Lecture 1- Introduction

Data Structure and Algorithm Analysis

Introduction

- Program is written in order to solve a problem
- A solution to a problem actually consists of two things:
  - A way to organize data (data structures)
  - Sequence of steps needed to solve the problem (Algorithm)
- A famous quote:
  - Program = Algorithm + Data Structure.
- A program prescribes 'what' is to be done with 'which' data

Example Program

- Read two integers and output their sum

```
#include <iostream>
using namespace std;
int main()
  {
        int i, j,sum;
        cin >> i >> j;
        sum=i+j;
        cout <<sum << endl;
        return 0;
  }
```

Data structures

- Variables i and j are used to represent integers
  - They form the data structure
  - int data type is available as part of the C++ programming language
  - A data structure is also called a data type in a programming language
  - The data type of a variable defines its possible values and the operations that can be performed on it

Algorithm

- The ***main*** function defines the algorithm
  - This uses the built-in operation + for adding int variables
  - It also uses functions defined in the iostream library for reading and printing int variables
  - Many commonly required data structures and algorithms are available as built-in types or as part of libraries

What will you learn from this course

- As we said, in this course, we will study:
  - *Algorithms*
    - Sequence of steps the needs to be followed to solve problems
  - *Data structures*
    - *A means* for efficiently storing, accessing, and modifying data
- Specifically, you are going to learn
  1. A collection of more commonly used data structures and algorithms-"programmers' basic toolkit"
  2. Tradeoffs associated with data structures and algorithms preferred
     - Usually, done by comparing space and time required by each DS and AL
  3. How to measure quality of a given data structure and algorithms
     - Allow you to judge the merits of new data structures that you or others might invent

Computer program design goals

- There are two basic design goals(sometimes conflicting)
  1. To design algorithm that is easy to understand, code and debug
  2. To design algorithm that makes efficient use of the computer's resources
- "Elegant program" satisfies both of the above goals
- The codes we write for this course needed to be elegant but our primary concern is goal 2 (i.e. efficiency).
  - Goal 1 is primarily the concern of software engineering

Efficiency

- A solution is said to be efficient if it solves the problem within its resource constraints or less cost
  - Cost is the mount of resources that the solution consumes such as time
- Constraints
  - Space (typical for many programs)
  - Time (specially for real time systems)
  - Bandwidth
- However, that does not mean we always strive for the most efficient program.
  - If the program works well within resource constraints, there is no benefit to making it faster or smaller.

A philosophy of data structures

- Question: -processor speed and memory size still continue to improve, will not today's efficiency problem be solved by tomorrow's hardware ?
  - The answer is no, our history proved it.
- Reasons:- as we develop more powerful computers, that addition is being used to tackle more complex problems
  - More sophisticated user interface
  - Bigger problem sizes
  - New problems previously deemed unfeasible

Data structure

- A data structure is any data representation and its associated operations.
  - Example int and float can be viewed as simple data structures
  - Operations support similar operations +,*,/,% etc
- ***Commonly, people use the term "data structure" to mean an organization or structuring of collection of data items***
  - Example, List of integers stored in array
- Data can be represented in computer using different data structures
  - Example, list of integers can be represented using array or another data structure called linked list
  - However, using the proper data structure can make the difference.

How to select a good data structure ?

- There are different ways to organize data in computer
  - In other words, Data structures
- And, there is no ultimate data structure that fits to every problem
  - Each data structure has associated costs and benefits(trade-offs)
- The choice to use a particular data structure depends on our requirements

Steps to select data structure

1. Analyze your problem to determine the basic operations that must be supported. Examples
   - inserting a data item into the data structure,
   - deleting a data item from the data structure, and
   - finding a specified data item etc…

- 2. Quantify the resource constraints for each operation.
  - Such as Time
- 3. Select the data structure that best meets these requirements.
  - the "simplest" that meets requirements
- Note:-Resource constraints on key operations such as search, insert and delete drives the data structure selection.

Abstract Data types-Definitions

- A type is a collection of values.
  - Example
    - Boolean type consists of values true and false
- Simple type is a type/values that contains no sub parts
  - Example, int, float,…
- Aggregate/composite type: its value has subparts.
  - Example student type has parts like name, idno, gpa…
- A data item is a member of a type.
- A data type is a type together with a collection of operations to manipulate the type.
  - Example, int variable is a member of the integer data type and addition is example operation on int data type

Abstract Data Type (ADT)

- Abstract Data Types(ADT) Consists of data to be stored and operations supported on them.
- ADT is a specification that describes a data set and the operation on that data.
  - An ADT doesn’t specify how the data type is implemented
  - Rather it only specifies a set of values and a set of operations on that data type
  - Each ADT operation is implemented by a function/method.
- A data structure is the implementation of an ADT.
- In OOP languages, an ADT and its implementation together makes up a class.
  - Each operation of ADT is implemented by the member methods.

Example

- Integer
  - Values are …., -3, -2, -1, 0, 1, 2, 3, …..
  - Operations are +, -, *, /, % …
- The abstract data type Integer is an infinite set
- The built-in data structure int is a particular implementation of the abstract data type Integer
- Another built-in data structure long long int also implements the same abstract type

Data structure Vs File structure

- **Data structure:** usually refers to an organization for data in main memory.

- **File structure:** an organization for data on peripheral storage, such as a disk drive or tape.

Problems, Algorithms and Programs

- Problem is a task to be performed.
    - Best thought of as inputs and matching outputs.
        - Example given id, find the detail of students
    - Problem definition should include constraints on the resources that may be consumed by any acceptable solution.
- ***Algorithms*** are steps that need to be followed to solve a problem.
    - A recipe(a set of instructions for preparing a particular dish, including a list of the ingredients required)
- An algorithm takes the input to a problem and transforms it to the output.
    - A mapping of input to output

Algorithm Design

- You have a problem to solve
  - Analyze the problem and identify the requirements
  - Design an efficient algorithm
    - Use good data structures
  - Show that your algorithm works!
    - Prove its correctness
  - Study the efficiency of your algorithm
    - Run time
    - Storage required
- For a problem given we might come up with different algorithms

Example

- Two algorithms for computing the Factorial
- Which one is better?

```
1)
   int factorial (int n)
 {
    if (n <= 1) return 1;
    else
    return n * factorial(n-1);
}
```

```
2)
int factorial (int n)
{
if (n<=1) return 1;
else {
fact = 1;
for (k=2; k<=n; k++)
      fact *= k;
return fact;
}
```

Properties of Algorithm

- **Finiteness**:
  - Algorithm must complete after a finite number of steps.
- **Definiteness**:
  - Each step must be clearly defined, having one and only one interpretation. At each point in computation, one should be able to tell exactly what happens next.
- **Sequence**:
  - Each step must have a unique defined preceding and succeeding step. The first step (start step) and last step (halt step) must be clearly noted.
- **Feasibility**:
  - It must be possible to perform each instruction.
- **Correctness**:
  - It must compute correct answer for all possible legal inputs.
- **Language Independence**:
  - It must not depend on any one programming language.

Properties of Algorithm

- **Completeness**:
  - It must solve the problem completely.
- **Efficiency**:
  - It must solve with the least amount of computational resources such as time and space.
- **Generality**:
  - Algorithm should be valid on all possible inputs.
- **Input/Output**:
  - There must be a specified number of input values, and one or more result values.

Program

- A computer program is an instance, or concrete representation, for an algorithm in some programming language.
  - We frequently interchange use of "algorithm" and "program" though they are actually different concepts

End of lecture 1

Next Lecture:- Algorithm and Algorithm Analysis