CS 234: Data Types and Structures

Naomi Nishimura

Module 1

Date of this version: July 12, 2019

WARNING: Drafts of slides are made available prior to lecture for your convenience. After lecture, slides will be updated to reflect material taught. Check the date on this page to make sure you have the correct, updated version.

WARNING: Slides do not include all class material; if you have missed a lecture, make sure to find out from a classmate what material was presented verbally or on the board.

Introduction

Course goal: Solve problems using complex manipulation of data.

Topic	**First-year CS**	**Beyond first year**
Code	From scratch by one person	From multiple sources integrated together
Design recipe	Plan/code separation	Plan/code and user/provider separation
Data	Simple ways of structuring data, often built-in	Complex ways of manipulating data, often customized
Resources	Introduction to efficient use of time	Analysis of time use

Key ideas

We can handle complexity by taking the following steps:

- Separate work done by different people or groups (roles)
- Separate planning and coding (stages)
- Use analysis to determine differences among options before any coding is done

Understanding roles

Example: Building a house

Component-user role:

- Decide what parts are needed
- Specify functionality

Component-provider role:

- Agrees on a contract for both user and provider
- Provides a part that satisfies the functionality requirements

Roles in CS 234

Users and providers agree on the functionality of complex ways of manipulating data.

An **abstract data type (ADT)** specifies one or more types of data items and operations on those items.

Comparing roles:

- Both user and provider agree on the ADT.
- A user makes use of the ADT operations without knowing how they are implemented.
- A provider implements the ADT operations without knowing how they are used.

Note: In this course, the term **implementation** does not imply coding has taken place.

Ideas used in separation into roles

Modularity is the division of a program into independent, reusable pieces.

Data hiding is the protection of data from other parts of the program.

Modularity and data hiding in CS 234

For each ADT operation, preconditions and postconditions are given.

A **precondition** is a requirement that must be satisfied for an operation to be guaranteed to work.

A **postcondition** is a guarantee of the outcome of an operation being executed.

An ADT does not include details of how items are stored or how the operations are implemented.

Note: In contrast, a **data type** is a data storage format for a programming language, such as a string.

View of ADTs by users and providers

ADTs are like contracts or interfaces between users and providers:

- The user ensures the preconditions are satisfied when an operation is used.
- The provider ensures the postconditions are satisfied in the implementation of the operation.
- The user of an ADT does not need to know details of how the ADT is implemented.
- The provider of an ADT does not need know details of how the ADT is used.

An ADT allows:

- division of labour between user and provider;
- separation of manipulation of data from the rest of the solution;
- code reuse (many problems use the same common ADTs); and
- updates to an implementation without requiring the user to make changes.

Separation into both roles and stages

You will learn what steps to take in each of the four table entries:

- Roles: user and provider
- Stages: planning and coding

Each set of steps will be summarized as a “recipe” for that entry.

	User	**Provider**
Planning		
Coding		

Make sure you know the role and the stage of each task you complete.

Planning using models

Planning takes place before any coding is done.

Both user and provider choose among options, where the options for the user are generated by the provider.

To analyze options without using coding, both user and provider determine costs using the following mathematical models:

- a **model of computation**, which captures the essence of computation, and
- a **memory model**, which captures the essence of memory use.

The model of computation used in the course is **pseudocode**, a way of describing an algorithm without specifying a particular machine or programming language.

A memory model is a simplified representation of how computer memory is organized and accessed.

Planning by the user

Given a problem to solve, the user will:

- Determine types of data and operations used on the data
- Choose, modify, or create an ADT
- Develop an algorithm in pseudocode, using ADT operations
- Calculate the cost of the algorithm in terms of the costs of the ADT operations

To proceed further, now the user needs information from the provider.

Once the provider has provided various options (packages of costs of various operations), the user can choose which option is best.

Planning by the provider

The user will specify which ADT is needed.

The provider figures out how to store, access, and modify the data used by the ADT operations, before coding. There may be different options, with different costs for different operations.

Each option consists of:

- an **implementation**, which is a way of storing the data items, and
- an **algorithm** for each operation.

For our purposes, an **algorithm** is a sequence of steps.

A **data structure** is an implementation that specifies how to store data in the computer's memory.

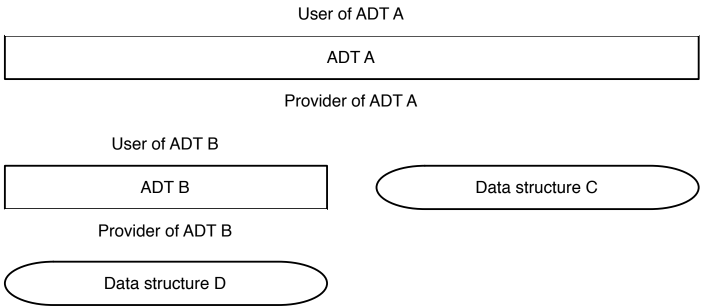
More than one possible algorithm may be possible for a particular operation on a particular data structure.

Layers of implementations

An ADT may be implemented using a single data structure, multiple data structures, a single ADT, multiple ADTs, or a combination of ADTs and data structures.

As a result, there may be layers of implementations, with data structures on the bottom layer.

The provider of one ADT may be the user of another ADT.



Roles/stages

	User	**Provider**
Plan	Agreement on ADT	
	Pseudocode using ADT	*ADT use unknown*
	Data structure unknown	Pseudocode for data structure
	Algorithms unknown	Pseudocode for operations
Code	Agreement on code interface	
	Code using ADT	*ADT use unknown*
	Data structure unknown	Code for data structure
	Algorithms unknown	Code for operations

Recipes for planning

Recipe for user/plan (solving a problem):

1. Determine types of data and operations.
2. For each type, choose/modify/create an ADT.
3. Develop a pseudocode algorithm using ADT operations.
4. Calculate the algorithm's cost with respect to costs of operations.
5. Using information from the provider, choose the best option.

Recipe for provider/plan (choosing among implementations):

1. Create pseudocode of various options for data structures and algorithms to implement the ADT and its operations.
2. Analyze the cost of each operation for each implementation.
3. Provide options for packages of operation costs.

Recipes for coding

Recipe for user/code (code solution using ADT operations):

1. Agree on the code interface.
2. Code a solution to the problem using the ADT.

Recipe for provider/code (code implementation of ADT operations):

1. Agree on the code interface.
2. Code the chosen data structure and algorithms implementing the ADT.

Course goals

Basic steps:

- [User-plan] Given a problem, determine a type of data and operations.
- [User-plan] Given a type of data and operations, choose an ADT.
- [User-plan] Given a problem, write pseudocode using ADT operations.
- [Provider-plan] Given an ADT, write pseudocode implementing operations using different data structures.
- [User/Provider-plan] Given pseudocode, analyze the worst-case running time.

Related topics:

- Common ADTs for different kinds of data
- Common data structures
- Analysis of pseudocode
- Understanding storage in memory
- Handling data features (general, orderable, “digital”); structure

Course logistics

Components:

- Lectures (all slides online but **without verbal and on-board explanations**)
- Readings (website materials; mini-textbook)
- Python sessions and materials
- Self-checks (only in class)
- Assignments
- Exams

Tools used:

- Course website: Schedule, resources, news; lecture summaries in advance and some updates after
- Piazza: Questions
- MarkUs: Hand in assignments; check marks

How to benefit from other sources

Different sources may:

- Use different names for ADTs and operations
- Use the same names for ADTs with different sets of operations
- Use the same names for operations with different preconditions and postconditions
- Assign different costs

Conclusion: Use the course materials for definitions of ADTs, data structures, and costs.

Most sources cover the key basic concepts:

- Separation between user and provider (not using those names)
- Choosing among ADTs
- Choosing among data structures and algorithms
- Analyzing choices

Conclusion: Use other sources, if needed, for additional examples on how to create ADTs and how to analyze and choose data structures and algorithms.

Work to do this week

- Start on Python review (see Python page on website and schedule for sessions).
- Read Python 2 versus Python 3 page if you studied Python a while ago.
- Sign up for Piazza.

Module summary

Topics covered:

- Roles: user and provider
- Abstract data types
- Stages: planning and coding
- Planning using models
- Course goals
- Course logistics