# Data Structures

## Basic Concepts

Data structure: Logical model of a particular organization of data.

- **Arrays**
- **Stacks**
- **Queues**
- **Linked lists**
- **Binary trees**
- **Heaps**
- **Graphs**

# Assignment

- What is the appropriate structure for the order in which the elements are processed in the following situations:
  - Batch computer programs are submitted to the computer center.
  - Program A calls subprogram B, which calls subprogram C and so on.
  - Employees have a contract which calls for a seniority system for hiring and firing.
  - iTunes playlist.
  - Documents submitted to the printer for printing.
  - People issued ticket at the counter.
  - To find the distance from city X to city Y.
  - Train cars are in a specific order so that they may be loaded, unloaded, transferred, dropped off, and picked up in the most efficient manner possible.

# Assignment

- Scavenger hunt: You have a clue, and that clue has a pointer to place to find the next clue. So you go to the next place and get another piece of data, and another pointer. To get something in the middle, or at the end, the only way to get to it is to follow this list from the beginning.
- Waiting times of customers in McDonalds .
- Unix/Linux file hierarchy
- Water flowing in a pipe
- A lady wearing  bangles
- Dialing a toll-free number for your bank
- Vehicles on toll-tax bridge
- CD's in the case

- Batteries in the flashlight
- Luggage checking machine
- Clothes in the trunk
- Patients waiting outside the doctor's clinic
- pixels in a drawing created onscreen by the user at run-time.
- Multiplication table(10 X 10)
- Test scores in a class of 20 students.
- A data record containing an address and phone number that will be input at run-time
- Number of alphabets
- Available flight paths between pairs of destination cities
- chess/checkerboards

# Operations on a data structure

- **Inserting:**
  - adding a new record
- **Deleting:**
  - removing a existing record
- **Searching:**
  - finding the location of the record
- **Traversing:**
  - accessing each record
- **Sorting:**
  - arranging the records in some logical order
- **Merging:**
  - Combining the records in two different sorted files into a single sorted file.

# Algorithm

- A precise rule (or set of rules) specifying how to solve some problem.
- Set of instructions in a computer program.
- step-by-step description of a solution to the given problem.
- logic of a program.

# Algorithm

- Algorithms can be expressed in many kinds of notation, including natural languages, pseudocode, flowcharts, programming languages
- It has clear starting and stopping point

# Characteristics of Algorithm

- Each instruction should be precise and clear.
- Each instruction should be executed in a finite time.
- One or more instructions should not be repeated Infinitely.
- After executing the instructions the desired result should be obtained.
- It ranges from simple to the complex set of instructions.

# Pseudocode

- It is an english –like representation of the algorithm logic.
- Part English , part structured code.
- English part:
  - Describes what must be done without unnecessary details such as error messages.
- Code part:
  - Contains extended version of basic algorithmic constructs-sequence ,selection and iteration.

# Pseudocode

- Data items do not need to be declared.It is automatically declared.
- Type is determined by context.
- Eg:
- Pseudocode is used to describe an algorithm.
- Eg:

# Example of a Computer Algorithm

**Algorithm sampleAlgo()**

1. **[initialize]**
   **set K=1 and loc=0**
2. **Repeat steps 3 and 4 while loc=0 and k<=n**
3. **If item= a[k] then**
   **1. set loc=k**
4. **set k=k+1**
   **end loop**

**5. [successful?]**

```
if loc=0 then
    print “item not in the array”
else
    print  “ item in location loc”
```

**6. exit**

# Algorithm parts

- **Header:**
  - Contains name of the algorithms and its parameters.
- **Purpose:**
  - Describes what the algorithm does.
- **Precondition:**
  - Describes the requirements for the parameters.
- **Postcondition:**
  - Identifies any action taken.
  - Describes the status of any output parameters.
- **Return:**
  - Determines whether a value is returned or not.

# Algorithm parts

- Eg:

algorithm search(list[],value_to_search)

Search array for specific item and return index location

Pre : list[]: array of data

value_to_search:value to be searched

Post: displays location where the data is found

Return true if found, false if not found

# Sample Linear Search Algorithm

Algorithm search(list[],value_to_search,N)
Search array for specific item and return index location
Pre : list[]: array of data
value_to_search:value to be searched ; N:No of elements
Post: displays location where the data is found
Return true if found, false if not found

**1. [initialize]**
**set K to 1 and loc to 0**
**2. Repeat steps 3 and 4 while loc=0 and k<=n**
**3. If item= a[k] then**
**1. set loc to k**
**4. set k =k+1**
**end loop**
**5. [successful?]**
**if loc=0 then**
**print "item not in the array"**
**else**
**print " item in location loc"**
**6. exit**