08 - Symbol Tables

Dr. Robert Lowe

Division of Mathematics and Computer Science
Maryville College

Outline

Outline

Purpose

- A symbol table tracks declarations within a program.

Purpose

- A symbol table tracks declarations within a program.
- If a language has more than one scope, it has more than one symbol table.

Purpose

- A symbol table tracks declarations within a program.
- If a language has more than one scope, it has more than one symbol table.
- When a symbol is used in a program, the symbol table(s) are checked to ensure that the symbol exists.

Purpose

- A symbol table tracks declarations within a program.
- If a language has more than one scope, it has more than one symbol table.
- When a symbol is used in a program, the symbol table(s) are checked to ensure that the symbol exists.
- Symbol tables account for the unpredictable nature of programmer symbols.

Error Detection

- Symbol tables are used to perform error detecting during syntax analysis.

Error Detection

- Symbol tables are used to perform error detecting during syntax analysis.
- **Re-declaration Errors** – when a symbol which is already in the table is declared again.

Error Detection

- Symbol tables are used to perform error detecting during syntax analysis.
- **Re-declaration Errors** – when a symbol which is already in the table is declared again.
- **Undefined Symbol Errors** – when a symbol is used before it is declared.

Error Detection

- Symbol tables are used to perform error detecting during syntax analysis.
- **Re-declaration Errors** – when a symbol which is already in the table is declared again.
- **Undefined Symbol Errors** – when a symbol is used before it is declared.
- **Type Error** – when a symbol's type makes it invalid in some context.

Error Detection

- Symbol tables are used to perform error detecting during syntax analysis.
- **Re-declaration Errors** – when a symbol which is already in the table is declared again.
- **Undefined Symbol Errors** – when a symbol is used before it is declared.
- **Type Error** – when a symbol's type makes it invalid in some context.
- Note that whether these are errors is dependent upon the programming language in question.

Symbol Table Functions

- Symbol tables have two basic operations:

Symbol Table Functions

- Symbol tables have two basic operations:

  `declare_symbol(s, t)` – Declare a symbol `s` of some type `t`.

Symbol Table Functions

- Symbol tables have two basic operations:

`declare_symbol(s, t)`	– Declare a symbol `s` of some type `t`.
`check_symbol(s)`	– Check to see if a symbol exists in the table.

Symbol Table Functions

- Symbol tables have two basic operations:

  `declare_symbol(s, t)` – Declare a symbol `s` of some type `t`.

  `check_symbol(s)` – Check to see if a symbol exists in the table.

- Note that both of these functions should raise an error should they detect one.

Language Implications

- When does declaration of symbols occur in a language?

Language Implications

- When does declaration of symbols occur in a language?
- What types exist? (We will do more with type checking layer)

Language Implications

- When does declaration of symbols occur in a language?
- What types exist? (We will do more with type checking layer)
- How does a language cope with undefined symbols?

Language Implications

- When does declaration of symbols occur in a language?
- What types exist? (We will do more with type checking layer)
- How does a language cope with undefined symbols?
- When are symbols used in the language?

Integration With Syntax Analyzer

- Any part of the grammar which declares a symbol must add the symbol to the appropriate symbol table.

Integration With Syntax Analyzer

- Any part of the grammar which declares a symbol must add the symbol to the appropriate symbol table.
- Any part of the grammar which uses a symbol must check the symbol table to see if the symbol exists.

Integration With Syntax Analyzer

- Any part of the grammar which declares a symbol must add the symbol to the appropriate symbol table.
- Any part of the grammar which uses a symbol must check the symbol table to see if the symbol exists.
- Errors in the symbol table should be detected and handled as the program is parsed.

Outline

Required Information

- The symbol table should maintain some information about the symbols in a program.

Required Information

- The symbol table should maintain some information about the symbols in a program.
- The name of the symbol.

Required Information

- The symbol table should maintain some information about the symbols in a program.
- The name of the symbol.
- The type of the symbol.

Example Symbol Table

- For example, consider the following C program:

```
int main()
{
    int var1;
    double var2;
}
```

Example Symbol Table

- For example, consider the following C program:

```
int main()
{
    int var1;
    double var2;
}
```

- The symbol tables would be:

Example Symbol Table

- For example, consider the following C program:

```
int main()
{
    int var1;
    double var2;
}
```

- The symbol tables would be:

Example Symbol Table

- For example, consider the following C program:

```
int main()
{
     int var1;
     double var2;
}
```

- The symbol tables would be:

Global Scope

Symbol	**Type**
`main`	Integer Function

Example Symbol Table

- For example, consider the following C program:

```
int main()
{
    int var1;
    double var2;
}
```

- The symbol tables would be:

Global Scope

Symbol	Type
`main`	Integer Function

Local Scope for `main`

Symbol	Type
`var1`	`int`
`var2`	`double`

Common Data Structures

- Symbols need to be added and searched efficiently.

Common Data Structures

- Symbols need to be added and searched efficiently.
- Common implementation strategies include:

Common Data Structures

- Symbols need to be added and searched efficiently.
- Common implementation strategies include:
    - Hash Table

Common Data Structures

- Symbols need to be added and searched efficiently.
- Common implementation strategies include:
  - Hash Table
  - Binary Search Trees

Common Data Structures

- Symbols need to be added and searched efficiently.
- Common implementation strategies include:
  - Hash Table
  - Binary Search Trees
  - Sorted Linked Lists

Common Data Structures

- Symbols need to be added and searched efficiently.
- Common implementation strategies include:
    - Hash Table
    - Binary Search Trees
    - Sorted Linked Lists
    - Sorted Vectors

Common Data Structures

- Symbols need to be added and searched efficiently.
- Common implementation strategies include:
  - Hash Table
  - Binary Search Trees
  - Sorted Linked Lists
  - Sorted Vectors
- Hash tables are by far the most common implementation method.

Common Data Structures

- Symbols need to be added and searched efficiently.
- Common implementation strategies include:
  - Hash Table
  - Binary Search Trees
  - Sorted Linked Lists
  - Sorted Vectors
- Hash tables are by far the most common implementation method.
- Discuss: Why use hash tables?

C++ `map`

- The STL contains a structure called `map`, which is an associated list of key value pairs.

C++ `map`

- The STL contains a structure called `map`, which is an associated list of key value pairs.
- Maps are declared as follows:

```
map<key_type, val_type> name;
```

C++ `map`

- The STL contains a structure called `map`, which is an associated list of key value pairs.
- Maps are declared as follows:

```
map<key_type, val_type> name;
```

- For a symbol tables `key_type` should probably be `string` and the `val_type` should be some appropriate representation of the symbol's type.

C++ `map`

- The STL contains a structure called `map`, which is an associated list of key value pairs.
- Maps are declared as follows:
  `map<key_type, val_type> name;`
- For a symbol tables `key_type` should probably be `string` and the `val_type` should be some appropriate representation of the symbol's type.
- There are two critical functions for symbol table use:

C++ `map`

- The STL contains a structure called `map`, which is an associated list of key value pairs.
- Maps are declared as follows:
```
map<key_type, val_type> name;
```
- For a symbol tables `key_type` should probably be `string` and the `val_type` should be some appropriate representation of the symbol's type.
- There are two critical functions for symbol table use:
    - `operator[]` – Index operations for insertion:
```
table["var1"] = integer_type;
```

C++ `map`

- The STL contains a structure called `map`, which is an associated list of key value pairs.
- Maps are declared as follows:

```
map<key_type, val_type> name;
```

- For a symbol tables `key_type` should probably be `string` and the `val_type` should be some appropriate representation of the symbol's type.
- There are two critical functions for symbol table use:
  - `operator[]` – Index operations for insertion:

    ```
    table["var1"] = integer_type;
    ```
  - `count(key)` – Count the number of elements matching the key (0 or 1).

Outline

Declarations

- When does looplang declare variables?

Declarations

- When does looplang declare variables?
- Looplang declares variables on the first assignment.

Declarations

- When does looplang declare variables?
- Looplang declares variables on the first assignment.
- What types can looplang symbols take?

Declarations

- When does looplang declare variables?
- Looplang declares variables on the first assignment.
- What types can looplang symbols take?
- Just one, integer.

Declarations

- When does looplang declare variables?
- Looplang declares variables on the first assignment.
- What types can looplang symbols take?
- Just one, integer.
- We could just use a bool to indicate presence of a symbol.

Declarations

- When does looplang declare variables?
- Looplang declares variables on the first assignment.
- What types can looplang symbols take?
- Just one, integer.
- We could just use a bool to indicate presence of a symbol.
- Here is pseudocode for when we do a declaration:

```
During the parsing of assignments:
if s does not exist in the table
    table[s] = true
```

Symbol Use

- When are symbols used in looplang?

Symbol Use

- When are symbols used in looplang?
- They can be in any operand, or on the left hand side of assignment.

Symbol Use

- When are symbols used in looplang?
- They can be in any operand, or on the left hand side of assignment.
- Operand handling:

```
In operand parsing:
if table.count(s) == 0
   error!
```