

## Hashes

by Dr. David J. Ritchie,  
Computer Club Advisor

Hashes are like regular arrays except that you use names for elements in place of numbers. % is used instead of @. {} are used instead of [].

### Try it:

Write this program:

```
$a{"tiger"} = "fur";  
$a{"seal"} = "skin";  
print "Animals: %a";
```

Each element has a key and a value. The key is like the array element number. The value is like value for an array.

### Questions:

1. What are the keys above?
2. What are the values?
3. What does this do:

```
%a = ();
```

## Associate

A hash is sometimes called an Associative Array because it **associates** the **key** with the **value**.

It takes two regular arrays to do the work of one hash--one to hold the keys and one to hold the values.

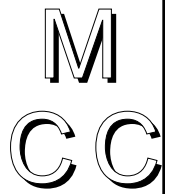
Issue 5

January 31, 2000

# Madison Computer Club News

Mrs. Gilmore

Dr. David J. Ritchie



## Students to learn about Hashes and Libraries Next!

This week's efforts are concerned with hashes and subroutine libraries.

A hash is similar to a regular array.

A regular array is specified with the @ sign. For example, a regular array of four elements is created by @animal = ("tiger", "seal", "lion", "dog"). The values of each element are \$animal[0], \$animal[1], \$animal[2], and \$animal[3].

A hash of four elements is specified with the % sign. For example, a hash of four elements is created by %coat = ("tiger", "fur", "seal", "skin", "lion", "fur", "dog", "fur").

The first item in the list becomes the **key** of the hash element. The second item becomes the **value** of that hash element.

The values can be obtained as follows:  
\$coat{"tiger"}, \$coat{"seal"},  
\$coat{"lion"}, etc.

The \$ indicates a *single* value is being obtained from the hash. The braces {} mean that the thing from which the value is being obtained is a hash.

## Libraries

People have written many libraries of subroutines to do useful things. Many come with Perl itself.

We don't have to do it all! We can use the work that others have done to make our lives easier!

## The Not So Great Escape Game

```
@room = ("r", "b", "g", "w");
%up = ("r", "b", "b", "g", "g", "r");
%down = ("r", "g", "g", "b", "b", "r");
%right = ("r", "w");
%left = ("g", "w");
$location = $room[0];
while ($location ne "w") {
    print "\nIn $location room.\n";
    $location = choice($location);
}
print "\nRoom: $location. You're out!\n";
```

Here's a list of some useful routines. You'll have to look at a book on Perl to find out exactly how to use them.

**abs** - returns absolute value  
**chomp** - chops off last character only if it is \n  
**chop** - chops off the last character regardless.  
**defined** - returns true if argument is defined.  
**exists** - returns true if the hash key exists.  
**join** - joins a list together into a string  
**keys** - returns all the keys of a hash.  
**last** - immediately exits the loop you are in.  
**lc** - makes string lower case  
**lcfirst** - lower cases the first letter of a string.  
**localtime** - returns local time as an array of seconds, minutes, hours...month, day, year given the number of seconds since January 1, 1970.  
**next** - immediately starts the next pass of a loop.  
**pop** - pops off the last element of an array.  
**printf** - prints values in a formatted way  
**push** - pushes a list onto the end of an array.  
**rand** - returns random number from 0 to 1  
**redo** - redoes the current pass of a loop.  
**return** - returns from a subroutine.

**reverse** - reverses the order of a list

**s** - substitute (like match)

**shift** - shifts out the *first* element of an array moving everything over.

**sort** - sorts a list alphabetically by default.

**splice** - cuts elements out of an array.

**split** - splits a string into array elements.

**printf** - converts values in a formatted way into a string.

**sqrt** - returns square root.

**substr** - extracts a substring

**time** - returns the number of seconds since January 1, 1970.

**tr** - translate.(like substitute)

**uc** - makes string upper case

**ucfirst** - upper cases first character of a string.

**undef** - makes argument

undefined.

**unshift** - shifts an argument into the *first* element of an array, moving everything over.

**values** - returns all the values of a hash.

## **A Choice ESCAPE Subroutine**

```
sub choice {
    $now = $_[0];
    $dir = "";
    while (not ($dir =~ m/[udrl]/) ) {
        print "Direction: udrl?";
        $dir = <STDIN>;
        chomp($dir);
        $dir = lc($dir);
        if ($dir eq "u") {
            $next = $up{ $now };
        } elsif ($dir eq "d") {
            $next = $down{ $now };
        } elsif ($dir eq "l") {
            $next = $left{ $now };
        } elsif ($dir eq "r") {
            $next = $right{ $now };
        }
        if (not $next) {
            print "Can't go there!\n";
            $dir = "";
        }
    }
    return $next;
}
```