编译第十三次作业

22373407 王飞阳

1.数组文法及其语义动作程序

数组文法

```
<program> ::= <declaration_list>

<declaration_list> ::= <declaration> <declaration_list> | <declaration>

<declaration> ::= <type> <identifier> '[' <number> ']' ';' | <type> <identifier>
'=' <array_initialization> ';'

<type> ::= 'int'  | 'char'

<identifier> ::= [a-zA-Z_][a-zA-Z0-9_]*

<number> ::= [0-9]+

<array_initialization> ::= '{' <initialization_list> '}'

<initialization_list> ::= <expression> ',' <initialization_list> | <expression>

<expression> ::= <number> | <identifier> '[' <number> ']'
```

语义动作程序

`<declaration_list> ::= <declaration> <declaration_list>`

- **语义动作**：将新声明的数组加入符号表，维护一个符号表（symbol table）来存储数组的信息。
- **伪代码**：

```
declaration_list → declaration declaration_list {
    append(declaration, declaration_list);
}
```

`<declaration> ::= <type> <identifier> '[' <number> ']' ';'`

- **语义动作**：记录数组类型、名称和大小，存储到符号表中。
- **伪代码**：

```
declaration → type identifier '[' number ']' ';' {
    symbol_table[identifier] = { "type": type, "size": number };
}
```

`<declaration> ::= <type> <identifier> '=' <array_initialization> ';'`

- **语义动作**：初始化数组，将初始化列表中的值与数组名关联。
- **伪代码**：

```
declaration → type identifier '=' array_initialization ';' {
    symbol_table[identifier] = { "type": type, "size":
len(array_initialization), "values": array_initialization };
}
```

`<array_initialization> ::= '{' <initialization_list> '}'`

- **语义动作**：构建一个包含初始化值的数组。
- **伪代码**：

```
array_initialization → '{' initialization_list '}' {
    $$ = initialization_list;
}
```

`<initialization_list> ::= <expression> ',' <initialization_list>`

- **语义动作**：将表达式的值添加到初始化列表中。
- **伪代码**：

```
initialization_list → expression ',' initialization_list {
    append(expression, initialization_list);
}
```

`<expression> ::= <number>`

- **语义动作**：处理数字常量并返回其值。
- **伪代码**：

```
expression → number {
    $$ = number;
}
```

`<expression> ::= <identifier> '[' <number> ']'`

- **语义动作**：处理数组元素的访问，并返回该元素的值。
- **伪代码**：

```
expression → identifier '[' number ']' {
    $$ = symbol_table[identifier]["values"][number];
}
```

2.结构体声明文法及其语义动作程序

数组文法

```
<program> ::= <struct_declaration_list>

<struct_declaration_list> ::= <struct_declaration> <struct_declaration_list> |
<struct_declaration>

<struct_declaration> ::= 'struct' <identifier> '{' <member_list> '}' ';'
```

```
<member_list> ::= <member_declaration> <member_list> | <member_declaration>

<member_declaration> ::= <type> <identifier> ';'

<type> ::= 'int' | 'float' | 'char' | 'struct' <identifier>

<identifier> ::= [a-zA-Z_][a-zA-Z0-9_]*

<struct_variable_declaration> ::= 'struct' <identifier> <identifier> ';'

<struct_access> ::= <identifier> '.' <identifier>
```

语义动作程序

`<struct_declaration_list> ::= <struct_declaration> <struct_declaration_list>`

- **语义动作**：将每个解析的结构体声明加入符号表。
- **伪代码**：

```
struct_declaration_list → struct_declaration struct_declaration_list {
    append(struct_declaration, struct_declaration_list);
}
```

`<struct_declaration> ::= 'struct' <identifier> '{' <member_list> '}' ';'`

- **语义动作**：记录结构体名称和其成员列表，将其添加到符号表中。
- **伪代码**：

```
struct_declaration → 'struct' identifier '{' member_list '}' ';' {
    symbol_table[identifier] = { "type": "struct", "members": member_list };
}
```

`<member_list> ::= <member_declaration> <member_list>`

- **语义动作**：构建结构体成员列表。
- **伪代码**：

```
member_list → member_declaration member_list {
    append(member_declaration, member_list);
}
```

`<member_declaration> ::= <type> <identifier> ';'`

- **语义动作**：记录成员的类型和名称。
- **伪代码**：

```
member_declaration → type identifier ';' {
    $$ = { "name": identifier, "type": type };
}
```

`<struct_variable_declaration> ::= 'struct' <identifier> <identifier> ';'`

- **语义动作**：声明结构体变量，将其类型和名称加入符号表。

- **伪代码**:

```
struct_variable_declaration → 'struct' identifier identifier ';' {
    if (symbol_table[identifier].type != "struct") {
        error("Unknown struct type: " + identifier);
    }
    symbol_table[identifier2] = { "type": identifier, "struct":
symbol_table[identifier] };
}
```

`<struct_access> ::= <identifier> '.' <identifier>`

- **语义动作**：通过符号表查找结构体的成员并返回对应的值。
- **伪代码**:

```
struct_access → identifier '.' identifier {
    struct_info = symbol_table[identifier];
    if (struct_info.type != "struct") {
        error(identifier + " is not a struct");
    }
    member_info = find_member(struct_info, identifier2);
    if (member_info == null) {
        error("Unknown member " + identifier2 + " in struct " + identifier);
    }
    $$ = member_info;
}
```