COP-3402 Systems Software
10.15 Tue

Reminder: **Project2 - Variables**

SimpleC Project2 only has one type
No need to implement true type checking
Symbol table only needs name and LLVM IR var
Symbol tables are dictionaries: map from key to value; linked list, hash table, dynamic array.
Ex:
emacs
read y
x = 1
if (y>10) {
        x = x+1
else 	*{*
        x = x -1
}
print x;
%x = 1; not valid llvm
br
%x = add nsw i32 %x, 1;
Reassigning the variable is not allowed by LLVM.
Optimize this allocation of registers to variables so you use the most number of variables and least number of stores and loads to memory.

Types:
We use types to prevent errors during runtime
In C, the compile has to check the type before generating machine instructions.
In C, you cannot assign a floating point to an integer.

A type is a set of values and operations on those values.
int: set of integers, arithmetic operations
bool: true/false, logic connectives (and, or, not)

Typed languages restrict variables range of values (Python, C, Java, etc.)
Untyped do not; Lisp, assembly

Runtime errors are
Trapped; terminated by machine, e.g., divide-by-zero,..
Untrapped; program continues...

Errors: Check happens;

Compile-time (C, Java)
Run-time (python)

Python type system. Ex:

```
x = “degrees”
x = 1.7 + “ degrees”
y = 1.7 * 2
print y
```

Run the program and gives type error: unsupported operand type(s) for +: ‘float’ and ‘str’

```
python
x = “degrees”
type(x)
<type ‘str’>

type(2)
<type ‘int’>
type(1.7)
<type ‘float’>
y = 1.7 * 2
print y
3.4
```

Python converts the type int to float for us.

```
10*(float(1)/10 + float(2)/10)
3.0000000000004
```

Floating-point returns this way because of scientific notation of it.

C type system. Ex:

```
double temp = (double)2;
int y = 1.7 * temp;
printf (“%x\n”, y);
```

Compile: This examples compiles and runs successfully and return 3.

```
gcc -Wall -o static static.c
./static
```

If the char has a pointer, it is a memory address and if we add it to an integer number, the program still compiles with a warning(makes integer from pointer without a cast). And it runs successfully.

Ex:

```
float x = 1.7;
int normalcast = x;
printf("%d\n", normalcast);

float *p = &x;

void *ip = (void *)p;
printf("%d\n", *(int *)ip);
return 0;
```

//First printf prints out 1. The second one prints the floating-point representation of 1.7 as an integer

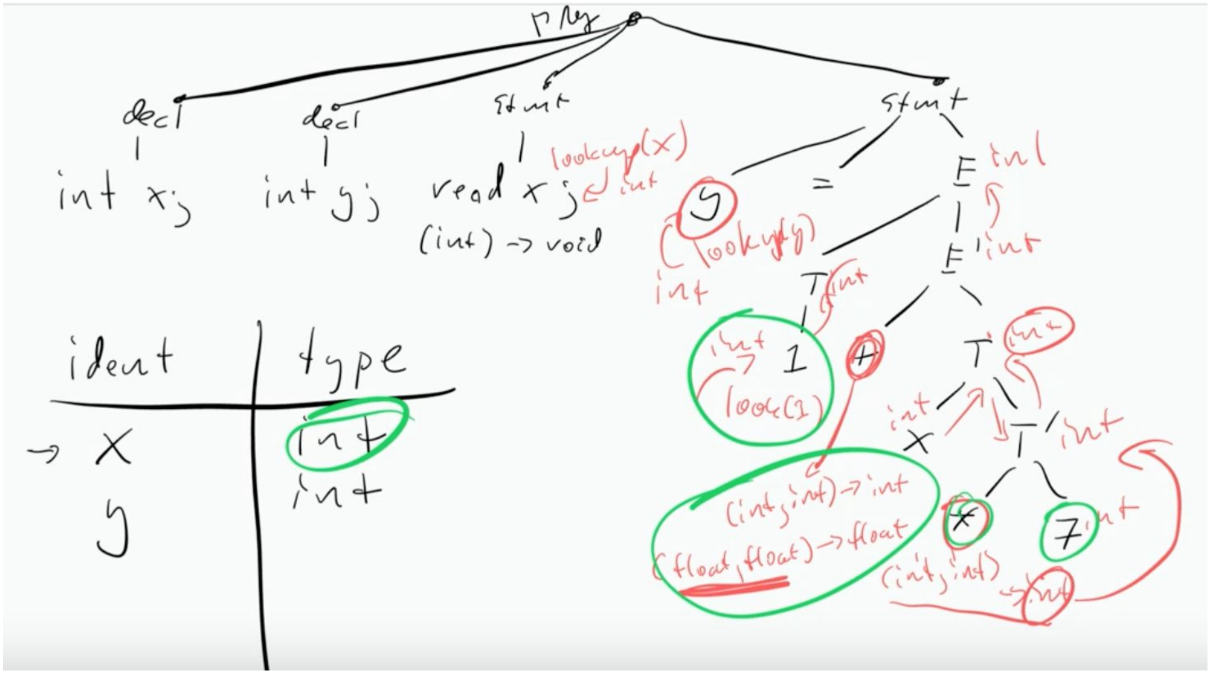
Static Type Checking
Record types of identifiers in symbol table
Post-order tree traversal
Check identifiers used in arithmetic operations, function calls, assignments.
Lookup type in symbol table
Constants have a fixed type

Function Types:
Scalar values have a primitive types: int, char, long, etc.
x: int
f: (int, int) -> bool

Symbol table holds variable name, variable type, and address.

Ex.:

```
int x;
int y;
read x;
y = 1 + x * 7;
```



ident	type
x	int
y	int