Data Structure Algorithms & Applications

CT-159

Prepared by

Muhammad Kamran

Course Requirements

- Two kinds of homework:
  - Assignments (**10 Marks**)
  - Mini Project + Competition (**5+5 Marks**).
- Two exams:
  - Midterm (**20 Marks**).
  - Final Exam (**60 Marks**).


Quiet Please
Sshhhhh!
Teaching and Learning in Progress
SmartSign.com • 800-952-1457 • S2-1465


NOTICE
NO TEXTING
NO CELL PHONE USE IN THIS CLASSROOM
Reorder: ONEP-14113 www.ComplianceSigns.com

Data Structure

- Data Structure is a systematic approach to organize data in order to use it efficiently.

OR

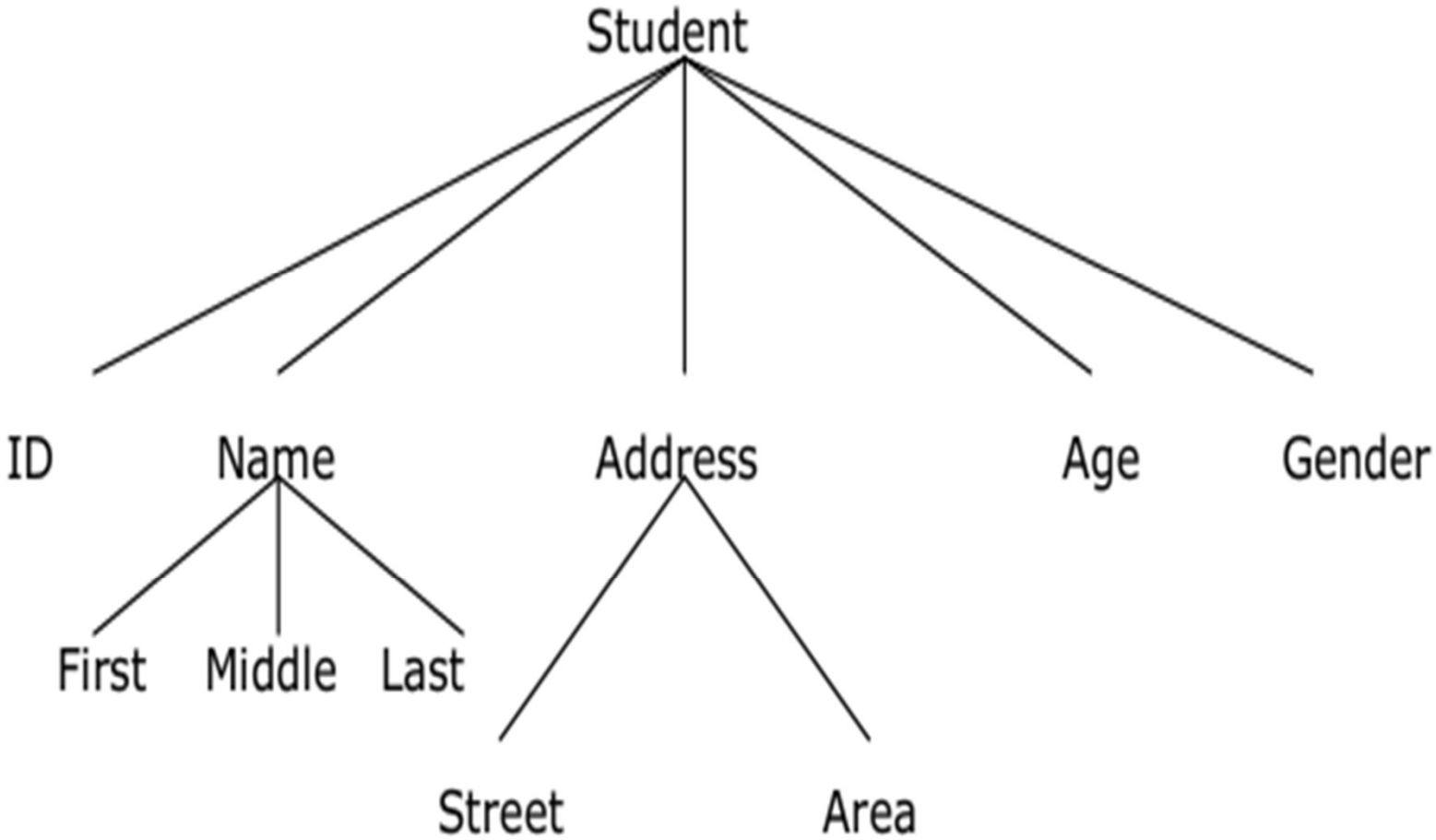
- Data Structure can be defined as the group of data elements which provides an efficient way of storing and organising data in the computer so that it can be used efficiently.

Data

Data Definition defines a particular data with the following characteristics.

- **Atomic** − Definition should define a single concept.
- **Traceable** − Definition should be able to be mapped to some data element.
- **Accurate** − Definition should be unambiguous.
- **Clear and Concise** − Definition should be understandable.

Data Terminology



Terminology

- **Data:** Data can be defined as an elementary value or the collection of values, for example, student's name and its id are the data about the student.
- **Attribute and Entity:** An entity represents the class of certain objects. It contains various attributes. Each attribute represents the particular property of that entity.
- **Field:** Field is a single elementary unit of information representing the attribute of an entity.

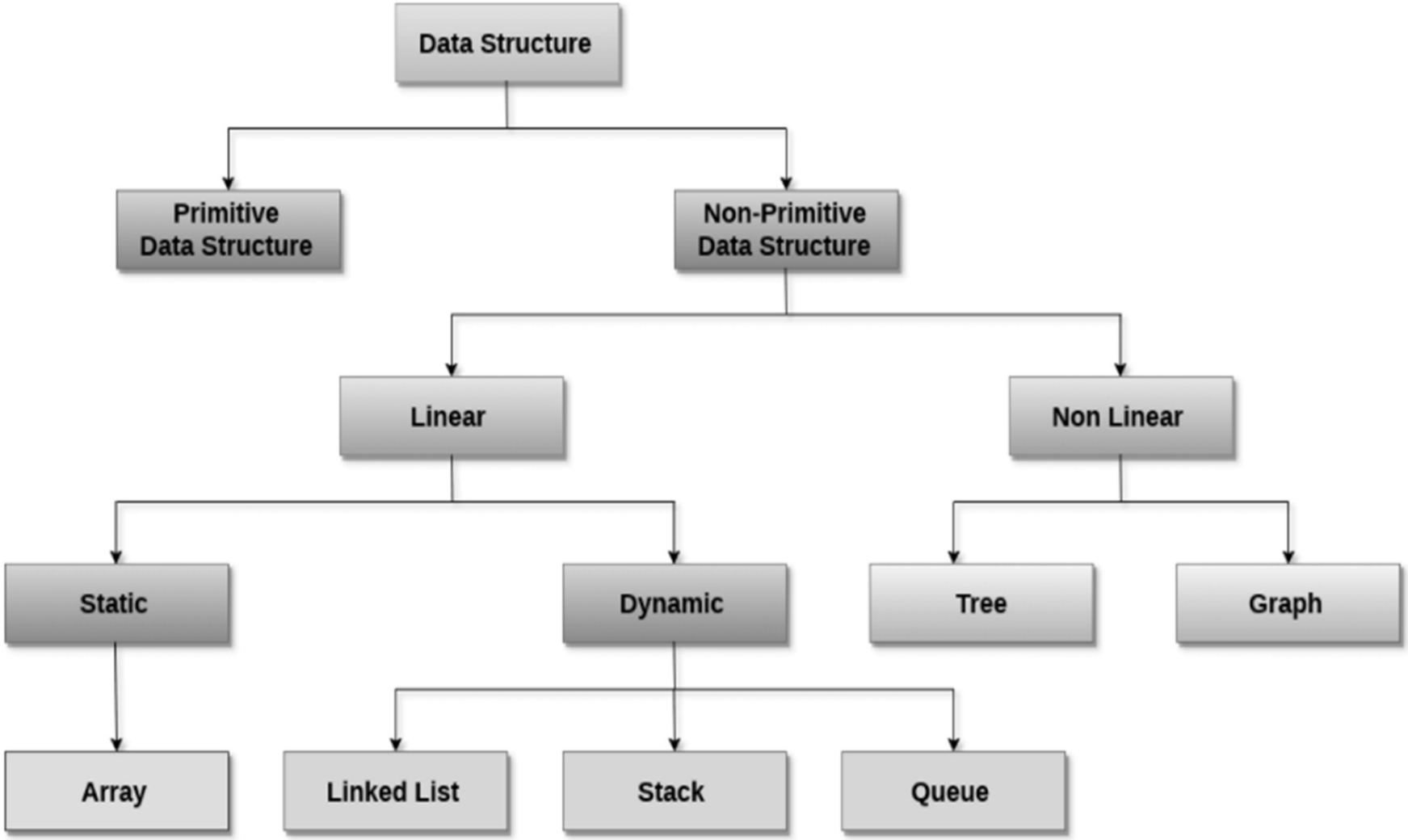
Terminology

- **Group Items:** Data items which have subordinate data items are called Group item, for example, name of a student can have first name and the last name.
- **Record:** Record can be defined as the collection of various data items, for example, if we talk about the student entity, then its name, address, course and marks can be grouped together to form the record for the student.
- **File:** A File is a collection of various records of one type of entity, for example, if there are 60 employees in a department, then there will be 20 records in the related file where each record contains the data about each employee.

Data Structure Operations

- **Traversing, Insertion, Deletion.**
- **Searching, Sorting and Merge.**

Data Structure Types



Primitive

- Primitive data structure is a data structure that can hold a single value in a specific location whereas the non-linear data structure can hold multiple values either in a contiguous location or random locations. The following are the four primitive data structures:
- **Integer:** The integer data type contains the numeric values. It contains the whole numbers that can be either negative or positive. When the range of integer data type is not large enough then in that case, we can use long.
- **Float:** The float is a data type that can hold decimal values. When the precision of decimal value increases then the Double data type is used.
- **Boolean:** It is a data type that can hold either a True or a False value. It is mainly used for checking the condition.
- **Character:** It is a data type that can hold a single character value both uppercase and lowercase such as 'A' or 'a'.

Non-Primitive

- The non-primitive data structure is a kind of data structure that can hold multiple values either in a contiguous or random location.
- The non-primitive data types are defined by the programmer.
- The non-primitive data structure is further classified into two categories, i.e., linear and non-linear data structure.
- Examples are **Array, Stack, Queue, Linklist, Tree** and **Graph**.

Data Structure

- Interface / Implementation
- Characteristics of a Data Structure
  - **Correctness**
  - **Time Complexity**
  - **Space Complexity**
- Execution Time Cases
  - **Best**
  - **Worst**
  - **Average**

Why Need Data Structure ?

- **Processor speed:** To handle very large amount of data, high speed processing is required.
- **Data Search:** Consider an inventory size of 1006 items in a store, If our application needs to search for a particular item, it needs to traverse 1006 items every time, results in slowing down the search process.
- **Multiple requests:** If thousands of users are searching the data simultaneously on a web server, then there are the chances that a very large server can be failed during that process.

Algorithms

- An algorithm is a process or a set of rules required to perform calculations or some other problem-solving operations especially by a computer.

OR

- An algorithm is that it contains the finite set of instructions which are being carried in a specific order to perform the specific task.

Why Need Algorithm

- **Scalability:** It helps us to understand the scalability. When we have a big real-world problem, we need to scale it down into small-small steps to easily analyze the problem.
- **Performance:** The real-world is not easily broken down into smaller steps. If the problem can be easily broken into smaller steps means that the problem is feasible.

Example Algorithm

- Step 1: Start
- Step 2: Declare three variables a, b, and sum.
- Step 3: Enter the values of a and b.
- Step 4: Add the values of a and b and store the result in the sum variable, i.e., sum=a+b.
- Step 5: Print sum
- Step 6: Stop

Characteristics of An Algorithm

- **Input:** An algorithm has some input values. We can pass 0 or some input value to an algorithm.
- **Output:** We will get 1 or more output at the end of an algorithm.
- **Unambiguity:** An algorithm should be unambiguous which means that the instructions in an algorithm should be clear and simple.

Characteristics of An Algorithm

- **Finiteness:** An algorithm should have finiteness. Here, finiteness means that the algorithm should contain a limited number of instructions, i.e., the instructions should be countable.
- **Effectiveness:** An algorithm should be effective as each instruction in an algorithm affects the overall process.
- **Language independent:** An algorithm must be language-independent so that the instructions in an algorithm can be implemented in any of the languages with the same output.

Algorithm Complexity

- **Time complexity:** The time complexity of an algorithm is the amount of time required to complete the execution.
- The time complexity of an algorithm is denoted by the big **O** notation. Here, big **O** notation is the asymptotic notation to represent the time complexity.
- The time complexity is mainly calculated by counting the number of steps to finish the execution.

Algorithm Complexity

- **Space complexity:** An algorithm's space complexity is the amount of space required to solve a problem and produce an output. Similar to the time complexity, space complexity is also expressed in big **O** notation.
- Instruction space,
- Data space and
- Environment space.

Computer Program

- Program = Algorithm + Data Structure

- Learning data structure and algorithm is needed to develop efficient and optimized computer programs.

Computer Program (C++ Language)

```
#include <iostream>

int main() {
  int first_number, second_number, sum;

  cout << "Enter two integers: ";
  cin >> first_number >> second_number;

  // sum of two numbers in stored in variable sumOfTwoNumbers
  sum = first_number + second_number;

  // prints sum
  cout << first_number << " + " <<  second_number << " = " << sum;
  return 0;
}
```

Abstract Data Types

- An abstract data type (ADT) is a model of a data structure that specifies:
  - the characteristics of the collection of data
  - the operations that can be performed on the collection
- It's abstract because it doesn't specify how the ADT will be implemented.
  - does not commit to any low-level details
- A given ADT can have multiple implementations.

Example : A Bag

- A bag is just a container for a group of data items.

  - analogy: a bag of candy

- The positions of the data items don't matter (unlike a **list**).

  - {3, 2, 10, 6} is equivalent to {2, 3, 6, 10}

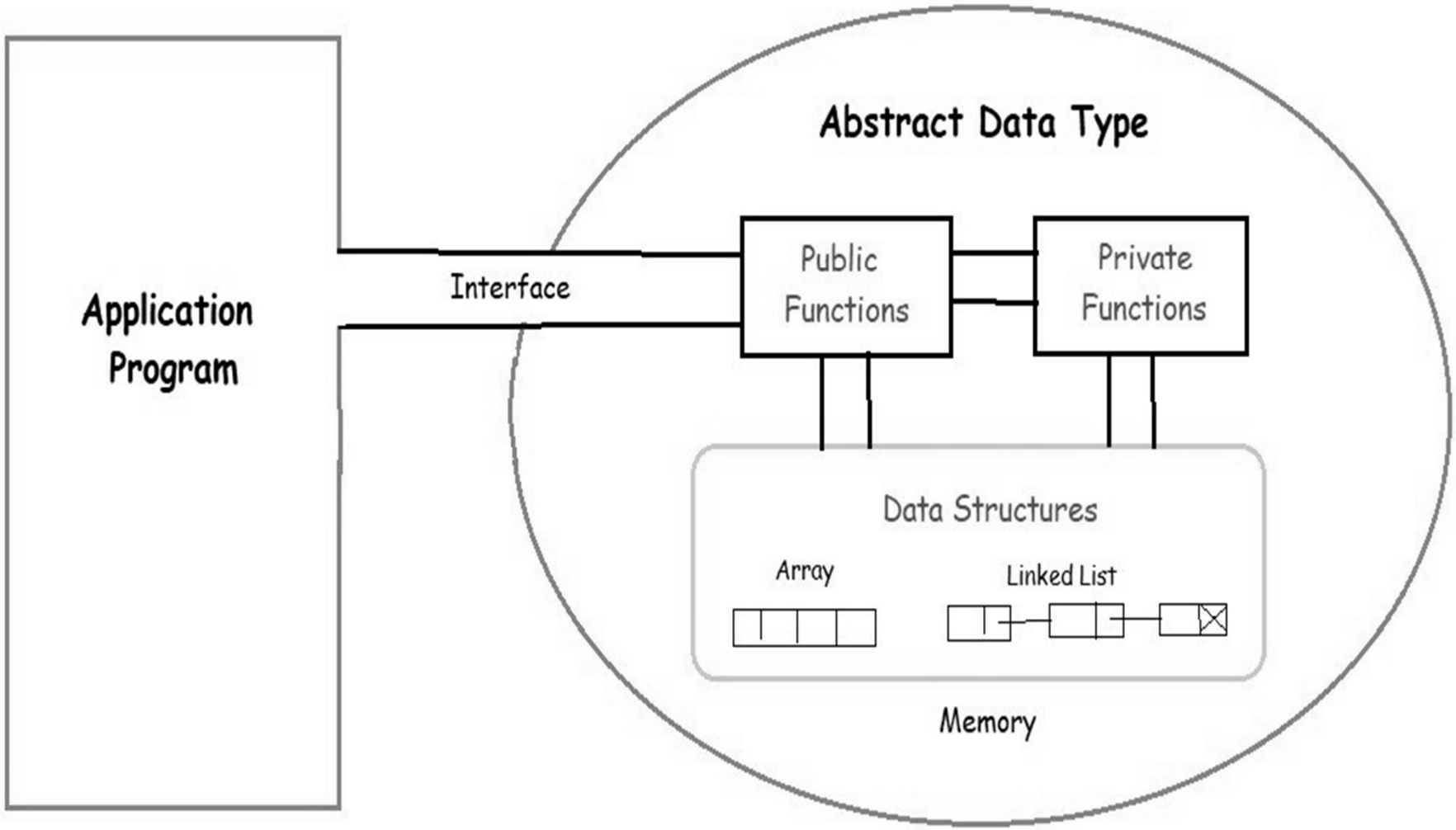
- The items do not need to be unique (unlike a **set**).

  - {7, 2, 10, 7, 5} isn't a set, but it is a bag

A Simple ADT: A Bag

- The operations we want a Bag to support:
  - **add(item)**: add item to the Bag
  - **remove(item)**: remove one occurrence of item from the Bag
  - **contains(item)**: check if item is in the Bag
  - **numItems():** get the number of items in the Bag
  - **grab():** get an item at random, without removing it
  - **toArray():** get an array containing the current contents of the bag.

Abstraction Process



Features of ADT

- **Abstraction**
- **Better Conceptualization**
- **Robust**
- **Encapsulation**
- **Data Structure Independence:**
- **Information Hiding**
- **Modularity**

Thank You