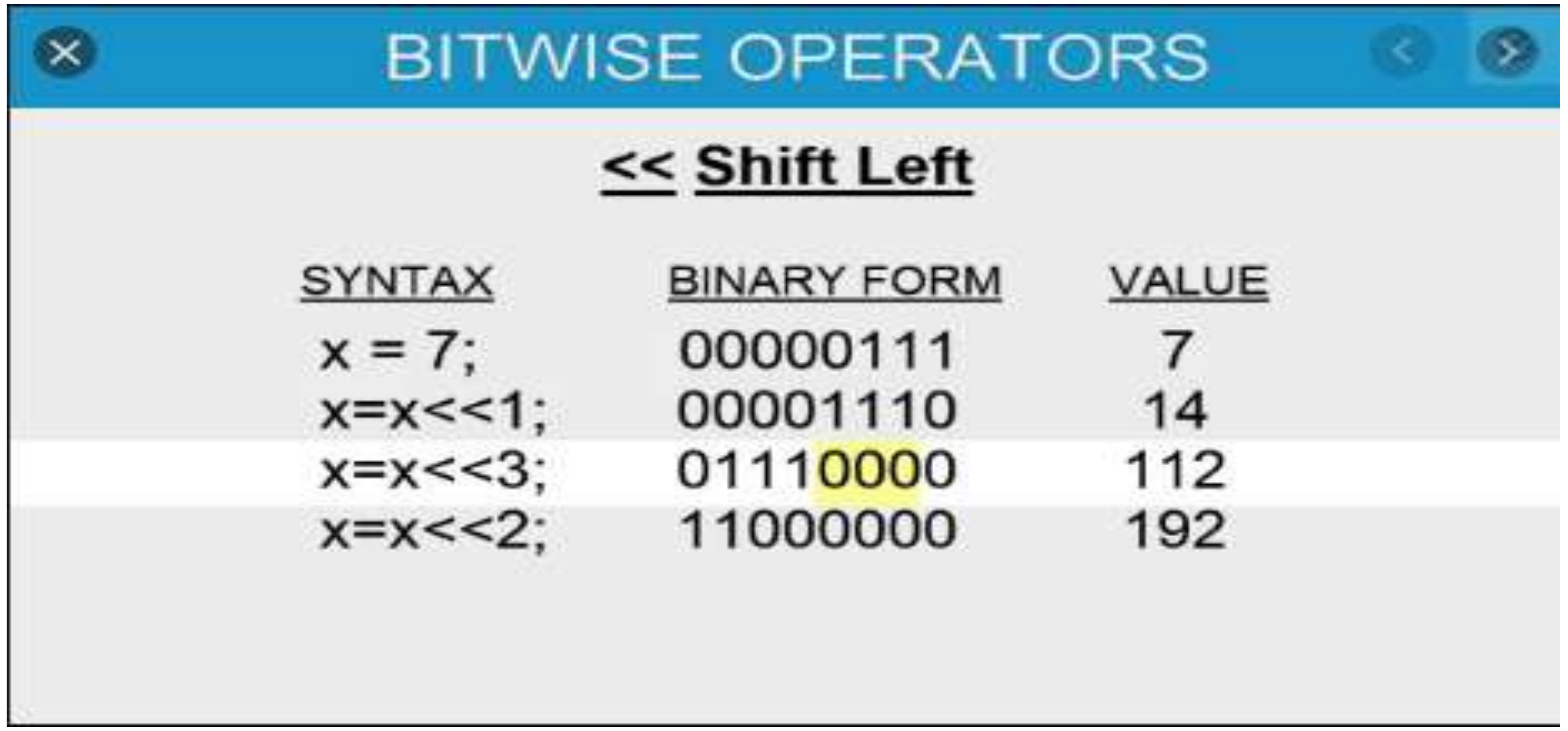


```
Output:

error: invalid operands to binary & (have 'float' and 'float')
```

Example for Shift Operators



Thus, the value of 40<<1 is 01010000. The decimal equivalent of this binary value is 80.



BITWISE OPERATORS

<< Shift Left

SYNTAX	BINARY FORM	VALUE
x = 7;	00000111	7
x=x<<1;	00001110	14
x=x<<3;	01110000	112
x=x<<2;	11000000	192

Assignment Operators

Operator	Description	Example
=	Simple assignment operator. Assigns values from right side operands to left side operand	C = A + B will assign the value of A + B to C
+=	Add AND assignment operator. It adds the right operand to the left operand and assign the result to the left operand.	C += A is equivalent to C = C + A
-=	Subtract AND assignment operator. It subtracts the right operand from the left operand and assigns the result to the left operand.	C -= A is equivalent to C = C - A
*=	Multiply AND assignment operator. It multiplies the right operand with the left operand and assigns the result to the left operand.	C *= A is equivalent to C = C * A
/=	Divide AND assignment operator. It divides the left operand with the right operand and assigns the result to the left operand.	C /= A is equivalent to C = C / A

Assignment Operators

Operator	Description	Example
%=	Modulus AND assignment operator. It takes modulus using two operands and assigns the result to the left operand.	C %= A is equivalent to C = C % A
<<=	Left shift AND assignment operator.	C <<= 2 is same as C = C << 2
>>=	Right shift AND assignment operator.	C >>= 2 is same as C = C >> 2
&=	Bitwise AND assignment operator.	C &= 2 is same as C = C & 2
^=	Bitwise exclusive OR and assignment operator.	C ^= 2 is same as C = C ^ 2
|=	Bitwise inclusive OR and assignment operator.	C |= 2 is same as C = C | 2

Misc Operators ↦ sizeof & ternary

Operator	Description	Example
sizeof()	Returns the size of a variable.	sizeof(a), where a is integer, will return 4.
&	Returns the address of a variable.	&a; returns the actual address of the variable.
*	Pointer to a variable.	*a;
? :	Conditional Expression.	If Condition is true ? then value X : otherwise value Y

Operator Precedence in C

A single expression in C may have multiple operators of different types. The C compiler evaluates its value based on the operator precedence and associativity of operators.

The precedence of operators determines the order in which they are evaluated in an expression. Operators with higher precedence are evaluated first.

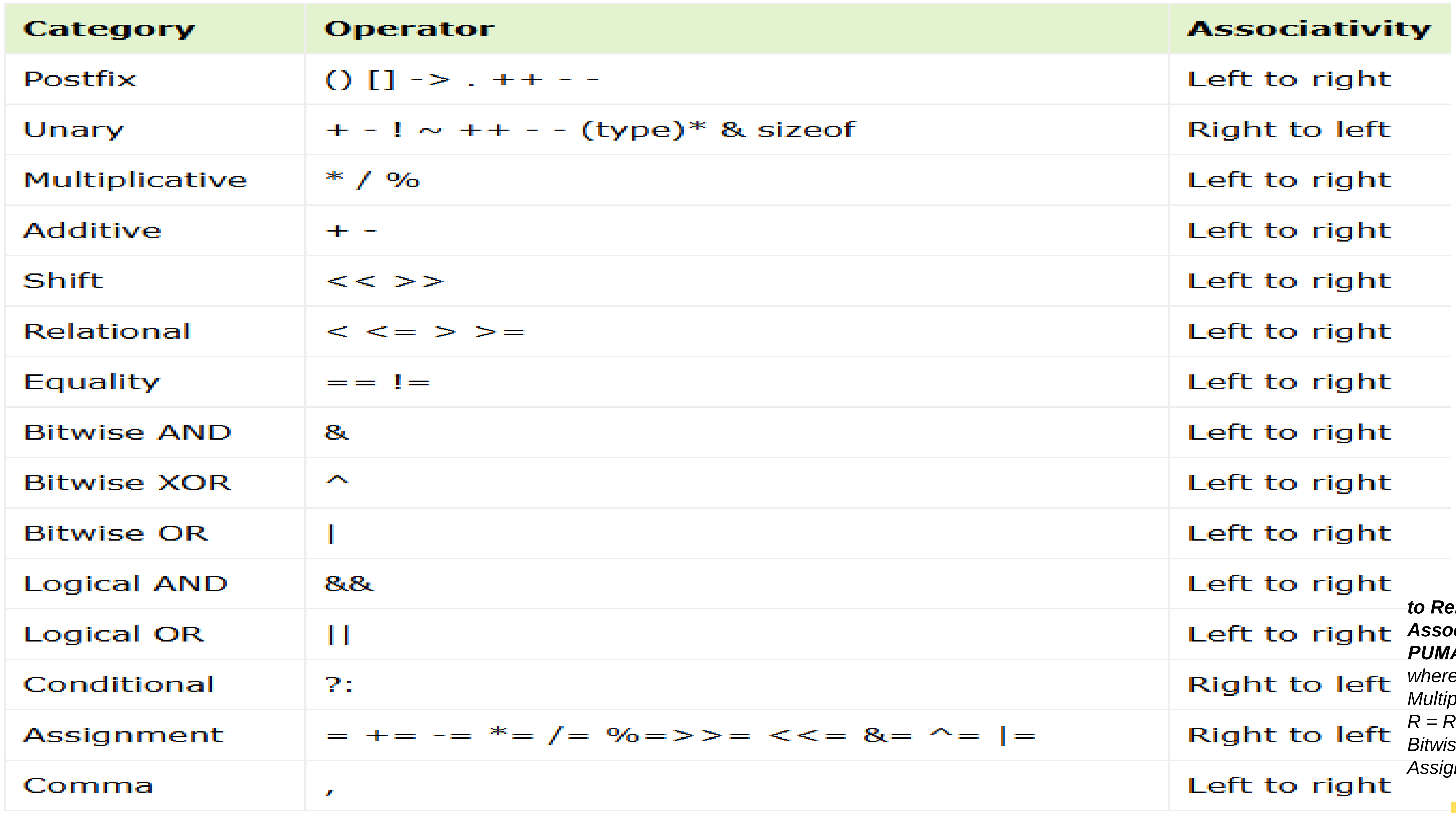
For example,

```
x = 7 + 3 * 2;
```

Here, the multiplication operator "*" has a higher precedence than the addition operator "+". So, the multiplication 3*2 is performed first and then adds into 7, resulting in "x = 13".

The following table lists the order of precedence of operators in C. Here, operators with the highest precedence appear at the top of the table, and those with the lowest appear at the bottom.

Operator precedence



Category	Operator	Associativity
Postfix	() [] -> . ++ - -	Left to right
Unary	+ - ! ~ ++ - - (type)* & sizeof	Right to left
Multiplicative	* / %	Left to right
Additive	+ -	Left to right
Shift	<< >>	Left to right
Relational	< <= > >=	Left to right
Equality	== !=	Left to right
Bitwise AND	&	Left to right
Bitwise XOR	^	Left to right
Bitwise OR	|	Left to right
Logical AND	&&	Left to right
Logical OR	||	Left to right
Conditional	?:	Right to left
Assignment	= += -= *= /= %=>>= <<= &= ^= |=	Right to left
Comma	,	Left to right

to Remember the Operators Associtivity and Precedence: PUMA'S REBL TAC
where, P = Postfix, U = Unary, M = Multiplicative, A = Additive, S = Shift, R = Relational, E = Equality, B = Bitwise, L = Logical, T = Ternary, A = Assignment and C = Comma

Operators and expressions

```
#include <stdio.h>
 void main()
{
int a = 20;
int b = 10;
 int c = 15;
int d = 5;
 int e;
 e = a + b * c / d;
printf("e : %d\n" , e );
}
```

Result?

```
#include <stdio.h>

Void main(){

   int a = 20;
   int b = 10;
   int c = 15;
   int d = 5;
   int e;
   e = (a + b) * c / d;
   printf("e:  %d\n",  e);
}
```

Result?

Assessment 1

1. Write operators and expression?

Ans : ____

2. Write about operators precedence?

Ans : ____e=50_________________________________

References

TEXT BOOKS

1.E.Balagurusamy, “Fundamentals of Computing and Computer Programming”, 2nd Edition Tata McGRaw-Hill Publishing Company Limited, (2012). (UNIT – I, II, III, IV, V)

2.Ashok.N.Kamthane,“ Computer Programming”, Pearson Education (India) (2010). (UNIT –II, III IV, V)

3.Reema Thareja, “Programming in C”, 2nd Edition, Oxford University Press,(2015). (UNIT –I,II, III, IV, V)

REFERENCES

1.Byron Gottfried, “Programming with C”, 2nd Edition, (Indian Adapted Edition), TMH Publications, (2006). (Unit II, III, IV)

2.Stephan G kochan, “Programming in C” Pearson Education (2008), (UNIT II, III, IV, V)

3.P.Sudharson, “Computer Programming”, RBA Publications (2008), (UNIT I, II, III, IV)

4.Yashavant P. Kanetkar. “Let Us C”, BPB Publications, 2014.(Unit II, III, IV, V)

5.Anita Goel and Ajay Mittal, “Computer Fundamentals and Programming in C”, Dorling Kindersley (India) Pvt. Ltd., Pearson Education in South Asia, 2011. (UNIT – I, II, III, IV, V)

Thank You