WHAT IS DATA STRUCTURE
IN C++

The Need for Data Structures

Data structures organize data
$\Rightarrow$ more efficient programs.

More powerful computers $\Rightarrow$ more complex applications.

More complex applications demand more calculations.

Complex computing tasks are unlike our everyday experience.

Organizing Data

Any organization for a collection of records can be searched, processed in any order, or modified.

The choice of data structure and algorithm can make the difference between a program running in a few seconds or many days.
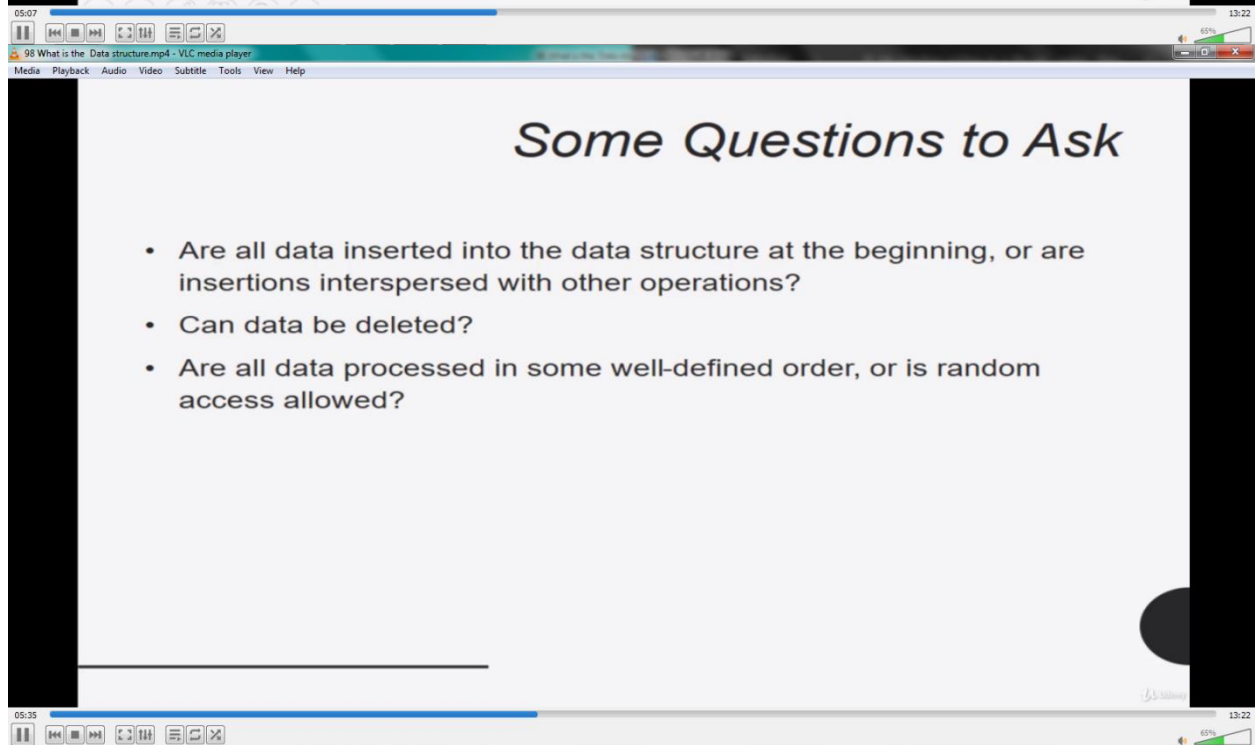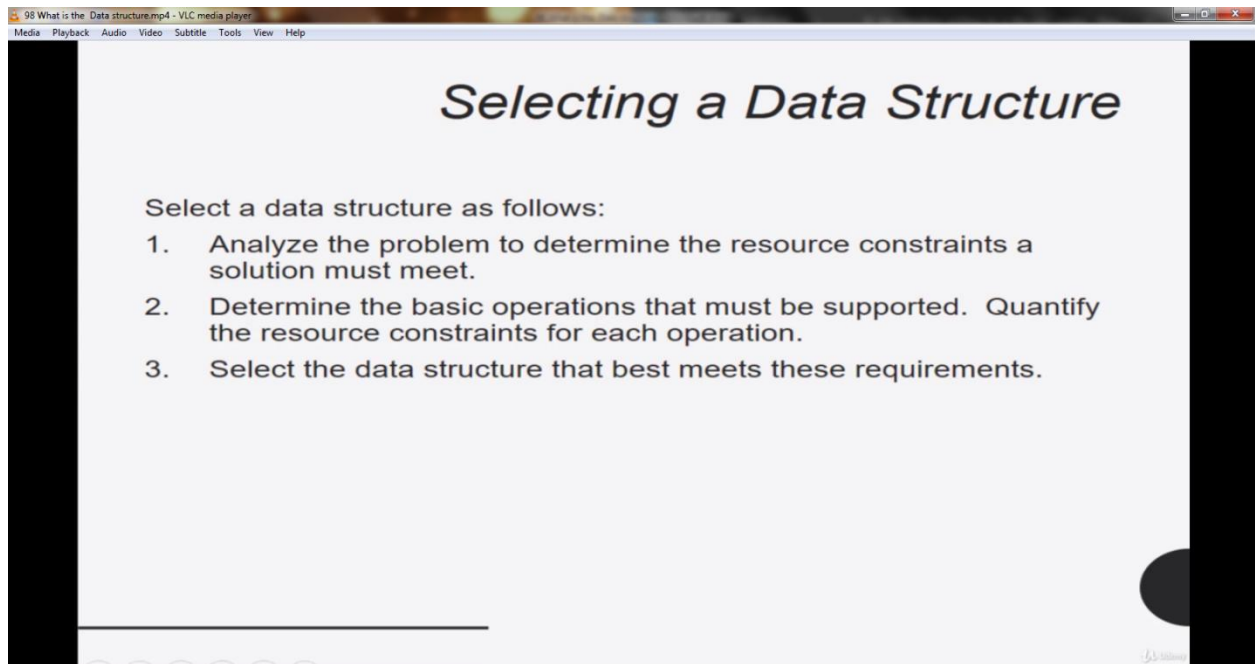

Efficiency

A solution is said to be _efficient_ if it solves the problem within its _resource constraints_.

- Space
- Time

- The _cost_ of a solution is the amount of resources that the solution consumes.

Selecting a Data Structure

Select a data structure as follows:

1. Analyze the problem to determine the resource constraints a solution must meet.
2. Determine the basic operations that must be supported. Quantify the resource constraints for each operation.
3. Select the data structure that best meets these requirements.

Some Questions to Ask

- Are all data inserted into the data structure at the beginning, or are insertions interspersed with other operations?
- Can data be deleted?
- Are all data processed in some well-defined order, or is random access allowed?

Answer one → at the beginning all data cannot be inserted into the data structure at the beginning, And no insertions will not interspersed with other operations.

Answer two → yes data can be deleted.

Answer three → random access is allowed.

Data Structure Philosophy

Each data structure has costs and benefits.

Rarely is one data structure better than another in all situations.

A data structure requires:

- space for each data item it stores,
- time to perform each basic operation,
- programming effort.

Data Structure Philosophy (cont)

Each problem has constraints on available space and time.

Only after a careful analysis of problem characteristics can we know the best data structure for the task.

Bank example:

- Start account: a few minutes
- Transactions: a few seconds
- Close account: overnight

Data Structure

- A <u>data structure</u> is the physical implementation of an ADT.
  - Each operation associated with the ADT is implemented by one or more subroutines in the implementation.
- <u>Data structure</u> usually refers to an organization for data in main memory.
- <u>File structure</u> is an organization for data on peripheral storage, such as a disk drive.

09:16   13:22

Logical vs. Physical Form

Data items have both a <u>logical</u> and a <u>physical</u> form.

<u>Logical form</u>: definition of the data item within an ADT.
  - Ex: Integers in mathematical sense: +, -

<u>Physical form</u>: implementation of the data item within a data structure.
  - Ex: 16/32 bit integers, overflow.

09:46   13:22

Stacks

LIFO: Last In, First Out.

Restricted form of list: Insert and remove only at front of list.

Notation:
- Insert: PUSH
- Remove: POP
- The accessible element is called TOP.

Queues

FIFO: First in, First Out

Restricted form of list: Insert at one end, remove from the other.

Notation:
- Insert: Enqueue
- Delete: Dequeue
- First element: Front
- Last element: Rear

Queue Implementation (1)