# Advanced Programming

Functions

# Topics

- Functions
- Parameter Passing to Functions
  - Call by value
  - Call by reference
  - Passing arrays to a function
- Returned values
  - Pointer
  - Struct
- Recursion

# Example

- Using arrays store polynomials
- Read two polynomials and add them into a third polynomial
- Solve the problem using dynamic allocation

# Function

- In general, **functions** are blocks of code that perform a number of pre-defined commands to accomplish a task.
- We can either use the built-in library **functions** or we can create our own **functions**.
- **Functions** that a programmer writes (user defined) require a prototype.

# Parameter Passing to a Function

- The number, type, and order of the parameters passed to a function should be declared in it prototype.

- Parameter names are local to the function
  - int  * Add(int a, int b);

# Call by Value

- In call by value only the value of the parameter is passed to the functions
- Any change in the parameter variable is local to the function (not visible from the calling function)

# Call by Reference

- In call by reference, the address of the variable is passed to the function
- Any change in the location referred to by the address is visible from the calling function
- The pointer variable is still local to the function

# Example

- Write a function to swap the values of two variables. Call the function from the main program.

- Write a function to get a struct as input. Discuss which fields are passed as call by reference

# Passing Arrays to Functions

- Array names are pointers to the first element of the array. Therefore, arrays are always passed as call by reference.

- It is not necessary to mention the number of elements of the **first dimension** in the prototype of the function

# Return Values

- A function can return only one value (or none)
- It is possible to return a struct as the return value of a function
- Functions can return pointers
  - float *CreateFloatArray( int size)

# Recursion

- A function is called a recursive function if:
  - It calls itself
  - It calls a function which in turn call the calling function
- A recursive function should have a condition to terminate the function

# Examples

- Write a function to find the sum of the number from 1 to n (n is given as a parameter)
- Write a function to find the factorial of a positive integer N

# Towers of Hanoi

- Assume N plates of different sizes are slid onto a rod  in the order of their size.  The goal is moving these plates to the second rod one plate at a time, so that the smaller plates are always on top of the large plates. A third rod can be used as auxiliary