SRM

INSTITUTE OF SCIENCE AND TECHNOLOGY, CHENNAI.

18CSS101J – Programming for Problem Solving

Unit II

SRM

INSTITUTE OF SCIENCE AND TECHNOLOGY, CHENNAI.

UNIT II

Relational and logical Operators - Condition Operators, Operator Precedence - Expressions with pre / post increment Operator - Expression with conditional and assignment operators - If statement in expression - L value and R value in expression - Control Statements – if and else - else if and nested if, switch case - Iterations, Conditional and Unconditional Branching - For loop - While loop - do while, goto, break, continue – Array - Initialization and Declaration-

SRM

INSTITUTE OF SCIENCE AND TECHNOLOGY, CHENNAI.

UNIT II

- Initialization: one Dimensional Array-Accessing, Indexing one Dimensional Array Operations - Array Programs – 1D

2. 1 Operators in C

a) Relational Operators

b) Logical Operators

c) Conditional Operators

2. 1 Operators in C Contd…

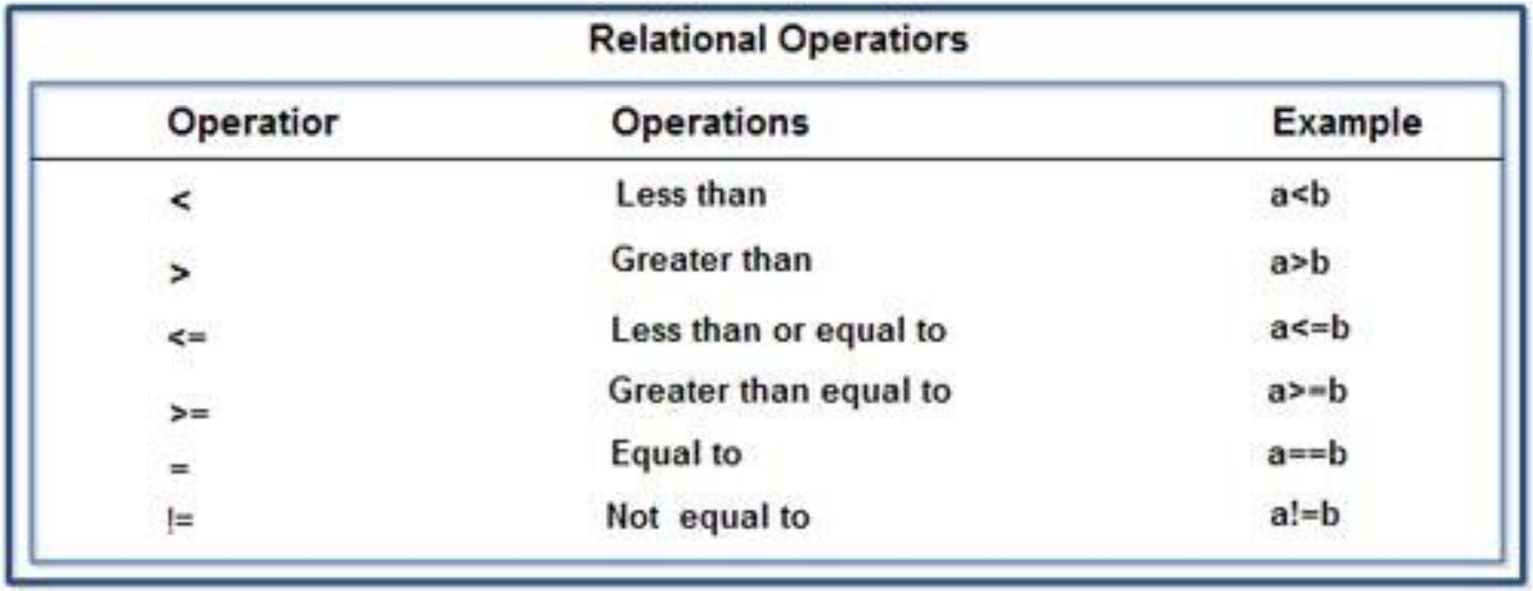
a) Relational Operators

- ❑ Binary Operators (or) Boolean Operators
- ❑ Produces an integer result
    - ❑ **Condition True :** Integer value is 1
    - ❑ **Condition False :** Integer value is 0
- ❑ Compares
    - ❑ Values between two variables
    - ❑ Values between variables and constants

2. 1 Operators in C Contd…

a) Relational Operators Contd…

- ❑ **Relational Expression / Boolean Expression :** An expression containing a relational operator



Relational Operators

Operatior	Operations	Example
<	Less than	a<b
>	Greater than	a>b
<=	Less than or equal to	a<=b
>=	Greater than equal to	a>=b
=	Equal to	a==b
!=	Not equal to	a!=b

2. 1 Operators in C Contd…

a) Relational Operators Contd…

- Consider a = 10 and b =4. The relational expression returns the following integer values

Relational Expression	Result	Return Values
a < b	False	0
a > b	True	1
a < = b	False	0
a > = b	True	1
a = = b	False	0
a ! = b	True	1

/* Program for Relational Operations*/

```
#include<stdio.h>
int main( )
{
        int a,b;
        printf("Enter the two Values\n");
        scanf("%d%d", &a, &b);
        printf(“a>b is %d\n“, (a>b));
        printf(“a<b is %d\n“, (a<b));
        printf(“a>=b is %d\n“, (a>=b));
        printf(“a<=b is %d\n“, (a<=b));
        printf(“a==b is %d\n“, (a==b));
        printf(“a!=b is %d\n“, (a!=b));
        return 0;
}
```

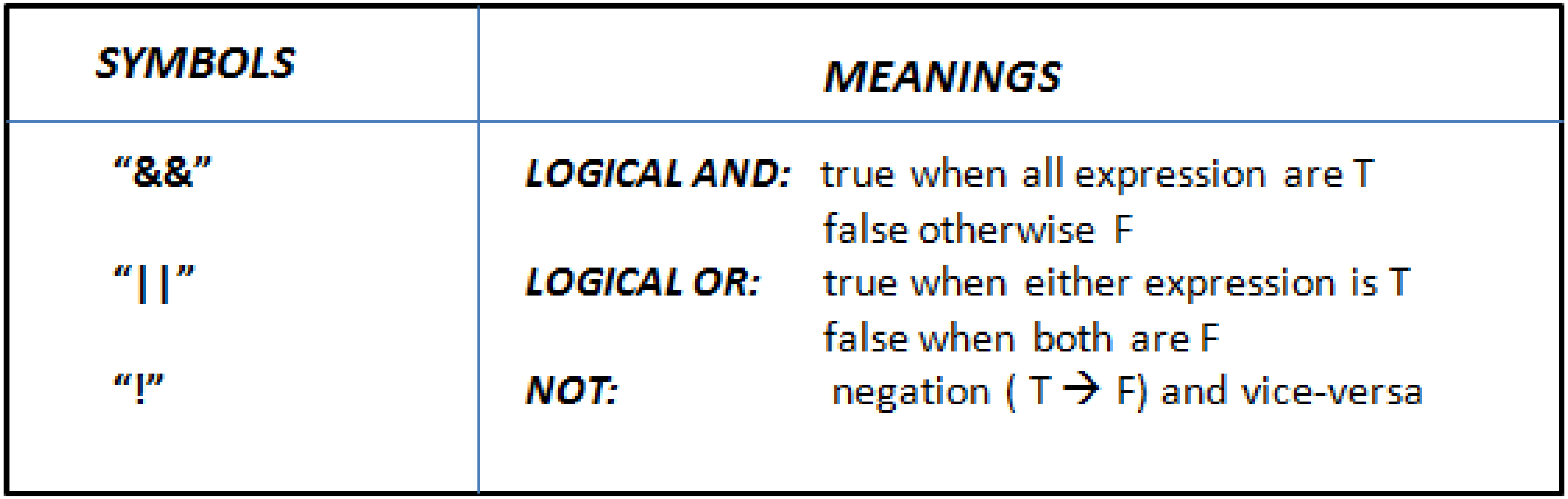
Output

```
4
2
a > b is 1
a < b is 0
a > = b is 1
a < = b is 0
a = = b is 0
a ! = b is 1
```

2. 1 Operators in C Contd…

b) Logical Operators

- ❑ Combines two or more relations
- ❑ Used for testing one or more conditions



SYMBOLS	*MEANINGS*
"&&"	***LOGICAL AND:*** true when all expression are T false otherwise F
"||"	***LOGICAL OR:*** true when either expression is T false when both are F
"!"	***NOT:*** negation (T → F) and vice-versa

2. 1 Operators in C Contd…

b) Logical Operators Contd…

- ❑ **Logical Expression / Compound Relational Expression :**

  An expression which combines two or more relational expression

Op1	Op2	Op1 && Op2	Op1 || Op2
F (0)	F (0)	F (0)	F (0)
F (0)	T (1)	F (0)	T (1)
T (1)	F (0)	F (0)	T (1)
T (1)	T (1)	T (1)	T (1)

2. 1 Operators in C Contd…

b) Logical Operators Contd…

❑ Consider a = 10 and b =4. The Logical expression returns the following integer values

Relational Expression	Result	Return Values
a < 5 && b > 2	True	1
a < 5 && b < 2	False	0
a >5 && b < 2	False	0
a >5 || b < 2	True	1
a <5 || b < 2	False	0
a > 5 || b < 2	True	1

/* Program for Logical Operations*/

```
#include<stdio.h>
int main( )
{
    int age,height;
    printf("Enter Age of Candidate:\n");
    scanf("%d", &age);
    printf(“Enter Height of Candidate:\n“);
    scanf("%d", &height);
    if ((age>=18) && (height>=5))
        printf(“The Candidate is Selected”);
    else
        printf(“Sorry, Candidate not Selected”);
    return 0;
}
```

Output 1

```
Enter Age of Candidate: 18
Enter Height of Candidate: 6
The Candidate is Selected
```

Output 2

```
Enter Age of Candidate: 19
Enter Height of Candidate: 4
Sorry, Candidate not Selected
```

2. 1 Operators in C Contd…

c) Conditional Operators

- ? and are the Conditional Operators
- Also called as Ternary Operators
- Shorter form of if-then-else statement
- ***Syntax***

  Expression 1 ? Expression 2 : expression 3

- If expression 1 is true then the value returned will be expression 2
- Otherwise the value returned will be expression 3

```
#include<stdio.h>
int main( )
{
        int x, y;
        scanf("%d", &x);
        y=(x > 5 ? 3 : 4);
        printf(“%d”, y);
        return 0;
}
```

```
#include<stdio.h>
int main( )
{
        int x, y;
        scanf("%d", &x);
        if(x >5)
                y=3;
        else
                y=4;
        printf(“%d”, y);
        return 0;
}
```

```
                /* Program for Addition (or) Multiplication*/
#include<stdio.h>
int main( )
{
        int   a, b, result, choice;
        printf(“Enter  first number \n”);
        scanf(“%d”,&a);
        printf(”Enter second number\n”);
        scanf(“%d”,&b);
        printf(“Enter 1 for addition or 2 for multiplication\n”);
        scanf(“%d”,&choice);
        result = (choice==1)?a+b:(choice==2)?a*b:printf(“Invalid”);
        if(choice==1||choice==2)
        printf(“The result is %d\n\n”,result);
        return 0;
}
```

Output

```
Enter first number
10
Enter second number
3
Enter 1 for addition or 2 for multiplication
2
The result is 30
```

/* Program to find the maximum of 3 Numbers*/

```
#include <stdio.h>
 int main( )
 {
         int a, b, c, max;
        printf("Enter three numbers: ");
        scanf("%d%d%d",&a, &b, &c);
        max = (a > b && a > c) ? a : (b > c) ? b : c;
        printf("\n Maximum between %d, %d and %d = %d", a, b, c, max);
        return 0;
}
```

Output

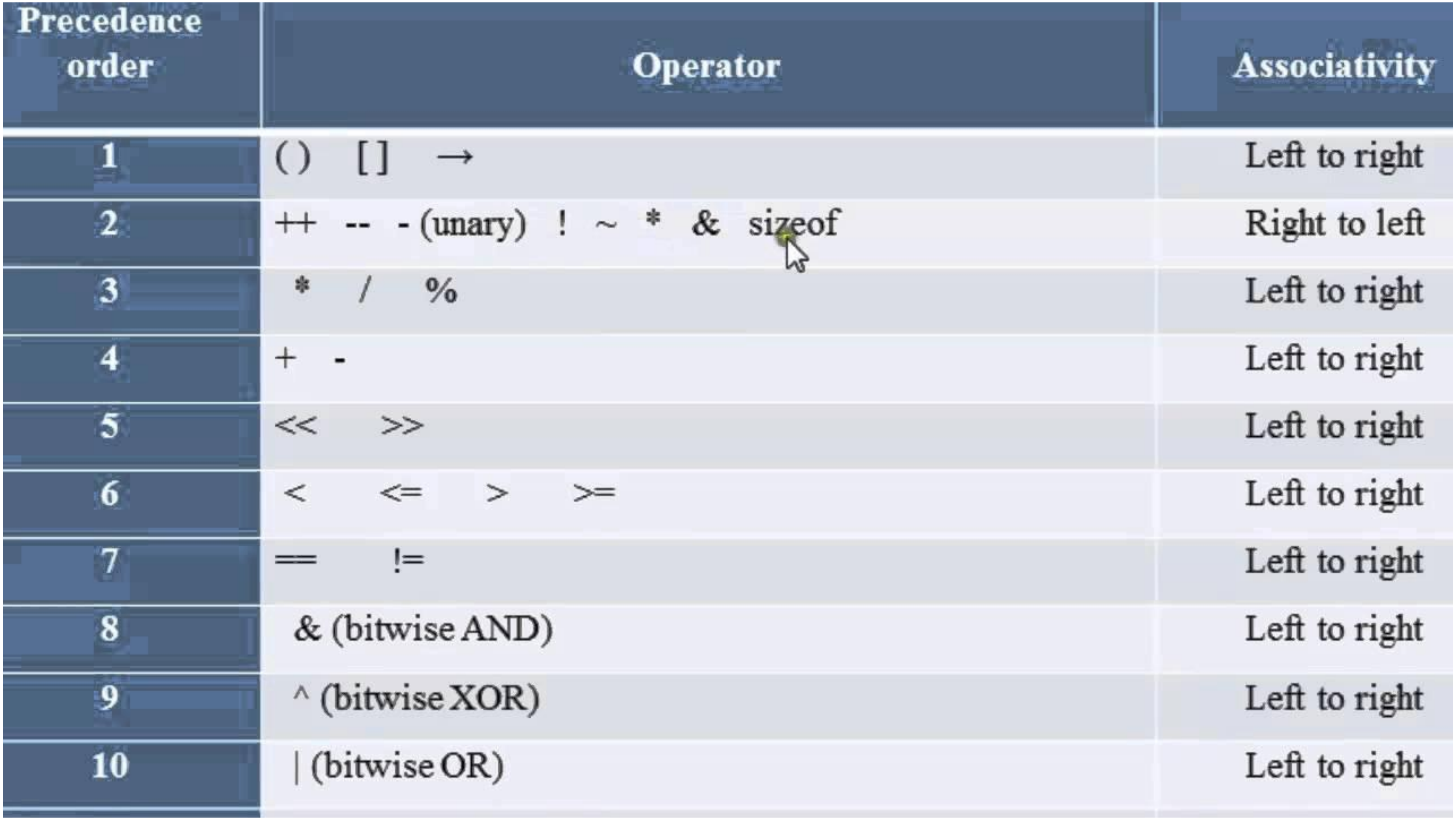
```
Enter three numbers: 30 10 40

Maximum between a, b and c = 40
```

2. 2 Operator Precedence

- ***Operator Precedence*** is used to determine the order of operators evaluated in an expression
  - Every operator has precedence (Priority)
  - Operator with higher precedence is evaluated first and the operator with least precedence is evaluated last
- ***Associativity*** is used when two operators of same precedence appear in an expression
  - Determines the order of evaluation of those operators
  - Associativity can be either Left to Right or Right to Left

2. 2 Operator Precedence Contd…

- Operators are listed in descending order of precedence
- An Expression can contain several operators with equal precedence
  - Evaluation proceeds according to the associativity of the operator i.e.,
    - From Right to Left (or)
    - From Left to Right
- *Note:* Order of operations is not defined by the language



Precedence order	Operator	Associativity
1	( ) [ ] →	Left to right
2	++ -- - (unary) ! ~ * & sizeof	Right to left
3	* / %	Left to right
4	+ -	Left to right
5	<< >>	Left to right
6	< <= > >=	Left to right
7	== !=	Left to right
8	& (bitwise AND)	Left to right
9	^ (bitwise XOR)	Left to right
10	| (bitwise OR)	Left to right

2. 3 Expressions using Pre/Post Increment Operator

- □ Increment operators increase the value of the variable by one
- □ Decrement operators decrease the value of the variable by one
- □ *Syntax*

Increment operator: ++var_name; (or) var_name++;

Decrement operator: – -var_name; (or) var_name – -;

- □ Example

Increment operator : ++ i ; i ++ ;

Decrement operator : – – i ; i – – ;

/* Expresssions using Pre-Increment Operator*/

```
#include<stdio.h>
int main( )
{
        int x,i;
        i=10;
        x=++i;
        printf("x: %d",x);
        printf("i: %d",i);
        return 0;
}
```

Output

```
x: 11

i: 11
```

/* Expresssions using Post-Increment Operator*/

```
#include<stdio.h>
int main( )
{
        int x,i;
        i=10;
        x=i++;
        printf("x: %d",x);
        printf("i: %d",i);
        return 0;
}
```

Output

```
x: 10
i: 11
```

/* Expresssions using Pre-Decrement Operator*/

```
#include<stdio.h>
int main( )
{
        int x,i;
        i=10;
        x=--i;
        printf("x: %d",x);
        printf("i: %d",i);
        return 0;
}
```

Output

```
x: 9

i: 9
```

/* Expresssions using Post-Decrement Operator*/

```
#include<stdio.h>
int main( )
{
        int x,i;
        i=10;
        x=i--;
        printf("x: %d",x);
        printf("i: %d",i);
        return 0;
}
```

Output

```
x: 10
i: 9
```

/* Expresssions using Increment / Decrement Operators*/

```
#include<stdio.h>
int main( )
{
        int  p,q,x,y;
        printf(“Enter the value of x \n”);
        scanf(“%d” ,&x);
        printf(“Enter the value of y \n”);
        scanf(“%d” ,&y);
        printf(“x=%d\ny=%d\n”,x,y);
        p=x++;
        q=y++;
        printf(“x=%d\ty=%d\n”,x,y);
        printf(“x=%d\tq=%d\n”,p,q);
        p=--x;
        q=--y;
        printf(“x=%d\ty=%d\n”,x,y);
        printf(“p=%d\tq=%d\n”,p,q);
        return 0;
}
```

Output

```
Enter the value of x 10

Enter the value of y 20

x = 10

y = 20

x = 11    y  =  21

p = 10    q  = 20

x =  10    y  =  20

p = 10     q  =  20
```

2. 4 Expressions using Conditional Operator

- ❑ Any operator is used on three operands or variable is known as **Ternary Operator**
- ❑ It can be represented with **? :** . It is also called as **conditional operator**

/* Program for Printing Odd or Even Number*/

```c
#include<stdio.h>
int main( )
{
int num;
printf("Enter the Number : ");
scanf("%d",&num);
(num%2==0)?printf("Even\n"):printf("Odd");
 }
```

Output

```
Enter the Number : 10
Even
```

/* Program for Eligibility to Vote*/

```
#include<stdio.h>
 int main( )
{
  int age;
   printf(" Please Enter your age here: \n ");
  scanf(" %d ", &age);
 (age >= 18) ? printf(" You are eligible to Vote ") : printf(" You are not eligible to Vote ");
 return 0;
}
```

Output

```
Please Enter your age here: 19

You are eligible to Vote
```

/* Program for Finding Biggest of 2 Numbers*/

```
#include<stdio.h>
int main( )
{
int a, b, max;
printf("Enter a and b: ");
scanf("%d%d", &a, &b);
max = a > b ? a : b; printf("Largest of the two numbers = %d\n", max);
return 0;
}
```

Output

```
Enter a and b: 10 20

Largest of the two numbers = 20
```

2. 5 Expressions using Assignment Operator

- Assignment Operator is used to assign value to an variable
- Assignment Operator is denoted by equal to sign
- Assignment Operator is binary operator which operates on two operands
- Assignment Operator have Two Values – L-Value and R-Value
    - Operator = copies R-Value into L-Value
- Assignment Operator have lower precedence than all available operators but has higher precedence than comma Operator

Operator	Example	Equivalent Expression
=	m = 10	m = 10
+=	m += 10	m = m + 10
−=	m −= 10	m = m − 10
*=	m *= 10	m = m * 10
/ =	m / =	m = m/10
% =	m % = 10	m = m%10
<<=	a <<= b	a = a << b
>>=	a >>= b	a = a >> b
>>>=	a >>>= b	a = a >>> b
& =	a & = b	a = a & b
^ =	a ^ = b	a = a ^ b
| =	a | = b	a = a | b

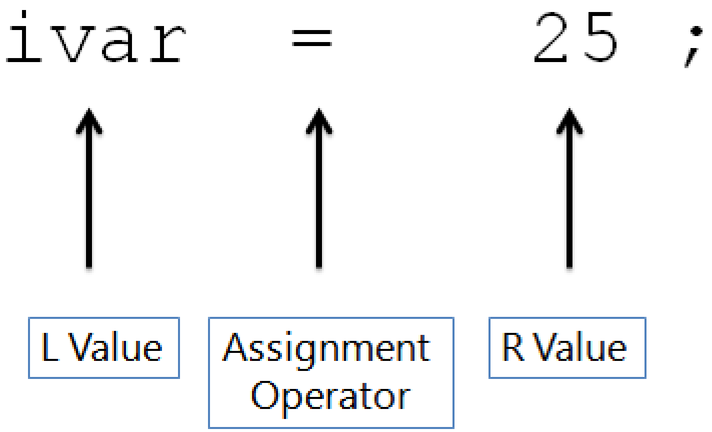
2. 6 L-Value and R-Value of Expression

a) L-Value stands for **left value**

- L-Value of Expressions refer to a memory locations
- In any assignment statement L-Value of Expression must be a container(i.e. must have ability to hold the data)
- Variable is the only container in C programming thus L Value must be any Variable.
- L Value cannot be a Constant, Function or any of the available data type in C

2. 6 L-Value and R-Value of Expression Contd…

- ❑ Diagram Showing L-Value of Expression :



ivar = 25 ;

L Value

Assignment Operator

R Value

2. 6 L-Value and R-Value of Expression Contd…

```
#include<stdio.h>
int main( )
{
      int num;
      num = 5;
      return(0);
}
```

Example of L-Value Expression

```
#include<stdio.h>
int main( )
{
      int num;
      5 = num; //Error
      return(0);
}
```

L-value cannot be a Constant

```
#include<stdio.h>
int main( )
{
      const num;
      num = 20; //Error
      return(0);
}
```

L-value cannot be a Constant Variable

2. 6 L-Value and R-Value of Expression Contd…

```
#include<stdio.h>
#define MAX 20
int main( )
{
    MAX = 20; //Error
    return(0);
}
```

L-value cannot be a MACRO

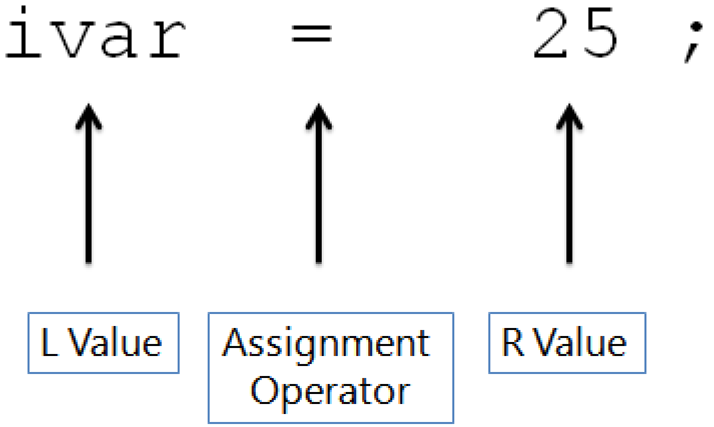
```
#include<stdio.h>
enum {JAN,FEB,MARCH};
int main( )
{
    JAN = 20; //Error
    return(0);
}
```

L-value cannot be a Enum Constant

2. 6 L-Value and R-Value of Expression Contd…

b) R Value stands for **Right value** of the expression

- ❑ In any **Assignment statement** R-Value of Expression must be anything which is capable of returning Constant Expression or Constant Value



```
ivar  =   25 ;
```

L Value

Assignment Operator

R Value

2. 6 L-Value and R-Value of Expression Contd…

Examples of R-Value of Expression	
Variable	Constant
Function	Macro
Enum Constant	Any other data type

- R value may be a Constant or Constant Expression
- R value may be a MACRO
- R Value may be a variable

2. 7 Control Statements

- Also called as Conditional Statement
- Decides order of execution based on conditions
- Helps repeat a group of statements
- Modifies control flow of program
- Decision Making
- Branching

2. 7 Control Statements Contd...

❑ ***Types of Branching Statements***

a) if statement

i. Simple if

ii. if…else statement

iii. nested if…else statement

iv. else…if statement

b) switch statement

c) goto statement

2. 7 Control Statements Contd...

a) if statement

- Condition “True" - Statement block will be executed
- Condition “False“ - Statement block will not be executed.
- **Variations**

  i. Simple if

  ii. if…else statement

  iii. nested if…else statement

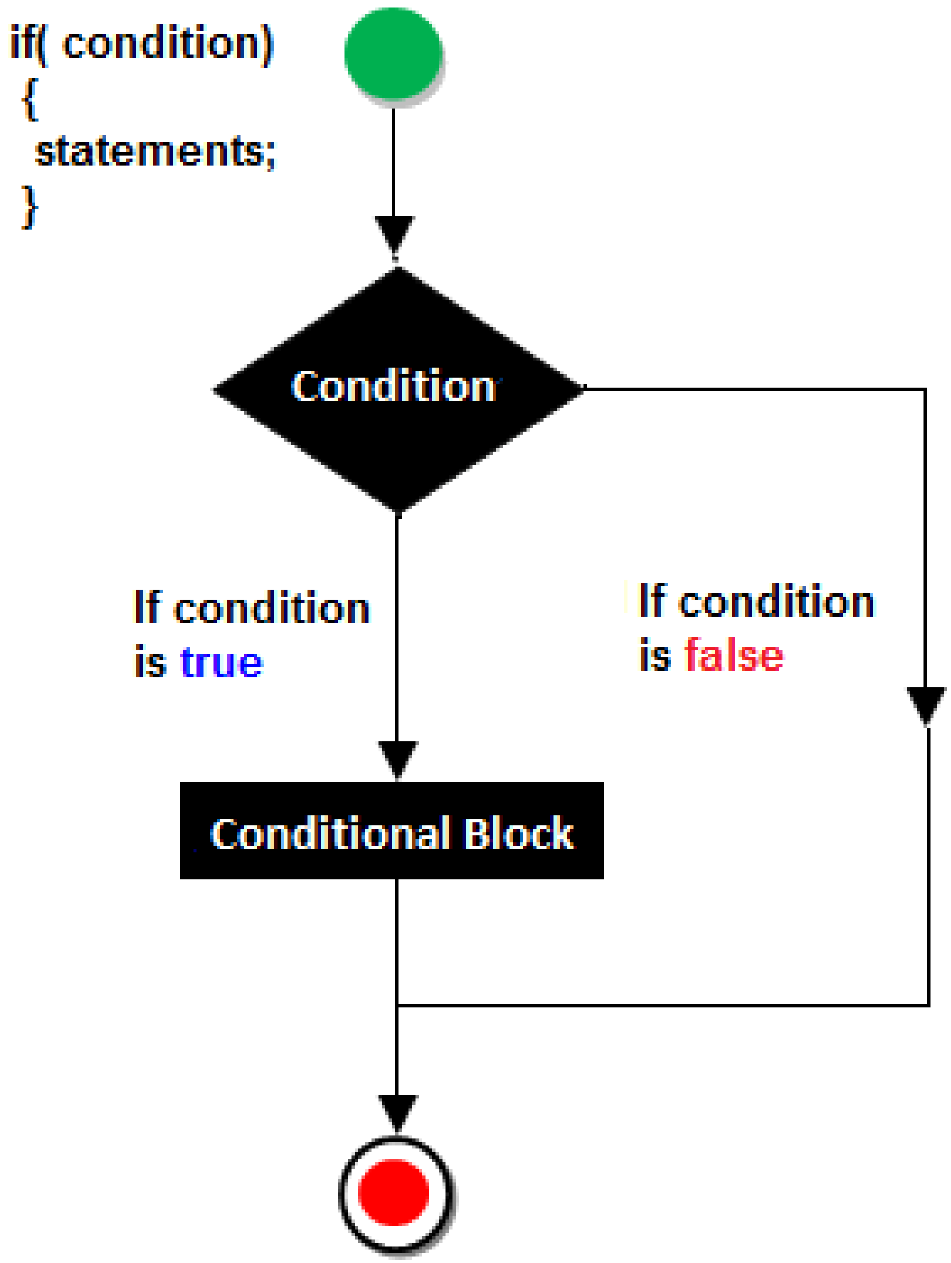
  iv. else…if statement

2. 7 Control Statements Contd…

i. Simple if statement

- Basic if statement
- What is a condition?
- Executes statement block only if condition is true
- **Syntax**

```
if (condition)
{
        Statements;
}
```



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

Condition

If condition is true

If condition is false

Conditional Block

/* Simple if – Program to check whether a number is Odd*/

```
#include<stdio.h>
int main( )
{
    int number;
    printf(“Enter the Number: ”);
    scanf(“%d, &number);
    if(number%2==0)
    {
        printf(“The Number is Even”);
    }
    return 0;
}
```

Output

```
Enter a value : 10342

The number is Even
```

2. 7 Control Statements Contd...

❑ **Try it Out Yourself ! Write a C program to:**

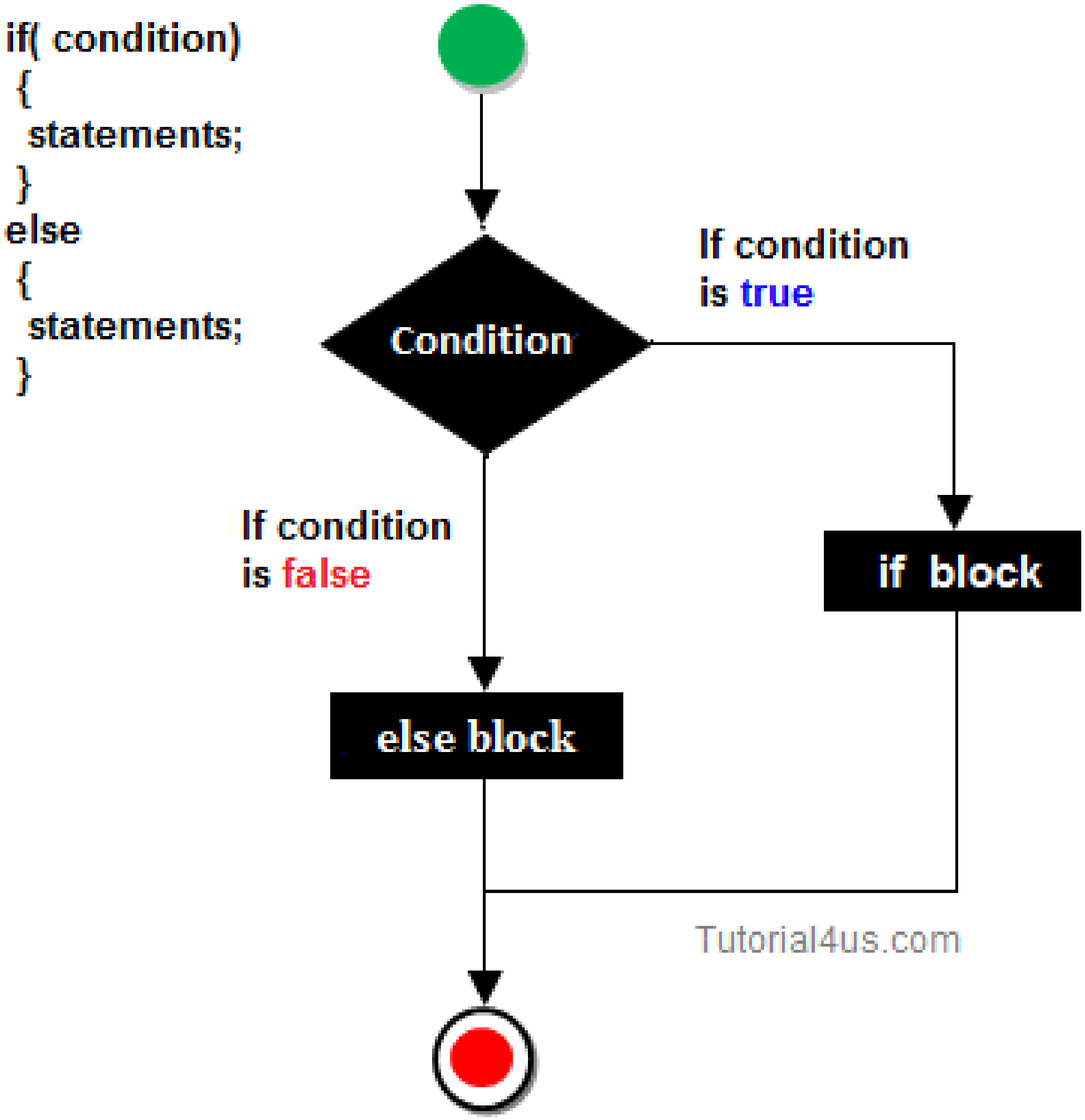
1) Check whether the given number is Even
2) To check whether the given number is Greater
3) To check whether the given number is Smaller
4) To check whether the given number is positive
5) To check whether the given number is negative
6) To check whether the given number is zero
7) To check whether two numbers are equal

2. 7 Control Statements Contd...

ii. If else statement

- Extension of basic if statement
- Takes care of True and False condition
- Number of Statement Blocks - 2
  - Block 1 – True Condition
  - Block 2 – False Condition

```
if (condition)
{
            Statements;
}
Else
{
            Statements;
}
```



```
if( condition)
 {
  statements;
 }
else
 {
  statements;
 }
```

/* if else –To check whether a number is Odd or Even*/

```
#include<stdio.h>
int main( )
{
    int number;
    printf(“Enter the Number: ”);
    scanf(“%d, &number);
    if(number%2==0)
    {
        printf(“The Number is Even”);
    }
    else
    {
         printf(“The Number is Odd”);
    }
    return 0;
}
```

Output 1

```
Enter the Number : 10341
The number is Odd
```

Output 2

```
Enter the Number : 10342
The number is Even
```

2. 7 Control Statements Contd...

- **Try it Out Yourself ! Write a C program to:**

    1) To check whether the given number is Greater or Smaller

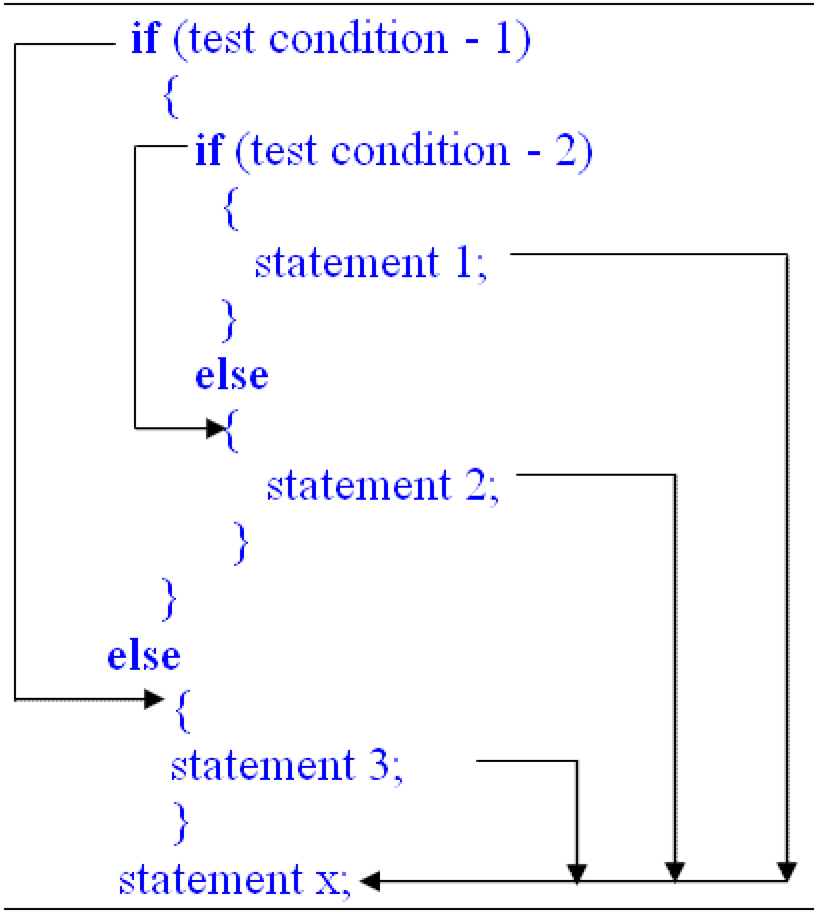
    2) To check whether the given number is +ve or -ve

    3) To check whether two numbers are equal or not

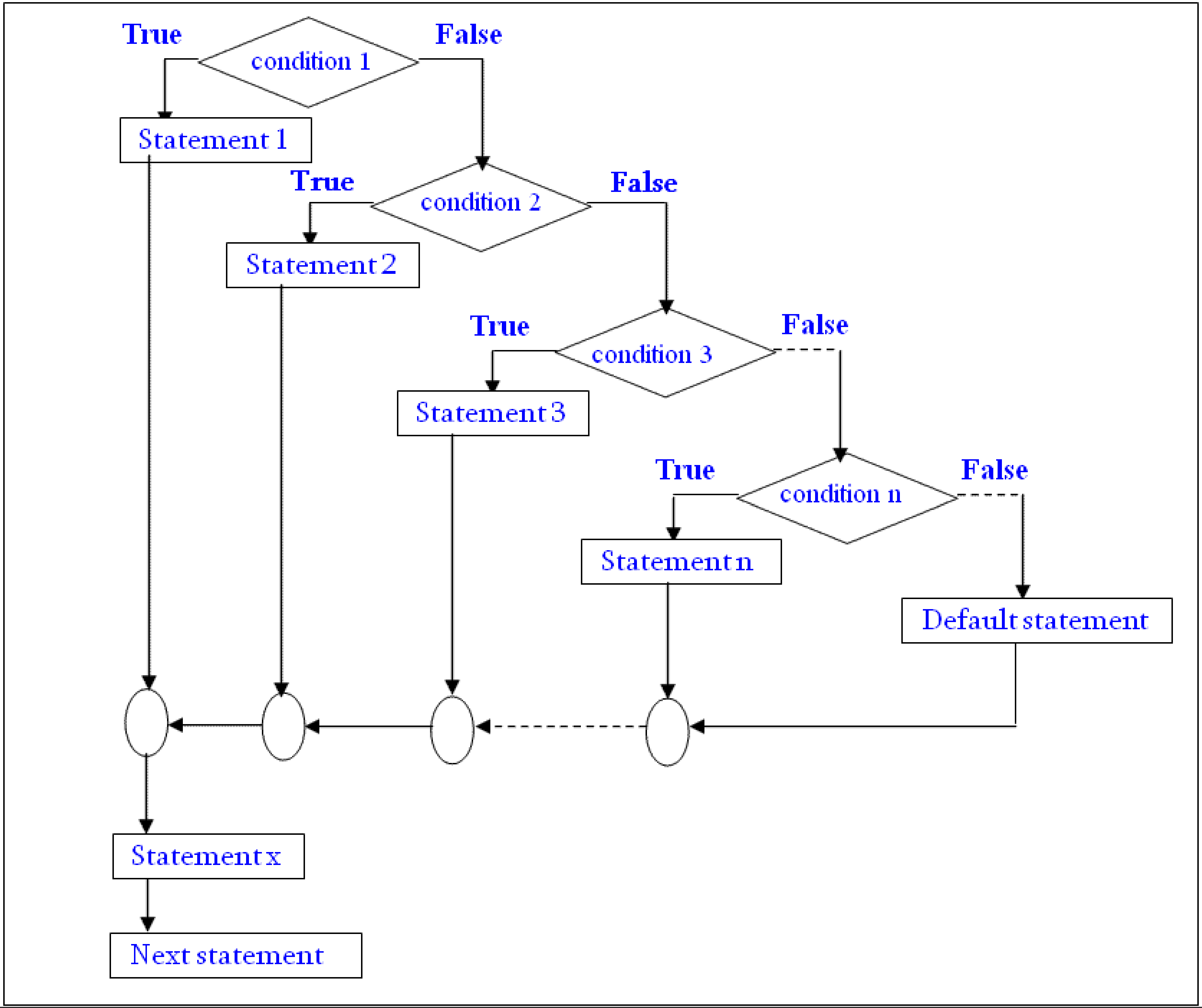
2. 7 Control Statements Contd...

iii. Nested if else statement

- ❑ Used when a series of decisions are involved
- ❑ Makes a choice between several alternatives
- ❑ New if else statement block is used within existing if else statement block



```
if (test condition - 1)
 {
   if (test condition - 2)
    {
      statement 1;
    }
   else
    {
      statement 2;
    }
 }
else
 {
  statement 3;
 }
statement x;
```


True
condition 1
False
Statement 1
True
condition 2
False
Statement 2
True
condition 3
False
Statement 3
True
condition n
False
Statement n
Default statement
Statement x
Next statement

/*Program for Nested if else */

```
#include <stdio.h>

void main( )

{

      char username;

      int password;

      printf("Username:");

      scanf("%c",&username);

      printf("Password:");

      scanf("%d",&password);
if(username=='a')
        {
                if(password==12345)
                {
                        printf("Login successful");
                }
                else
                {
                printf("Password is incorrect, Try again.");
                }
        }
else
        {
        printf("Username is incorrect, Try again.");
        }
return 0;
}
```

Output 1

```
Username: a
Password: 12345
Login Successful
```

Output 2

```
Username: a
Password: 54321
Password is incorrect, Try again.
```

Output 3

```
Username: b
Password: 54321
Username is incorrect, Try again.
```

2. 7 Control Statements Contd...

- ❑ **Step 1:** First if condition will be true, if the user has typed 'a' as a username then the program control moves to second if condition and checks for the password
  - ❑ if it true it will print 'login successful’
  - ❑ else it will execute block statement 'Password is Incorrect, Try again.‘
- ❑ **Step 2:** If the first if condition is false then it executes last else block thus printing 'Username is Incorrect, Try again.‘

2. 7 Control Statements Contd...

- □ **Step 3:** In this above example we have use username as single character to use multiple character username we need to use string data type

2. 7 Control Statements Contd...

b) Switch statement

- Allows to make decisions from a number of choices
- Also called as Switch-Case-Default Statement
- Faster than nested if else statement
- Easier to understand
- **Rules for writing switch ( ) statement**
  - Expression in switch must be an integer value or a character constant
  - No real numbers used in Expression

2. 7 Control Statements Contd…

- Each case block and default block must end with break statements
- Default is optional
- Case keyword must end with colon (:)
- Default may be placed anywhere in the switch
- No two case constants are identical

2. 7 Control Statements Contd…

```
switch(variable or expression)
{
    case constant 1:
            statements;
    break;
    ….
    case constant N;
            statements;
    break;
    default:
            statements;
}
```

/* Program for Switch Case*/

```
#include<stdio.h>
int main( )
{
    int a, b, choice;
    printf(“\nEnter Two Numbers: ”);
    scanf(“%d%d”, &a,&b);
    printf(“\n Enter 1 for Addition”);
    printf(“\n Enter 2 for Subtraction”);
    printf(“\n Enter 3 for Multiplication”);
    printf(“\n Enter 4 for Division”);
    printf(“ Enter your Choice”);
    scanf(“%d”,&choice);
```

```
switch (choice)
{
    case 1:
            printf(“Sum is : %d”, a+b);
            break;
    case 2:
            printf(“Difference is : %d”, a-b);
            break;
    case 3:
            printf(“Multiplication is : %d”, a*b);
            break;
    case 4:
            printf(“Difference is : %d”, a/b);
            break;
```

```
        default:
                printf(“Invalid Choice:”);
    }
getch( );
}
```

```
Enter two numbers
20
10
Enter 1 for Addition
Enter 2 for Subtraction
Enter 3 for Multiplication
Enter 4 for Division
Enter your Choice: 3
Product is : 200
```

2. 7 Control Statements Contd...

- **Nested Switch statement**
    - Inner switch( ) can be a part of an outer switch( )
    - Inner switch( ) and outer switch( ) case constants may be the same

/* Program for Nested Switch Case*/

```
#include<stdio.h>
int main( )
{
        int square, i, n, fact = 1,choice;
        printf(“\n Enter Any Number: ”);
        scanf(“%d”, &n);
        printf(“ 1. Square \n”);
        printf(“ 2. Factorial \n”);
        printf(“ 3. Find Odd or Even \n”);
        printf(“ 4. Exit \n”);
        printf(“ Enter your Choice”);
        scanf(“%d”, &choice);
```

```
switch (choice)
{
    case 1:
        square = n * n;
        printf(“The Square of the Given number is %d\n”, square);
        break;
    case 2:
        for(i=1;i<=n;i++)
        {
            fact = fact * i;
        }
        printf(“The Factorial of a given number is %d\n”, fact);
        break;
    switch (n%2)
        {
                case 0:
                printf(“Given Number is Even\n”);
                case 1:
                printf(“Given Number is Odd\n”);
        }
        case 3:
        exit(0);
        default:
        printf(“Invalid Choice. Please try again\n”);
    }
return 0;
}
```

```
Enter any number
5
1.   Square
2.   Factorial
3.   Find Odd or Even
4.   Exit
Enter your choice
2
The factorial of a given number is: 120
```

2. 7 Control Statements Contd...

c) The goto statement

- Transfers control from one point to another
- **Syntax**

```
goto label;
          statements;
          ..............
goto label
          statements;
```

2. 8 Looping Statements

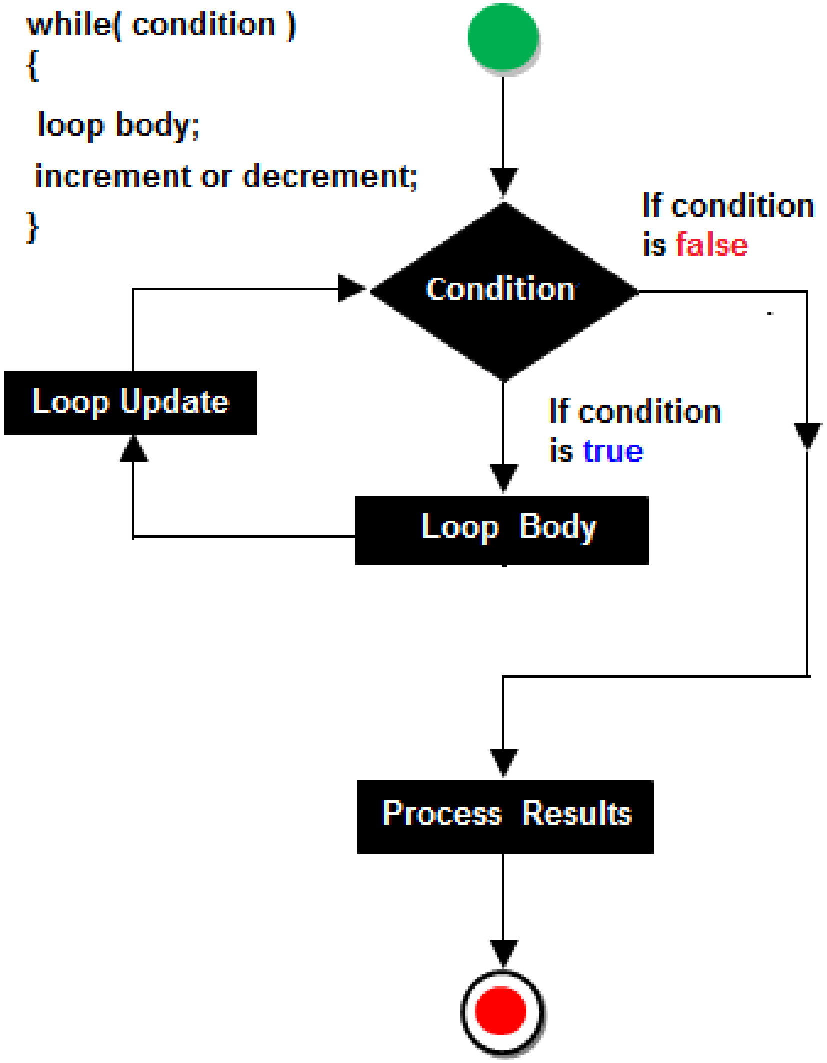
- Loop – A segment of the program that is executed repeatedly until a condition is satisfied
- Classification – Entry Controlled & Exit Controlled
- **Types**
    - a) while do loop
    - b) do while loop
    - c) for loop
        - i. Nested for loop

2. 8 Looping Statements Contd...

a) The While Loop

- ❑ Simplest looping structure in C
- ❑ Statements in the program may need to repeat for many times. e.g., calculate the value of *n*!
- ❑ Loop consists of two segments
  - ❑ Control Statement
  - ❑ Body of the Loop
- ❑ How while loop works?

```
while( condition )
{
 loop body;
 increment or decrement;
}
```



```
Initialize loop counter variable;

while (condition)

{

        Statements;

        increment / Decrement loop

                counter variable;

}
```

/* Program to Add 3 Numbers*/

```
#include<stdio.h>
int main( )
{
     int a, b, c, sum;
    printf(“\n Enter the Three Numbers: ”);
    scanf(“%d%d%d”, &a,&b,&c);
    sum = a+b+c;
    printf(“The sum of 3 Numbers is %d”, sum);
    return 0;
}
```

Output

```
Enter the Three Numbers: 10 20 30

The sum of 3 Numbers is: 60
```

/* Program to Add n Numbers*/

```
#include<stdio.h>
int main( )
{
         int i=1,n, sum=0;
        printf(“\n Enter the value for n: ”);
        scanf(“%d”, &n);
        while (i<=n)
        {
                sum = sum + i;
                i++;
        }
        printf(“The sum of n Numbers is: %d”, sum);
        return 0;
}
```

Output

```
Enter the value for n: 5

The sum of n Numbers is: 15
```

2. 8 Looping Statements Contd...

❑ **Try it Out Yourself ! Write a C program to:**

1) To print all even numbers from 1 to 100
2) To print all even numbers from 1 to n
3) To print table for any number
4) To calculate the sum of its digits
5) To check whether the entered number is Prime or not
6) To get a number as input and print it in reverse.
7) To check whether the number is Armstrong number

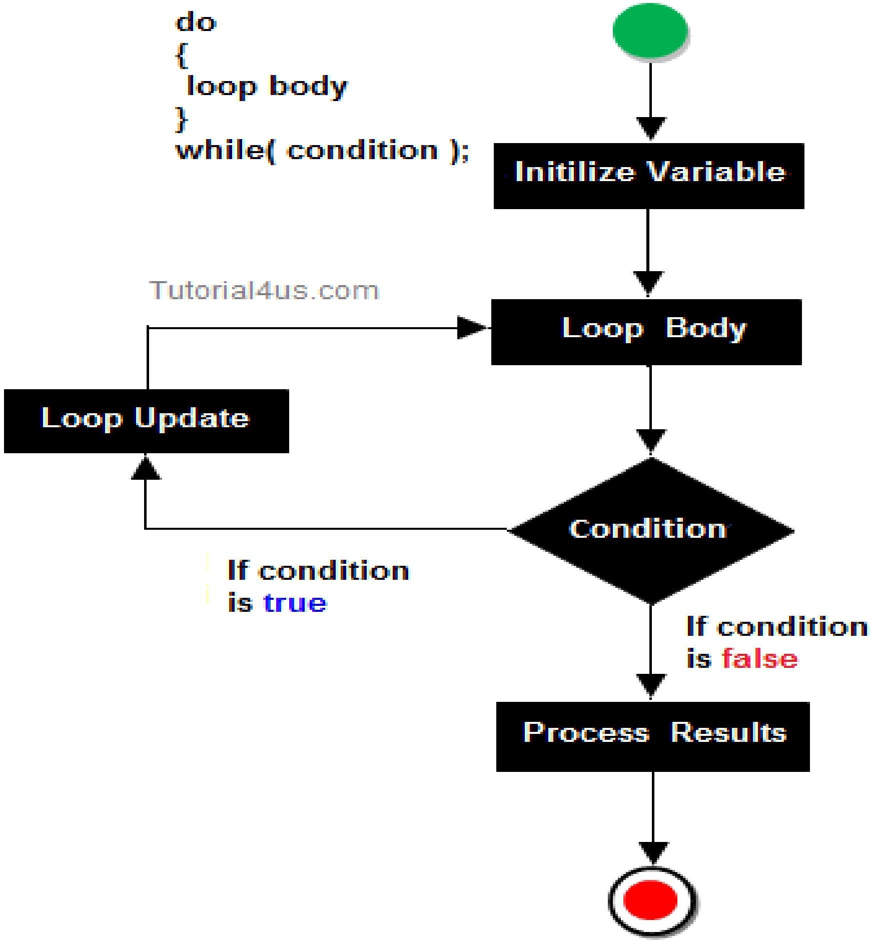
2. 8 Looping Statements Contd…

b) The Do While Loop

❑ The body of the loop is executed at least once

❑ **Syntax**

```
do
{
        statements;
}
while (condition);
```



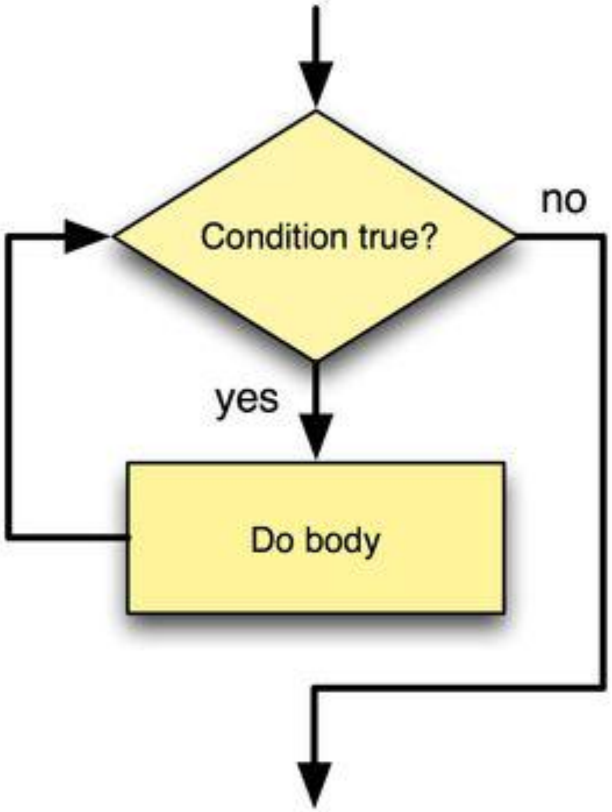
```
do
{
 loop body
}
while( condition );
```

```
Initialize loop counter variable;

do

{

        Statements;

        increment / Decrement loop

                counter variable;

}

while (condition)
```

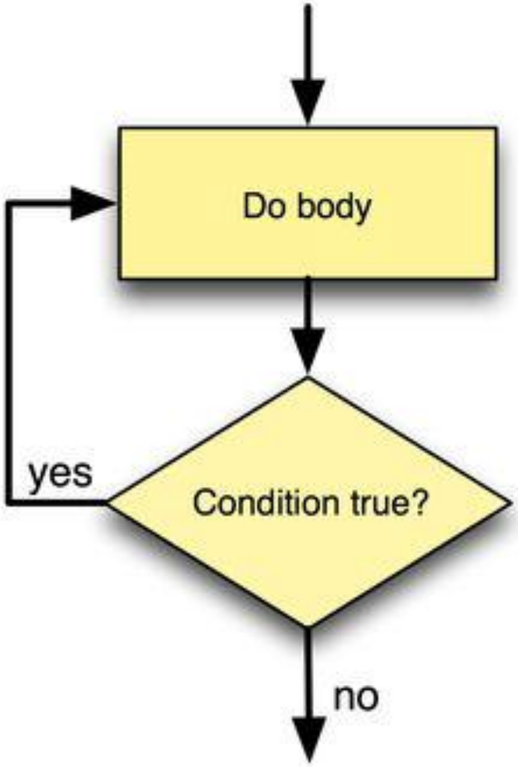
2. 8 Looping Statements Contd…

While Do Loop	Do While Loop
Entry Controlled Loop	Exit Controlled Loop
Test condition is checked before body of the loop is executed	Test condition is checked after the body of the loop is executed
Loop will not be executed if condition is false	Loop will be executed at least once even if condition is false
Top tested loop	Bottom tested loop

2. 8 Looping Statements Contd...



while flowchart



do/while flowchart

2. 8 Looping Statements Contd...

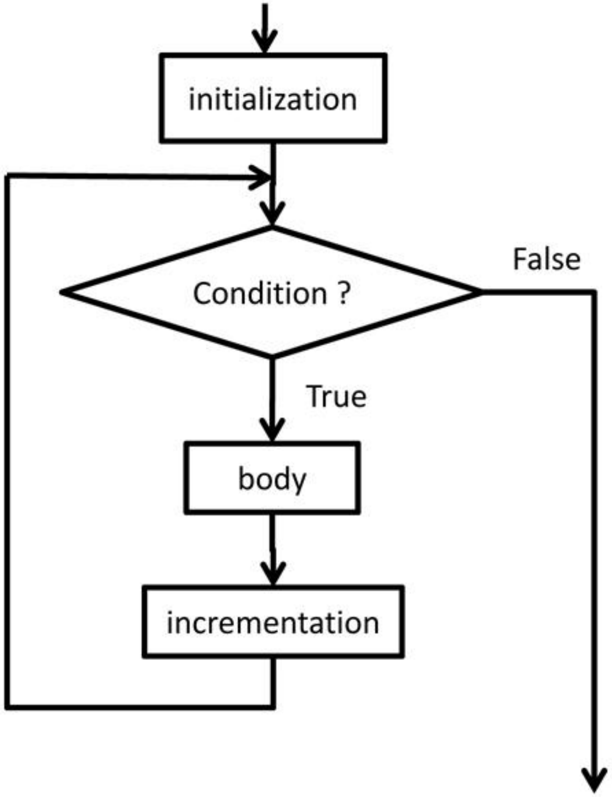
c) The for loop

- ❑ Most commonly and popularly used loop structure
- ❑ Structure of the for loop
  - ❑ Initialize loop counter variable
  - ❑ Check for condition
  - ❑ Increment / Decrement the loop counter variable
- ❑ **Syntax**

  for(initialization; condition; increment / decrement)

2. 8 Looping Statements Contd…

```
for( initialization; condition; incrementation )
        body;
```



2. 8 Looping Statements Contd...

❑ Examples

```
i.    for(i = 0; i < n; i++)
              {
                      Statements;
              }
ii.   for(count = 0; count > n; count--)
              {
                      Statements;
              }
```

/* Program to Add n Numbers using for loop */

```
#include<stdio.h>
int main( )
{
        int i, n, sum=0;
        printf(“\n Enter the value for n: ”);
        scanf(“%d”, &n);
        for (i =1; i<=n; i++)
        {
                sum = sum + i;
        }
        printf(“The sum of n Numbers is: %d”, sum);
        return 0;
}
```

Output

```
Enter the value for n: 5
The sum of n Numbers is: 15
```

2. 8 Looping Statements Contd...

- **Try it Out Yourself ! Write a C program to:**

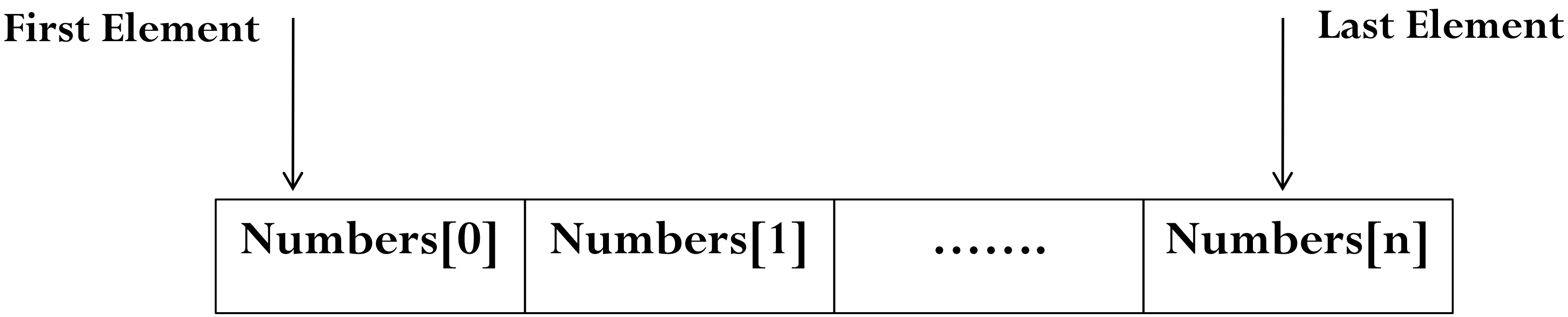
1) To print all even numbers from 1 to 100
2) To print all even numbers from 1 to n
3) To print table for any number
4) To calculate the sum of its digits
5) To check whether the entered number is Prime or not
6) To get a number as input and print it in reverse.
7) To check whether the number is Armstrong number

2. 9 Arrays

❑ Definition

An array is defined as **finite ordered collection of homogenous** data, stored in contiguous memory locations.

- ✓ Array is used to store a collection of data
- ✓ Array is a collection of variables of the same type.



First Element

Last Element

Numbers[0]	Numbers[1]	…….	Numbers[n]

2. 9 Arrays Contd...

- □ Need for Arrays
    - □ Used to represent a list of numbers / names
    - □ Used to represent tabular data in 2, 3 or more dimensions
    - □ Important Data Structure in any programming language
- □ **Definition**
    - □ Collection of elements of similar data types
    - □ Each element is located in separate memory locations
    - □ Each Array element share a common name

2. 9 Arrays Contd…

Characteristics of Arrays

- All elements in the arrays share a common name
- Elements distinguished by index number
- Index (or) element number of an array plays vital role for calling each element
- Specific array elements can be modified
- Value of array element can be assigned to variables
- Array elements stored in continuous memory locations

2. 9 Arrays Contd...

- Storage space for array depends on its data type and size

Total bytes = sizeof (Data type) x Size of Array

  - Example

    int a [5];

    Total bytes = sizeof (int) x 5 = 2 x 5 = 10 bytes

2. 9 Arrays Contd...

a) Array Declaration

- Syntax

Datatype arrayname [size/subscript];

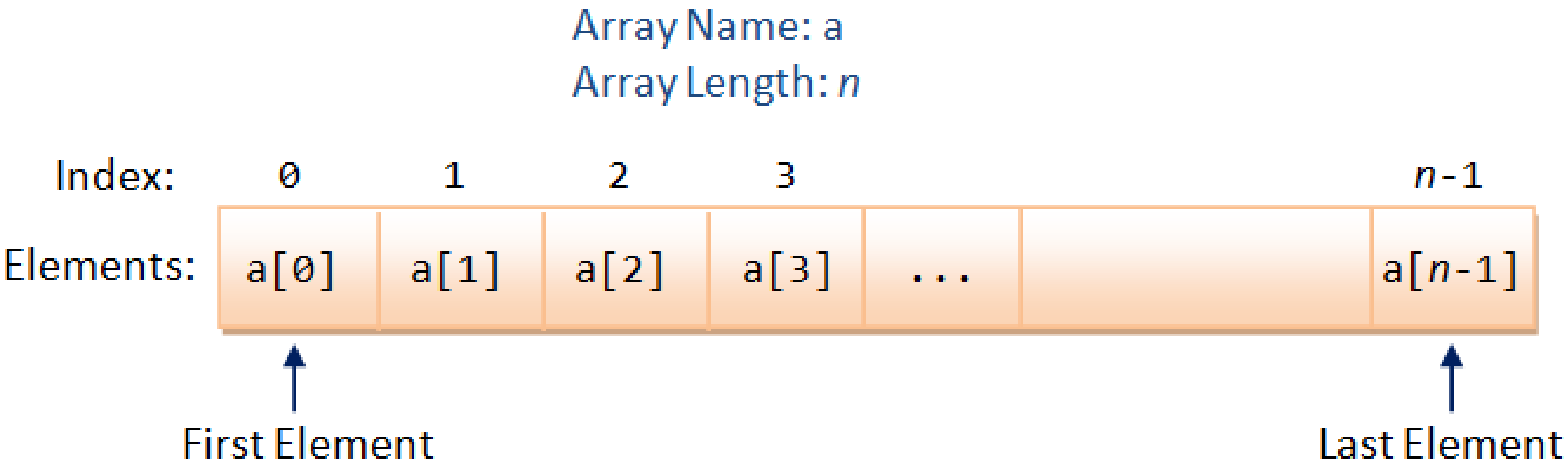
- **Data Type:** int, float, double, char, structure, union
- **Array Name:** Name given to the Array variable
- **Size / Subscript:** Number of values an Array can hold
- **Examples**

```
int numbers[5];        float marks[50];
char name[20];         double a[i];
```

2. 9 Arrays Contd...

- Illustration

int a[n];



Array Name: a
Array Length: *n*

Index:	0	1	2	3			*n*-1
Elements:	a[0]	a[1]	a[2]	a[3]	...		a[*n*-1]

First Element

Last Element

2. 9 Arrays Contd...

- ***Static Array:*** Array size (range) declared in the program
- ***Dynamic Array:*** Array size given during execution

STATIC ARRAYS	DYNAMIC ARRAYS
Range / Size of an array included in the Array definition	Range / Size of an array not included in the Array definition
Static Arrays cannot be changed	Dynamic Arrays can be changed

2. 9 Arrays Contd...

b) Array Initialization

- ❑ **Initialization:** Assigning values to array elements
  - ❑ Values specified in curly braces separated by commas
- ❑ Examples

```
int a[ 5] = {1, 2, 3, 4, 5};
float  b[3] = { 40.5, 59.0, 98.5};
char name[6] = ” SRMIST”;
```

- ❑ Array element index start from 0

2. 9 Arrays Contd...

- Array elements are called by array names followed by the element numbers
- int a[5] = {1, 2, 3, 4, 5};

  a[0] refers to 1st element i.e., 1

  a[1] refers to 2nd element i.e., 2

  a[2] refers to 3rd element i.e., 3

  a[3] refers to 4th element i.e., 4

  a[4] refers to 5th element i.e., 5

2. 9 Arrays Contd…

c) Getting Input for Arrays

- Use for loops to get input in arrays
- Use for loops with regard to the Array’s dimension
  - Input for One Dimensional Arrays – 1 for loop

```
for(i = 0; i < 5; i++)
{
        scanf(“%d”, &a[i]);
}
```

2. 9 Arrays Contd...

- Input for Two Dimensional Arrays – 2 for loops

```
for(i=0;i<5;i++)
     {
               for(j=0;j<5;j++)
               {
                         scanf(“%d”,&a[i][j]);
               }
     }
```

2. 9 Arrays Contd...

d) Printing Output in Arrays

- □ Use for loops to print array output
- □ Use for loops with regard to the Array’s dimension
  - □ Printing One Dimensional Array Output – 1 for loop

```
for(i=0;i<5;i++)
{
        printf(“%d”,a[i]);
}
```

2. 9 Arrays Contd...

- Printing Two Dimensional Array Output – 2 for loops

```
for(i = 0; i <  5; i++)
     {
               for(j=0; j < 5; j++)
               {
                         printff("%d", a[i][j]);
               }
     }
```

/* Program 1 : Array Declaration & Initialization*/

```
#include<stdio.h>
int main( )
{
    int i, arr[5];
    arr[0] = 10;
    arr[1] = 20;
    arr[2] = 30;
    arr[3] = 40;
    arr[4] = 50;
    for(i=0; i<=n; i++)
    {
        printf(“%d”\n, a[i]);
    }
    return 0;
}
```

Output

```
10
20
30
40
50
```

/* Program 2 : Array Declaration & Initialization*/

```
#include<stdio.h>
int main( )
{
    int i, arr[5];
    arr[5] = {10, 20, 30, 40, 50};
    for(i=0; i<=n; i++)
    {
        printf(“%d”, a[i]);
    }
    return 0;
}
```

Output

```
10
20
30
40
50
```

/* Program 3 : Array Declaration & Initialization*/

```
#include<stdio.h>
int main( )
{
    int i, n, arr[5];
    scanf(“%d”, &n);
    printf(“Enter the Elements of Array\n”);
     for(i=0; i<n; i++)
    {
        scanf(“%d”, &a[i]);
    }
    printf(“The Elements of the Array are”\n”);
    for(i=0; i<n; i++)
    {
        printf(“%d”, a[i]);
    }
    return 0;
}
```

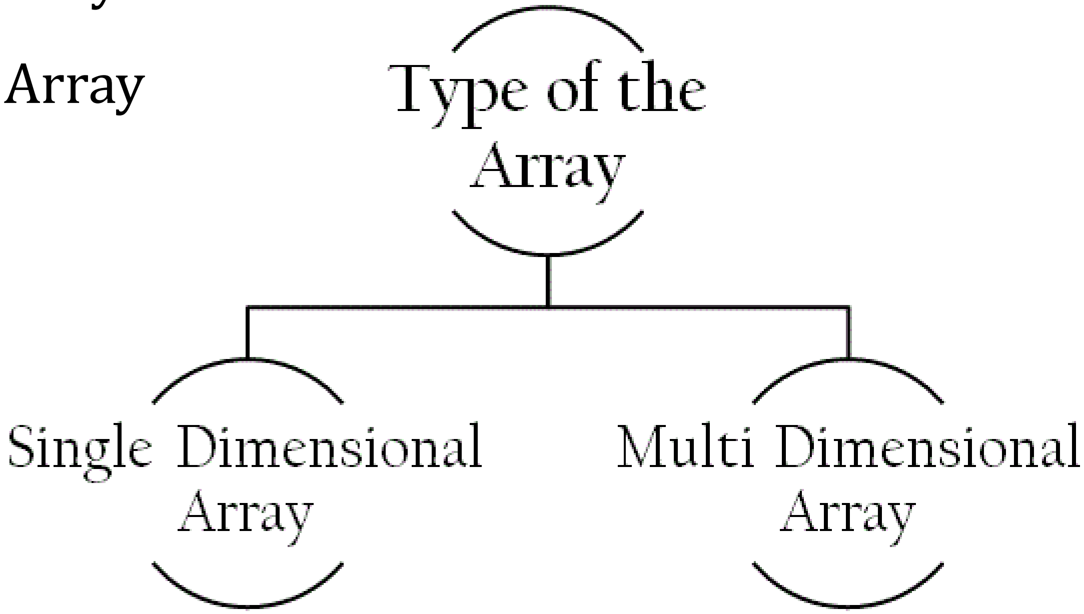
Output

```
Enter the Elements of the Array
10 20 30 40 50
The Elements of the Array are
10 20 30 40 50
```

2. 9 Arrays Contd...

e) Classification of Arrays

- i. One-Dimensional Array
- ii. Two-Dimensional Array
- iii. Multi-Dimensional Array



2. 9 Arrays Contd...

i. One Dimensional Array

- Data stored under a single variable using one subscript
- 1-D Array Declaration – Syntax

  datatype arrayname [size/subscript];

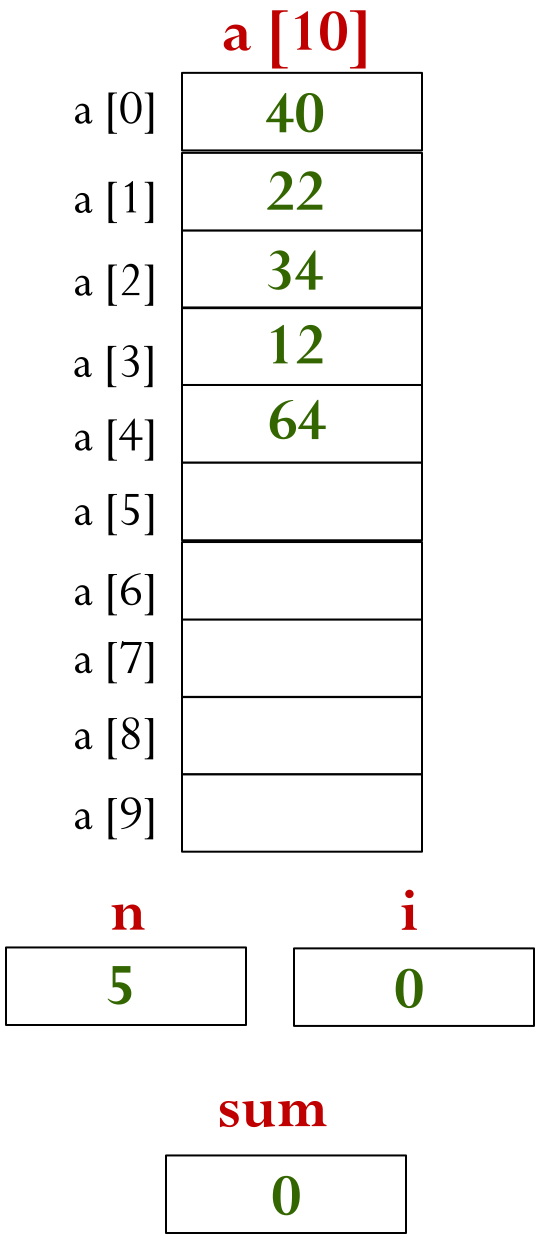
  - **Example:** int a [5];
- 1-D Array initialization – Syntax

  datatype arrayname [size] = { list of values};

  Example: int a [5] = { 10, 20, 30, 40, 50};

/* Program 1 : One Dimensional Array*/

```
#include<stdio.h>
int main ( )
{
int a[10], n, i, sum;
clrscr( );
printf(“Enter the Number of Elements\n”);
scanf(“%d”, &n);
for(i = 0; i <  n; i++)
{
      scanf(“%d”, & a [i]);
}
sum = 0;
for(i = 0; i < n; i++)
```



a [10]

a [0] 40
a [1] 22
a [2] 34
a [3] 12
a [4] 64
a [5]
a [6]
a [7]
a [8]
a [9]

n 5

i 0

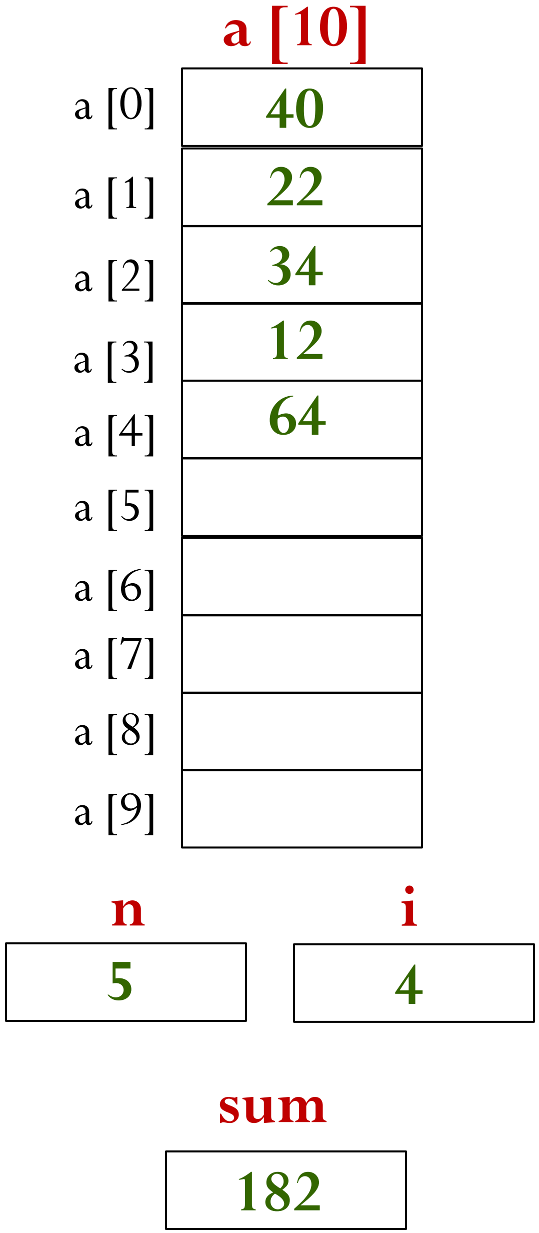
sum 0

/* Program 1 : One Dimensional Array*/

```
        {
                sum = sum + a[i];
        }
printf(“The Sum is: %d”, sum);
return 0;
}
```

Output

```
Enter the Number of Elements
5
40 22 34 12 64
The Sum is 182
```



a [10]

a [0]	40
a [1]	22
a [2]	34
a [3]	12
a [4]	64
a [5]	
a [6]	
a [7]	
a [8]	
a [9]	

n 5

i 4

sum 182

/* Program 2 : 1-D Array for Sorting*/

```
#include<stdio.h>
int main( )
{
        int i, j, temp, n, a[10];
        printf(“Enter the Number of Elements:”);
        scanf(“%d”, &n);
        printf(“Enter the Elements to be Sorted\n”);
        for(i=0;  i<n;  i++)
        {
                scanf(“%d\n”, &a[i]);
        }
        for(i=0;  i<n;  i++)
        {
                 for(j=i+1;  j<n; j++)
                {
```

```
                if(a[i] >a[j])
                {
                        temp = a[i];
                        a[i] = a[j];
                        a[j] = temp;
                }
        }
}
print(“The Sorted Elements are: \n”);
for(i=0;  i<n;  i++)
    {
        printf(“%d\n”, a[i]);
    }
return 0;
}
```

Output

```
Enter the Number of Elements:5
Enter the Elements to be Sorted
25
12
45
68
7
The Sorted Elements are:
7
12
25
45
68
```

THANK YOU