Overview:
- More about CGI
- Perl data types and operations
- Perl control structures
- Perl I/O
- More input element types

More about CGI
- Form elements associate value with name specified
- When form is posted, each element's data is sent as encoded "name=value" pairs separated by &
- CGI script receives this data as one long string through standard input

What is Perl?
- Perl is a portable (UNIX, Windows, MAC) scripting language with more high-level features than shell languages or C/C++.
- Perl is ideally suited for applications involving text manipulation and system administration tasks. It is commonly used for CGI programming.

Perl Variables
Three types: scalars, arrays and hashes (associative arrays).

Scalar Operations
- Numeric Operations
  $amp = 2**.5; # sqrt of 2
  $pi = atan2(1,1)*4;
  $val = $amp*cos(2*$pi*0.1/60); ++$amp;
- String Operations
  $msg = "Hi there ". $name; # concatenation
  $a = "ba"."("na" x 4) . "\n"; # remove '\n'
  chomp($a);

Perl is a portable (UNIX, Windows, MAC) scripting language with more high-level features than shell languages or C/C++. Perl is ideally suited for applications involving text manipulation and system administration tasks. It is commonly used for CGI programming.
String tricks

```perl
$string = "abc:def:ghijkl:mnop";
@parts = split(":\", $string);  # assign to array
($b, $c, $d) = @parts[2..4];  # array slice
($first, $second) = split(":", $string);  # first two
$str1 = substr($string, 3, 3);  # c:d
substr($string, 3, 4) = "wxyz";  # c:de -> wxyz
$str2 = join("-", @parts);  # back to string
```

Array Operations

```perl
$array = ("zero", "one", "two", "three", "four");
$length = scalar(@array);  # $length = 5
$lastind = @$array;  # $lastind = 4
# pop and shift change the array
$last = pop(@array);  # get right element
$first = shift(@array);  # get first element
# sort doesn't change the array
@srt_array = sort(@array);
```

Hash Operations

```perl
# print out the environment values
@env = sort(keys(%ENV));
for($i=0; $i<=$#env; $i++) {
    print $env[$i], "=" , $ENV{$env[$i]} , "\n";
}
# Another way (unsorted)
while (($var, $val) = each(%ENV)) {
    print $var, "=" , $val , "\n";
}
# Yet another way (TMTOWTDI = tim toady)
foreach $var (sort(keys(%ENV))) {
    print $var, "=" , $ENV{$var} , "\n";
}
# Create new hash with values as keys
%numname = reverse(%phlist);
```

Control Structures

- "" (empty string), "0", and 0 are false.

 numeric comparison: like C++, <= >. String comparison: eq, ne, lt, gt, le, ge, cmp.

```perl
if($city eq "New York") {
    $state = "New York";
} elsif ($city eq "Miami") {
    $state = "Florida";
} elsif ($city eq "Dallas") {
    $state = "Texas";
} else {
    $state = "";
}
```

while, for, and foreach examples are on the previous slide. In loops, next and last are equivalent to C++’s continue and break.

I/O

```perl
$line = <STDIN>;  # read a line from STDIN
open(GRD, "grds.db");  # open for reading
open(OUT, ">out.txt");  # open for writing
open(OUT, ">>out.txt");  # open for appending
$line = <>;  # read from filehandle
print $line;  # print to STDOUT
print OUT $line;  # print to filehandle
```

HERE documents

HERE documents are useful for large amounts of static output like headers

```perl
print <<EndOfHTTPHeader;
Content-type: text/html
EndOfHTTPHeader
```

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

```perl
print <<EndOfHTMLHeader;
```

```
<!--http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd-->
```

```perl
EndOfHTMLHeader
```
Subroutines

- Subroutines can return a scalar or list. Parameters are passed in the @_ array:

  ```perl
  sub process {
    my @parms = @_; # local var
    # do stuff
    if(wantarray) { return @parms; }
    else {return @parms[0]; }
  }
  ``

- Calls made using &:

  ```perl
  &process(arg1, arg2, arg3);
  ```

More Input Element Types

- Radio buttons

  ```html
  <input type="radio" name="size" value="Small" />
  <input type="radio" name="size" value="Large" checked="checked" />
  ``

- Check boxes

  ```html
  <input type="checkbox" name="pepperoni" value="Pepperoni" />
  ``

- Reset button

  ```html
  <input type="reset" value="Reset this form" />
  ```

Select boxes

- Pull-down menu

  ```html
  <select name="size">
    <option value="Small">Small</option>
    <option value="Large" selected="selected">Large</option>
  </select>
  ``

- Multi-selection

  ```html
  <select name="toppings" multiple="multiple">
    <!-- options the same as above -->
  </select>
  ```

Purple Pizza Parlor v2-v4

- Copy files -hwang/cs350/lecture4/*.* to your cgi-bin directory

- Edit the HTML files to use the CGI scripts in your web space.

- Edit the response in the CGI programs so you can tell it's your program running.

- Make sure they are all running correctly.

In-class Exercise

- Modify one of the versions (v2, v3, v4):
  - Add an input element for a phone number
  - Add more sizes and/or toppings
  - Split the toppings into two lists: meats and veggies.
  - Add pricing information: for size, for meats, for veggies.
  - Compute the price of the ordered pizza

- Add a link to the modified HTML page in your cs350.html page

Programming Notes

- Errors in CGI Perl code cause an "internal server error" message to be displayed in the browser. Error messages can be viewed on cserver by using

  ```bash
  tail -20 /var/log/httpd/error_log
  ```

- Perl has implicit variable declarations, so watch out for typos. All variables are initialized to the NULL value.