CSCI3180 Principle of Programming Languages

Tutorial 8:
Dynamic Scoping

TA: LIN Huaijia
email: linhj@cse.cuhk.edu.hk

Useful Sources

- Build System of Perl in Sublime Text3
  - https://gist.github.com/kbdhero/d4f2a299c0207bef8eda
- A good tutorial
  - https://qntm.org/files/perl/perl.html

Several Features

- No boolean types
  - The following scalar is equivalent to False in an “if” statement.
  - undef, number 0, “”, “0”

- The type of a Scalar depends on the operator

```
my $a = "4H";
my $b = "4G";
print $a + $b; # "8" with two warnings
print $a.$b; # "4H4G"
print $a eq $b; # ""
print $a == $b; # "1" with two warnings
```

An Overview

- Declared with keyword “my”
  - lexical scoping
  - local variable
- Declared with keyword “our”
  - lexical scoping
  - global variable
- Declared with keyword “local”
  - dynamic scoping
  - “local” variable

my in Perl

- lexical scoping
  - The scoping is determined at compile time
- local variable
  - be local in current enclosing block
  - can not be accessed from anywhere outside its enclosing scope.

Example 1 (my)

```
my $x= “hello”;
{
    my $x= “world ”;
    print $x; # “world”
}
print $x; # “hello”
```

“my” declare the variable to be local to the enclosing block or file.

Example 2 (my)

```
my $x= “hello”;
sub funA{
    my $x= “world ”;
    funB ();
    print $x; # world
}
sub funB {
    print $x; # hello
}
funA();
```

lexical scoping: If no declaration found in the enclosing block, try to find the declaration in the outside block

Decided in compile time

Package: `main`

- All subroutines and package variables must have a package.
- The root namespace is **`main`**. In effect, the top of every source file in Perl is prefixed by an implicit statement:

```
package main;
```

our in Perl

- alias to a package variable
  - variables without any prefix is a package variable
    - Eg. “$a” is equal “our $a”
- lexical scoping
- global variable

Example (our)

```
our $x= “hello”;
sub funA{
    $x= “world”;
    funB(); # world
}
sub funB{
    print $x; # world
}
print $x; # hello
funA();
print $x; # world
```

Within a package, they behave like ‘global’ variables of other languages.

Value of $x declared in the current package is eternally modified.

Example (our & my)

```
our $x= “hello”;
sub funA{
    my $x = “world”;
    funB();
    print $x; # world
}
sub funB{
    print $x; # hello
    $x = “key”
}
funA();
print $x; # key
```

local in Perl

- temporarily changes the local value of a global variable.

- dynamic scoping
  - depends on the calling sequences

Example1 (local)

```
our $x= “hello”;
sub funA{
    local $x= “world”;
    funB();
}
sub funB {
    print $x;
}
funA(); # world
funB(); # hello
```

You search in the local function first, then search in the function called the local function, and so on, up the call stack.

-- “http://wiki.c2.com/?DynamicScoping”

Example2 (local & our)

```
our $x= “hello”;
sub funA{
    local $x= “world”;
    funB();
    print $x; # key
}
sub funB {
    print $x; # world
    $x = “key”
}
funA(); #
print $x; # hello
```

Q&A