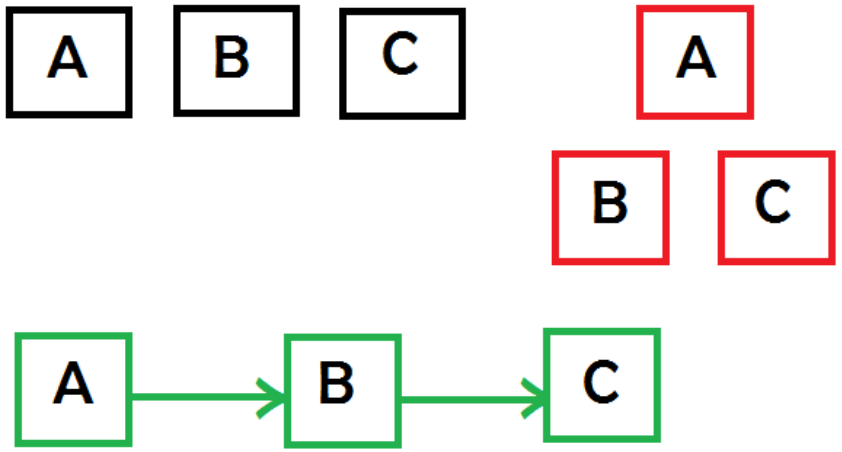
# CS-2001 Data Structures

## Week 2 | Lecture 1

*Special thanks to Dr. Douglas DH*

# Data Structures

- ***Data structure*** is a particular way of storing and arranging data (in a computer) so that it can be used efficiently



# Data Type

- Set of data with predefined values

**OR**

- Something that can store particular “type” of values

# Data Type

- At the top level, there are two types of data types:
  - Primitive Data Types (System-defined Data Types)
  - User-defined Data Types

# Primitive Data Types

- Data types defined by the system
- **Examples:** *int, float, char, double, etc*.

# User-defined Data Types

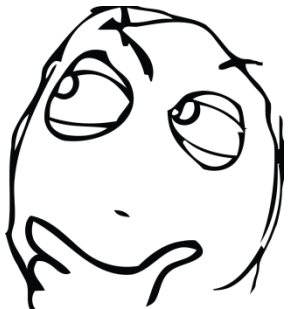
- Most programming languages allow users to define their own data types

- **Examples:** *structures in C, classes in C++ & JAVA*

# Abstract Data Type (ADT)

- All primitive data types (int, float etc.) support basic operations like addition, subtraction

- But what about user-defined data types???



# Abstract Data Type (ADT)

- For user-defined data types, we define operations and provide implementations for these operations whenever we want to use these data types

# Abstract Data Type (ADT)

- We combine data types with their operations and call it ***Abstract Data Types***

- An **ADT** consists of:
  - Declaration of data
  - Declaration of operations

# Example

- A class represent a new type, with user-defined operations represented by member functions

- Create a list that contains different types of elements such that only last added element can be read (LIFO)

# Abstract Data Type (ADT)

- Commonly used ADTs *include: Linked Lists, Stacks, Queues, Trees*, Graphs, and many others

- While defining an ADT, we do not worry about the implementation details until we need to use it

# Memory Allocation

- Memory allocation can be classified as either
  - Contiguous
  - Linked
  - Indexed

# Contiguous Allocation

- Allocation of memory as a single contiguous block of memory
- **Example:** Arrays

***Contiguous, adj.*** *Connected throughout in an unbroken sequence (Meriam Webster)*

# Linked Allocation

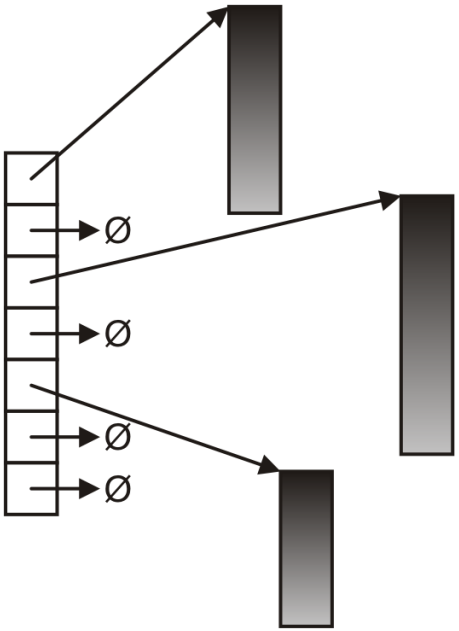
- Linked storage such as a linked list associates two pieces of data with each item being stored:
  - The object itself
  - A reference to the next item

- In C++ that reference is the address of the next node

- **Example:** Linked List

# Indexed Allocation

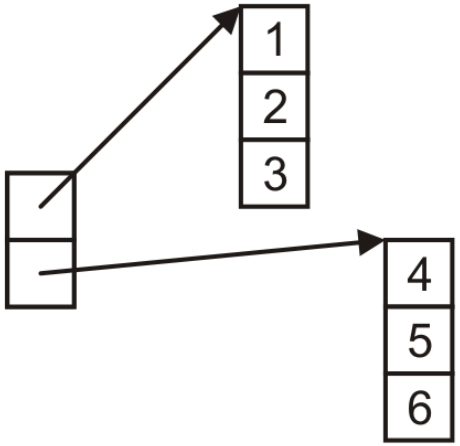
- With indexed allocation, an array of pointers (possibly NULL) link to a sequence of allocated memory locations

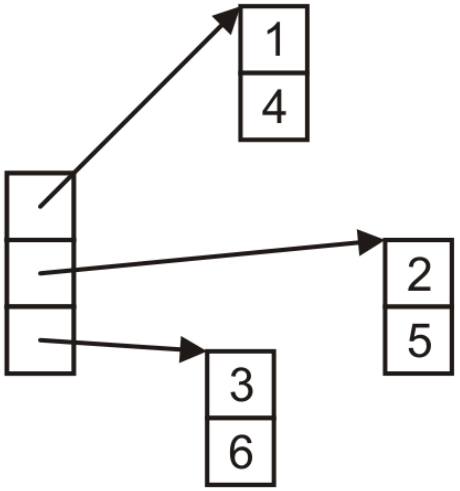


# Indexed Allocation

- Matrices can be implemented using indexed allocation:

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}$$



# Next Lecture

- Iteration vs Recursion
- Arrays & List ADT