

Data Structures on the Web

NGI PROJECT PRESENTATION for SUMMER 2002

Presented by Neha Kumar
Advised by Prof. Michael Clancy

Agenda

- Data Structures today
- What is WISE?
- Data Structures in WISE
- Why WISE?
- Long-term objectives

Data Structures today

- 3 hours of lecture - They listen
- 2 hours of lab - They start to experiment...
- 1 hour of discussion - They listen
- hours of homework - They struggle

What they don't do enough

- Learn by experimenting
- Test their learning
- Recognize gaps in their knowledge

What is WISE?

- Web-Inquiry Science Environment
- Collection of activities for students
- Online database of learning material

Activities in WISE

- Online reading
- Assessment exercises
- Quizzes
- Online discussions
- Brainstorming sessions

The WISE CS3 Course Portal

The screenshot shows a Mozilla browser window displaying the course portal for CS 3: Introduction to Symbolic Programming. The browser's address bar shows the URL <http://www.ucwise.org/coursePortal/>. The page title is "CS 3: Introduction to Symbolic Programming" and it is for the "Summer 2002 Section 1". A welcome message reads "Welcome, cs3-aw!".

On the left side, there is a navigation menu with the following items:

- Home
- General Info
- Announcements
- Group
- My Grade
- Calendar
- Manage My Files
- Reading Assignment
- Special Event
- Newsgroup: cs3
- Software Help Desk
- Logout

The main content area features an "Announcement Lists" section with a message dated 2002-08-07: "Slides and links relating to this week's CS culture lectures and other topics of interest are accessible [here](#)."

Below the announcement is a "Calendar section:1" with a "Week View" selected. The calendar shows the following schedule:

Monday, 07-08	Tuesday, 07-09	Wednesday, 07-10	Thursday, 07-11	Friday, 07-12
<p>Starting to think recursively</p> <ul style="list-style-type: none">Reviewing material in chapter 11Design individual procedures to make their common pattern more obvious.Produce recursive procedures from the individual procedures.How do I know if work?Design some more recursive procedures.Homework for Tuesday <p>"Difference between Dates" miniproject</p>	<p>Working with more complicated recursive procedures</p> <ul style="list-style-type: none">Here are important things to note for todayReviewing "Starting to think recursively"Design a recursion using simple cases and clues.Design recursive procedures with two arguments.Design recursions that examine more than one word in a sentence at a time.Try some accumulating recursionsHomework for Wednesday	<p>exam 1</p> <p>Working with even more complicated recursions!</p> <ul style="list-style-type: none">Do some sorting.Analyze a complicated (buggy) recursion.Think about tree recursion.Here's a procedure to do thorough, several.Homework for Thursday	<p>Working with the "Roman Numerals" case study</p> <ul style="list-style-type: none">Reviewing recursion and the "Roman Numerals" case studyHere's more information about the Modeler.Work with the case study program.Have a bit more recursion practice.Homework for tomorrow: reflect on the case study.Homework for Tuesday	<p>Working with the "Roman Numerals" case study continues from last time</p> <p>July 12 quiz</p> <ul style="list-style-type: none">Quiz questions are here. <p>"Number spelling" miniproject</p>

The Learning Environment

Course Portal - Mozilla (Build ID: 2002052917)

WISE CS3: Summer 2002

Design a recursive procedure using simple cases and clones.

Design recursive procedures with two arguments.

Design recursive procedures that examine more than one word in a sentence at a time.

- Consider the letter-pairs example from this book.
- Design wanted?

Try some accumulating procedures.

Homework for Wednesday

(c) 2002 WISE

Simple Scheme describes the procedure `letter-pairs`, given a word, `letter-pairs` returns a sentence that contains all pairs of consecutive letters in the word. For example, `(letter-pairs 'george)` returns `(ge eo or rg ae)`. Here's their code.

```
(define (letter-pairs wd)
  (if (<= (count wd) 1)
      '()
      (sentence
        (first-two wd)
        (letter-pairs (butfirst wd)))))
```

Helper procedure:
RETURN a sentence that contains the first two letters of the given word (which must contain at least two words).

```
(define (first-two wd)
  (word (first wd) (first (butfirst wd))))
```

The interesting thing about this example is not the recursive call, in which the argument is `(butfirst wd)` as in many of the other examples. It's the base case, which tests for a word with 0 or 1 character rather than just an empty word. The reason for this base case is to protect the call to `first-two`, which is going to use the second character in its argument. If the procedure were coded as

```
(define (letter-pairs wd)
  (if (empty? wd)
      '()
      (sentence
        (first-two wd)
        (letter-pairs (butfirst wd)))))
```

a crash in `first-two` would result.

search paths more obvious.

- Produce recursive procedures from the individual procedures.
- How do I know it works?
- Design some more recursive procedures.
- Homework for Tuesday

Recursive starting to the recursive!

- Design a recursion using simple cases and clones.
- Design recursive procedures with two arguments.
- Design recursive that examines more than one word in a sentence at a time.
- Try some accumulating recursions.
- Homework for Wednesday

Complicated recursion!

- Do some sorting.
- Analyze a complicated (buggy) recursion.
- Think about the recursion.
- Have a procedure to do through several.
- Homework for Thursday

- Have more information about the Master.
- Work with the case study programs.
- Have a bit more recursion practice.
- Homework for tomorrow.
- reflect on the case study.
- Homework for Tuesday

Summer 2002 Section 1

Welcome, cs3-aw!

ossible [here](#).

Syllabus

Friday, 07-12

Working with the "Roman Numerals" case study continues from last time

July 12 quiz

- Quiz questions are here

"Number spelling" miniproject

Document: bone (0.003 secs)

Data Structures in WISE

- Course divided into topics called “Projects”
- Each “Project” has its set of “Activities”
- Each “Activity” is divided into “Steps”

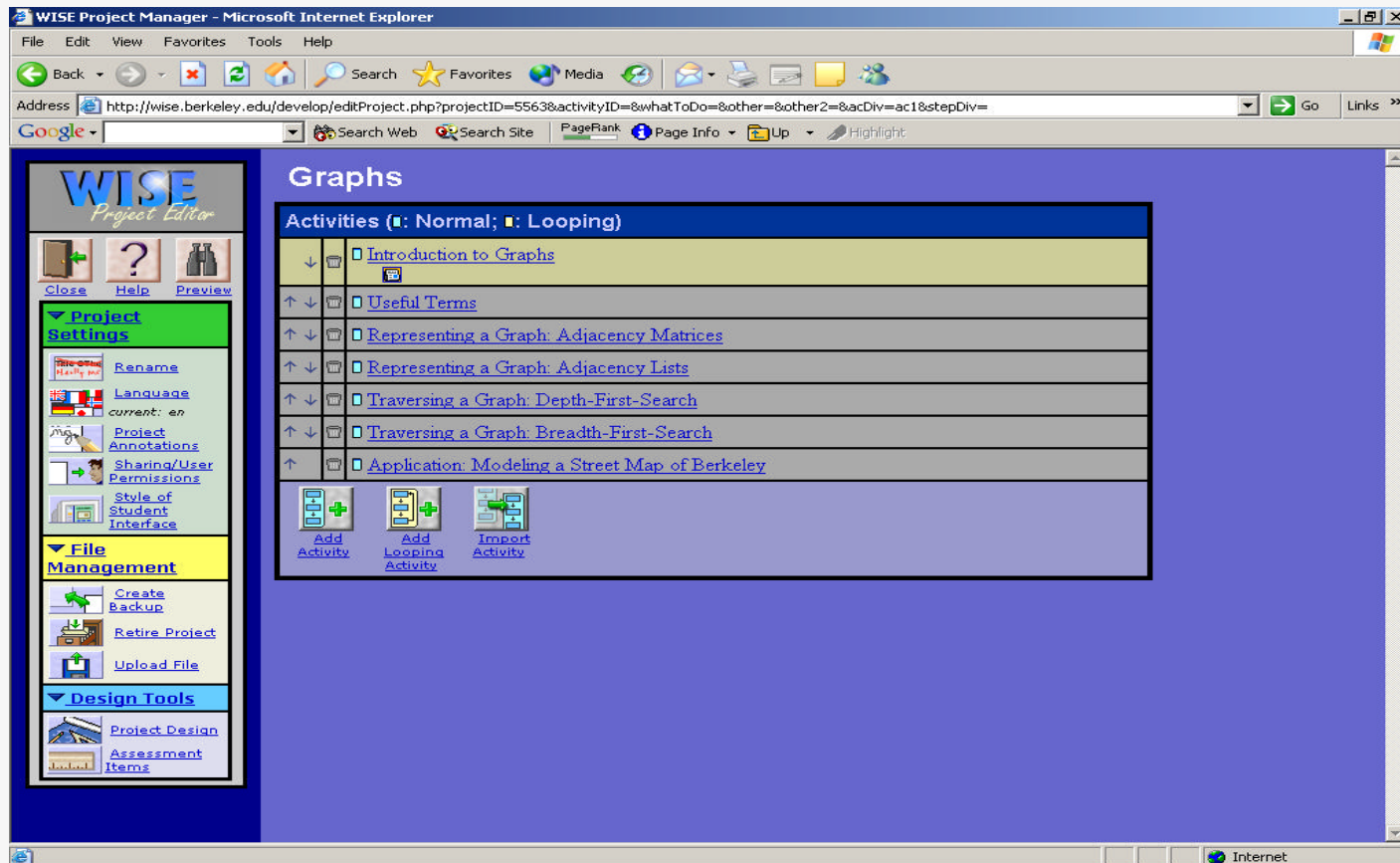
WISE Projects for 61B

- Java Programming
- Activation Records
- Inheritance
- Exceptions
- Testing
- Linked Lists & Arrays
- Game-tree Search
- Asymptotic Analysis
- Hash Tables
- Trees
- Binary Heaps
- Binary Trees
- 2-3-4 Trees
- Graphs
- Sorting Algorithms
- Randomized Analysis

Close-up on Graphs

- Project: Graphs
 - Introduction
 - Terms to Know
 - Representation: Adjacency Matrices and Lists
 - Traversal: DFS and BFS
 - Homework: Modeling a Map of Berkeley

Creating a project in WISE



WISE and 61B: how they fit

- Activities proceed in small steps
- WISE activities enable thorough learning
- Progress is easy to monitor
- Weaker students get individual attention
- Students benefit from group discussions

- They learn by doing!

Why WISE?

Studies done over the summer show

- Students find WISE more enjoyable
- They perform better overall

“I like the way lectures are combined with problems online. I am learning more about CS than I ever did before. This method of teaching is very practical and works a lot better than regular lecture/lab/discussion sections...”
- Summer CS3 Student

Long-term Objectives

- Adapt WISE to all lower-division CS courses
- Export curriculum to UC Merced