Operators

Operators

- Perl has MANY operators.
  - Covered in Chapter 3 of Prog.Perl
- Most operators have numeric and string version
  - remember Perl will convert variable type for you.
- Go through them in decreasing precedence

Increment/Decrement

- ++ and --. Postfix and Prefix work as in C.
- ++ is “magical”.
  - if value is purely numeric, works as expected
  - if string value, or ever used as string, magic happens
  - ‘99’++ ➔ ‘100’
  - ‘a9’++ ➔ ‘b0’
  - ‘Az’++ ➔ ‘Ba’
  - ‘zz’++ ➔ ‘aaa’
- Try it, see what happens.

Exponentiation

- ** → Exponentiation. Calls C's pow() function
  - works on floating points
  - 2**3 == pow(2, 3) == "2 to the power of 3" == 8
  - NOTE: higher precedence than negation
    - -2**4 → -(2**4) → -16

Unary Operators

- ! – logical negation
  - 0, "0", "", undef → all false
  - anything else → true
- - – arithmetic negation (if numeric)
  - if non-numeric, 'negates' the string
  - ex: `$foo = "-abc"; $bar = -$foo;`
  - $bar gets value "+abc";
- ~ – bitwise negation

Multiplicative

- / -- Division. Done in floating point.
- % -- Modulus. Same as in C.
- * -- Numeric multiplication
- x -- String multiplication (aka repetition).
  - 123 * 3 → 369
  - 123 x 3 → '123123123' (scalar context)
  - (123) x 3 → (123, 123, 123) (list context)

Additive

- + – normal addition
- - – normal subtraction
- . – string concatenation
  - `$var1 = “hello”; $var2 = “world”;`
  - `$var3 = $var1 . $var2;`
    - $var3 contains “helloworld”
  - `$var3 = “$var1 $var2”;`
    - $var3 contains “hello world”

Shift operators

- << and >> - work as in C.
- 1 << 4 ➔ 16
- 32 >> 4 ➔ 2

Relational Operators

Numeric	String	Meaning
>	gt	Greater Than
>=	ge	Greater Than or Equal
<	lt	Less Than
<=	le	Less Than or Equal

Equality Operators

Numeric	String	Meaning
==	eq	Equal to
!=	ne	not equal to
<=>	cmp	comparison

•<=>

- -1 if left < right
- 0 if left == right
- 1 if left > right

Bitwise Operators

- & -- AND. | -- OR ^ -- XOR
  - & has higher precedence
- if either value numeric:
  - convert to integer,
  - bitwise comparison on integers
- if both values strings:
  - bitwise comparison on corresponding bits from the two strings

"C-Style" Logical Operators

- && - AND || - OR
  - && has higher precedence
- operate in short-circuit evaluation
  - ie, evaluate only what's needed
  - creates this common Perl line:
- `open (FILE, "file.txt") || die "Can't open file.txt";`
- return last value evaluated, not 0 or 1.

Conditional Operator

- ?: -- Trinary operator in C.
- like an if-else statement, but it's an expression
  - `$a = $ok ? $b : $c;`
  - if $ok is true, $a = $b. if $ok is false, $a = $c

Assignment operators

- =, **=, *=, /=, %=, x=, +=, -=, .=,
- &=, |=, ^=, <<=, >>=, &&=, ||=
- In all cases, all assignments of form
- `TARGET OP= EXPR`
- evaluate as:
- `TARGET = TARGET OP EXPR`

Comma Operator

- Scalar context:
  - evaluate each list element, left to right. Throw away all but last value.
  - $a = (fctn(), fctn2(), fctn3());
    - fctn() and fctn2() called, $a gets value of fctn3()
- Array context:
  - list separator, as in array assignment
  - @a = fctn(), fctn2(), fctn3());
    - @a gets return values of all three functions

Logical and, or, not, xor

- Functionally equivalent to &&, ||, !
- BUT, a lower precedence.
- `$xyz = $x || $y || $z;`
- `$xyz = $x or $y or $z;`
- What's the difference?

Incomplete list

- ALL operators found in Chapter 3 of PP.
- some skipped over, we'll talk about them later. (arrow, file test, range)