Course Code: ESC108A

Course Title: Elements of Computer Science and Engineering

Course Leaders:

Roopa G.

Ami Rai E.

Chaitra S.

Contents

Tutorial 1
Algorithms

- At the end of this tutorial, student will be able to
  - Write algorithms to solve simple problems
  - Identify algorithmic constructs such as *variables, begin, end, parameter list* and *data types*

Tutorial Questions

- Write an algorithm to
  1. Find sum of two numbers
  2. Find average of 3 numbers
  3. Find largest of three numbers
  4. Find factorial of a number
  5. Generate Fibonacci series till 1000

Tutorial 2
Input-Output Statements

- At the end of this tutorial, student will be able
  - Identify format specifiers used with integer, character, float and double data types for input and output
  - Identify library functions and format used to perform input/output operations

Tutorial Contents

- Integer Data Type-Format specifiers
- Character Data Type –format specifiers
- Floating Point Data Type-Format specifiers
- Library functions for input/output operations
- Format for input/output statements

Format Specifiers

- Integer Data Type

  %d : signed decimel integer

  %ld : long integer

  %hd : short integer

  %i : signed decimal, octal or hexadecimal integer

  %u : unsigned decimal integer

  %x or %X : hexadecimal integer

  %o : octal integer

- Character Data Type

  %c : single character

  %s : string

Format Specifiers contd.

- Floating Point Data Type

  %e or %E : a floating point value in exponential notation

  %f : floating point values in fixed-point notation

  %g or %G : a floating-point value in either the floating-point form f or the exponential form e (or E), based on the magnitude of the value

  %l or %L : double

Library Functions

- C provides a library function to perform input/output operations – stdio
- Header file containing such library functions – stdio.h
- Some functions
  - printf()
  - scanf()
  - getchar()
  - putchar()
  - gets()
  - puts()

Input-output Statements

- Formatted I/O statements
  - Enable the user to specify the type of data and the way in which it should be read in or written out
  - Example : printf(), scanf()

- Unformatted I/O statements
  - Do not specify the type of data and the way in which it should be read in or written out
  - Example : getchar(), gets()

Formatted Input

- To read the value for the variable in a program from the keyboard use

    scanf()

- General format

    Scanf("control string", address_list);

- Example:

    scanf("%d%f", &x ,&y); /*make sure to use & before variable */

Input Formats

- scanf(“%3d”, &num);
  - If the input is 12345, then only 123 is stored in the memory location of num

scanf(“%d %d”, &num1, &num2);
  - Programmer has to input one extra character to end the input operation

scanf(“%d %*d %d”, &a, %b, &c);
  - If input is 10 20 30, then a=10, 20 is skipped and c=30

scanf(“%5f”, &n);
  - If input is 12.3456, then 12.34 is stored in the memory

Formatted Output

- C provides printf() function to display data on the monitor included in stdio.h
- General format

```
printf(“control string”,var_list);
```

- Examples:

```
printf(“C Programming”;
printf(“%d”,num);
printf(“Product is %f”,result);
printf(“value1=%d value2=%d”,num1,num2);
```

Output Formats

- printf(“%d”, num);
  - Print the value stored in ‘num’
- printf(“%8.4f”, &num1);
  - Print the number having a maximum of 8 characters including a decimal point and there must be 4 digits after the decimal point
- printf(“%5d”, &n);
  - Width of the output will be 5

Unformatted Input

getchar()

- Reads a single character from the standard input device
- In stdio.h
- No parameter within the parenthesis

- Syntax

```
var=getchar();          //var is a character type variable
```

- Example:

```
main(){
        char ch;
        ch=getchar();
}
```

Unformatted Output

putchar()

- Prints a single character on the screen
- In stdio.h

- Syntax

```
putchar(var);        //var is a character type variable
```

- Example:

```
main(){
        char ch;
        putchar(ch);
}
```

Tutorial Questions

Write C program to

1. Print your address
2. Print “Hi! How are you?”
3. Print the integer entered by the user
4. Find the sum and product of two numbers
5. Find the area and circumference of a rectangle
6. Calculate simple interest
7. Convert temperature from Fahrenheit to Celsius

Tutorial 3
Operators

- At the end of this tutorial, student will be able to
  – Develop programs that require *operators*
  – Identify and use algorithmic, flowchart and C programming constructs for choice

Operators and Operands

- Expressions consist of variables, operators and operands

  a = b + 5

- Operator: Symbol representing the operation

  = and +

- Operand: The data items (variables and constant) on which the operation is performed

  a, b and 5

  - In case of + operator, operands are b and 5
  - In case of = operator, operands are a and the value of expression b+5

Different Types of Operators

- Assignment operators
- Arithmetic operators
- Increment and decrement operators
- Relational operators
- Logical operators
- Bitwise operators
- Conditional operators

Tutorial Questions

Write an algorithm and develop a program to

1. Demonstrate the working of
   - Increment and decrement operators
   - Logical operators
2. Compute quotient and remainder
3. Extract last two digits of the given year
4. Find the average of three numbers
5. Swap two numbers Without using third variable
6. Check the given number is odd or even using conditional operator
7. Multiply a number by two using shift operator

Tutorial 4
Condition Statements

- At the end of this tutorial, student will be able to
  - Develop programs that require *choice*
  - Identify and use algorithmic, flowchart and C programming constructs for choice

Types of Control Structures

- Condition statements
  - if statement
  - if-else statement
  - switch-case statement

- Iteration statements
  - for statement
  - while statement
  - do-while statement

Tutorial Questions

Write an algorithm and develop a program to

1. Check given number is positive or not
2. Check given number is odd or even
3. Check a given number is divisible by 5 or not
4. Find the maximum of two numbers
5. Find the biggest of given three integer numbers
6. Check the given year is leap year or not
7. Accept a month in digits from the user and Display the month in words. If number is not between 1 and 12 display message “invalid month” using switch statement.
8. Enter 6 marks, find average and grade the students

Tutorial 5
Iteration Statements

- At the end of this tutorial, student will be able to
  - Develop programs that require *iteration*
  - Identify and use algorithm, flowchart and C programming constructs for iteration

Types of Control Structures

- Condition statements
  - if statement
  - if-else statement
  - switch-case statement

- Iteration statements
  - for statement
  - while statement
  - do-while statement

for, while and *do-while*

for statement

```
        for (i=0; i<10; i++)
                printf(“hi”);
```

while statement

```
i=0;
while(i<10){
   printf(“hi”);
   i++;
}
```

do-while statement

```
i=0;
 do{
        printf(“hi”);
        i++;
 } while(i<10);
```

Tutorial Questions

Write an algorithm and develop a program to

1. Print numbers from 1 to 100 using
   - for
   - while
   - do-while
2. Print even numbers from 2 to 100 (without using % operator)
3. Print multiplication table
4. Print the pyramid

```
*
* *
* * *
```

Tutorial Questions contd.

Write an algorithm and develop a program to

5. Create a menu driven interface using do-while loop to find the sum of two numbers
6. Find factorial of a number
7. Check whether the number is Armstrong or not
8. Generate Fibonacci series
9. Reverse a given number
10. Generate value for cos(x), given that
    - cos(x) = 1-x2/2! + x4/4! - ...for all x

Tutorial 6
Arrays

- At the end of this tutorial, student will be able to
  - Develop programs that require arrays
  - Identify and use algorithm, flowchart and c programming constructs for arrays

One Dimensional Array

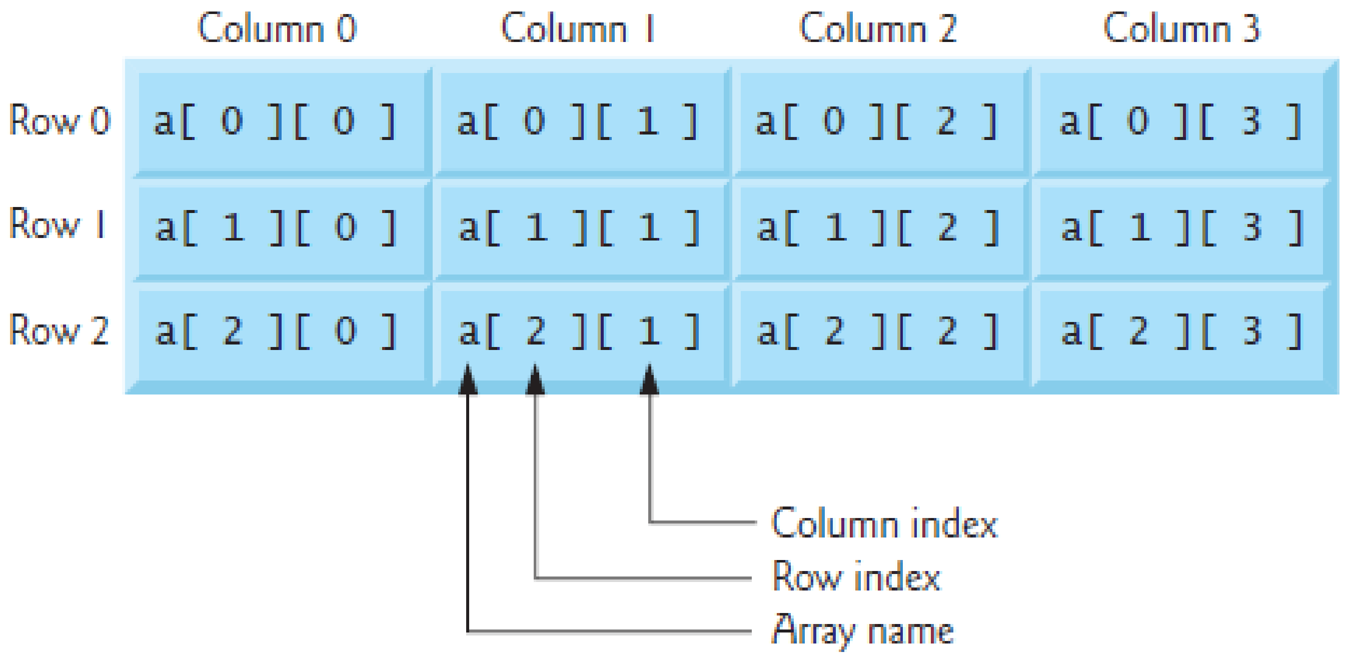
- Array - Collection of elements of same data type
- All these elements are stored in consecutive memory locations

- Example

```
int numArray[4]= {10,20,30,40}; /*4 is size. numArray[0]=10,...
                                   numArray[3]=40*/
  or
int numArray[4];
numArray[0]=10;
numArray[1]=20;
```

Two-Dimensional Arrays

- The array contains three rows and four columns, so it's said to be a 3-by-4 array
- 2D Array declaration - int a[3][4];



Tutorial Questions

- Write an algorithm and develop a program to
  1. Read and display the elements of 1D array
  2. Find the smallest element in an array. Use #define to define the size of the array
  3. Reverse the elements in an array
  4. Print alternate elements in an array
  5. Print even and odd elements in an array
  6. Replace the even elements in an array with zeros
  7. Delete an element from the array

Tutorial Questions contd.

- Write an algorithm and develop a program to

7. Read the elements of 2D array and display in matrix format
8. Find sum and subtraction of two matrices
9. Find the transpose of a given matrix
10. Multiply two matrices and
11. Find the sum of elements in the second row of a 4x4 matrix
12. Display the diagonal elements of a square matrix

13. Store values entered by the user in a three-dimensional array and display it

Tutorial 7
Strings

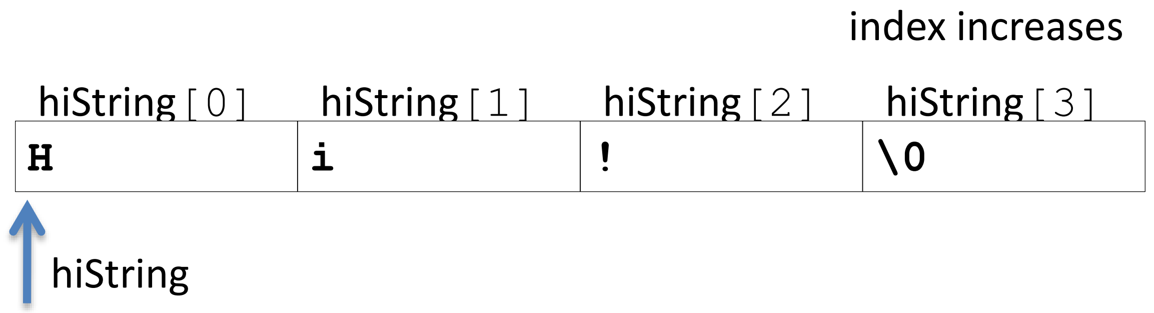
- At the end of this tutorial, student will be able to
  - Develop programs that require Strings
  - Identify and use algorithm, flowchart and c programming constructs for arrays

Strings

- Strings are 1-Dimensional ***char*** arrays
  - Always end with '\0' character

- Representation of string in memory

char hiString[4] = "Hi!"; //creates a 4-element array hiString containing the characters 'H', 'i', '!', and '\0



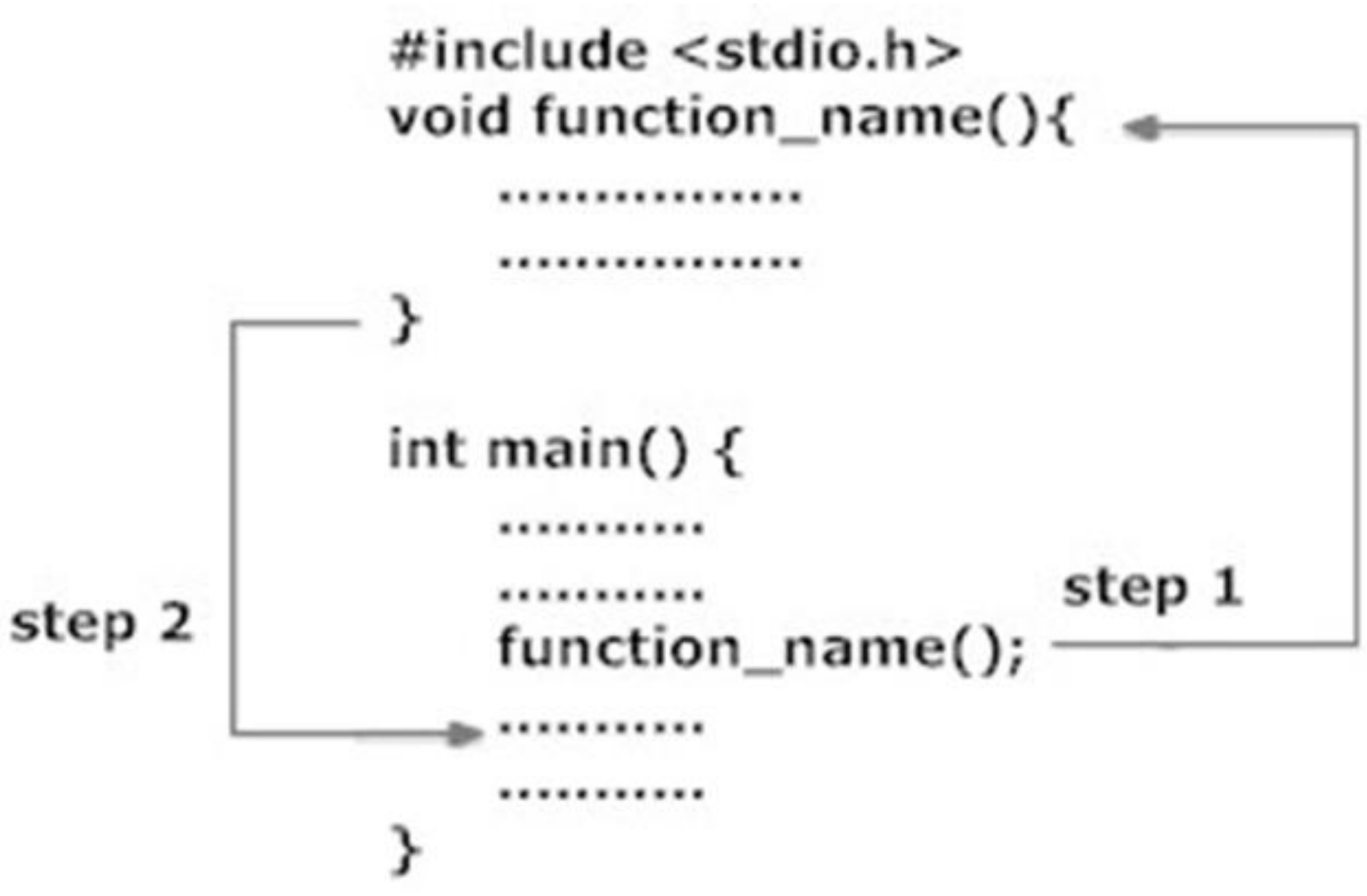
Tutorial Questions

- Write a program to
  1. Read and display a string using printf and scanf
  2. Read and display a string using gets and puts
  3. Check whether the given letter is lower case or not using built in function
  4. Demonstrate the working of String functions
     - strlen()
     - strcpy()
     - strcat()
     - strcmp()
  5. Count the number of vowels, consonants, digits, and spaces in a string
  6. Check if the Substring is present in the given String or not

Tutorial 8
Functions

- At the end of this tutorial, student will be able to
  - Develop programs that use functions

User Defined Function



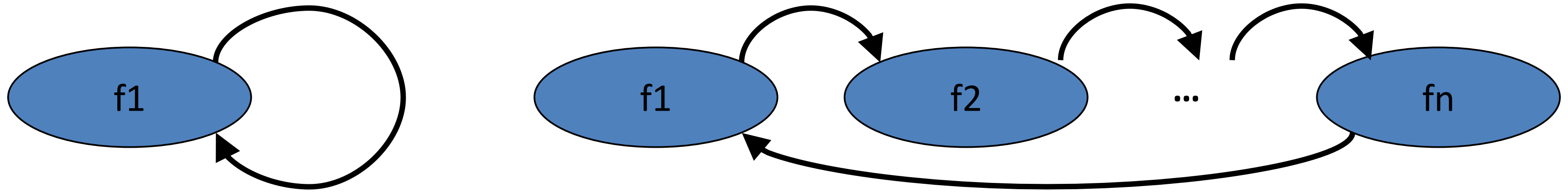
```
#include <stdio.h>
void function_name(){
        ................
        ................
}

int main() {
        ...........
        ...........
        function_name();
        ...........
        ...........
}
```

Recursion

- C functions can be used recursively
  - A function may call itself either directly or indirectly

- When a function calls itself recursively, each invocation gets a fresh set of all the automatic variables, independent of the previous step



Tutorial Questions

- Write a program to
  1. Find sum of two numbers using function
     - Pass arguments and return value
     - Pass arguments and don’t return value
  2. Find average of three numbers using function (Pass three numbers as argument)
  3. Check the given number is prime or not (Pass the number as argument and return the result)
  4. Find factorial of a number using function (Pass the number as argument)
  5. Convert binary number to decimal number using function (pass the binary number as the argument)

Tutorial Questions contd.

- Write a recursive function to
  6. Find the sum of first *n* natural numbers
  7. Find factorial of a number
  8. Generate the Fibonacci series upto a limit
  9. Reverse the given string

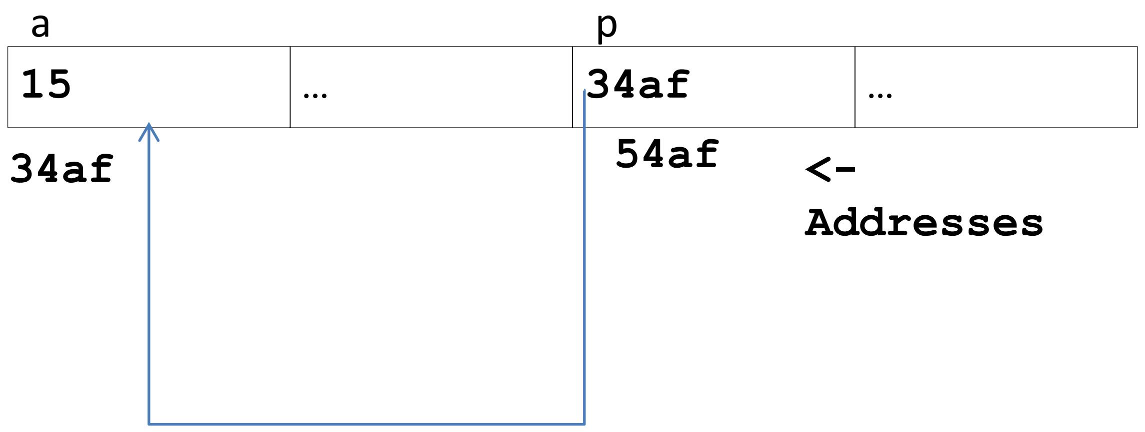
Tutorial 9
Pointers

- At the end of this tutorial, student will be able to
  - Develop programs that use pointers

Pointers

- Pointers hold the address of elements rather than a data value
- Example:

```
int  a = 15;
int *p ;
p = &a;
```



Tutorial Questions

- Write a program to
  1. Create, initialise, assign and access a pointer variable
  2. Change the value of an integer using pointers
  3. Initialise a pointer to the first element of array and access all the elements of array. Print the value of pointer and the elements of array using the pointer variable
  4. Compute sum of all elements stored in array using pointers
  5. Display the characters of a string using pointers
  6. Find the length of a string using pointers

Tutorial Questions contd.

- Write a program to

7. performs division operation on type int using generic pointers

   division (char type, void*operand1, void*operand2, void*result)

8. Demonstrate the working of
   - Call by value
   - Call by reference

Tutorial 10
Structures and Union

- At the end of this tutorial, student will be able to
  - Identify the operators used with structures
  - Develop programs with *structures* and *Union*

Structure

- Structure - Collections of related variables under one name

```
struct student{
        char name[10];
        char rollNum[10];
         int age;
 };
```

  - student is the structure name and is used to declare variables of the structure type
  - student contains three members
    - two members of type char - name and rollNum
    - a member of type int - age

Unions

- Derived data type
- Members of a union share the same storage space
  - Each member within a structure is assigned its own memory location
- Size allotted is the size of largest member
- Members can be of any type, but only one data member can be referenced at a time
- Only the last data member’s value can be accessed

Tutorial Questions

- Write a program to
  1. Store and Display the details of a student
     - Initialize the values
     - Take the details from the user
  2. Store and Display the details of 5 students
  3. Demonstrate the size of structure and union
  4. Add Two Complex Numbers by Passing Structure to a Function

Tutorial 11
Random Number Generation and Files

- At the end of this tutorial, student will be able to
  – Develop programs that random number generation
  – Develop programs using files in C

Random Number Generation

- Getting a random number

- Consider the following statement

  int i = rand();

➢ The function prototype for rand() is in <stdlib.h>

➢ The value produced directly by rand() are always in the range

  0 ≤ rand() ≤ RAND_MAX (constant in <stdlib.h>)

  – Standard C states that the value of RAND_MAX must be at least 32767, which is the maximum value for a 16-bit integer

File I/O functions

Prototype	Description
FILE	Data type (a structure) to store file information permanently
FILE *fopen(const char *path, const char *mode)	Opens a file and returns FILE pointer. Path is the path to the file. Mode can be combinations of “r”, “w” or “a”. Other possibilities include “+”.
int fclose(file *FP)	Closes a file that was already opened in the program
int fscanf(FILE *stream, const char *format,…)	Similar to scanf for file I/O
int fprintf(FILE *stream, const char *format,…)	Similar to printf for file I/O

File Opening Modes

File Mode	Description
r	Open a text file for reading
w	Create a text file for writing, if it exists, it is overwritten
a	Open a text file and append text to the end of the file
r+	Open a text file both in reading and writing mode
rb	Open a binary file for reading
wb	Create a binary file for writing, if it exists, it is overwritten
ab	Open a binary file and append data to the end of the file

Tutorial Questions

1. Write a program to
   - Generate one random number
   - Generate 10 random numbers
   - Generate 10 random numbers between 1 and 100
   - Generate 5 random numbers between 1000 and 2000

2. Write a C program to read a number from the keyboard and store it in a file “DATA.txt”. Read the number from the file “DATA.txt”, find its square and display on the monitor.

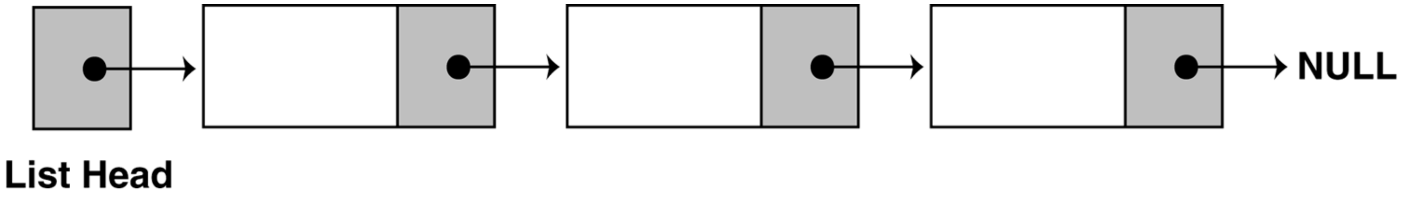
Tutorial 12
Linked List

- At the end of this tutorial, student will be able to
  - Develop programs that use List ADT using pointers

Linked List

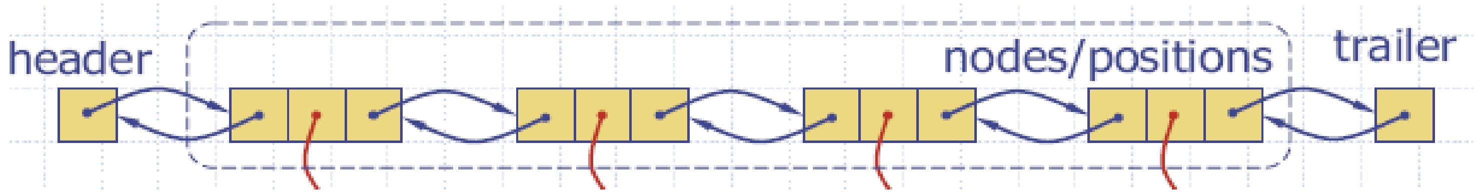
- A linked list can grow or shrink in size as the program runs
- Does not require the shifting of items during insertions and deletions

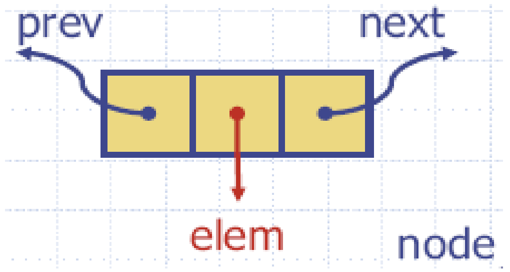
- A linked list is called "linked" because each node in the series has a pointer that points to the next node in the list



Doubly Linked List

- Doubly Linked List provides a better implementation of the Link ADT (than Singly Linked List)





Tutorial Questions

- Write a program to implement
  1. Singly linked list
  2. Doubly linked list