Overview

- Object lifetime and storage management
- Scopes
- Aliasing and overloading
- Binding of referencing environments
Scoping Rules

- **Scoping rules**: Define which bindings are active
  - I.e., what’s the meaning of a name at a given program point?

- Each PL defines its scoping rules
  - E.g., Basic has only one scope
  - Most PLs have nested scopes for subroutines
Nested Scopes

- Common for nested subroutines
- Each subroutine has its own scope
- Closest nested scope rule
  - Name is known in scope where it is declared and all internally nested scopes
  - Inner scopes can hide names from outer scopes
Example

```javascript
fun f1(a1) {
    var x
    fun f2(a2) {
        fun f3(a3) {
        }
    }
    ...
    }
    fun f4(a4) {
        fun f5(a5) {
            var x
        }
    }
}
```
Static vs. Dynamic Scoping

Static scoping

- Binding of a name can be derived from program text
- Most common in today’s PLs

Dynamic scoping

- Binding of a name depends on control flow
  - I.e., not known statically (in general)
Example

Pseudo code:

global x = 1
fun a() {
    local x = 3
    b()
}
fun b() {
    y = x
}
a()
Example

Pseudo code:

global x = 1
fun a() {
    local x = 3
    b()
}
fun b() {
    y = x
}
a()

Static scoping:
y gets value 1 because
b doesn’t have a local
variable called x and
the surrounding static
scope provides the
global variable x
Example

Pseudo code:

global x = 1
fun a() {
    local x = 3
    b()
}
fun b() {
    y = x
}
a()

Dynamic scoping:
y gets value 3 because
b doesn’t have a local
variable called x and
the dynamically closest
scope provides the
local variable x of a
Quiz: Dynamic Scoping

What does the following Perl code print?

```perl
$b = 0;
sub foo {
    return $b;
}
sub bar {
    local $b = 1;
    return foo();
}
print bar();
```

Please vote via Ilias.
Quiz: Dynamic Scoping

What does the following Perl code print?

```perl
$b = 0;
sub foo {
    return $b;
}
sub bar {
    local $b = 1;
    return foo();
}
print bar();
```

**Result: 1**

*Please vote via Ilias.*
Quiz: Dynamic Scoping

What does the following Perl code print?

```perl
$b = 0;
sub foo { return $b; }
sub bar { local $b = 1; return foo(); }
print bar();
```

Result: 1

Global variable that would be used with static scoping

Please vote via Ilias.
Quiz: Dynamic Scoping

What does the following Perl code print?

```perl
$b = 0;
sub foo {
    return $b;
}
sub bar {
    local $b = 1;
    return foo();
}
print bar();
```

- But: Perl has dynamic scoping
- Uses last encountered definition of $b

Result: 1

Please vote via Ilias.
Function Stack vs. Static Scopes

- Push allocation frames on calls
- Pop frames on returns

How to resolve bindings outside of current scope?

- Each allocation frame has a static link to its parent scope
- Not affected by which functions get called
Example:
Nested functions

A
   B
      C
          L

E

Function stack

... -> ... stack

C
D
E
A
Built-in Objects

Many PLs have built-in (or predefined) objects

- E.g., for built-in types and APIs
- Invisible, outer-most scope
- Accessible from all scopes, except if hidden
Quiz: Scopes

What does this Python code print?  
(Hint: Python uses static scoping)

```python
x = "a"
def f():
    def g():
        print(x)
    def h():
        g()
        x = "c"
        print(x)
        x = "b"
        h()
        print(x)
    print(x)
f()
print(x)
```
What does this Python code print?
(Hint: Python uses static scoping)

```python
x = "a"
def f():
    def g():
        print(x) # (1) x in f : "b"
    def h():
        g()
        x = "c"
        print(x) # (2) x in h : "c"
        x = "b"
        h()
        print(x) # (3) x in f : "b"
    f()
    print(x) # (4) x in main: "a"
```

Please vote in Ilias.