# CSc 520

# Principles of Programming Languages

## *24 : Names, Scope, Bindings — Dynamic Scope*

Christian Collberg

collberg+520@gmail.com

Department of Computer Science

University of Arizona





## Static vs. Dynamic Scope

- Pascal is lexically scoped. We can look (textually, or at compile-time) at a procedure and determine to which object an identifier refers.
- Some languages (Snobol, APL, Perl, some dialects of LISP) are dynamically scoped. The binding between an identifier and the object it refers to is not decided until run-time.



## Dynamic Scope

- The current binding for an identifier is the one last seen during execution and whose scope has yet to be destroyed.
- Consider the example on the next slide.

  **static scope:** the program prints 1.

  **dynamic scope:** the program prints 2.
- Static scope rules match the use of an identifier with the closest lexically enclosing declaration.
- Dynamic scope rules choose the most recent active declaration at runtime.



## Dynamic Scope. . .

```
var a :  integer;

procedure first();
   a := 1;

procedure second();
   var a :  integer;
   first();

begin
   a := 2;
   second();
   write(a);
end
```

# Dynamic Scope — Problems

```
var max : integer;

procedure scale(x : integer) : real;
   return x/max;

procedure compute(y : integer);
   var max : integer;
   write(scale(y));
```

- Dynamic scope makes it is easy to accidentally redefine a variable.



# Dynamic Scope — Advantages

```
procedure A(base : integer)
   printInt(base, 245);

procedure B(base : integer)
   A();

procedure C(base : integer)
   B();

begin C(16); end
```

- We often have to pass around state so that deeply nested procedures can make use of it. `DEBUG`-flags is a common example.



# Dynamic Scope — Advantages. . .

```
var base : integer := 10;
procedure A()
   printInt(base, 245);
procedure B()
   A();
procedure C()
   B();

begin
   var last_base := base;
   base := 16; C();
   base := last_base;
end
```

- We can, of course, use global variables.



# Dynamic Scope — Advantages. . .

```
procedure A()
   printInt(base, 245);
procedure B()
   A();
procedure C()
   B();

begin
   var base : integer := 16;
   C();
end
```

- Dynamic scope makes it is easy customize the behavior of procedures.

# Readings and References

- Read Scott, pp. 115, 131-135
- *Dynamic Variables*, David R. Hanson and Todd A. Proebsting, PLDI 2001.

  `www.microsoft.com/~drh/pubs/dynamic.pdf`.