

To be covered today

- Defining functions (subroutines)
- Calling functions
- Returning values from functions
- Passing arguments
 - ► the a_array

CSE2395/CSE3395

- Local variables
- ▶ my and local keywords
- Sorting arbitrarily

Subroutines

2

Notes and terminology

- In Perl, called functions or subroutines
- ▶ no difference in meaning
- All subroutines can take parameters
 - called arguments or actual parameters by caller
 - called (formal) parameters by function
- No type checking
 - of parameters or return value
 - optional prototypes allow rudimentary type checking
- No formal naming of parameters
- programmer can do this if desired

4

Defining a function Also called a subroutine subroutines are defined with sub keyword name of subroutine goes here greet { CSE2395/CSE3395 print "Hello there.\n"; body of subroutine can contain anything braces are required Llama2 pages 92-93; Camel3 pages 217-218 Camel2 pages 111-112

Subroutines Subroutines may be declared anywhere in the definitions are skipped on execution CSE2395/CSE3395 ▶ by convention, definitions go first or last in code Subroutines can access all global variables ► can declare localized variables with my keyword

Llama2 pages 93, 96; Camel3 page 218, 223 Camel2 page 189

Subroutines

Naming conventions

- Technically, subroutine names begin with special character &
 - variables \$days, @days, %days and function &days are all separate
- In practice, no leading character is needed
- when declaring, not used in sub definition
- when calling, parentheses after the subroutine name identify it as a function call
- days(2000, 1, 1) # or could say &days(2000, 1, 1)



CSE2395/CSE3395

CSE2395/CSE3395

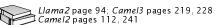
CSE2395/CSE3395

1

Subroutines

Returning values

- Subroutines return to their caller the last expression evaluated
- ▶ sub pi { 3.1415926535898; }
- Can use return keyword to return sooner
 - sub abs_a { if (\$a >= 0) { return \$a; }
 else { return -\$a; } }
- Return value can be scalar or list
 - ▶ sub first_two_args { return @ARGV[0,1]; }
- return value is interpreted according to context subroutine was called in
- ► can use wantarray function to determine context



8

Parameter passing

Passing values to a function

- Caller names arguments in parentheses after function name
 - \$ \$dayname = weekday(\$year, \$month, \$day);
 - as with built-in functions, parentheses can be omitted if subroutine is pre-declared
- Arguments are formed into list and placed in special local array variable a
- Subroutine can access a or individual members
 - sub weekday { (\$y, \$m, \$d) = a_; ... }
 sub weekday { \$y = \$ [0]; ... }
- Llama2 pages 94-96; Camel3 pages 219-221
 Camel2 page 112

9

Example

Calculating the hypotenuse

```
# Declare the hypotenuse function.

sub hypotenuse {
    # Assign parameters meaningful names.
    # Could also have done:
    # ($x, $y) = a_;
    $x = $_[0]; $y = $_[1];
    return sqrt($x * $x + $y * $y);
}

# Read two numbers on one line.
print "Enter two numbers: ";
($a, $b) = split /\s+/, <STDIN>;

print "Hypotenuse is: ",
hypotenuse($a, $b), "\n";
```

10

Example

Summing a list

```
# Read some numbers into @nums.
while (<>)
{
   chomp; push @nums, $_;
}
print "Sum is ", sum(@nums), "\n";

sub sum {
   $sum = 0;
   # Iterate $_ over parameter list @__,
   foreach (@_)   # $_ is the default iterator.
   {
        $sum += $_;   # Add this list element to @sum
   }
   return $sum;
}
```

11

Local variables with my

Protecting variables from accidental modification

- By default, all variables are global
- Variables can be declared local (lexical scoping) with my keyword
- ▶ my (\$sum); # Protects old value of \$sum.
- Old value is restored at end of enclosing block (often end of subroutine)
- Can localize and assign in one step
- ► my (\$x, \$y) = a;

CSE2395/CSE3395

▶ parentheses needed because of precedence



Example

Finding all elements in a list that match a pattern

```
# Almost same as the Perl builtin function grep.
sub filter {
    # Declare meaningful names for parameters.
    my ($pattern, @values);
    # Declare a temporary array for return value.
    my (@result);
    # shift in a function defaults to using @_.
    $pattern = shift;
    @values = @_;

foreach (@values) {
    if /$pattern/ {    # Test $_ against $pattern.
        push @result, $_; # Save this string.
    }
    @aresult; # Return value.
```

local versus my

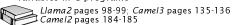
Use this only if you have to

- Perl has another kind of localizing of variables, using local keyword
- Use local where my does not work
- ►local \$_; # my \$_ isn't allowed.
- Otherwise, use my

CSE2395/CSE3395

- Local causes dynamic scoping of variables (they are visible inside all called functions); this is unfamiliar to C programmers
- my causes lexical scoping, which behaves like local variables in C programs

14



13

Global variables: our

Explicitly declaring globals

- Perl 5.6 has our keyword
- declares global variable to be visible in current scope
- ▶ our \$house;

CSE2395/CSE3395

- Not needed unless programming under use strict 'vars'
 - undeclared variable is normally automatically global
 - use strict is recommended for modules or large programs



Parameter passing

Passing arrays and hashes

- Arrays and hashes are unwound into single list before being stored in a
 - ▶ sizes of arrays are lost in unwinding
- If passing one array, make it the last argument
- Passing more than one array can't usually be done
 - diff(@a, @b) will pass one list containing all elements in both @a and @b to the diff function
 - if diff does (ax,ay) = a_then ax gets all elements, and ay is empty
 - ► can be solved with references (Topic 11)

Camel3 pages 221, 224-225; Camel2 page 114

15

CSE2395/CSE3395

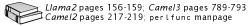
Sorting

Sorting a list numerically

- sort function normally sorts items alphabetically (lexicographically)
- Can sort by other criteria by providing comparison function
 - ▶ does not use normal parameter-passing mechanism
 - inside comparison function, \$a and \$b are aliases of two list elements
 - function must return

CSE2395/CSE3395

- less than zero if \$a precedes \$b
- zero if \$a and \$b are the same
- greater than zero if \$a follows \$b



Example

16

Sorting a list numerically

```
# For code readability, use adverbs for names.
sub numerically
{

    # $a and $b are automatically localized
    # in this function.

    # Could also have said: return $a <=> $b;
    if ($a < $b) { return -1 }
    elsif ($a > $b) { return 1 }
    else { return 0 }
}

alist = (1, 128, 16, 2, 32, 4, 64, 8);

# Note name of function between keyword
# and list; also no comma after function name.
anewlist = sort numerically alist;
```

17

Covered today

- Defining functions (subroutines)
- ▶ sub keyword
- Calling subroutines
- Returning values from subroutines
 - ► last expression evaluated in function
 - ► return keyword
- Passing arguments
- ▶ the a array
- Local variables
 - ▶ my and local keywords
- Sorting lists arbitrarily
- ▶ comparison functions

19

Going further

More things related to today's topic

- Prototypes
- making user-defined subroutines behave more like builtins
- Camel3 pages 225-228; Camel2 pages 118-121
- Code generation
- ▶ building Perl code on-the-fly with eval
- Camel3 pages 705-707; Camel2 pages 161-163
- Built-in functions
 - ▶ a myriad of standard subroutines provided in Perl
- Camel3 pages 677-830; Camel2 pages 141-242
- ■BEGIN and END
 - special functions that run before or after other code
 - Camel3 pages 480-485; Camel2 pages 283-284

20

Next time

To be covered in Topic 8

- File operations
- ▶ open, close
- Reading from and writing to files
- File tests
- Scanning directories

CSE2395/CSE3395

Reading: Learning Perl 2nd edition chapters 10, 12, 13 pages 108-115, 129-133, 134-141 Programming Perl 3rd edition pages 20-22, 28-29, 97-100, 747-755, 770 Programming Perl 2nd edition pages 12-14, 19-20, 85-87, 191-195

perlfunc, perlopentut Manpages

CSE2395/CSE3395

CSE2395/CSE3395 lecture notes copyright © 2000-2001 Deborah Pickett.
Reproduction of this presentation for nonprofit study use is permitted. All other reproduction, including for other educational courses, must be authorized in writing by the author.

21

22