Data Structure and Lab (CSE123)

Introduction to Data Structures

Muhammad Tariq Mahmood

tariq@koreatech.ac.kr
School of Computer Science and Engineering

Note: These notes are prepared from the following resources.

- Data Structures and Algorithms Using Python by Rance D. Necaise
- Data Structures Using Python by Y.K. Choi and I.G. Chan (Korean)
- Data Structures and Algorithms in Python by Michael T. Goodrich , Roberto Tamassia , Michael H. Goldwasser
- https://www.geeksforgeeks.org/data-structures/

Contents

Course Overview

- Data Structures course is a very important course.
- It is usually considered as the foundation course in computer science.
- The topics covered in this course will be encountered again and again in more advanced courses.
- We will cover well-known data structures such as dynamic arrays, linked lists, stacks, queues, trees and graphs.
- We will Implement these data structures in Python

Course Overview (cont...)

Course Objectives

- The objectives of the class include to:
  1. Study and understand well-known linear data structures (List, Stack, Queue)
  2. Understand well-known nonlinear data structures (Tree, Graph)
  3. Study various applications related to specific data structures
  4. Implement the data structures using Python or any other language
  5. Write Python programs to test various data structures

Course Overview (cont...)

Grading

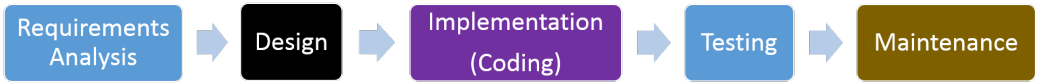
Item	% of the the total grade
Assignments/Practice/Labs	30%
Midterm Exam	30%
Final Exam	30%
Quizzes/Attendance/QA sessions	10%

Textbooks and References

1. Data Structures and Algorithms Using Python by Rance D. Necaise
2. Data Structures Using Python by Y.K. Choi and I.G. Chan (Korean)
3. Data Structures and Algorithms in Python by Michael T. Goodrich , Roberto Tamassia , Michael H. Goldwasser
4. https://www.geeksforgeeks.org/data-structures/

Problem Solving Process

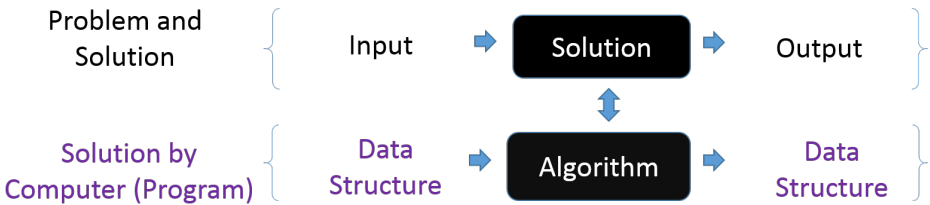
- Phases in Problem Solving Process(Software Development)



- Requirements analysis: involves in determining input and output required to solve the problem including other informations such as software usage, feasibility of computer solution
- Design: involves in selecting appropriate **data structures** to hold input/output and designing efficient algorithms, needed to process the data
- Implementation: involves in selecting appropriate programming language and translating algorithms into well-structured, well-documented and readable code
- Testing: involves in executing the software, correcting errors, verifying output of algorithms.
- Maintenance: involves in modifying software to improve performance and adding new features, etc

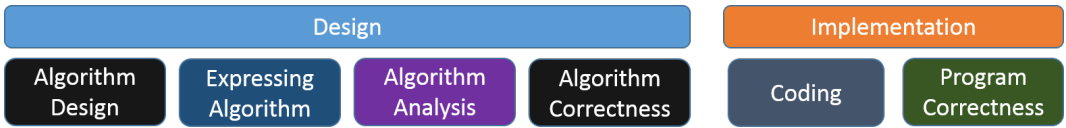
What is an Algorithm?

- Relation between a problem and its solution (algorithm)



- A computer **algorithm** is a detailed **step-by-step method** for solving a **problem** using a computer.
- An algorithm is a **well-defined computational procedure** that takes some value or a set of values as **input** and produces some value or a set of values as **output**.
- Applying a technique to solve a **problem** results in a **step by step procedure**. This step by step procedure is called an algorithm for the problem.

- Design and Implementation Phases



- Algorithm design techniques
- There are general approaches to construct efficient solutions to problems. They provide templates suited to solving a broad range of diverse problems.
  - Brute Force
  - Divide and conquer
  - Dynamic programming
  - Greedy approach
  - State space search techniques

- Expressing Algorithms (Pseudocode)
- Pseudocode specifies the steps required to accomplish the algorithm.
- Pseudocode cannot be compiled nor executed, and there are no syntax rules.
- Example

```
1: procedure ArrayMax(A,n)
2:     tmp=A[0]
3:     for i=1 to n-1 do
4:         if tmp < A[i] then
5:             tmp = A[i]
6:         end if
7:     end for
8:     return tmp
9: end procedure
```

Introduction to Data structures (cont...)

- Algorithm Analysis
- The analysis of algorithms is estimated in the asymptotic sense, i.e., to estimate the **complexity function** for arbitrarily large input.
- Usually, the efficiency or running time of an algorithm is stated as a function relating the input size to the number of steps, known as **time complexity**, or volume of memory, known as **space complexity**.
- Generally, following types of analysis are performed
  - Worst-case The maximum number of steps taken on any instance of size a.
  - Best-case The minimum number of steps taken on any instance of size a.
  - Average case An average number of steps taken on any instance of size a.
  - Amortized A sequence of operations applied to the input of size a averaged over time.
- Algorithm Correctness Analysis
- This means to verify if the algorithm leads to the correct solution of the problem after a finite number of processing steps. Two common strategies: Experimental analysis and Formal analysis

What is Data Structure?

What is Data Structure?

- Data structure is the structural representation of logical relationships between elements/items of data
- In other words a data structure is a way of organizing data items by considering its relationship to each other
- Organizing data means that the data should be arranged in a way that it is easily accessible and we may perform operations on it.

  $Data \quad Structure = Organized \quad data \quad items + Operations$

- Data structure affects the design of both the structural and functional aspects of a program

  $Algorithm + Data \quad Structure = Program$

What is Data Structure? (cont...)

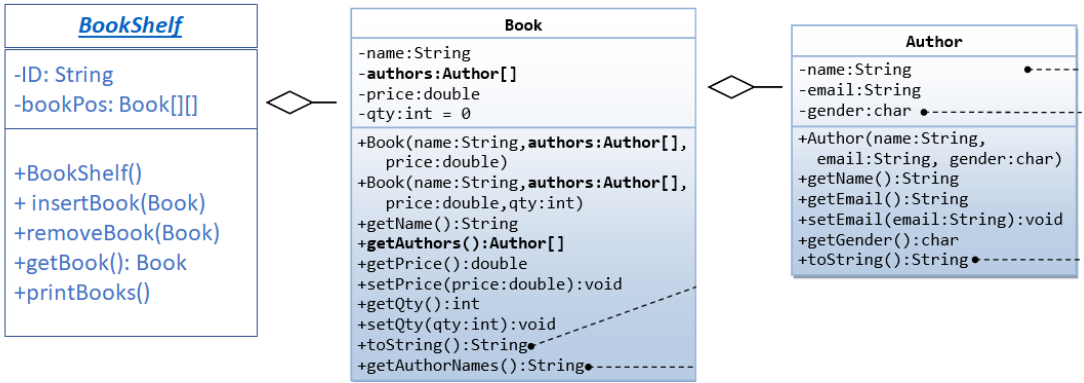
What is data organization?

- Computer memory stores value of a particular type
- Organizing items: examples from the real life

- Data types
  - Book
  - Shoes
  - Key
  - Necklace
  - Bookshelf
  - Shoes-case
  - Key-holder
  - Necklace-case

What is Data Structure? (cont...)

- Data Structure for BookShelf Type



What is Data Structure?

Data and Data Type

- Data means value or a set of values stored in computer memory
- Data may be in single or plural form
- A data type is a term which refers to the type/class/kind of data.
- Some examples are shown in Table 1.

Table 1: Data and their types

Data	Data Type
34	Integer
5.6	Real
$12, 39, 23, 90, 78, 3, 6, 17$	Array of integers
$2015/06/23$	Date
$ISBN - 2011 - 203 - 11111 - 11$	Book number

What is Data Structure? (cont...)

- Data types can be divided into two main categories
  - Built-in Data Types
  - User Defined Data Types

Built-in Data Types

- Every programming language contains a set of data types called built-in data types
- Programming language provides many advantages/facilities regarding the processing of these data types
- For example, if a user declares a variable of type double then several things automatically implied by the compiler.
  1. Mechanism is provided to storage of the value
  2. Required memory is allocated
  3. Different operations are possible on the data type

- Built-in Data Types in C++ are shown in Table 2

Table 2: Data types in C++

Data Type	Sub type	Size	Domain
Integer	short	2 bytes (16 bit)	-32,768 to 32,767
	unsigned short	2 bytes (16 bit)	0 to 65,535
	int	4 bytes (32 bit)	-2,174,483,648 to 2,174,483,647
	unsigned int	4 bytes (32 bit)	0 to 4,294,967,295
	long	4 bytes (32 bit)	-2,174,483,648 to 2,174,483,647
	unsigned long	4 bytes (32 bit)	0 to 4,294,967,295
Real	float	4 bytes (32 bit)	1.2E-38 to 3.4E38 (7 digits)
	double	8 bytes (64 bit)	2.2E-308 to 1.8E308 (15 digits)
Character	char	1 bytes (8 bit)	-127 to 127
	unsigned char	1 bytes (8 bit)	0 to 255
Boolean	bool	1 bytes (1 bit)	0 or 1

What is Data Structure? (cont...)

- One important aspect of data structure/type is how much memory is taken by it
- **The sizeof Operator:** C/C++ provides an unary sizeof operator to get the size of the operand (in bytes). The following program uses sizeof operator to print the size of the fundamental types.

```cpp
#include <iostream>
using namespace std;

int main() {
   cout << "sizeof(char) is " << sizeof(char) << " bytes " << endl;
   cout << "sizeof(short) is " << sizeof(short) << " bytes " << endl;
   cout << "sizeof(int) is " << sizeof(int) << " bytes " << endl;
   cout << "sizeof(long) is " << sizeof(long) << " bytes " << endl;
   cout << "sizeof(long long) is " << sizeof(long long) << " bytes " << endl;
   cout << "sizeof(float) is " << sizeof(float) << " bytes " << endl;
   cout << "sizeof(double) is " << sizeof(double) << " bytes " << endl;
   cout << "sizeof(long double) is " << sizeof(long double) << " bytes " << endl;
   cout << "sizeof(bool) is " << sizeof(bool) << " bytes " << endl;
   return 0;
}
```

What is Data Structure? (cont...)

- **header <climits>** The climits header (ported to C++ from C's limits.h) contains information about limits of integer type. For example,

```cpp
#include <iostream>
#include <climits> // integer limits
using namespace std;

int main() {
   cout << "int max = " << INT_MAX << endl;
   cout << "int min = " << INT_MIN << endl;
   cout << "unsigned int max = " << UINT_MAX << endl;
   cout << "long long max = " << LLONG_MAX << endl;
   cout << "long long min = " << LLONG_MIN << endl;
   cout << "char max = " << CHAR_MAX << endl;
   cout << "char min = " << CHAR_MIN << endl;
   cout << "signed char max = " << SCHAR_MAX << endl;
   cout << "signed char min = " << SCHAR_MIN << endl;
   return 0;
 }
```

What is Data Structure? (cont...)

Non-primitive Data Types

- When an application requires a special kind of data (special data type) which is not available as built-in data type then programmer defines and implements it
- Programmer's own data type is termed as abstract data type
- Abstract data types also known as user-defined data types
- User defined types (New types) can be defined through classes.
- A class provides a set of behaviors in the form of **member functions** (also known as methods), and a set of state information for each instance is represented in the form of **attributes** (also known as **fields**, **instance variables**, or **data members**).

- Examples
  1. Time

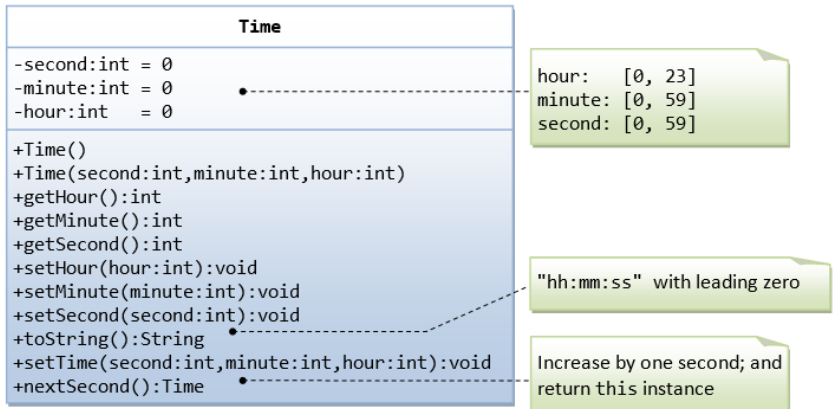


Figure 1: Date Class Diagram

2. Date

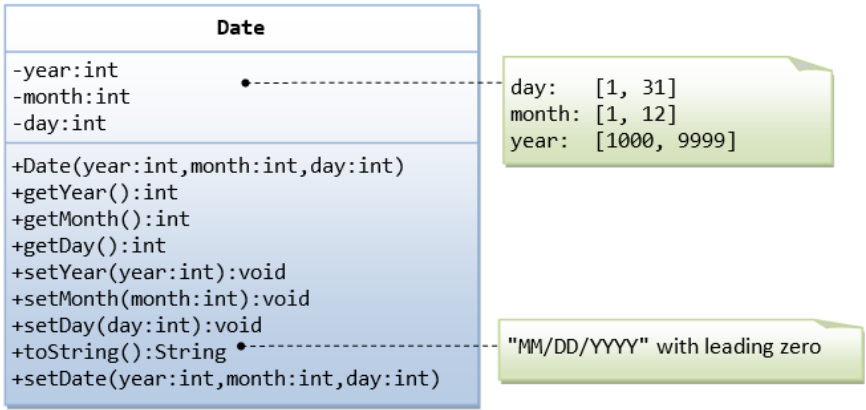


Figure 2: Date Class Diagram

3. Employee

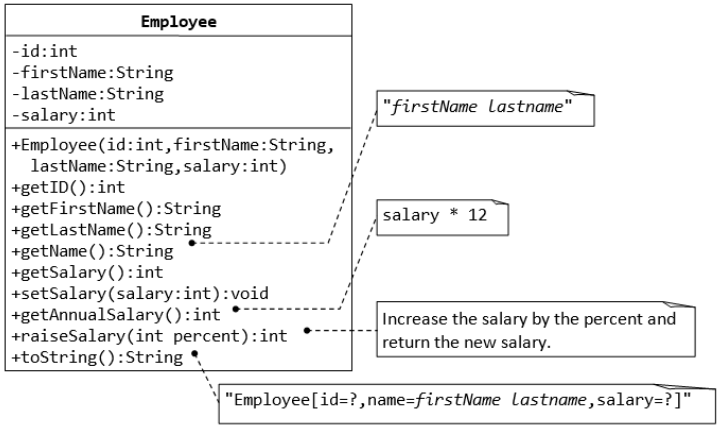


Figure 3: Employee Class Diagram

4. Student

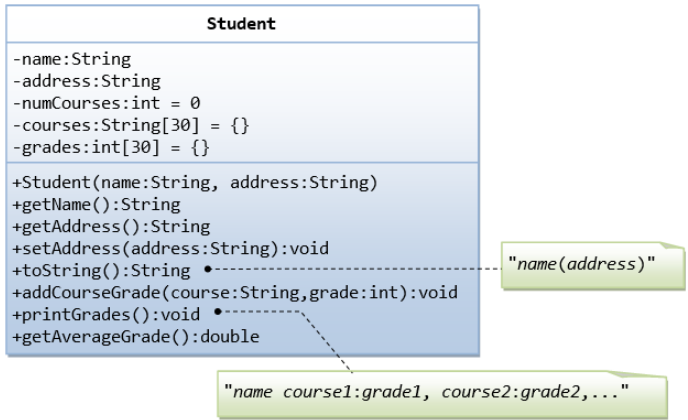


Figure 4: Student Class Diagram

5. Complex Number

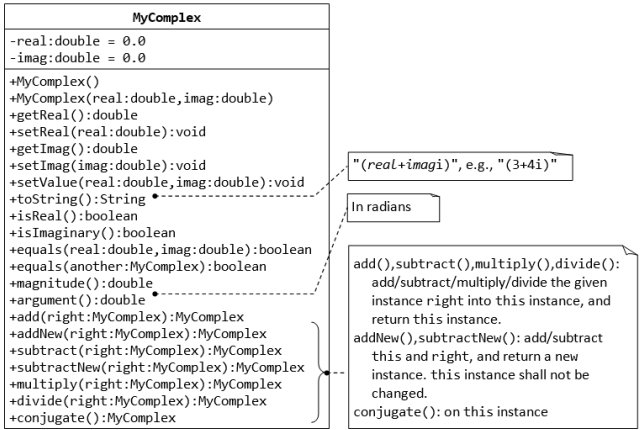


Figure 5: Complex Number Class Diagram

6. Polynomial

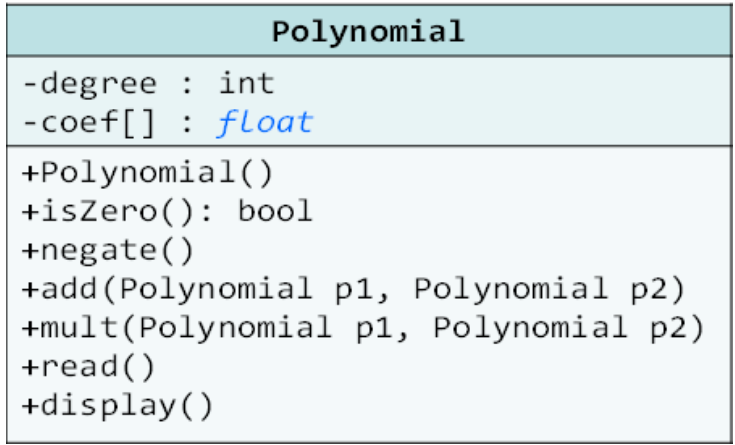


Figure 6: Polynomial Class Diagram

Classification of Data structures

Data structures can be classified into different categories

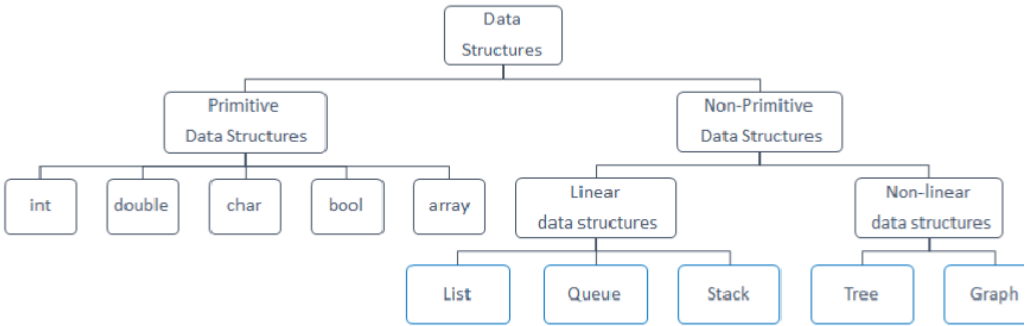


Figure 7: Classification of Data structures

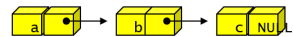
Classification of Data structures (cont...)

- **Simple Data Structures:** Simple data structures are used to store a single element. Usually primitive data types of any one of the computer languages. Variables, pointers, structures, unions, etc. are examples of simple primitive data structures.

- **Compound Data structures:** Compound data structures organize multiple data items. Data items may have primitive data types or user defined data types. It can be classified into (1) Linear data structures (2) Non-linear data structures
  - **Linear data structure :** Collection of data elements which are logically adjacent. In the linear Data Structures the relation ship of adjacency is maintained between the Data elements. *Array, List, Stack and Queue* are linear data structures

Classification of Data structures (cont...)

Examples from daily life	Data Structure
List of things to buy	List
A stack of books	Stack
Cinema box office line	Queue



- **Non-linear data structure:** Non-linear data structure can be constructed as a collection of randomly distributed set of data elements joined together by using a special pointer (tag). In non-linear Data structure the relationship of adjacency is not maintained between the Data items. *Tree and Graph* are non-linear data structures

Classification of Data structures (cont...)

Figure 8: Compound Data structures(Non-linear data structures:)

- **Operations applied on linear and non-linear data structures:**
  - *create a data structure*
  - *add/insert* element(s)
  - *remove/delete* element(s)
  - *display* the elements
  - *sort* the elements
  - *search* for a data element