Abstract Syntax Tree

Compiler Construction

2024-11-12

Prof. Dr. Michael Kuhn
michael.kuhn@ovgu.de

Parallel Computing and I/O
Institute for Intelligent Cooperating Systems
Faculty of Computer Science
Otto von Guericke University Magdeburg
https://parcio.ovgu.de

Outline

Abstract Syntax Tree

- Abstract syntax tree (AST) represents the structure of a program
- Leaves out certain details of parsing
  - For example, it does not matter whether prefix, postfix or infix expressions are used
- For B-Minor, we care about five structures

Structures

1. Declarations
2. Statements
3. Expressions
4. Types
5. Parameters

Declarations

- Declares the existence of a variable or function
- A program is a list of declarations

```
1 struct decl {
2     char* name;
3     struct type* type;
4     struct expr* value;
5     struct stmt* code;
6     struct decl* next;
7 };
```

decl				
name	type	value	code	next

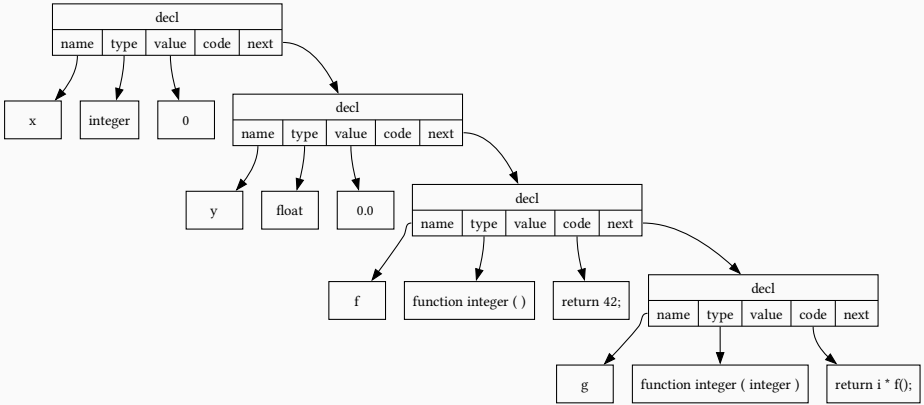
Group Exercise

- Draw structs for the following program.

```
1 x: integer;
2 y: float = 0.0;
3 f: function integer ( ) = { return 42; }
4 g: function integer ( i: integer ) = { return i * f();  }
```

Group Exercise

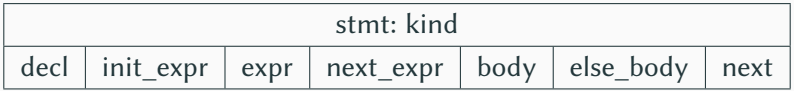
- Draw structs for the following program.



Statements

- Statements specify actions the program should perform
- The body of a function is made up of a list of statements

```
1  struct stmt {
2      stmt_t kind;
3      struct decl* decl;
4      struct expr* init_expr;
5      struct expr* expr;
6      struct expr* next_expr;
7      struct stmt* body;
8      struct stmt* else_body;
9      struct stmt* next;
10 };
```



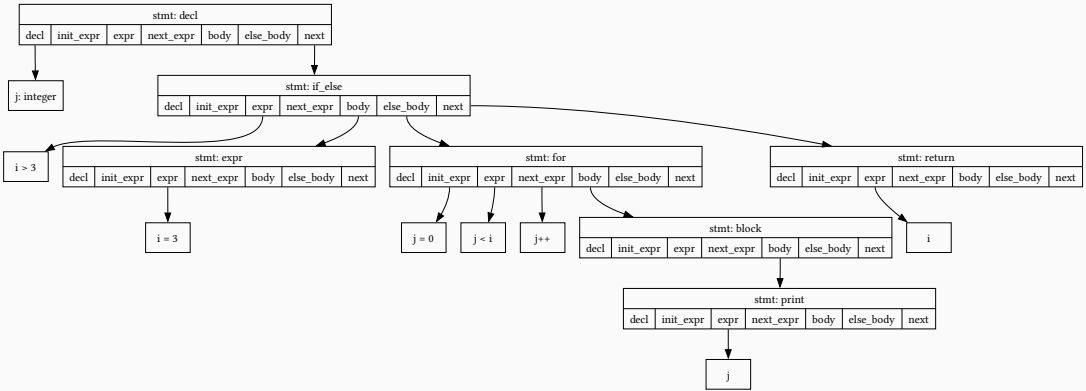
Group Exercise

- Draw structs for the body of the following function.

```
1  f: function integer ( i: integer ) = {
2      j: integer;
3      if (i > 3)
4          i = 3;
5      else
6          for (j = 0; j < i; j++) {
7              print j;
8          }
9      return i;
10 }
```

Group Exercise

- Draw structs for the body of the following function.



- Could the struct be optimized?

- Could the struct be optimized?

```
1  struct stmt {
2      stmt_t kind;
3      union {
4          struct {
5              struct decl* decl;
6          } decl;
7          struct {
8              struct expr* expr;
9              struct stmt* body;
10             struct stmt* else_body;
11         } if_else;
12         ...
13     }
14 };
```

Expressions

- Expressions can be used in declarations, statements etc.

```
1 struct expr {
2     expr_t kind;
3     struct expr* left;
4     struct expr* right;
5     const char* name;
6     int integer_value;
7     const char* string_literal;
8 };
```

expr: kind					
left	right	name	integer_value	string_literal	...

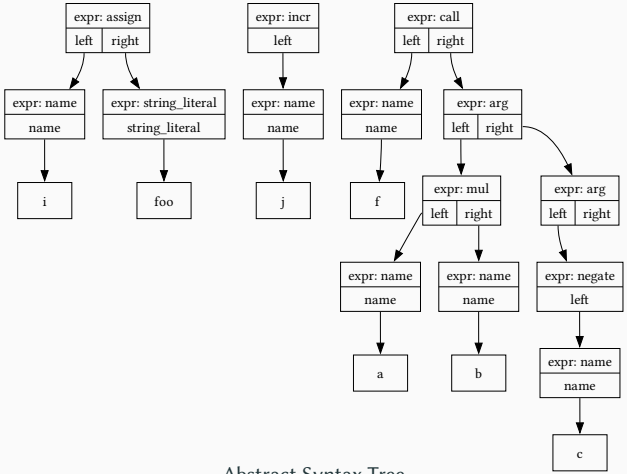
Group Exercise

- Draw structs for the following expressions.

```
1 i = "foo"
2 j++
3 f(a * b, -c)
```

Group Exercise

- Draw structs for the following expressions.



Types

- Types are used to specify the data types of variables and function return values

```
1  struct type {
2      type_t kind;
3      struct type* subtype;
4      struct param_list* params;
5  };
6  struct param_list {
7      char* name;
8      struct type* type;
9      struct param_list* next;
10 };
```

type: kind	
subtype	params

param_list		
name	type	next

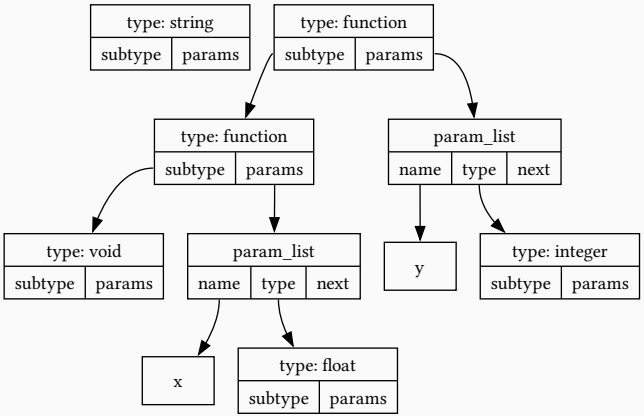
Group Exercise

- Draw structs for the following types.

```
1 string
2 function function void ( x: float ) ( y: integer )
```

Group Exercise

- Draw structs for the following types.



Outline

Summary

- AST represents a program's structure
  - Some parsing details are not visible from the AST
- Declarations, statements, expressions, types and parameters are relevant
  - Each consists of a struct for internal management
- Structures can be connected to represent complex programs
  - Some more infrastructure code is necessary to manage everything

References

[Thain, 2020] Thain, D. (2020). ***Introduction to Compilers and Language Design: Second Edition.*** http://compilerbook.org/.