ECOL 553L

Hashs, and Dynamic Programming

One more note about special indexes

- We can negative index, from the end of an array

```
$array[scalar(@array)-1]
$array[$#array]
$array[-1]
```

A special array @ARGV

- @ARGV is the array of argument variables
- it contains the extra stuff passed to perl on the command line

```
./test.pl 8
```

$\Rightarrow$

```
$ARGV[0] = 8
```

(note: `$0` -- the script name, i.e. “test.pl”)

Perl Hashes (Associative Arrays)

- We've seen arrays and the use of integers as index values: `$items[0], $items[1]`, etc.
- Sometimes it is useful to store `<Key, Value>` pairs rather than using integers to index an array
- Perl Hashes do just that. Another name for a Hash is an Associative Array
- You can build a 'dictionary', containing keywords and definitions associated with these keywords
- Hash syntax is similar to array syntax, but employs different symbols

Perl Hash Example

- Array variables begin with the `@` sign, and to index an individual item, use `[ ]`: `@arr = (1,3,5); $arr[3] = 7;`
- Hash variables begin with the `%` sign. Key,value pairs are connected with the double arrow `=>`
- To index an individual item, use `$hash{'key'}`
- Example:

```
# define species key, value pairs
%species = (‘human’ => ‘H.sapiens’,
            ‘mouse’ => ‘M.musculus’,
            ‘fruitfly’ => ‘D.melanogaster’);

print $species{‘mouse’}, “\n”;
```

Adding to and Removing from a Hash

- Adding a key, value pair to a hash is easy. Of course, each key must be distinct.
  - Example:
    - `$species{‘blowfish’} = 'T.rubripes';`
- Removing a key, value pair from a hash is done by the delete function.
  - Example:
    - `delete $species{‘human’};`
- The exists function can be used to check for existing hash entries.
  - Example:
    - `if (exists $species{‘human’}) { … }`

Looking up Keys or Values in a Hash

- You can get a list of all values in a hash using the values function:
  - `@values = values (%hash);`
- The `values()` function takes a hash as an argument and returns an array of values.

- Similarly, a list of all keys in a hash can be obtained by using the keys function:
  - `@keys = keys (%hash);`

- The `keys()` function takes a hash as an argument and returns an array of values.

Stepping through Key, Value pairs in a Hash

- To step through each key, value pair in a hash, use a foreach loop and the keys function:

```
while ( my ($key, $value) = each(%hash) )
{
        print "$key => $value\n";
}
```

- TIMTOWTDI:

```
foreach my $key ( keys %hash ) {
      my $value = $hash{$key};
      print "$key => $value\n";
}
```

- Note that hash elements are not ordered!

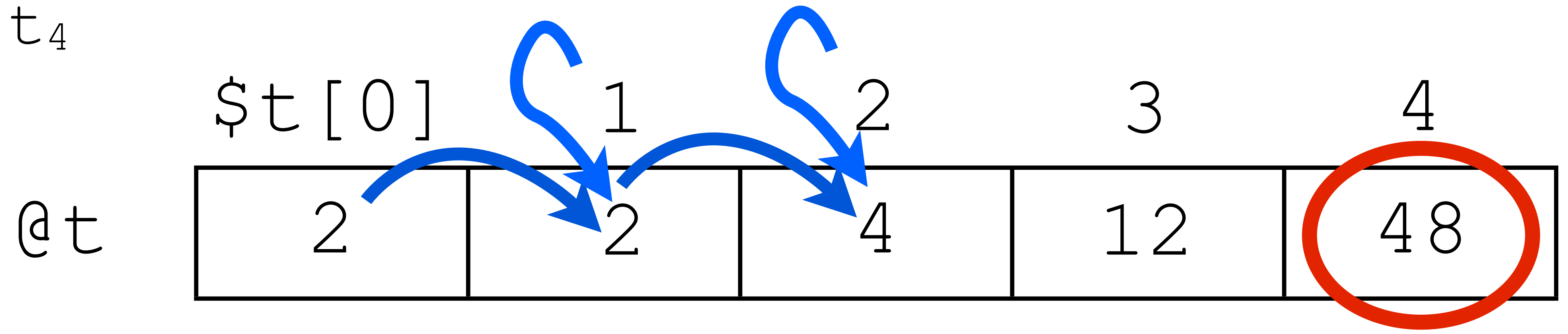
Simple Example

- Get the unique values in an array, and count occurrences

```
my %hash;
foreach my $item (@array){
   $hash{$item}++;
}
foreach my $key (keys %hash){
   print "$key\t$hash{$key}\n";
}
```

Dynamic Programming

- Using previous results to solve a new problem
- lets solve the problem
  - $t_i = t_{i-1} * i$
  - let $t_0 = 2$
  - find t_4



```
my @t;
$t[0] = 2;
foreach my $i (1 ... 3){
    $t[$i] = $t[$i-1] * $i;
}
```

System Commands

- The system command `system($cmd)` runs a unix command from within perl
  - **(bad)** example: `system("rm *");`
  - removes all files from the folder the script is run in
- To run and CAPTURE the output use the `` operator
  - `my @list = `ls`;`
  - runs the `ls` command, and puts the results in an array