Paper: Data Structures
Unit No : II, Linear Structures,
Lesson : Introduction to Linear structures
Lesson Developer : Vandana Kalra, Associate Professor
College/Department: College of Vocational Studies , University of Delhi

Table of Contents

Chapter 1: Introduction to Linear Structures

As the computer operates on bits, representation of the data is done in group of bits. There are many different data types depending upon the number of bits required for representing them.

1.1 Data Types

This term specifies the different kinds of data used as an object in the programming languages. Data types can be categorized depending upon their definition structures.

1.1.1 Primitive Types

Primitive data types consist of those data types which are defined as basic units in bits. They are also known as *Standard data types* or *Built-in data types*. The representationof data types internally required certain number of bits which vary from language to language. In C++ ,these bit representation for different data types are shown in Table 1.1.

Data Type	Bits required for representation
Character	1
Integer	16
Double	64
Float	32

Table 1.1 Data types and representation in C++

1.1.2 Abstract Data Types (ADT)

Abstract data types refers to those data types that are described with primitive data types as their basic units. It consists of the set of values (domain) and the specifications of the operations that are provided to create and manipulate the data. ADTs are nothing to do with time and space. Object oriented programming languages like C++ has a direct link to ADTs by implementing them as a *class.* Some of the ADTs with their operations are given below:

ADT	Operations
List	Insertion, Deletion
Tree	Traversal, Insertion, Deletion
Queue	Insertion, Deletion
Stack	Push, Pop

Table 1.2 ADTs and Operations

1.2 Data Structure

A single integer can be used for various purposes as sum , counter or an index in a program, but generally we also need data with various parts such as list containing number of elements. We describe the logical properties of such a collection of data as an ADT. *A Data structure is a collection of data elements whose organization is characterized by accessing operations that are used to store and retrieve the individual data elements.*

Type

Int

Value range: min…….max

Operations

addition subtraction division multiplication

negation comparison ++ (increment)

-- (decrement)

Fig 1.1 Representation of Integers

Source: self

A real life Example : A Library, Library can be decomposed into its composite elements – Books. The collection of individual books can be arranged in a number of ways. The Library data structure is composed of elements (books) arranged in some physical arrangement. The operations (*set of operations are also referred as Interface*) can be adding a book, removing a book , searching a book, issuing a book, returning a book etc.

Data Structure

Library

Elements: Books

Operations

addition removal searching

issue return generation of fine

Fig 1.2 Library Data Structure

Value addition: Frequently asked question
How does the ADTs are related to encapsulation and abstraction ?
we can define an abstract data type (ADT) as a data type, that is a type and the set of operations that will manipulate the type. The set of operations are only defined by their inputs and outputs. The ADT does not specify how the data type will be implemented, all of the ADT's details are hidden from the user of the ADT. This process of hiding the details is called encapsulation. The ability to create new data types when needed and then use these data types is called data abstraction . We extend the example of the integer data type to an abstract data type, the operations might be delete an integer, add an integer, print an integer, and check to see if a certain integer exists. Notice that we do not care how the operation will be done but simply how do invoke the operation. We are thinking of data and the operations in a logical sense and consider their use without worrying about implementation details. We are doing the abstraction of data by encapsulating it.
Source:self

We categorize data structures in different types:

1.2.1 Linear Structures

Linear data structure defines a set of operations which do not create hierarchical structure among the elements of its data objects e.g array, singly Linked list . The linear data structure array has number of elements arranged in a column and operations on these elements can be storing, retrieval, searching etc.

1.2.2 Non-Linear Structures

Non-Linear or hierarchical data structure defines such a operations which create an hierarchical structures among the elements of its data objects e.g binary trees, graphs etc. Generally these types of data structures can not be implemented without the help of pointers.

1.2.3 Static Structures

A data structure whose organizational characteristics are invariant throughout its lifetime. Such structures are well supported by high-level languages . Some examples are arrays and records.In static data structure , the elements of an allocated structure are physically contiguous, held in a single segment of memory; all descriptive information i.e size ,name other than the physical location of the allocated structure, is determined by the structure definition; type of all elements remain same. They are created during compilation time. Elements are accessed using some programming name given to it.

1.2.4 Dynamic Structures

A data structure whose organizational characteristics may change during its lifetime. The adaptability afforded by such structures, e.g. linked lists, is often at the expense of decreased efficiency in accessing elements of the structure. Two main features distinguish dynamic structures from static data structures. Firstly, it is no longer possible to infer all structural information from a header; each data element will have to contain information relating it logically to other elements of the structure. Secondly, using a single block of contiguous storage is often not appropriate, and hence it is necessary to provide some storage management scheme at run-time. Elements of dynamic structure can be accessed using their addresses and not by name.

Value addition: Do you know

How data structures are represented in programming Languages?

Data structures were well represented in the object oriented programming languages like C++. The ADTs are implemented in such languages in a form of a ***Class .*** Due to data encapsulation property in OOPs , data is well protected from outside world.

```
Class student

{  char  Name[30];
   char  Address;
   int  age;
   int marks;

   void student_detail();
    int  result();
}
```

1.3 Basic Linear structures

A total linear structure is a collection of items ordered by a single property so that each item, except possibly for the first or last, has a unique "predecessor" and a unique "successor". It is the most commonly used structure and appears under a variety of names depending on storage representation and its intended use. Some examples are arrays, lists, stack, queue etc.

We have two commonly used ADTs stacks and queues in which the representation and operations of data are done in different manner. Stacks are linear structures in which data is arranged in piles and accessed from one end whereas in Queues the data can be arranged in linear or circular row and accessed from two ends.

These data structures are having many advantages in terms of their representation and operations in the field of software and hardware in computer science. There is a need to understand the concept of these ADTs and their operations to know the architecture of computer parts and implementations of various constructs.

Summary

- set of elements such as Integers are used as data objects.
- Data objects are represented using basic data types available in various programming languages.
- Abstract data type related with abstraction of data is a mathematical concept which defines the complete well defined entity using its related components data structure and accessed with defined operations.
- The set of operations can be called as interface. Then we categorize the data structures as linear, Non-linear, static and dynamic.

Exercises

1.1 Design an ADT fraction which describes the properties of fraction:

a) What data structure can be used? What are its elements?
b) What does the interface look like?

1.2 Write an ADT specification in terms of its elements and operations for

a) Complex numbers a+ib, its absolute value(square of complex number), addition, subtraction, multiplication and negation.

b) rational numbers with operations as addition, subtraction, multiplication and division.

1.3 " ADT supports abstraction of data" Justify the statement.

Glossary

Abstract data type

A data type whose properties (domain and operations) are specified independently of any particular representation : a class of data objects with a defined set of operations that process the data objects while maintaining its properties.

Algorithm

A logical sequence of discrete steps that describes a complete solution to a given problem.

Constructor

An operation that builds new instances of an abstract data type .

Data Structure

A Data structure is a collection of data elements whose organization is characterized by accessing operations that are used to store and retrieve the individual data elements; the implementation of the composite data members in an ADT

Dynamic data structure

A data structure that can expand and contract during program execution.

Implementation

Coding and testing an algorithms

Primitive data types

These data types consists of those data types which are defined as basic units in bits

stack

A stack is a last in, first out (LIFO) abstract data type and data structure. A stack can have any abstract data type as an element, but is characterized by only two fundamental operations: *push* and *pop*

Queue

A queue is a particular kind of collection in which the entities in the collection are kept in order and the principal (or only) operations on the collection are the addition of entities to the rear terminal position and removal of entities from the front terminal position. This makes the queue a First-In-First-Out (FIFO) data structure.

References

Suggested Readings

1. Fundamentals of Data Structures - Ellis Horowitz
2. Data Structure And Algorithms In C++ 2nd ed - Adam Drozdek
3. Algorithms and Data Structures in CPlusPlus - Alan Parker
4. C++ plus Data Structures 4th ed - Nell Dale
5. Data Structures and Algorithms - Alfred V. Aho
6. Data Structures and Program Design in C++ - Robert L. Kruse
7. Data Structure using C and C++ - Langsam ,Augenstein and Tanenebaum

Web Links

1. www.codefords.wordpress.com
2. www.Wikipedia .org
3. www. read.cs.ucla.edu
4. www.cs.usfca.edu