DATA STRUCTURES AND ALGORITHM

LILY E. BENSAH

Goals:

- Learn the commonly used data structures
- To learn how to measure the merits of these data structures
- To learn/reinforce the concept that every data structure has costs and benefits

SECTION 1-INTRODUCTION

- Introduction to Data Structures
- Abstract Data Types
- Algorithm

SECTION 1.1-INTRODUCTION TO DATA STRUCTURES

What is Data Structure?

- In computer science**, data structure** is a particular way of storing and organizing data in a computer so that it can be used efficiently.
- A well-defined data structure allows a variety of critical operations to be performed on the data using as little resources, both execution time and memory space, as possible.

SECTION 1.1-INTRODUCTION TO DATA STRUCTURES

Why Study Data Structures?

Data structures organize data:

- Good choice: more efficient programs
- Bad choice: poor program performance

The choice can make a difference between the program running in a few seconds or many days

INTRODUCTION TO DATA STRUCTURES(CONT.)

CLASSIFICATION OF DATA STRUCTURE

Data structures are broadly divided into two :

- *Primitive data structures :*A primitive data type are predefined types of data that are built into the programming language - int, char, long, etc
  - Also referred to as Basic, Fundamental or simple data types
  - variables of that type can store only one value at a time

INTRODUCTION TO DATA STRUCTURES(CONT.)

CLASSIFICATION OF DATA STRUCTURE

- *Non-primitive data structures :* a more sophisticated data structure consisting of a group of homogeneous (same type) or heterogeneous (different type) data items - Array, Linked list, Stacks etc
  - not defined by the programming language, but are instead created by the programmer.
  - Sometimes called structured data type,
  - Simple data types are building blocks of structured data types.

INTRODUCTION TO DATA STRUCTURES(CONT.)

- A data structure is a group of data elements grouped together under one name. These data elements, known as *members*, can have different types and different lengths. Data structures are declared in C++ using the following syntax:

```
struct structure_name {
    member_type1 member_name1;
    member_type2 member_name2;
    member_type3 member_name3;
    .
    .
    } object_names;
```

INTRODUCTION TO DATA STRUCTURES(CONT.)

- where structure_name is a name for the structure type,
- object_name can be a set of valid identifiers for objects that have the type of this structure.
- Within braces { } there is a list with the data members, each one is specified with a type and a valid identifier as its name.

INTRODUCTION TO DATA STRUCTURES(CONT.)

- data structure creates a new type: Once a data structure is declared, a new type with the identifier specified as structure_name is created and can be used in the rest of the program as if it was any other type. For example:

INTRODUCTION TO DATA STRUCTURES(CONT.)

```
• struct product {
    int weight;
    float price;
  } ;
  product apple;
  product banana, melon;
```

INTRODUCTION TO DATA STRUCTURES(CONT.)

- We have first declared a structure type called product with two members:

  weight and price, each of a different fundamental type.
- We have then used this name of the structure type (product) to declare three objects of that type:

  apple, banana and melon as we would have done with any fundamental data type.

INTRODUCTION TO DATA STRUCTURES(CONT.)

- Once declared, product has become a new valid type name like the fundamental ones int, char or short and from that point on we are able to declare objects (variables) of this compound new type, like we have done with apple, banana and melon.

INTRODUCTION TO DATA STRUCTURES(CONT.)

- Right at the end of the struct declaration, and before the ending semicolon, we can use the optional field object_name to directly declare objects of the structure type. For example, we can also declare the structure objects

  apple, banana and melon at the moment we define the data structure type this way:

  – struct product {

    int weight; float price;

    } apple, banana, melon;

INTRODUCTION TO DATA STRUCTURES(CONT.)

- It is important to clearly differentiate between what is the structure type name, and what is an object (variable) that has this structure type. We can instantiate many objects (i.e. variables, like apple, banana and melon) from a single structure type (product).

INTRODUCTION TO DATA STRUCTURES(CONT.)

- Once we have declared our three objects of a determined structure type (apple, banana and melon) we can operate directly with their members. To do that we use a dot (.) inserted between the object name and the member name.
- For example, we could operate with any of these elements as if they were standard variables of their respective types:

INTRODUCTION TO DATA STRUCTURES(CONT.)

- apple.weight
- apple.price
- banana.weight
- banana.price
- melon.weight
- melon.price

Each one of these has the data type corresponding to the member they refer to:

- apple.weight, banana.weight andmelon.weight are of type int,

  while apple.price, banana.price and melon.price are of type float.

ABSTRACT DATA TYPES(ADT) (CONT.)

- Abstract data types consist of collections of data elements together with basic operations on the data.
- Abstract data type: a type and a collection of operations to manipulate that type
- ADTs are mathematical abstractions; an ADT only mentions what is to be done, not how.

ABSTRACT DATA TYPES(ADT) (CONT.)

- “Abstract” in that implementation of operations not specified in ADT definition

E.g., List

- Insert, delete, search, sort

- Can change ADT implementation details without breaking code using ADT

Assignment

Implement the following algorithms (1 & 2) in C++.

Algorithm 1

```
1. Accept an integer for n.
2. Initialize sum to O.
3. For each integer i in the range 1 to n:
      Assign sum + i to sum.
4. Return the value of sum.
```

Assignment

Algorithm 2

1. Accept an integer for *n*.
2. Return the value of $n \times \frac{(n+1)}{2}$.