Principles of Programming Languages
Fundamental Semantics

Quan Thanh Tho, Ph.D.
CSE – HCMUT
qttho@hcmut.edu.vn

Outline

- Binding
- Declaration, Block,
- Scope and Referencing Environment
- Symbol Table
- Name
- Allocation, Lifetime and Environment
- Variable & Constant
- Alias, Dangling Reference and Garbage

Binding

- Association between a *program element* and a *property*
- Binding time:
  - Execution time
  - Translation time
  - Language implementation time
  - Language definition time

Binding

- X = X + 10
  - Possible types of X?
  - Type of X?
  - Possible values of X?
  - Value of X?
  - Representation of 10?
  - Properties of +?

Binding

- X = X * 10
  - Possible types of X?
  - Type of X?
  - Possible values of X?
  - Value of X?
  - Representation of 10?
  - Properties of +?
- Answer: Not now yet!

Binding

- Static Binding
- Dynamic Binding

Declaration

- A binding between an *identifier* and a *type*
  - Implicit (default) declaration
  - Explicit declaration

Grammar Exercise (1)

- Construct a (CF) grammar for a language:
  - Declaration (C-like)
  - Arithmetic expression
    - Operators: +,-,*,/ (left associativity, same precedence)
    - Operands: ID, integers, floating points
    - Type: int, float
  - No declaration allowed after first expression

Block

- A part of program
- May be nested
- Containing local declarations and statements
- Defining local referencing environment
- May be named

Visibility of Declaration

- A declaration is visible in a block if it can be referenced in the block

```
void main {
        int x;
        for (;;) { int x; cout << x;}
        for (;;) {cout <<x;}
        cout << x;
}
```

Visibility of Declaration

- A declaration is visible in a block if it can be referenced in the block

```
void main {
        int x;
        for (;;) { int x; cout << x; int y; }
        for (;;) {cout <<x; cout << y;}
}
```

Scope

- Scope of a declaration (or *variables* involved in the declaration):
  - Blocks in which the declaration is visible
- Scope of declaration: static scope

Static scope rules

1. Declaration D in Block B: B in scope S of D
2. D in B, $B \supset B_1 \supset \ldots \supset B_n$: B_i in S
3. If B in S, B consists $D' \neq D$ having same variable names with D: eliminating B (and sub-block of B)from S
4. If $B \subset B'$ is named $N \Leftrightarrow$ Declaration of N in B'

Example

```
var a:integer
    procedure sub(a:integer)        //sub-1
        procedure sub(a:integer)  //sub-2
        begin
                …
                a := 1;
                sub(1); //1
        end
    begin
        …
        a:= 2;
        sub(1);  //2
    end
Begin
    a:=3;
    sub(1);  //3
end
```

Referencing Environment

- Set of visible declarations: Static referencing environment
  - Program
  - Subprogram
  - Block
- Local, non-local and global referencing environment

```
program main;
var A, B, C: real;
procedure Sub1 (A: real);
    var D: real;
    procedure Sub2 (C: real);
            var D: real;
            begin
            … C:= C+B; …
            end;
    begin
    … Sub2(B); …
    end;
begin
… Sub1(A); …
end.
```

	Local	Non-local	Global
Sub2	C, D	A,Sub2 B,Sub1	B,Sub1
Sub1	A,D,Sub2	B,C,Sub1	B,C,Sub1
main	A,B,C,Sub1		A,B,C,Sub1

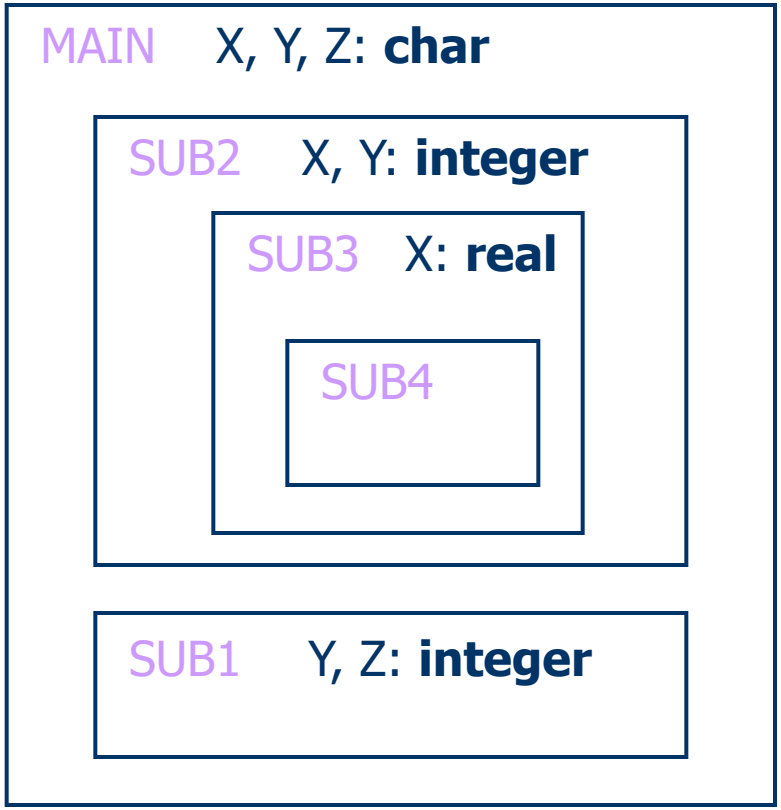
Static referencing environment

Grammar Exercise (2)

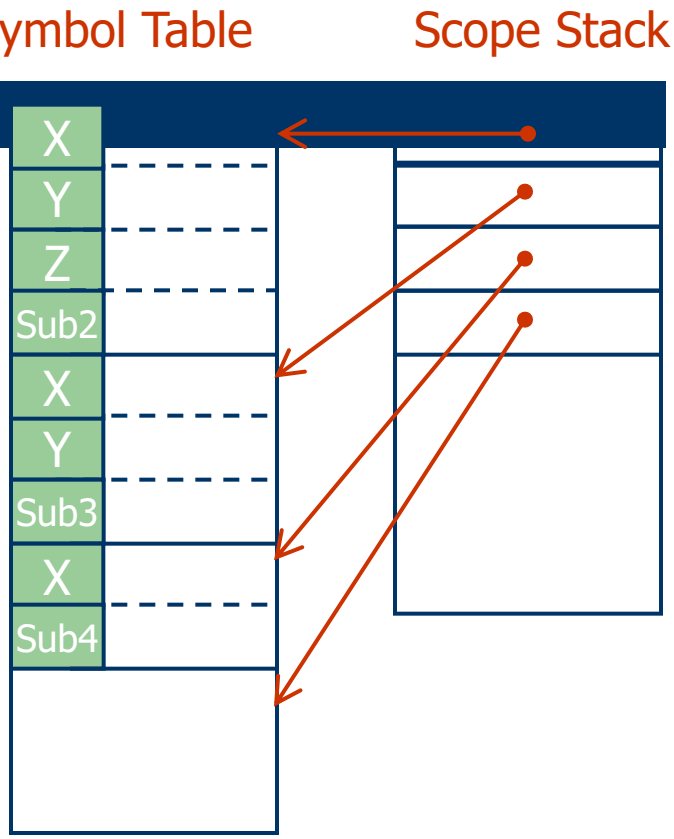
- Construct a (CF) grammar for a language:
  - Declaration (C-like)
  - Arithmetic expression
    - Operators: +,-,*,/ (left associativity, same precedence)
    - Operands: ID, integers, floating points
    - Type: int, float
    - Parentheses included
  - Declaration allowed after first expression

Symbol Table

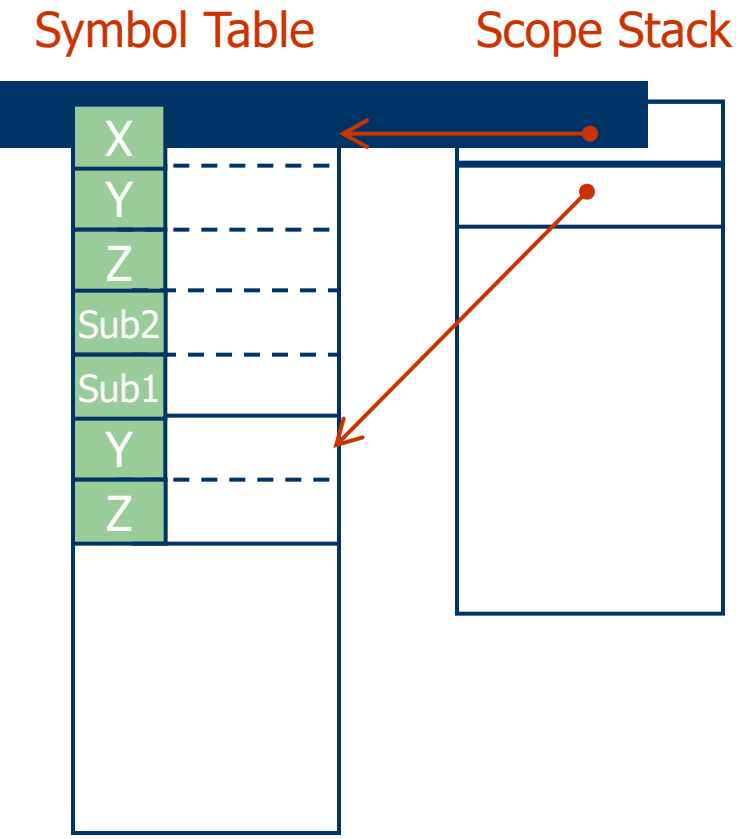
```
program MAIN;
    var X, Y, Z: char;
    …
    procedure SUB2;
    var X, Y: integer;
    …
        procedure SUB3;
          var X: real;
          …
              procedure SUB4;
              …
    procedure SUB1;
    var Y, Z: integer;
    …
```



```
program MAIN;
    var X, Y, Z: char;
                SUB2
    var X, Y: integer;
          procedure SUB3;
           var X: real;
                 procedure SUB4;
                 begin …end
           begin … end
    begin … end
    procedure SUB1;
    var Y, Z: integer;
    begin …end
begin … end
```



```
program MAIN;
    var X, Y, Z: char;
                SUB2
    var X, Y: integer;
          procedure SUB3;
           var X: real;
                 procedure SUB4;
                 begin …end
           begin … end
    begin … end
    procedure SUB1;
    var Y, Z: integer;
    begin …end
begin … end
```



Names

- User-defined names
  - Names of variables in declarations
- Design issues for names:
  - Maximum length?
  - Are connector characters allowed?
  - Are names case sensitive?
  - Are special words reserved words or keywords?
- Name = identifier?

Names

- Simple names

  a = 1; b = a+1;

- Composite names:

  a[1] = b.c[2];

Allocation

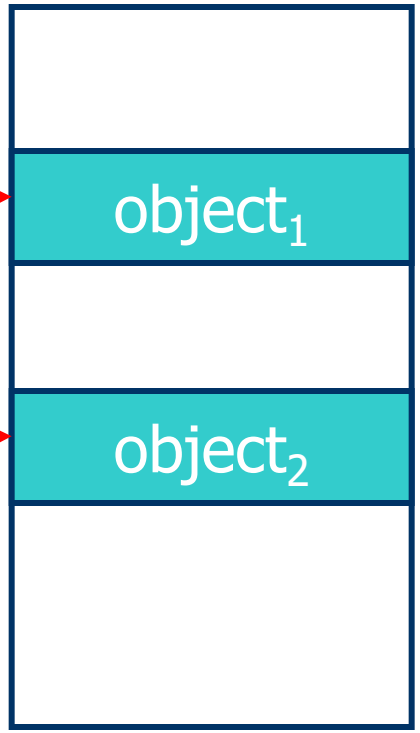
- Data object: a block of storage, containing data value
- Allocation: reserving a block for a data object
- Lifetime of a data object: the time its block is reserved.
- A data object may or may not have a name:

```
a = b +1*2
```

Association

- A binding between a name and a data object

```
program MAIN;
var X: integer;
...
    procedure SUB1;
    var X: real;
    ...
    procedure SUB3;
    ...
    X := 3;
```



Variables and Constants

- Variables: named data object whose values are changeable
- Constants: named data object whose values cannot be changed
  - Literals
  - User-defined constant