Module 1: Data Organization

Knowledge of the ways that data is commonly organized on a computer will be helpful to you, both as you work with new types of data and applications and as you develop your html and programming skills. In this module, you'll learn about ways that data is categorized (data types), organized (data structures), and named (including variables). You'll apply your knowledge in designing HTML web pages and in searching for data on a computer and on the world wide web.

Learning Goals. After this module you should be able to

1.1 recognize examples of data types in every-day life and in the computing context and illustrate how properties associated with familiar data types on a computer can influence the behaviour of computer applications which act on these data types;

1.2 recognize examples of data structures in every-day life and in the computing context and classify data structures as networked, hierarchical, and/or tabular when applicable

1.3 use html to design networked, hierarchical and tabular structures in webpage content, and use analysis and debugging skills to correct and avoid html errors;

1.4 explain how names convey properties and/or structure of data, with particular attention to names in computing environments such as domain names, file names, URL's or email addresses; and

1.5 explain how variables are used to ease data management and to describe actions on data, and be able to use variables for these purposes in familiar contexts.

1.1 Data types: classifying data

We come to know things by differentiating them from other things. The traditional game "Animal, vegetable, mineral" comes to mind: one person thinks of an object, the other tries to determine the object by asking questions about its type:

Q: "Is it animal, vegetable, or mineral?"

A: "Animal."

Q: "Is it a mammal?"

A: No.

Q: "Is it a fish?"

A: Yes.

...

Successive questions narrow down the type of the object until the questioner succeeds in determining the object or has asked 20 questions.

Plants, animals, food, diseases— virtually everything around us— has been classified into types. Knowing the type of a thing can help us know about its properties (an adult insect has three pairs of legs and a body segmented into head, thorax, and abdomen) and predict its behavior (is the disease genetic? is the food perishable?). Knowing the type of an object can also influence our behaviour with respect to that object (how should we treat this disease? should we put the food in the refrigerator?).

Similarly, data on a computer is often naturally distinguished by type. For example, as we discuss in more detail below, files could be of several types: gif files, jpg files, txt files, html files and so on. Knowing the type of a data item can help you understand what **properties** (sometimes called **attributes** or **modifiers**) it has, and can also help you predict what happens when you do something with the data (for example, when you click on it). Following are several examples. We use the word "type" informally throughout but note that in certain contexts (such as programming languages, as we will see in a later module), the notion of data type is quite formal.

- **Types of data in folders.** Open up some folders on your computer. What do you see? Most likely some files, perhaps some more folders, and possibly some applications. These are three types of data objects that can reside in folders. Can you distinguish objects of each type from each other? If so, how?

- **Properties of files.** Properties associated with files are quite different than properties associated with folders or applications. On your computer you can probably figure out some properties associated with your files. (On a mac, select (click once on) the file, then choose File > Get Info. On Microsoft Windows, select the file, then choose File > Properties.) Examples of properties typically associated with a file include: the date when it was last modified, its type or kind (e.g., txt or pdf document), its size, its name and extension and what application it should be opened with. Other properties of a file determine what users can read the file or modify (write to) the file.

- **Types of data in documents.** A document may contain different types of data: text, images, or tables, for example. Consider the text within a document. One property associated with the text is its font. What are some other properties that the text has? How would you change these properties, using your favourite document formatting system?

- **Preferences.** Preference settings are associated with typical applications that you use, such as a web browser. Your preference settings influence how the application behaves when you use it. Many of your preferences can be set by either checking a box or leaving it unchecked. Your preference is indicated as checked or unchecked. You could think of this as choosing between yes or no; true or false. Data like this is considered to be **Boolean** data. That is, preference settings are examples of data and the type of each preference setting is Boolean.

Exercises:

1.1.1 Can you figure out how to determine properties of folders or applications on your computer? Compare the actions available under the File menu for files with the actions available for folders. What differences do you notice?

1.1.2 Tired of being confused by your computer? Try to confuse your computer for a change! Choose a file, let's say a pdf file on your computer. Make a copy (so nothing happens to the original file). Now, change the copy to a txt file by changing its extension to .txt instead of .pdf. (You may need to ensure that preferences are set so that you can indeed see and edit the extension.) Does this action change your view of the file on the computer? Has the default application for opening the file changed? Have any other properties of the file changed?

1.1.3 Shapes are a type of data you work with in many contexts, including on a computer. What properties do you associated with simple shapes, such as circles, squares, spheres, and so on? You may already be familiar with applications that create and

manipulate shapes. Do you recall any properties associate with shapes in these applications?

Curves or ovoids are more complex shapes. Experiment with properties of such shapes when you use paint or image creation programs.

1.2 Data structures: organizing data

Organizing data, whether on a computer or in a file cabinet, helps when we later need to find information. Data organization also helps to clarify relationships between data items, by putting related items in the same folder, for example. Figure 1.1 shows some visual examples. Although the examples are from very different contexts, some of the ways in which the data is structured, or organized, share common attributes.

How does the way the data is structured reflect relationships between individual data items? What do the data structures tell us about the underlying data? Do some classification yourself! Find pairs of these data structures that you think share common attributes. List those attributes. Group your examples into clusters. Is there any overlap? Why or why not?

A *data structure* describes relationships among data. The examples of Figure 1.1 include some the following:

- **Networked data structure:** describes general relationships between pairs of data items. Visually, links (lines or edges) are often used to indicate a relationship between a pair of items. A networked data structure is often called a network or a graph.

  Links may be **directed** or **undirected**, depending on the type of the relationship. Figure 1.1 parts (a) and (d) show representations of networks with undirected links. Directed links are often represented as arrows. The world wide web is a gigantic example of a network with web pages being data items and links indicating relationships between pages. In this example links are directed: if page A points to page B it may not necessarily be the case that page B points back to page A.

- **Hierarchical data structure:** in which relationships between data items resemble those in a tree. That is, branches emanate from a single "root"; data items are organized at the branch points and/or at the "leaves" of the tree. A hierarchical data structure is often called a tree or a directed tree.

  Examples include phylogenetic trees, organizational charts, or family trees. Phylogenetic trees illustrate relationships among biological species or groups of species. Extant species (i.e. species which are not extinct) are arranged as "leaves" in the

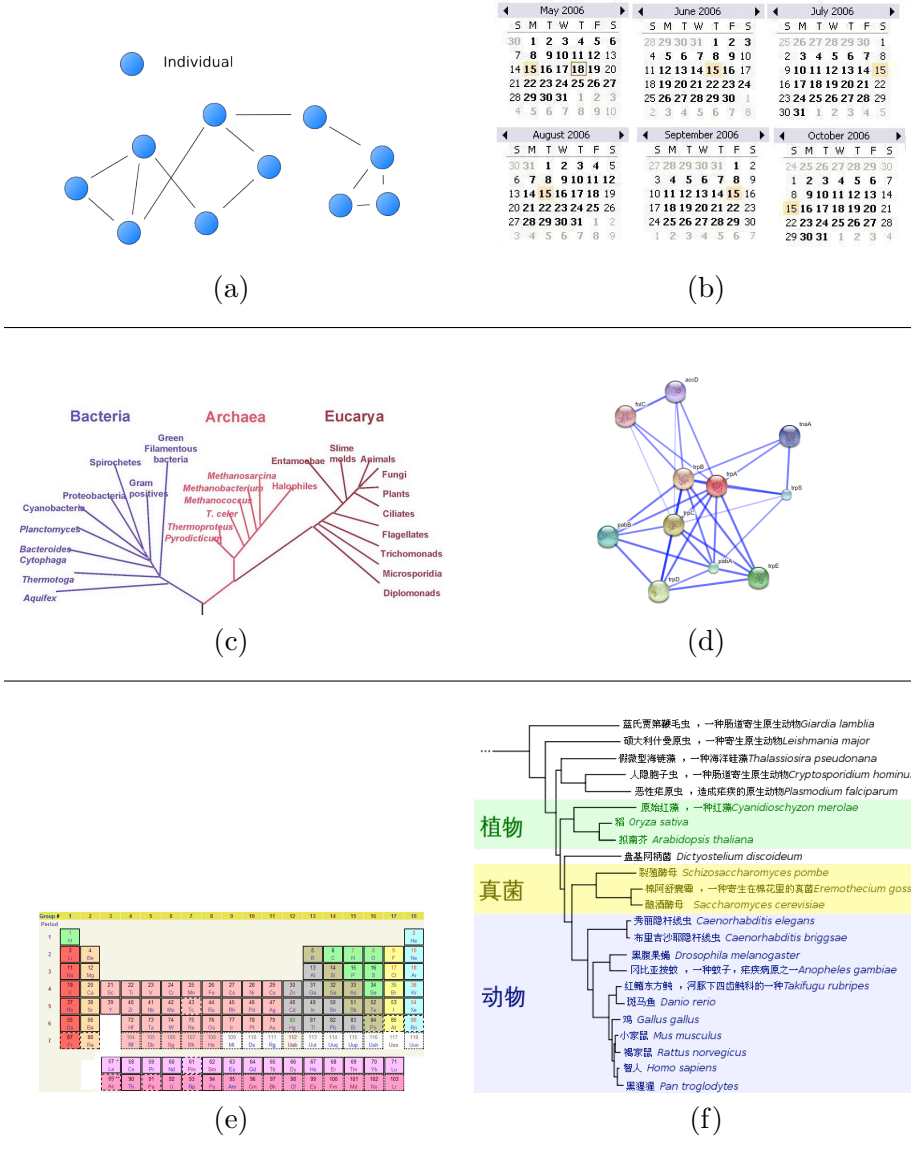


Figure 1.1: Visual images of data structures. Images were obtained from the Wikimedia Commons (http://commons.wikimedia.org/wiki/). (a) Social network diagram (File:Social-network.png). (b) Calendar (File:2006_Calendar.JPG). (c) Phylogenetic tree of life (File:PhylogeneticTree.png). (d) Protein-protein interaction network (File:STRING_network_image.png). (e) Periodic Table (File:IUPAC_Periodic_Table.PNG). (f) Eukaryote tree (File:Eukaryote tree.svg).

tree, and internal nodes or branch points represent ancestral species from which the extant species evolved (via genetic mutations, for example).

The tree of Figure 1.1 (c) is oriented with the root at the bottom and leaves at the top. Trees are often oriented with the root at the top or at the left (as in Figure 1.1 (f)) or even with the leaves arranged in circular fashion[1]. (There are many visually distinct ways of representing hierarchical data structures, such as Treemaps[2]).

- **Tabular data structure:** in which data items are organized into a table. The table may be 1-dimensional, 2-dimensional, or have more than two dimensions. A tabular data structure is often called an **array** or a **table**. A 1-dimensional table may be called a *linear* data structure because its elements can be organized along a line. Calendars are often displayed in tabular format and in fact can be laid out in a **nested** fashion. For example, part (b) of Figure 1.1 has a 2-dimensional table for the six months with three rows and two columns. Each entry in this table is itself a 2-dimensional table, which shows the dates of each day of the week for one month. A timeline can be thought of as a 1-dimensional table.

This list is by no means exhaustive. Think about other ways that data or objects you are familiar with are commonly organized.

Exercises:

1.2.1 Books and articles typically organize their content in more than one way. Can you think of examples of networked, hierarchical, or tabular (linear) organization in books? How is each type of orderings useful when you use the book?

1.2.2 How would you describe the organization of files and folders on your computer?

1.2.3 A mathematical expression can be viewed as having a hierarchical structure. To see this, consider the expressions

$$(((1+2)+(3-4))*((5+6)+(7-8))) \tag{1.1}$$

and

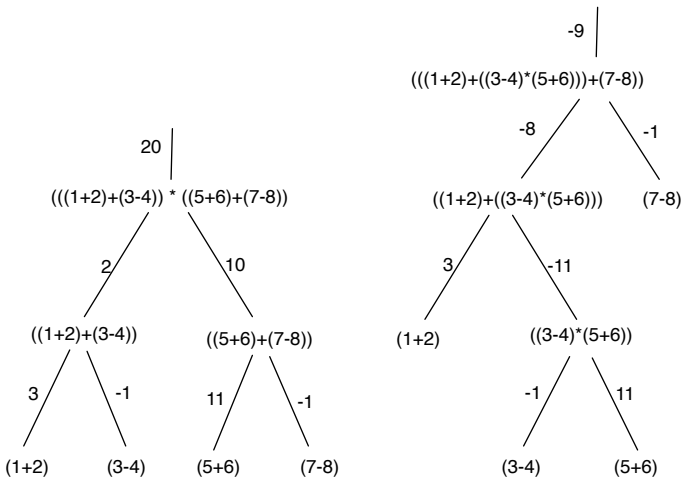
$$(((1+2)+(((3-4))*(5+6)))+(7-8)). \tag{1.2}$$

(Here, the $*$ denotes "multiply".) To evaluate these expressions, you have to first evaluate the innermost terms, then work outwards, guided by the parentheses (i.e., ('s and)'s). Figure 1.2 shows the corresponding hierarchical structures.

[1] see `http://www.utexas.edu/features/graphics/2008/tree/tree3.jpg`
[2] see `http://www.cs.umd.edu/hcil/treemap/`

Can you draw a diagram which illustrates the hierarchical structure of the following mathematical expression?

$$((1+2)+(((3-4)*(5+6))+(7-8))).$$



(1.1) (1.2)

Figure 1.2: Hierarchical view of two mathematical expressions. Labels on the links, or edges, show the value of the expression at the node just below.

1.3 From concepts to practice: HTML

HTML (hypertext markup language) is used to specify how information (data) is organized in a web page. In this sense, HTML can be used to describe data structures. HTML also provides ways to describe properties of the data such as the colour and font of text in the web page. You can use **HTML tags** to specify networked, hierarchical and tabular structures and can use **HTML attributes** to specify properties of data such as images and text.

The HTML lab provides a hands-on introduction to HTML and the lecture slides provide several examples of the use of HTML tags and attributes.[3]. Another way to learn HTML is to look at the source code of web pages you like. You can do this by right-clicking (or

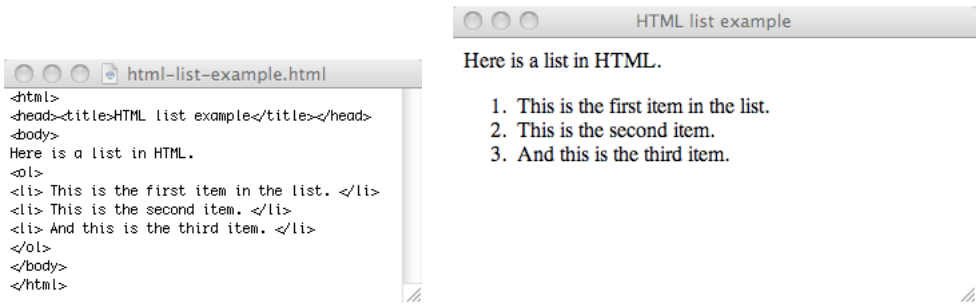
[3]You can find a reference guide of tags and attributes at:
`http://www.devx.com/projectcool/Article/19816`

ctrl-clicking) on the page and choosing "View Source". The following very brief notes focus on specifying lists and tables in HTML.

1.3.1 HTML lists

Figure 1.3 shows an example of a numbered list and its specification in HTML. The tags `<ol>` and `</ol>` delimit (i.e., denote the start and end of) the list and the tags `<li>` and `</li>` delimit each item in the list. Can you guess how to add a fourth item to the list?

Figure 1.4 shows an example in which one list is nested inside another. How you might you try to write the specification of such a nested list in HTML?



(a) (b)

Figure 1.3: Lists in HTML. (a) HTML specification of a list with three items. (b) The list specified in (a), as viewed in a browser.

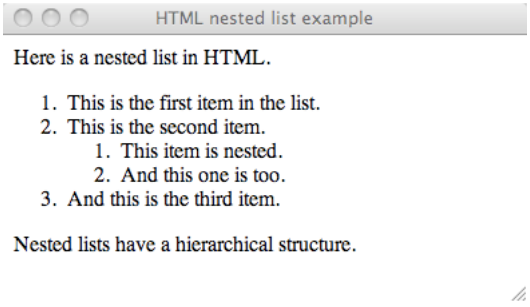
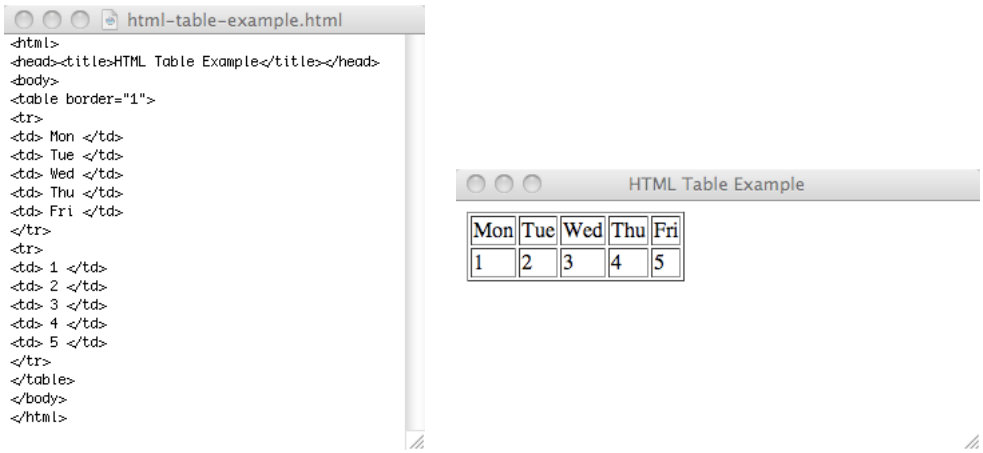


Figure 1.4: Browser view of a nested list. Can you guess at its specification?

1.3.2 HTML tables

Figure 1.5 shows an example of a table and its specification in HTML. The overall table is delimited by the tags `<table>` and `</table>`. Rows are delimited by the tags `<tr>` and `</tr>`, while individual table data items are delimited by the tags `<td>` and `</td>`. How might you make a nested table?



(a) (b)

Figure 1.5: Tables in HTML. (a) HTML specification of a table with five columns and two rows. (b) The table specified in (a), as viewed in a browser.

1.3.3 Analyze and Debug

When designing HTML web pages, you can hone your analysis and debugging skills to correct and avoid html errors.

You will likely design some of your own unique HTML data structures by generalizing from examples you've already seen. In this process, you **build a conceptual model** of specific HTML tags and attributes from the examples available to you. You then use this model to design your own page. If your page does not turn out as expected it might be that you need to refine your conceptual model or it might be that there is an error in your HTML specification. If you are not sure how to so something, you can **search on the web to get more information** on HTML tags and their attributes. You can **experiment** until the page looks as you intended.

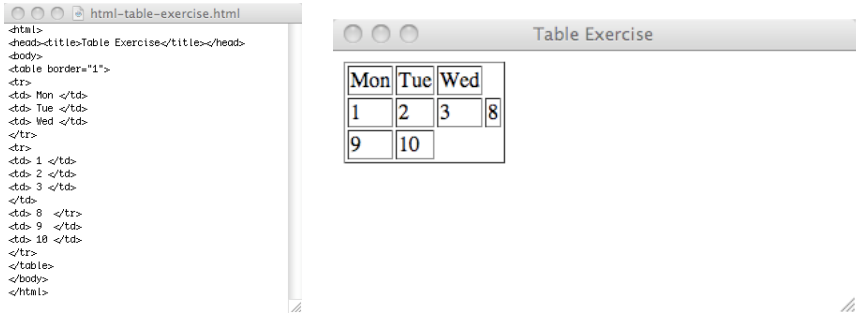
An advantage of working with HTML is that you can **compare your HTML source and**

the formatter web page often as you write the specification, making it easier to debug than when writing all of the specification at once. As the lab emphasizes, if you **lay out your specification cleanly** it will be easier to find missing or misspelled tags.

Exercises:

1.3.1 Figure 1.6 (a) shows an HTML specification of the formatted web page in (b). Something is wrong - the "8" should be aligned under the "1" in the Mon column, the "9" under the "2" and the "10" under the "3". Can you identify the error in the HTML specification?

1.3.2 When writing your own HTML code, take a few moments occasionally to experiment. See what happens when you leave out a tag. Sometimes the results might surprise you. Share your most surprising examples with your TAs and classmates - they provide valuable learning opportunities.



(a) (b)

Figure 1.6: (a) HTML specification of a web page containing a 2-dimensional table. (b) View of the table in a web browser.

1.4 Names: describing data

"What's your name?" This is often the first thing we want to know when meeting someone for the first time. When you see an unfamiliar object, the first thing you're likely to learn in response to the simple question "What's that?" is its name. Indeed, peoples' interest in naming things is probably even more ingrained than our tendency to classify and organize things. Names are often synonymous with **IDs** or **titles**.

Here, we interpret "name" rather broadly and consider the many different ways in which names are useful when describing data in computing contexts.

- *Names are useful in identifying data.* Let's start with names of files on your computer. Typically, a file has a **name** and an **extension**, usually separated from the file name by a dot. (Sometimes, the extension is also considered to be part of the name, sometimes not.) Within a folder, it's not possible on typical systems to give two distinct files the same name and extension. So, names and extensions uniquely identify files in a folder. Can folders have the same names as files? (Can you confuse your computer to be inconsistent on this point? Experiment!)

- *Names can reflect the structure (organization) of data.* Consider the names you use when you want to refer to a file on the world wide web. These names are called **URL's** (universal resource locators). For example, the course web page for this class is `cs.ubc.ca/~condon/wmst201/index.html`. You'll notice that the URL ends with what looks like a file name. On the computer I use, the file index.html for this course web page is stored within one of my folders called wmst201. (A convention in our department is that, by default, folders and files which are available on the world wide web should be kept within a top-level folder (directory) called public_html. In keeping with this convention, my wmst201 folder is within my public_html folder. You cannot tell this from the URL: since public_html is a default it turns out that it must always be omitted from the URL.)

  As another example, these notes on data organization can be found at `cs.ubc.ca/~condon/wmst201/notes/data-org-notes.pdf`. Where do you think the file data-org-notes.pdf is stored? I keep my lecture slides in a folder called "lectures", which is within my wmst201 folder. Slides from the first lecture is in a file called lec1.pdf. What do you think is the URL for this file?

  As you might guess, the term "`~condon`" is in the URL because the files are stored in folders within my account. My account is on a cluster of computers maintained by the Computer Science department at UBC. This cluster has **domain name** cs.ubc.ca, and so cs.ubc.ca appears at the left end of the URL.

  Many other people have accounts on the domain `cs.ubc.ca`. For example Steve Wolfman, who has taught this class in the past, has his web page at `cs.ubc.ca/~wolf`. Overall, as these URL's indicate, many web pages within the computer science department are organized hierarchically within the folders of individual accounts.

While names can be helpful in conveying meaning, it's important to remember that names may not always convey the true meaning. As a concrete example, suppose that I decided to use the name "lec41.pdf" for the lecture slides of the fortieth lecture of this class and to use "lec40.pdf" for the lecture slides of the forty-first lecture. (Don't worry— there won't be that many lectures; this is just a hypothetical example.) This would be a very confusing way to name the files. But if you were to click on

`cs.ubc.ca/~condon/wmst201/lectures/lec41.pdf`

you would get the lecture slides for the fortieth lecture of the class (and not the forty first lecture). In this case, the name and the meaning would not match up.

Exercises:

1.4.1 What features do email addresses and URL's have in common?

1.4.2 On typical computer systems, two files within the same folder cannot have both the same name and extension. It turns out, however, that one file have two different names. Why might that be useful? Can you think of examples in everyday life where something is referred to by more than one distinct name? In such situations, the second name for the file may be called an **alias**.

For example, the course web page for this class has both the name `cs.ubc.ca/~condon/wmst201/index.html` and the name `cs.ubc.ca/~condon/cpsc101/index.html`. Can you find a way to give a file two different names on your computer? Try searching on the web, or consulting with others in the class, for clues as to how to do this if you'd like help.

1.4.3 Recall that the computers within the computer science department have the domain name `cs.ubc.ca`. Computers within UBC have domain name `ubc.ca`. Computers within the department of Art History, Visual Art and Theory have the domain name `ahva.ubc.ca`. Computers at the University of Alberta have the domain name `ualberta.ca`. Can you guess how domain names are structured? (Networked, hierarchical, or tabular?)

Use your conceptual model of domain name structure to guess the domain name of computers in the computer science department at the University of Alberta. To check your model, check the actual URL of University of Alberta's computer science department. Try again, to see if you can guess the URL of the computer science department at U. Manitoba or U. Toronto. (One of them breaks the model...)

1.4.4 Names are useful in HTML to create links within a single web page. See if you can figure out how this is done to create a working table of contents in the course web page that contains logistic information:

`www.cs.ubc.ca/~condon/wmst201/logistic-info.html`.

In closing, we note that good naming practices and data organization are very helpful to humans who need to navigate file and folder structures on a computer. A file name can often be chosen to convey much information about its contents. For example, when making

multiple revisions to a document it can be helpful to save copies periodically and add the date at which the copy was saved to the file name.

1.5 Variables: describing data that changes over time

A name, or title, can be a place-holder. For example, the current Chancellor of UBC is Sarah Morgan-Silvester. But in some of UBC's organizational charts you won't find her name; rather you'll find the title "Chancellor". Often we use titles in this way when the person (or object) being named changes over time. This has the advantage, for example, that when there is a new chancellor, the organizational charts don't have to be redone. In the computing context, particularly when programming, we sometimes call such "placeholder" names **variables**. Here are two reasons why variables are useful in computing contexts:

- *Variables can make data management easier.* For example, email addresses can be placeholders: the head of the Computer Science Department can be reached at head@cs.ubc.ca. You can still reach the head at this address next year, even though it may be a new person by that time.

  As another example, the course home page uses variables to store colours. These variables are then used to set the colour of table rows in the course schedule. This makes it possible, for example, to change the colours of all of the "Monday" rows simply by changing one variable. See discussion in lecture for more details.

- *Variables provide a general way to express actions on data.* In particular, variables enable you to express simple calculations without reference to particular data values. You are probably already very familiar with this from high school algebra. For example, suppose the distance from Vancouver to Seattle is 140 miles. To convert this to kilometers (with room for a little approximation), multiply by 1.62. More generally, we can say "Distance in kilometers is 1.62 times the distance in miles". Or write:

  $$\text{distance in kilometers} = 1.62 * \text{distance in miles}$$

  (where $*$ denotes multiplication). We are using "distance in kilometers" and "distance in miles" as variables, or placeholders in describing the general rule for converting distance in miles to distance in kilometers. If we need to calculate the distance in kilometers to a new city, say Prince Rupert, given that we know the distance in miles— it's 933 miles— we can substitute the number 933 for the variable "distance in miles" to calculate the answer.

You may already be familiar with web pages which convert between currencies, kilometers to miles, and so on. Behind the scenes, these webpages use variables to express the needed calculations.

It is often convenient to use headings of the table rows and columns to refer to individual data items in a 2-dimensional table. For example, if the rows of a table are numbered consecutively starting at 1, and similarly for the columns, then "the data item at row 5 and column 8" uniquely identifies a particular data item in the table. If the table has a name, say T (a rather unimaginative name for a table) then a common convention is to use $T[5, 8]$ to refer to the table entry in row 5 and column 8 of the table. More generally, we use $T[i, j]$ could refer to the table entry in row i and column j of the table. Here, to explain the general convention for referring to table entries, we use variables i and j.

1.6 Perspective

An important aspect of data organization on a computer which we have not touched on much is how to present data to users in meaningful ways. Good data visualization techniques can enable people to uncover important relationships among data elements of interest to them.

An example of a challenge in data visualization is that of visualizing phylogenetic trees—trees that describe relationships among biological species. Biologists today want to understand relationships among thousands of species and ultimately to infer a complete tree of life with millions of species. There is still much controversy as to how species are related, making it useful to have ways to understand differences in trees that represent competing hypotheses as to how species are related.

Tamara Munzner in UBC's Computer Science department is at the forefront in development of methods that enable users to explore large phylogenetic trees and to compare different trees in meaningful ways. Innovations of her TreeJuxtaposer software include the visual cues that aid comparisons across trees, as well as a mode of interaction whereby users can explore a small subtree in detail while maintaining a view of how it fits in the overall tree.

You can read more about the challenges in visualizing phylogenetic trees in a New York Times article by Carl Zimmer[4]. The article includes comments from Tamara Munzner and also cites the Tree of Life web project whose "home team" includes Wayne Maddison, an expert on jumping spiders and faculty member in UBC's departments of Zoology and Botany.

[4]at `http://www.nytimes.com/2009/02/10/science/10tree.html?pagewanted=all`

Project idea:

1.6.1 Visualization of graphs is also a very important topic and an active area of research. Learn more about principles for drawing graphs in aesthetically appealing or useful ways. Experiment with graph drawing tools that you can download on the web.