CS151.02 2010S *Functional Problem Solving*

Class 36: Vectors

Held: Friday, April 9, 2010

Summary: We consider *vectors*, an alternative to lists for storing collections of data.

Related Pages:

- EBoard.
- Lab: Vectors.
- Reading: Vectors.

Notes:

- Reading for Monday: Pairs and Pair Structures.
- Assignment for Wednesday: Assignment 7: Fractals.
- Today will be a "Sam speaks for some time" day.
- Prospies! (I hope.)
- EC for today's overcrowded lunch with Dan Garcia.
- EC for tonight's opening reception and/or talk with Kluber and Simon.
- Any thing else of interest?

Overview:

- Problems with lists.
- A solution: Vectors.
- Behind the scenes: How Scheme implements vectors.
- Important vector procedures.

Data Types

- At the start of the semester, we decided that "basic values and operations on those values" are key to writing algorithms.
  - We tend to use the word "type" to express these two concepts.
- We've seen a variety of characteristics of types in 151.
  - Some types are defined in terms of a list of possible values or a simple construction method: Character, Boolean, RGB color, etc.
  - Some types that can potentially be defined recursively: Drawings-as-value, Maybe integers, Lists.
  - Some types are designed to collect other kinds of values: Lists. (Okay, maybe that's it for now.)
- We're about to learn a few more. Vectors today. Pair structures (particularly trees) on Monday.

List Deficiencies

- Now that we've worked with lists for a while, we've identified some things that make lists inappropriate for some situations.
  - List are *expensive* to use; to find the length of a list or to access an element late in the list, you need to cdr down the list.
  - Lists are *fixed*; you can't easily change an element of a list.
- At the same time, there are some advantages to lists:
  - Lists are *dynamic*; it is easy to grow and shrink a list.
  - Lists are *inherently recursive*; the type is defined recursively.
  - Lists are *simple*; you can build almost every list operation through just a few basic operations (`car`, `cdr`, `null`, and `null?`).

An Alternative: Vectors

- Vectors provide an alternative to lists.
- They have two primary advantages:
  - Vectors are *indexed*: You can quickly access elements by number.
  - Vectors are *mutable*: You can change the elements of a vector.
- In order to obtain these benefits, vectors lack some key features of lists. In particular,
  - Vectors are *static*: Once you've created a vector, you cannot change its length.
- Some key vector procedures:
  - `(vector` *`val1 ... valn`*`)`: Create a vector
  - `(make-vector` *`length val`*`)`: Make a vector of specified length, with duplicates of *val* as the contents.
  - `(vector-ref` *`vector position`*`)`: Extract a value from a vector.
  - `(vector-set!` *`vector position newvalue`*`)`: Change an element of a vector.
  - `(vector-length` *`vector`*`)`

Implementing Vectors

- To be done live in class: Memory layout and more.

Lab

- Do the lab.

 This material is based upon work partially supported by the National Science Foundation under Grant No. CCLI-0633090. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.