Tree++

PLT FALL 2018

Team Members

Allison Costa
Arc2211

MANAGER

Laura Matos
Lm3081

TESTER

Jacob Penn
jp3666

SYSTEM ARCHITECT

Laura Smerling
les2206

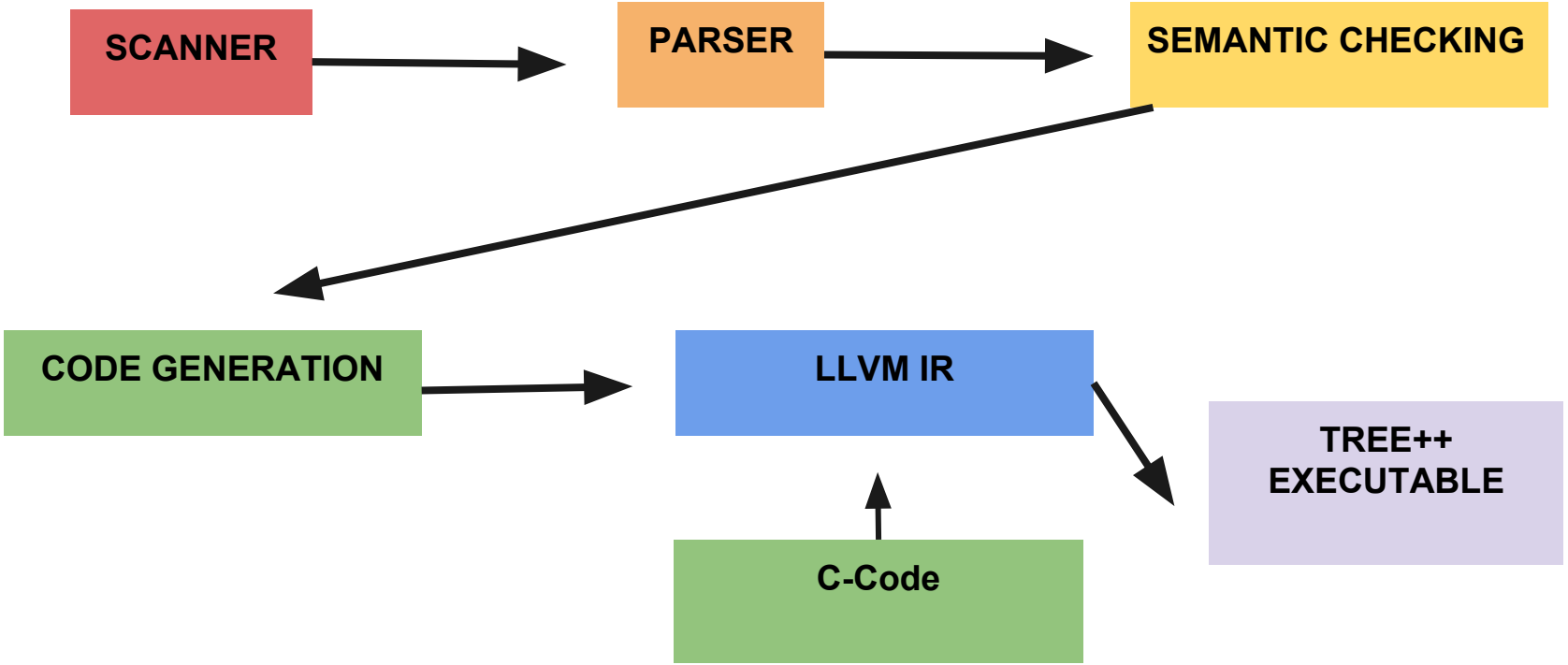
LANGUAGE GURU

TA: Justin Wong

OVERVIEW

- A general purpose programming language that allows easy manipulation of nodes in trees
- We wanted users to be able to use trees free from any other data structure including a function: easy, simple manipulation without wrappers
- To have user think in terms of trees
- Definition of our program: Our program is a list of items made up of statements and functions
- Gap between final output and semester long work

ARCHITECTURE



TYPES

- INT FLOAT BOOL STRING VOID
- All types allow for inline declaration and assignment
- Node<type> : node must have any of the above types
- Tree++ has explicitly typed declarations
- int x = 5; bool z= “true”; string node_t = “leaf”;
- Node<string> x = (node_t); or Node<string> = (“leaf”)
  - If the user tries to have children of different types we will throw an error

SYNTAX

CONTROL FLOW

```
node<string> hello_world = (“root”);
hello_world.root;
…
node<string> n = (“hello”);
hello_world.add_child(n);
node<string> m = (“world”);
hello_world.add_child(m);
printn(hello_world);
int x = 0;
while(x<1){
        hello_world <<;  /* shifts the child nodes left*/
        x = x+1;
}
printn(hello_world);
```

Output: root hello world root world hello

FUNCTION DECLARATION

```
node<string> h = (“hello”);
h.root;
node<string> m = (“world”);
h.add_child(m);
def node<string> rotate(node<string> root, node<string>
child ){
        root^child;
        return root;
}

printn(rotate(h));
```

Output: world hello /*the root is now the child and the child is now the root*/

Tree++ Features

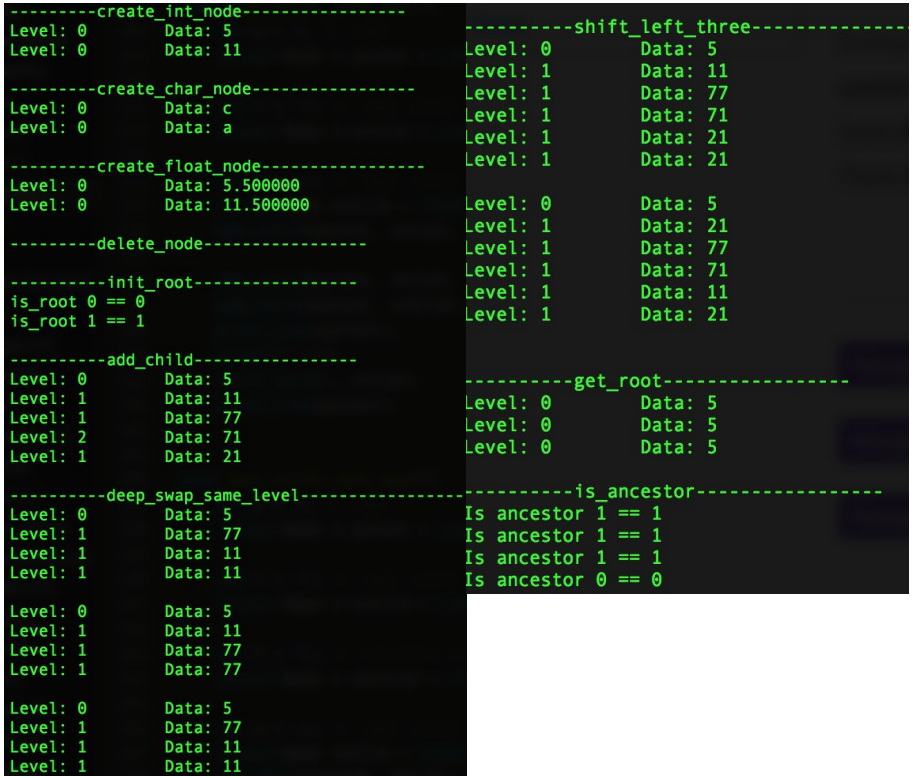
PARSER

```
| "node"   { NODE }
| ".root"   { ROOT }
| ".data"  { DATA }
| ".depth" { NODE_DEPTH }
| "<<"     { LSHIFT_NODE }
| ">>"     { RSHIFT_NODE }
| "^"      { SWAP_NODE  }
| ".add_child" { ADD_CHILD }
| ".delete_node" { DELETE_NODE }
```

C-Functions

```
void init_root(struct Node *node); // done
struct Node *create_int_node(int data); // done
struct Node *create_char_node(char data); // done
struct Node *create_float_node(float data); // done
void delete_node(struct Node *node); // done
void add_child(struct Node *parent, struct Node *child); // done
void deep_swap(struct Node *node_a, struct Node *node_b); // done
void shift_left(int index, struct Node *child); // done
void shift_right(int index, struct Node *child); // done
int is_root(struct Node *node); // done
int is_empty(struct Node *node); // done
void add_child(struct Node *parent, struct Node *child); // done
int is_root(struct Node *node); // done
int is_empty(struct Node *node); // done
int get_depth(struct Node *node); // done
struct Node *get_root(struct Node *node); // done
```

TESTING - C Backend

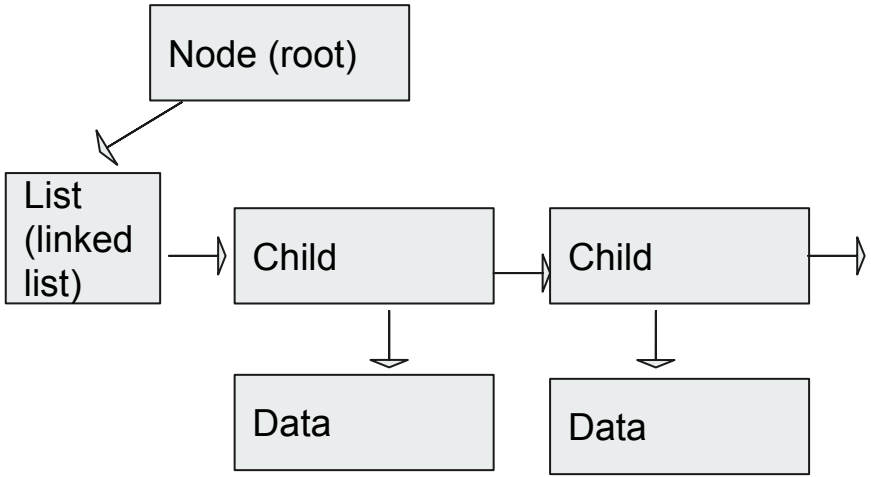


Unlike testing outside of the c_code directory, testing for the C backend is slightly different

Seperate test for C backend files managed by a separate Makefile exclusive to only the branches for modifying the C backend files.

Focused on unit tests and more verbose than regular tests

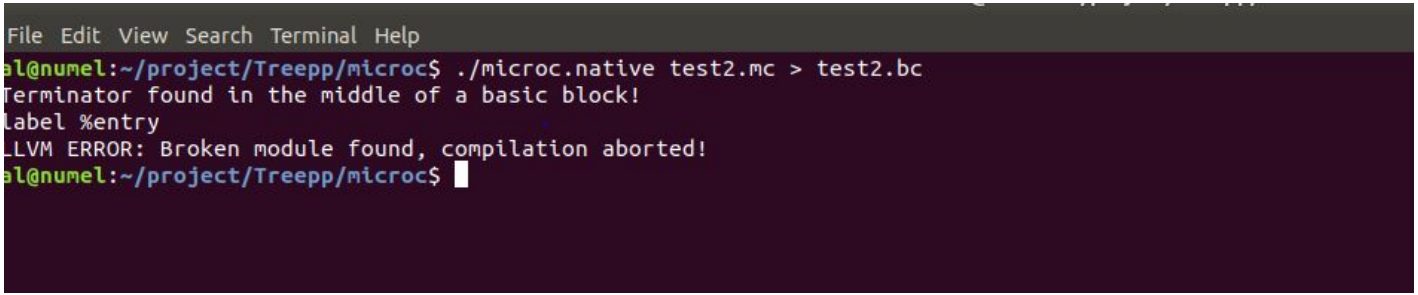
C Backend




```c
// CREATE HASH TABLE FOR STRING/VALUE AND POINTER
typedef enum {INT, CHAR, FLOAT, BOOL} data_type;
union data_u {
    int i;
    char c;
    float f;
};

struct Node {
    int32_t level;
    int32_t root;
    struct Node *next;
    struct Node *prev;
    struct Node *parent;

    struct List *children;
    int visited;

    data_type dtype;
    union data_u * data;
};
```

BEHIND THE SCENES

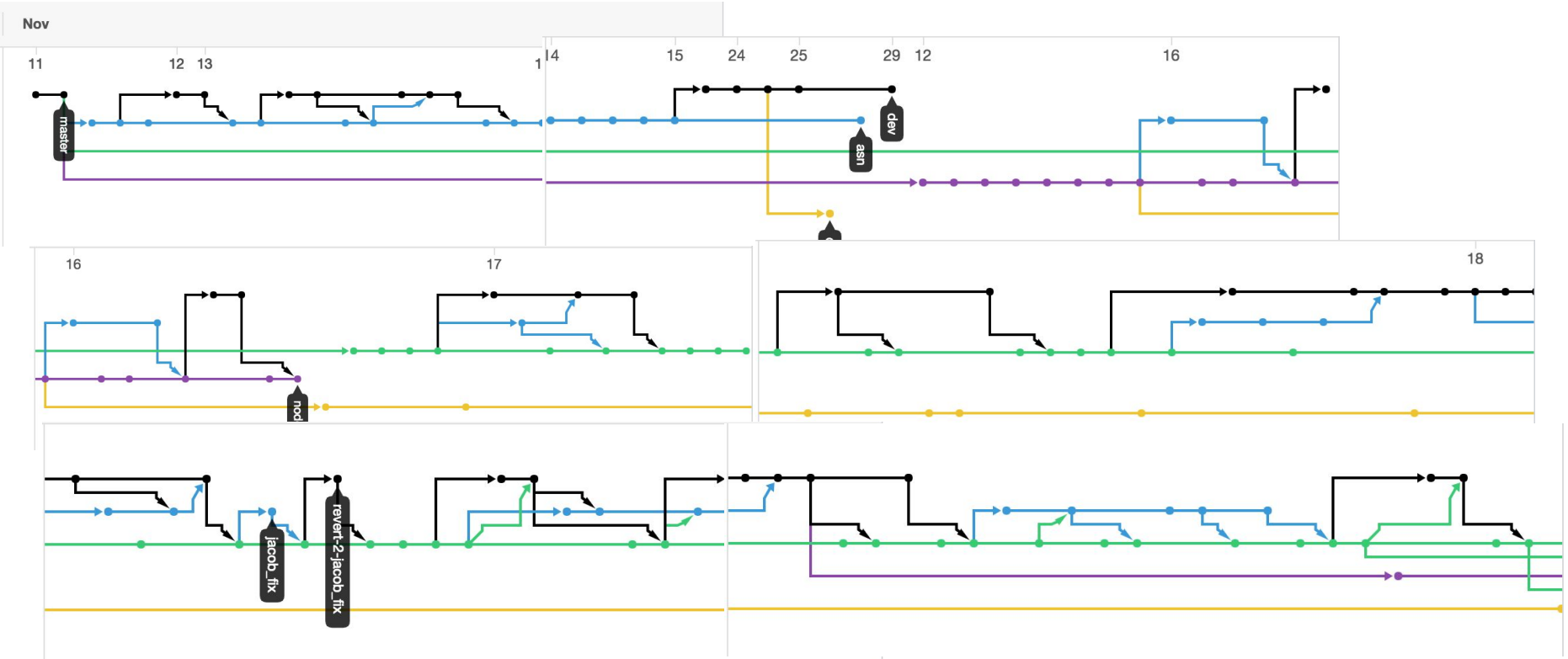
- Our main is hidden to give the user more access to manipulate functions without worry
- This ultimately lead to the major problem in our code



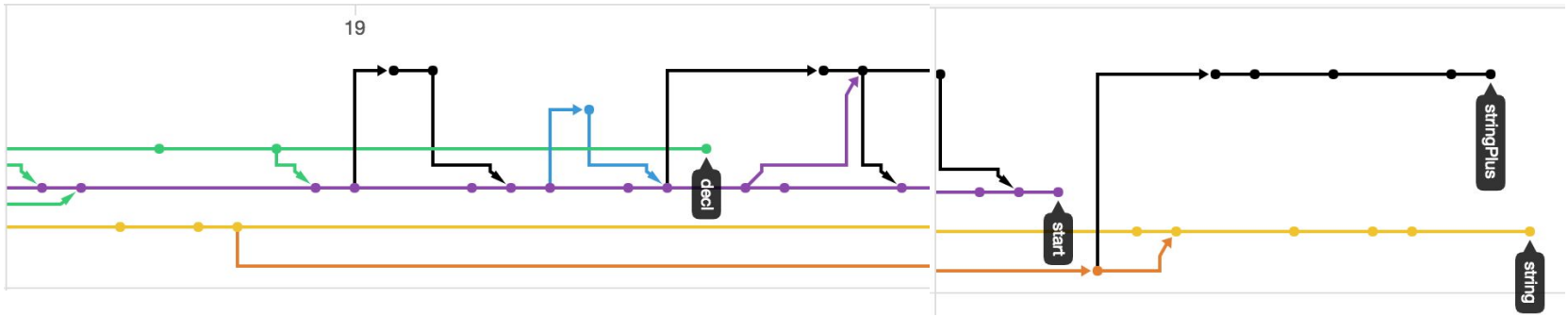
PROCESS

- Started coding from scratch
- Started anew with MicroC for Hello World
- Inspired by many past projects: especially Workspace, Giraph, BURGer, and PLTree
- Realization that code has fatale error
- Building up MicroC

Git Repository



Git Repository



LESSONS LEARNED

Don’t try to recreate the wheel when there are examples you can easily reference to help speed up understanding the process. -- Laura Matos

When you hit an error ask for help to see if there is an easy fix that you were unaware of --Laura Smerling

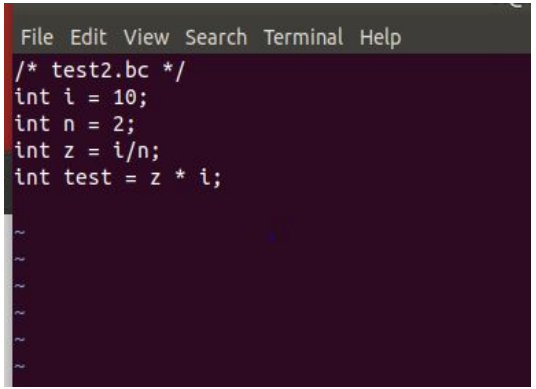
I gained a deep appreciation for the fact testing in isolation and compiling is not the same as testing a program as a whole. -- Allison Costa

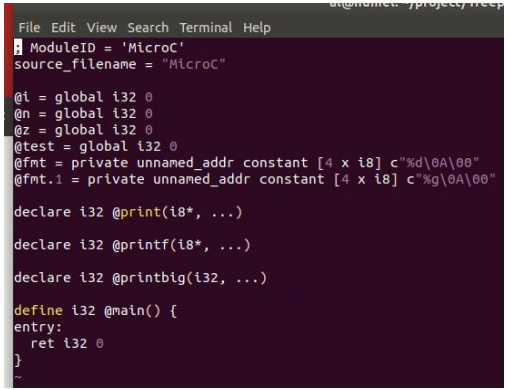
DEMO

To most accurately show our work we are presenting both our (not working) Tree++ code as well as working but unrepresentative MicroC+ code

Treepp Decl Branch

- Our most developed branch in terms of program structure and grammar
- We were ultimately unable to correct the LLVM basic block error for anything more advanced than the most basic expressions





Thank you!