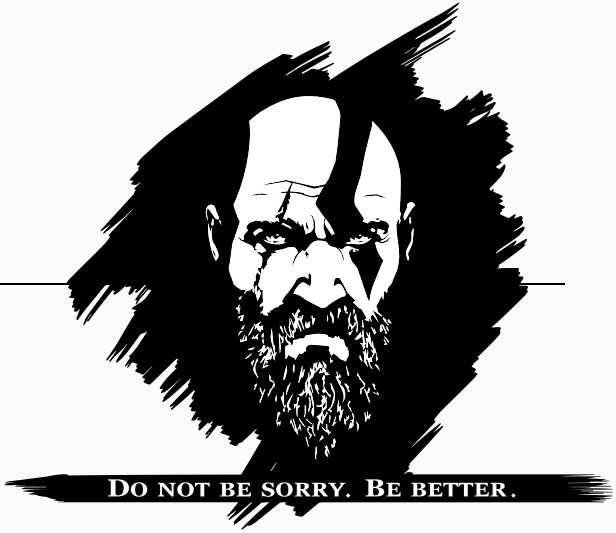
42sh - Presentation

ACU 2019 Team



Copyright

Rules

- You must have downloaded your copy from the Assistants' Intranet <https://intra.assistants.epita.fr>.
- This document is strictly personal and must **not** be passed on to someone else.
- Non-compliance with these rules can lead to severe sanctions.

Recap

Abstract Syntax Tree

Definition

In computer science, an abstract syntax tree (AST), or just syntax tree, is a tree representation of the abstract syntactic structure of source code written in a programming language. – Wikipedia

Example

```
while b != 0
    if a > b
        a = a - b
    else
        b = b - a
return a
```

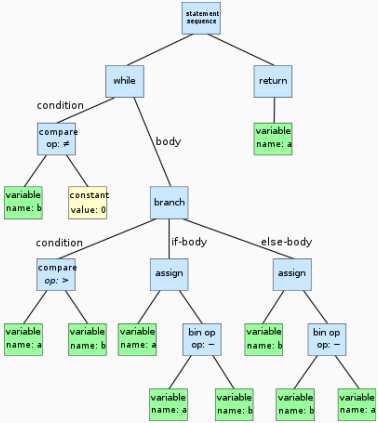


Figure 1: An AST example

Implementation of general tree

General Tree

```c
struct ast_node
{
    enum ast_node_type type;   // type of the node
    size_t nb_children;        // size of children
    struct ast_node *children; // array of children
};
```

General Tree

Pros

* Only one struct to write

Cons

* Very bug friendly
* Children can be any kind of node

Implementation of a “Object Oriented” Tree

Object Oriented Tree

```c
struct ast_node_if
{
    struct ast_node_compound_list *condition; // the condition
    struct ast_node_compound_list *if_body;   // the body of the if
    struct ast_node_compound_list *else_body; // the body of the else, may be NULL
};
```

Object Oriented Tree

```
shell_command:
        |   rule_for
        |   rule_while
        |   rule_case
        |   rule_if
```

Union Time!

```
union foo
{
    size_t a;
    struct my_struc b;
    struct my_struc *c;
};
```

- `foo` can be a size_t, a my_struct or a pointer on my_struct
- `sizeof(foo)` = MAX(sizeof(size_t), sizeof(my_struct), sizeof(my_struct *))

Object Oriented Tree

```c
enum shell_command_child_type
{
    FOR,
    WHILE,
    CASE,
    IF
};

union shell_command_child
{
    struct ast_node_for*;
    struct ast_node_while*;
    struct ast_node_case*;
    struct ast_node_if*;
};
```

```c
struct ast_node_shell_command
{
    enum shell_command_child_type type;
    union shell_command_child child;
};
```

Object Oriented Tree

Pros

* Easy to debug
* No void* or equivalent

Cons

* A lot of code to write
* You have ”useless” node

Create an AST

```
struct ast_node_if *create_node_if(ast_node_compound_list *condition,
                                   ast_node_compound_list *if_body,
                                   ast_node_compound_list *else_body);
```

Parsing

```c
struct ast_node_if *parse_rule_if(lexer_t *lexer)
{
    if (peek(lexer, TOKEN_IF) != 0)
        return NULL;
    pop(lexer);
    ast_node_compound_list *condition = parse_rule_compound_list(lexer);
    if (peek(lexer, TOKEN_THEN) != 0)
        return NULL;
    pop(lexer);
    ast_node_compound_list *if_body = parse_rule_compound_list(lexer);
    ast_node_compound_list *else_body = NULL;
    if (peek(lexer, TOKEN_ELSE) != 0)
    {
        pop(lexer);
        else_body = parse_rule_compound_list(lexer);
    }
    return create_node_if(condition, if_body, else_body);
}
```

Questions

Questions?