

9618 Paper 2 Memorizations

9 Algorithm Design and Problem-solving

- **Abstraction:**
 - To **filter out** information that is not necessary to solve the problem
 - To include **only essential** information
 - Reduce **complexity** of a program
- **Decomposition:**
 - **Breaking down** a problem into sub problems
 - In order to **explain** // **Easier** to solve the problem
 - Leading to the concept of **program modules**
- **Algorithm:** a **solution** to a problem expressed as a **sequence of defined steps**
- **Stepwise Refinement:**
 - Break the problem into **smaller steps**
 - Until steps can be directly translated into **lines of code**

10 Data Types and Structures

- **Data Types:**
 - integer, real, char, string, Boolean, date
 - **pseudocode:** INTEGER, REAL, CHAR, STRING, BOOLEAN, DATE, ARRAY, FILE
- **Record:**
 - A **set of data** of **different types** is held under a **single entity**
- **Arrays:** *use the technical terms of "index", "lower bound", and "upper bound"*
 - **lower bound:** the index of the first element in array
 - **upper bound:** the index of the last element in array
- **Files:** *Why files are needed?*
 - so the data is saved when the **program terminates**
 - so the data can be accessed **next time** the program is run
 - so the data can be "**permanently stored**"
- **Abstract Data Types:** a **collection of data** and a **set of operations** on those data
- **Queues:** 记得先判断队列是否为空后再取队头
 - *Features of a queue*
 - Each queue element contains one data item
 - A pointer to the front of the queue; A pointer to the end of the queue
 - Works on First-In-First-Out bases

- May be **circular**
- *How to implement a queue using an array? -> always declare the type + length of variables/arrays*
 - Declare an **array**
 - Declare **integer variables** for `FrontOfQueuePointer`, `EndOfQueuePointer`, `NumberInQueue`, `SizeOfQueue` (the maximum size of the queue)
 - Initialize `FrontOfQueuePointer` and `EndOfQueuePointer` to represent an empty queue
 - Initialize `NumberInQueue` to 0 and `SizeOfQueue` to the maximum size
- **Stack**: 记得先判断栈是否为空后再POP
 - *Features of a stack*
 - Each stack element contains one data item
 - A pointer to the top of the stack; A pointer to the bottom of the stack
 - Works on First-In-Last-Out
 - *How to implement a stack using an array?*
 - Declare an **array**, the number of elements corresponds to the size of the required stack
 - Declare **integer variables** for `TopOfStack`, `BottomOfStack`, `NumberInStack`, `SizeOfStack`
 - Initialize pointers and variables to indicate empty stack
 - Attempt to describe **Push** and **Pop** operations
 - Pop and Push routines need to check for **full** or **empty** conditions
- **Linked List**
 - *Features of a linked list*
 - Each node contains data and a pointer to the next node
 - A Pointer to the start of the list
 - Last node in the list has a null pointer
 - Data may be added / removed by manipulating pointers
 - Nodes are traversed in a specific sequence
 - Unused nodes are stored on a free list
 - *How to implement a linked list using an array?*
 - Define a **record** type with fields for data and pointer
 - Declare **one** array of the defined record type
 - Declare **integer variables** for `StartPointer`, `NextFreePointer`
 - Declare a **integer value** for `NullPointer`
 - Routines are needed to add / delete / search
 - *How to insert an element in the linked list?*
 - Check for a free node
 - Search for the correct insertion point
 - Assign data value to the node pointed by the start pointer of the free list

- Change of pointers of related nodes
- Start pointer in the free list moved to the point to next free node
- *Advantage*: pointers determine the ordering of data -> easier to add / delete data
- *Disadvantage*: need to store pointers as well as data -> more complex