LISP - VECTORS

Vectors are one-dimensional arrays, therefore a subtype of array. Vectors and lists are collectively called sequences. Therefore all sequence generic functions and array functions we have discussed so far, work on vectors.

Creating Vectors

The vector function allows you to make fixed-size vectors with specific values. It takes any number of arguments and returns a vector containing those arguments.

Example 1

Create a new source code file named main.lisp and type the following code in it.

```
(setf v1 (vector 1 2 3 4 5))
(setf v2 #(a b c d e))
(setf v3 (vector 'p 'q 'r 's 't))

(write v1)
(terpri)
(write v2)
(terpri)
(write v3)
```

When you execute the code, it returns the following result:

```
#(1 2 3 4 5)
#(A B C D E)
#(P Q R S T)
```

Please note that LISP uses the #. . . syntax as the literal notation for vectors. You can use this #. . . syntax to create and include literal vectors in your code.

However, these are literal vectors, so modifying them is not defined in LISP. Therefore, for programming, you should always use the **vector** function, or the more general function **make-array** to create vectors you plan to modify.

The **make-array** function is the more generic way to create a vector. You can access the vector elements using the **aref** function.

Example 2

Create a new source code file named main.lisp and type the following code in it.

```
(setq a (make-array 5 :initial-element 0))
(setq b (make-array 5 :initial-element 2))

(dotimes (i 5)
   (setf (aref a i) i))

(write a)
(terpri)
(write b)
(terpri)
```

When you execute the code, it returns the following result:

```
#(0 1 2 3 4)
#(2 2 2 2 2)
```

Fill Pointer

The **make-array** function allows you to create a resizable vector.

The **fill-pointer** argument of the function keeps track of the number of elements actually stored in the vector. It's the index of the next position to be filled when you add an element to the vector.

The **vector-push** function allows you to add an element to the end of a resizable vector. It increases the fill-pointer by 1.

The **vector-pop** function returns the most recently pushed item and decrements the fill pointer by 1.

Example

Create a new source code file named main.lisp and type the following code in it.

```
(setq a (make-array 5 :fill-pointer 0))
(write a)

(vector-push 'a a)
(vector-push 'b a)
(vector-push 'c a)

(terpri)
(write a)
(terpri)

(vector-push 'd a)
(vector-push 'e a)

;this will not be entered as the vector limit is 5
(vector-push 'f a)

(write a)
(terpri)

(vector-pop a)
(vector-pop a)
(vector-pop a)

(write a)
```

When you execute the code, it returns the following result:

```
#()
#(A B C)
#(A B C D E)
#(A B)
```

Vectors being sequences, all sequence functions are applicable for vectors. Please consult the sequences chapter for vector functions.