# Recitation Note 2

CSCI-GA.2110-001 Programming Languages

Yusen Su

June 6, 2019

## 1 Scoping

### 1.1 Definition

- Binding: a binding is an association of two things. Example: binding a variable name with a value.
- Static Binding: name binding performed before the program is running (compile time).
- Dynamic Binding: name binding performed when the program is executing (running time).
- Scope: the region of program text where a binding is active.
- Static Scoping: binding of a name is determined by rules that refer only to the program text (i.e. its syntactic structure).
- Dynamic Scoping: binding of a name is given by the most recent declaration encountered during run-time.
- Nested scopes: given nested subroutines (e.g. blocks, classes), the scope for a nested subroutine is inside another scope.
  Typically, bindings created inside a nested scope are not available outside that scope.

### 1.2 How to consider the variable bindings under static/dynamic scoping in the code segment?

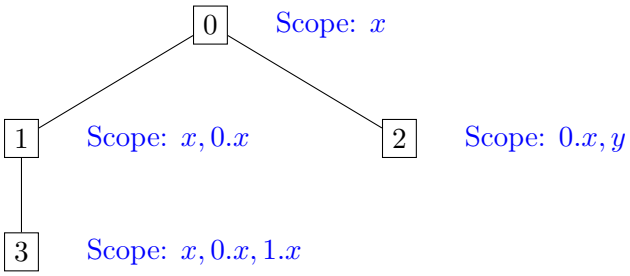
Global variable: declare outside any functions, could be accessed on any functions.
Local variable: declare inside a function, could only be accessed by that function.
Variable shadowing (hide): In a certain scope, if you redeclare a variable, the original binding is hidden, and has a hole in its scope.

- Static Scoping: depends on the code structure. The variable is bound by a data structure like tree. The nearest variable declaration of that variable in the code hierarchy (same level or upper levels).
- Dynamic Scoping: depends on the execution order. The variable is bound by a data structure like stack. The nearest variable declaration of that variable in the running order hierarchy (same level or upper levels).

For example, consider this code segment (assume variable should be declared before used):

```
0: var x = 1;
1: {
     x = x + 1;
     var x = 2;
    3:{
        x = x + 2;
        var x = 1;
        x = x + 1;
      }
  }
2:{
     var y = 0;
     y = x + 1;
  }
```

For static scoping, we could generate a scoping tree based on code structure:




For dynamic scoping, we could generate a scoping stack based on running order:

- consider the program is executing at line 8:

| Scoping stack for variable x | |
|---|---|
| | Var x in block 3: x = 1; |
| | Var x in block 1: x = 4; |
| | Var x in block 0: x = 2; |

- scope stack example when the program is running at line 10:

| Scoping stack for variable x | |
|---|---|
| | Var x in block 1: x = 4; |
| | Var x in block 0: x = 2; |

### 1.3 Sample question

Consider the following program (variable should be declared before used):

```
int x = 2;

```

```
void f(){
    int x = 3;
}

int g(){
    f();
    return x + 4;
}

int h(){
    int x = 5;
    return g();
}

printf("function g returns %d", g());
printf("function h returns %d", h());
```

#### 1.3.1 Assume program will run under static scoping, what does this program print?

**Solution:**
function g returns 6
function h returns 6

#### 1.3.2 Assume program will run under dynamic scoping, what does this program print?

**Solution:**
function g returns 6
function h returns 9

# 2 Control Flow

## 2.1 Code components

### 2.1.1 Expression

1. Definition: a combination of one or more constants, variables, operators, and functions that the programming language interprets and computes.
2. Evaluation order:

- Operator precedence: The order in which different infix operators are evaluated in an expression.
- Operator association: The order in which two consecutive infix operators with same precedence in an expression.
  Left-associative — evaluate from left to right

### 2.1.2 Statement

1. Definition: instructs the computer to take a specific action (usually a combination of the sequences of expressions).

2. Examples:

| Forms | Example |
|---|---|
| Assignment | $x := 5;$ |
| If statements | If ($expression$) then $statements_1$ else $statements_2$ |

## 2.2 Sequencing

Definition: execution statement and evaluation expression in sequential (or explicit specified) order.

## 2.3 Selection

1. Definition: executing one of two statements according to the value of a Boolean expression.
2. Short circuit evaluation: given a Boolean expression ($x == 0 \;||\; y > 0$), the second argument will not be evaluated if the first condition meets.

## 2.4 Iteration

1. Definition: execute a piece of statements repeatedly.
2. Breaking out: early exits the loop.

## 2.5 Questions

1. Why not using unstructured control flow mechanism such as *goto* in the modern programming language?
**Sample answer:**
Using *goto* is harmful because it could arrive at any particular locations of the code. It will break down the programming logic.