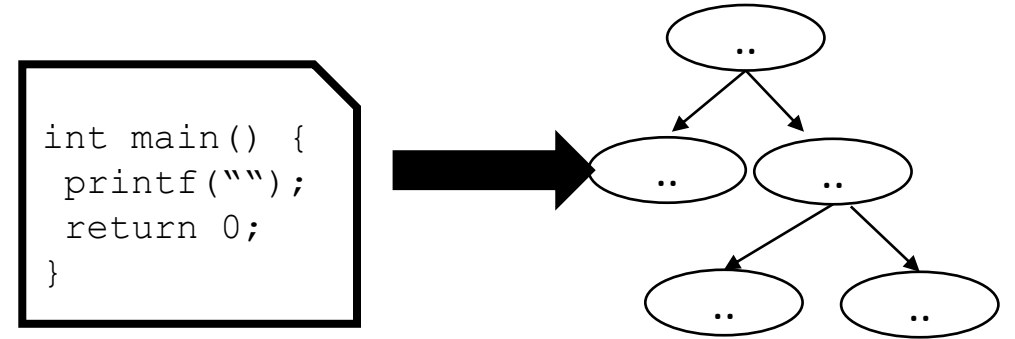


CSE110A: Compilers

Topics:

- *Scope*



Homework questions?

Scope

- What is scope?
- Can it be determined at compile time? Can it be determined at runtime?
- C vs. Python
- Anyone have any interesting scoping rules they know of?

One consideration: Scope

- Lexical scope example

```
int x = 0;
int y = 0;
{
    int y = 0;
    x+=1;
    y+=1;
}
x+=1;
y+=1;
```

What are the final values in x and y?

How to track scope?

- Symbol table object
- Two methods:
 - **lookup(id)** : lookup an id in the symbol table.
Returns None if the id is not in the symbol table.
 - **insert(id,info)** : insert a new id (or overwrite an existing id) into the symbol table along with a set of information about the id.

What information might we store about an id?

a very simple programming language

ID = [a-z]+

INCREMENT = “\+ \+”

TYPE = “int”

LBRAC = “{”

RBRAC = “}”

SEMI = “;”

```
int x;  
x++;  
int y;  
y++;
```

statements are either a declaration or an increment

a very simple programming language

ID = [a-z]+

INCREMENT = “\+ \+”

TYPE = “int”

LBRAC = “{”

RBRAC = “}”

SEMI = “;”

```
int x;  
{  
    int y;  
    x++;  
    y++;  
}  
y++;
```

statements are either a declaration or an increment

a very simple programming language

ID = [a-z]+

INCREMENT = "\+\+"

TYPE = "int"

LBRAC = "{"

RBRAC = "}"

SEMI = ";"

```
int x;  
{  
    int y;  
    x++;  
    y++;  
}  
y++;
```

error!

statements are either a declaration or an increment

How to track scope?

- `SymbolTable ST;`

Say we are matched the statement:
`int x;`

`declare_statement ::= TYPE ID SEMI`
`{ }`

lookup(id) : lookup an id in the symbol table. Returns None if the id is not in the symbol table.

insert(id,info) : insert a new id (or overwrite an existing id) into the symbol table along with a set of information about the id.

How to track scope?

- `SymbolTable ST;`

Say we are matched the statement:
`int x;`

`declare_statement ::= TYPE ID SEMI`

```
{  
    self.eat(TYPE)  
    variable_name = self.to_match[1] # lexeme value  
    self.eat(ID)  
    ST.insert(variable_name, None)  
    self.eat(SEMI)  
}
```

How to track scope?

- `SymbolTable ST;`

Say we are matched string:
`x++;`

`inc_statement ::= ID INCREMENT SEMI`
`{ }`

lookup(id) : lookup an id in the symbol table. Returns None if the id is not in the symbol table.

insert(id,info) : insert a new id (or overwrite an existing id) into the symbol table along with a set of information about the id.

How to track scope?

- `SymbolTable ST;`

Say we are matched string:
`x++;`

`inc_statement ::= ID INCREMENT SEMI`

```
{
    variable_name = self.to_match[1] # lexeme value
    if ST.lookup(variable_name) is None:
        raise SymbolTableException(variable_name)
    self.eat(ID)
    self.eat(INCREMENT)
    self.eat(SEMI)
}
```

How to track scope?

- SymbolTable ST;

statement : LBRAC statement_list RBRAC

```
int x;  
{  
    int y;  
    x++;  
    y++;  
}  
y++;
```

How to track scope?

- SymbolTable ST;

statement : LBRAC statement_list RBRAC

start a new scope S

remove the scope S

```
int x;
{
    int y;
    x++;
    y++;
}
y++;
```

How to track scope?

- Symbol table
- **four** methods:
 - **lookup(id)** : lookup an id in the symbol table.
Returns None if the id is not in the symbol table.
 - **insert(id,info)** : insert a new id into the symbol table along with a set of information about the id.
 - **push_scope()** : push a new scope to the symbol table
 - **pop_scope()** : pop a scope from the symbol table

How to track scope?

- `SymbolTable ST;`

statement : **LBRAC** statement_list **RBRAC**

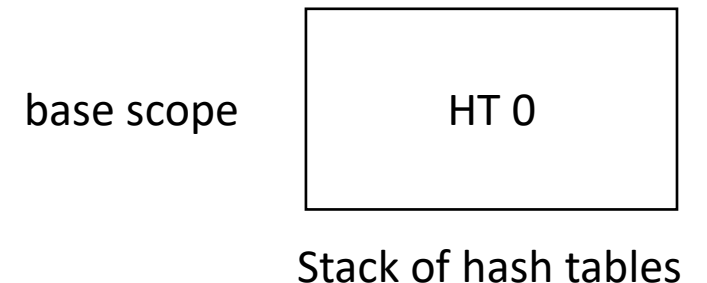
You will be adding the functions to push and pop scopes in your homework

How to implement a symbol table?

- Thoughts? What data structures are good at mapping strings?
- Symbol table
- **four** methods:
 - **lookup(id)** : lookup an id in the symbol table.
Returns None if the id is not in the symbol table.
 - **insert(id,info)** : insert a new id into the symbol table along with a set of information about the id.
 - **push_scope()** : push a new scope to the symbol table
 - **pop_scope()** : pop a scope from the symbol table

How to implement a symbol table?

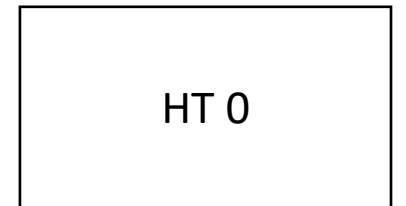
- Many ways to implement:
- A good way is a stack of hash tables:



How to implement a symbol table?

- Many ways to implement:
- A good way is a stack of hash tables:

`push_scope()`



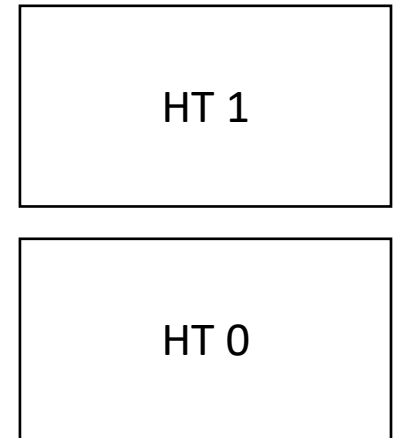
Stack of hash tables

How to implement a symbol table?

- Many ways to implement:
- A good way is a stack of hash tables:

*adds a new
Hash Table
to the top of the stack*

push_scope ()

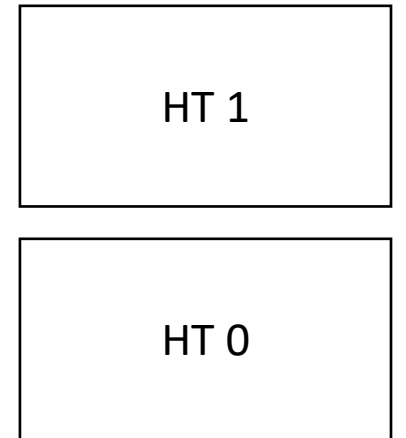


Stack of hash tables

How to implement a symbol table?

- Many ways to implement:
- A good way is a stack of hash tables:

`insert(id, data)`



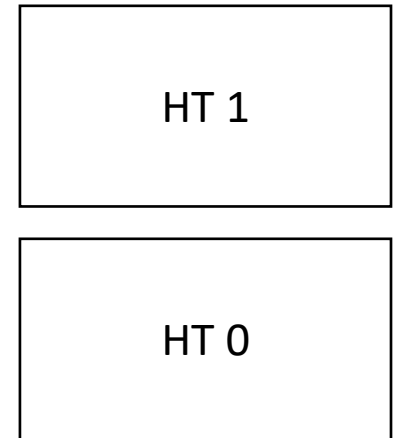
Stack of hash tables

How to implement a symbol table?

- Many ways to implement:
- A good way is a stack of hash tables:

`insert(id, data)`

`insert(id -> data)` at
top hash table

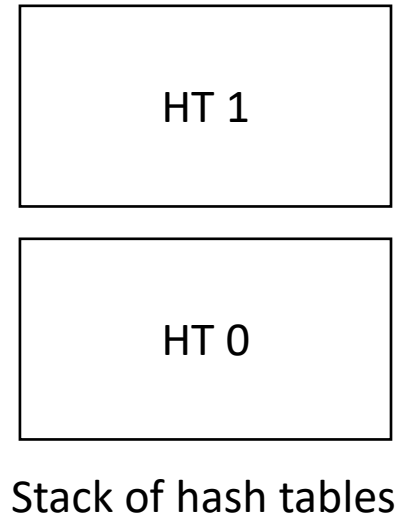


Stack of hash tables

How to implement a symbol table?

- Many ways to implement:
- A good way is a stack of hash tables:

`lookup(id)`

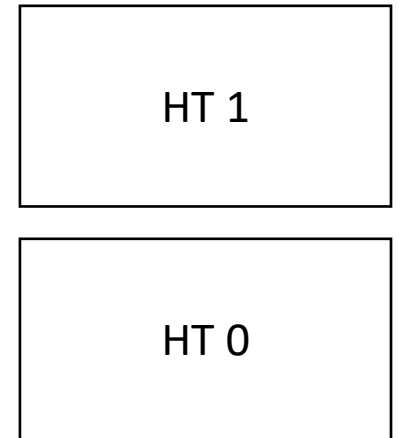


How to implement a symbol table?

- Many ways to implement:
- A good way is a stack of hash tables:

`lookup(id)`

check here
first



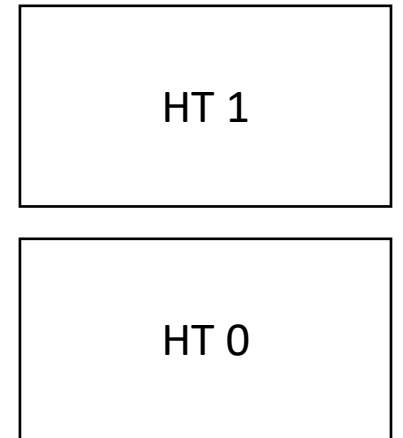
Stack of hash tables

How to implement a symbol table?

- Many ways to implement:
- A good way is a stack of hash tables:

`lookup(id)`

then check
here

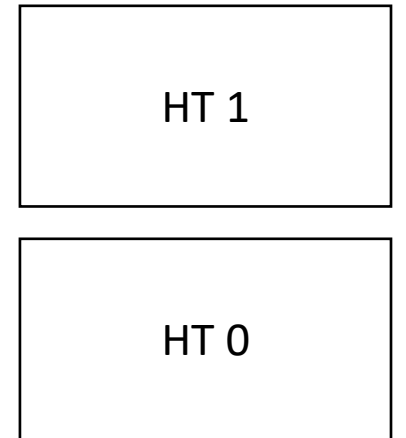


Stack of hash tables

How to implement a symbol table?

- Many ways to implement:
- A good way is a stack of hash tables:

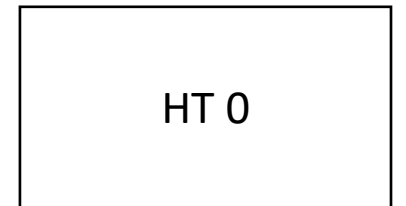
`pop_scope()`



Stack of hash tables

How to implement a symbol table?

- Many ways to implement:
- A good way is a stack of hash tables:



Stack of hash tables

How to implement a symbol table?

- Example

```
int x = 0;  
int y = 0;  
{  
    int y = 0;  
    x++;  
    y++;  
}  
x++;  
y++;
```



HT 0

Stack of hash tables

Next Topic:

- We will discuss parser generators