# 15 Static and Dynamic Typing and Scope

Monday, April 14, 2025 12:41 PM

"Static" - as in the text.
"Dynamic" - as the program runs.

- TYPE BINDING:

¿When does a variable gets its type?

1) Static Type Binding

Type is stated somewhere in the text:

- Explicit Statement

C++ `int x; string s;`

Pascal
```
VAR
  x: INTEGER;
  S: STRING;
```

- Implicit Declaration

FORTRAN I J K are integers
everything else is float
(unless otherwise specified)

PERL name carries type.

@x array
%x hashtable / dictionary
$x scalar.

2) Static Type Prediction

Type is deduced from initialization.

C++ auto

Go `VAR float x` `VAR x := 3.14`

Go

VAR float x
x = 3.14

VAR x := 3.14
use type deduction.

3) Dynamic Type Binding

- Type is bound on assignment as the program runs
- Type CAN change. by another assignment.

Python, JavaScript, Lua, Ruby.

- SCOPE:
  - The scope of a variable is the range of statements over which the variable is visible

1) Global BASIC

2) Static Scope:
range is determined by program text

C / C++
scope is based on "blocks"
{ block }

e.g

```
int z;                  <- global
int foo()
{
  int z;                <- local variable masks the global variable.
  {
    int z;              <- inner scope variable.
    z = 3+7
  }
}
```

Pascal scope is Module or Function based.

```
VAR  Z: INTEGER

FUNCTION  FOO( ) : INTEGER
VAR
    Z: INTEGER
BEGIN
    ≡
    Z := 3+7
    ≡
END.
```

} the scope of Z is the whole function

- How does the symbol table handles nested scopes?
  - Spit symbol table into "frames"
  - As the program is parsed:
    - new scope — push a new frame
    - end scope — pop a frame
  - Symbol table is now a stack

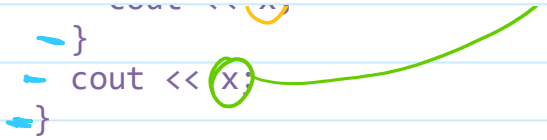
E.g C++

```
int x;
void foo ( int y )
{
  x = y * 10;
}

int main()
{
  int x, y;
  cin >> x >> y;
  if ( x<y ) {
    int x,z;
    cin >> z;
    x = y + z;
  }
  for( int x=0; x<y; x++) {
    cout << x;
  }
  cout << x;
}
```

Symbol table

| | | |
|---|---|---|
| | x | int |
| | foo | (int) |
| | main | () |
| → | y | int |
| main | | |
| | x | int |
| | y | int |
| if | | |
| | x | int |
| | z | int |
| for | | |
| | x | int |




3) Dynamic Scope:

- Used in "esoteric" interpreted languages

Lisp Bash