

MAZE



Alexander Brown (aab2212) | Alexander Freemantle (asf2161)
Michelle Navarro (mn2614) | Lindsay Schiminske (ls3245)

CONTENTS

1. **Introduction**
 - Related Work
2. **Language Tutorial**
 - Environment Setup
 - The basics
 - Control Flow
 - Methods
3. **Language Reference Manual**
 - Lexical Conventions
 - Data Types
 - Expressions and Operators
 - Statements
 - Methods
 - Classes
4. **Project Plan**
 - Planning Process
 - Specification Process
 - Development Process
 - Testing Process
 - Team Roles
 - Project Log
5. **Language Evolution**
 - Original Idea
 - Actual Implementation
6. **Architecture**
 - The Compiler
 - The Scanner
 - The Parser
 - The Analyzer
 - The Code Generator
7. **Test Plan**
 - Testing Phases
 - Unit Testing
 - Integration Testing
 - Automation

Test Suites

8. **Conclusions**

Lessons Learned

9. **Code Listings**

scanner.mll

ast.ml

parser.mly

analyzer.ml

maze.ml

testall.sh

Test Suite Code

1. Introduction

MAZE is an imperative language with Java-like syntax. It compiles down to LLVM IR.

Related Work

For reference, we spent a great deal of time with the MicroC example explained to us in class as well as Dice, a project completed last year that was similar to the idea we had in mind for our language.

2. Language Tutorial

Setup

The language was developed in OCaml which needs to be installed in order to use the compiler. The best way to install it is by using OPAM, the OCaml Package Manager.

Once inside the maze directory, type "make" to compile the different components of the compiler. From there running "./test_estall.sh" will build the corresponding .ll files from the .maze files. You can run an individual test within the "tests" directory using lli on the .ll file of the test you are running.

```
ex. lli test-binop.ll
```

You can also compile individual files by passing the maze executable the -c flag and the name of the file to be compiled.

```
ex. ./maze -c program.maze
```

The Basics

All variables and methods must be contained in classes.

Types

All types are declared with their type followed by an identification. All variables of all types must be declared before they are initialized. The following types are supported:

- integers (int)
- floating point (float)
- booleans (bool)
- characters (char)
- strings (string)

```
class example {
    void main(){
        int a;
        a = 3;
        float b;
        b = 4.2;
        bool c;
        c = true;
        char d;
        d = 'd';
        string e;
        e = "eeee";
    }
}
```

Operators

Maze supports the following binary operators:

- Arithmetic (+, -, *, /)
- Relational (==, !=, >, >=, <, <=)
- Logical (&&, ||)

It also supports the following unary operators:

- Negation (!)
- Negative (-)

```

class example2 {
    void main(){
        int x;
        x = 3 + 2;  (* x is 5 *)
        x = 10 - 3; (* x is 7 *)
        x = 5 * 2;   (* x is 10 *)
        x = 20 / 5; (* x is 4 *)

        int a;
        a = 8;
        bool b;
        b = true;
        bool f;
        f = false;

        a==8; a!=5; a>2; a<=10; b||f; (*evaluate to
        true*)
        b&&f; !b; (*evaluate to false*)
    }
}

```

Control Flow

Control flow statements break up the flow of execution by employing decision making, looping, and branching, enabling the program to conditionally execute particular blocks of code. Maze supports if statements, while loops and return.

```

class example3{
    void main(){
        int y;
        if(true){
            y = 6;
        }
        else{
            y = 4;
        }
    }
}

```

```

    (* y is 6 *)

while(y==6){
    if(false){
        y = 9;
    }
    else if(false){
        y = 2;
    }
    else{
        y = 1;
    }
    (* y is 1, while loop loops once *)
}
}
}

```

Methods

Maze supports two types of methods, ones that execute and return a value and those that execute without returning anything. Methods can also accept arguments which are computed in applicative order. Recursion is allowed.

```

class example4{
    void foo(){
        int x;
        x = 1;
        print(x);
    }

    void main(){
        foo();
    }
}

```

3. Language Reference Manual

Lexical Conventions

Comments

Comments begin with `(*` and end with `*)`

Identifiers

An identifier is any sequence of characters that can be used as a name for a variable, method, or new data type. Valid identifier characters include ASCII letters, decimal digits, and underscores. The first character of an identifier cannot be a digit. Identifiers cannot be the same sequence of characters as keywords.

Keywords

The following identifiers are reserved and cannot be used otherwise. They are case sensitive:

<i>int</i>	<i>while</i>	<i>null</i>
<i>char</i>	<i>bool</i>	<i>true</i>
<i>return</i>	<i>print</i>	<i>false</i>
<i>if</i>	<i>else</i>	<i>string</i>
<i>void</i>	<i>class</i>	

Literals

MAZE literals can be integers, booleans, floats, characters, and strings.

ex. `"Hello"` is a literal string.

`42` is a literal integer.

Data Types

Integers

int

An integer is a whole value between -2^{31} and $2^{31} - 1$.
The default value is 0.

Boolean

bool

A single byte that can have the value true or false.
The default value is false.

Float

float

A float is an integer followed by a decimal and a fractional part. The default value is 0.0 .

Void

void

Use the void type to signify a function that has no return value.

Character

char

A character is a single byte. It holds the value of a character in the set of allowed characters. The default value is an empty char '' .

The following are escape-sequence characters

- '\\\' - backslash
- '\"' - double-quote
- '\'' - single-quote
- '\n' - newline
- '\r' - carriage return
- '\t' - tab character

String

string

Strings are a sequence of zero or more

characters, digits, spaces or other ASCII characters. The default value is the empty string "" .
ex. "This is a string"

Expressions and Operators

Expressions

An expression is composed of:

- A literal value
- A variable identifier
- A reference
- An arithmetic expression

An arithmetic expression consists of one or more operands and zero or more operators. The arithmetic expression may include parentheses for grouping.

Arithmetic expression examples:

Operand, 0 operators: 100;

Two operands, 1 operator: 100 + 101;

Use of Parentheses: (100*(102-101));

- A call to a method that returns some value or reference
- A call to a constructor to create an object

Operators

Assignment Operators

The assignment operators assign values from the right hand operand to the left side operand.

```
ex. int x = 4;
    int y = 3 + 7;
    int z = null;
```

Arithmetic Operators

The arithmetic operators include + (addition), - (subtraction), * (multiplication), / (division) and negation.

ex.

Addition:

```
int x = 6 + 2;
```

Subtraction:

```
int x = 6 - 2;
```

Multiplication:

```
int x = 6 * 2;
```

Division:

```
int x = 6 / 2;
```

Negation:

```
int x = -6;
```

Relational Operators

```
expression < expression  
expression > expression  
expression <= expression  
expression >= expression
```

The operators are < (less than), >(greater than), <= (less than or equal to) and >=(greater than or equal to). The relational operators group left to right.

Equality Operators

```
expression == expression  
expression != expression
```

The == (equal to) and != (not equal to) operators evaluate the expression to determine if the two expressions are equal or not equal.

Logical Operators

```
expression && expression  
expression || expression
```

The && (logical AND) returns true if both expressions are met and false otherwise. The || (logical OR) returns true if at least one expression is true and false if no expressions are met.

Statements

Expression Statements

Expression statements are in the form:

```
expression ;
```

Usually expression statements are assignments or function calls.

```
ex.  int value;  
      int value = 14;
```

The if Statement

The two forms of conditional statements are:

```
if (expression) statement
```

AND

```
if (expression) statement1 else statement2
```

The expression is evaluated in both cases and if it is true then the first statement is executed, if it evaluates to false statement2 is executed in the second case.

The while Statement

The while statement has the form:

```
while (expression) statement
```

The statement is executed repeatedly as long as the Expression evaluates to true.

Return Statement

The return statement has the form:

```
return (expression) ;
```

A return statement is not necessary when a method's return type is void.

Methods

Method basics

A method is a collection of statements that are grouped together to perform an operation. A method can return a value or nothing when called. The void keyword is used to create a method that returns no value. If the void keyword is not used when creating a method then the method must return a value.

```
(* Method with return value *)
int myMethod (int a, int b){
    (* body *)

    return (*int*)
}
```

```
(* Method with no return value *)
void myMethod (int a) {
    (* body *)
}
```

The name of the method can not be the same as the name of the class it is defined in. (See Ch.7 for Classes)

The Main Method

The main method must appear once and only once in the program. The main method is the first method to be called, and in turn calls all of the other methods required to run the program.

```
Sample main method:
class example {
    void main(){
        (* do something *)
    }
}
```

Classes

Class Definitions

A class acts as a container for fields and methods that should be grouped together.

```
class MyClass {
    (*field, constructor and method declarations*)
}
```

Class Fields

Fields are variables containing data belonging to that class type.

Standard defining of a class

```
class myClass {
    int myVar;
    myMethod (float myVar1, string myVar2){
        ...
    }
}
```

4. Project Plan

Planning Process

Throughout the project the group met a minimum of three times a week. As our version control system we used Git and had working code on the master branch at any given time. The deadlines set for the language reference manual and Hello World were useful markers to keep us on track, and have a good feel for what we would realistically be able to accomplish. Once the scanner, ast and parser were complete, we alternated work between the analyzer, codegen and the test suite. Although we all had our specific roles, we tackled this project as a group, working with each other on the different components. This

allowed continuous work, especially if one or more teammates reached a standstill.

Specification Process

We had initially wanted our language to be one that was specifically designed to build games. Once we talked this through with our TA, we realized that we were limiting ourselves too much and turned our language into an object-oriented language similar to Java. We were directed towards DICE from a previous semester and took inspiration from them. We were not able to implement all of the features that we had hoped, such as arrays and inheritance, but we are able to compile some relatively complex and interesting programs including the gcd program.

Development Process

We began with the scanner, ast, and parser as a group. For the most part, we kept the same system going for the remainder of the project. Typically we had one member on the analyzer, and two to three on the codegen and test suite. Instead of dividing work having one person write the test suite and another write codegen, we worked on them simultaneously. We would write a test for different areas of codegen that we were trying to implement. This method worked well for us throughout the semester.

Testing Process

Every time we wrote a new component of the language, we would write a test for it. This allowed us to work out the bugs immediately, one component at a time. By adding tests as we added new features, we were able to monitor that the addition of different components would not break pre-existing tests that were known to work.

Team Roles

Lindsay Schiminske - Manager

Michelle Navarro - Tester

Alexander Freemantle - Systems Architect
Alexander Brown - Language Guru

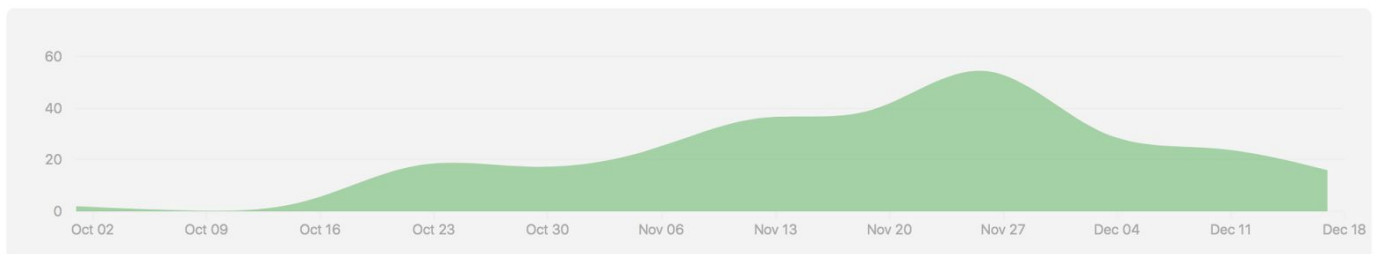
These roles were far from specific, as everybody contributed to the different areas of the project. The responsibilities were evenly distributed among all of the members.

Project Log

Oct 2, 2016 – Dec 19, 2016

Contributions: Commits ▾

Contributions to master, excluding merge commits



The image above shows our commit timeline. We noticed that our curve is shifted back. This shows that we worked diligently early on in the semester and had a big push in the last week of November. We anticipated that the end of the year would be busiest with the rest of our classes so we worked hard to make progress before the final weeks of the semester.

Git usernames

Lindsay Schiminske - Lindsay, Linkse36

Michelle Navarro - Michelle Navarro, michellenavarro

Alexander Freemantle - Alexander Freemantle, AFreemantle

Alexander Brown - Alex, Alexander Brown, abro9

Below is our git commit history.

```
1 commit 06e72f71f696d95405568247a87ec6c5c9f1b9c6
2 Author: AFreemantle <asf2161@columbia.edu>
3 Date: Mon Dec 19 13:06:17 2016 -0500
4
```


5 added recursive fib function test
6
7 commit d2c0cad321b249333acd8f9a78017bb53fc7c3cd
8 Author: AFree mantle <asf2161@columbia.edu>
9 Date: Mon Dec 19 12:22:22 2016 -0500
10
11 final update to README
12
13 commit eec85dfdc221e014fd358488a869c32abaa67eae
14 Author: Alexander Brown <aab2212@columbia.edu>
15 Date: Mon Dec 19 05:30:05 2016 -0500
16
17 final commit.
18
19 commit 411a5163717bac7bb996c8b026af89be92575d5e
20 Author: Alexander Brown <aab2212@columbia.edu>
21 Date: Mon Dec 19 05:02:50 2016 -0500
22
23 cruft n comments
24
25 commit 559a3e725a03197a32df590717ad0d900b2a0f9f
26 Merge: bce23f7 4d0ab45
27 Author: Michelle Navarro <mn2614@columbia.edu>
28 Date: Mon Dec 19 00:53:13 2016 -0800
29
30 Merge pull request #50 from AFree mantle/analyzer
31
32 forgot to commit this change last time
33
34 commit bce23f7a49bfe7b9200aae1123000580ee93fa00
35 Merge: 514c21b 52fb001
36 Author: Michelle Navarro <mn2614@columbia.edu>
37 Date: Mon Dec 19 00:48:56 2016 -0800
38
39 Merge pull request #49 from AFree mantle/analyzer
40
41 Analyzer
42

43 commit 4d0ab45a2183bfffbb85ff787df18e0b0a1ae4629
44 Author: Alexander Brown <aab2212@columbia.edu>
45 Date: Mon Dec 19 03:33:58 2016 -0500
46
47 forgot to commit this change last time
48
49 commit 52fb001d279b0814a7ea0865e91ae633b37402bf
50 Author: Alexander Brown <aab2212@columbia.edu>
51 Date: Mon Dec 19 03:28:05 2016 -0500
52
53 check for main actually works now
54
55 commit 60940d16816c1444f4f6756f2f0859a27200a646
56 Author: Alexander Brown <aab2212@columbia.edu>
57 Date: Mon Dec 19 02:47:59 2016 -0500
58
59 checks for exactly 1 main fcn
60
61 commit 7787a7cdbda21bfff214a44021e1ca14e7dfc01c7
62 Author: Alexander Brown <aab2212@columbia.edu>
63 Date: Mon Dec 19 02:10:09 2016 -0500
64
65 finished all type checking, tests to come
66
67 commit 4ae237f08e5e98c5a9a2e5fd7ceb66fbde0efe2d
68 Author: Alexander Brown <aab2212@columbia.edu>
69 Date: Sun Dec 18 22:24:33 2016 -0500
70
71 saving changes before branch switch
72
73 commit 05c6e8174a9c32f6aa7af3edc7d58a762631e51e
74 Author: Alexander Brown <aab2212@columbia.edu>
75 Date: Sun Dec 18 21:26:12 2016 -0500
76
77 now checks assignment exprs to ensure left side same type as
right side
78
79 commit 99fbf9a4d278e3543ca1aff9605de2a4edc292eb

80 Author: Alexander Brown <aab2212@columbia.edu>
81 Date: Sun Dec 18 21:20:24 2016 -0500
82
83 check assign fcn for comparing rt and lt
84
85 commit 0370ace8eb3f4fc001005623a24d6cfc3b13294c
86 Author: Alexander Brown <aab2212@columbia.edu>
87 Date: Sun Dec 18 21:16:27 2016 -0500
88
89 clarified some binop type checking
90
91 commit da074a1d659fa4e75df51cd2864f6c910946b87a
92 Author: Alexander Brown <aab2212@columbia.edu>
93 Date: Sun Dec 18 20:53:23 2016 -0500
94
95 test binop with diff operand types
96
97 commit 453441d42cc0806a71f0d295c6b5943e63aa1eda
98 Author: Alexander Brown <aab2212@columbia.edu>
99 Date: Sun Dec 18 20:19:01 2016 -0500
100
101 Better binop checking, error message
102
103 commit 9f68c92dc99f042f08e1146376bd51580912ed54
104 Author: Alexander Brown <aab2212@columbia.edu>
105 Date: Sun Dec 18 19:48:07 2016 -0500
106
107 undeclared ID test
108
109 commit aa4b06f2b1871abeea758d2cbec7815c1d483115
110 Author: Alexander Brown <aab2212@columbia.edu>
111 Date: Sun Dec 18 19:42:12 2016 -0500
112
113 fixed uop expr checking, passes all tests
114
115 commit 6b6b46f44fdf7f8993e5687e39f07b6998ae8f84
116 Author: Alexander Brown <aab2212@columbia.edu>
117 Date: Sun Dec 18 03:58:36 2016 -0500

118
119 small cruft
120
121 commit 6ed6f2a9d19bb0c3851564c1b038d1caf723982f
122 Author: Alexander Brown <aab2212@columbia.edu>
123 Date: Sun Dec 18 03:56:54 2016 -0500
124
125 checks almost everything, look at BINOP, UNOP and FIX
126
127 commit fbb61eb4156afb850b4cd5bc839f9c962df8ed80
128 Author: Alexander Brown <aab2212@columbia.edu>
129 Date: Sun Dec 18 00:06:47 2016 -0500
130
131 type of identifier fcn
132
133 commit 943552bb2783084f9cff68e1e8d91f5a5940b6cb
134 Author: Alexander Brown <aab2212@columbia.edu>
135 Date: Sun Dec 18 00:01:44 2016 -0500
136
137 Full symbol map per class, helper fcns
138
139 commit 7acd7e359e54467095e7af9f82affd6f67fc7599
140 Author: Alexander Brown <aab2212@columbia.edu>
141 Date: Sat Dec 17 23:53:42 2016 -0500
142
143 beginning symbol checking, helper fcns
144
145 commit 2a839594dc366c0ce4db175a0be967ef7203264d
146 Author: Alexander Brown <aab2212@columbia.edu>
147 Date: Sat Dec 17 22:26:46 2016 -0500
148
149 duplicate formal test
150
151 commit 317059f1300a12680d69a9815bd7b9db89b14ace
152 Author: Alexander Brown <aab2212@columbia.edu>
153 Date: Sat Dec 17 22:24:14 2016 -0500
154
155 checks for duplicate formals

156
157 commit 2a5fc2eea8259a9ea2dd8394bbf9c1510fd1f737
158 Author: Alexander Brown <aab2212@columbia.edu>
159 Date: Sat Dec 17 22:07:24 2016 -0500
160
161 void formal test
162
163 commit b44b6c06b36067b067a8a9e02866134b5458402b
164 Author: Alexander Brown <aab2212@columbia.edu>
165 Date: Sat Dec 17 22:01:35 2016 -0500
166
167 Checks for void formals
168
169 commit 69d565c710eed94fb09d333c3ef1598f797d3d47
170 Author: Alexander Brown <aab2212@columbia.edu>
171 Date: Sat Dec 17 19:37:37 2016 -0500
172
173 started checkin func formals
174
175 commit 86933358cf63654397a1acd4ad6200cc5c925788
176 Author: Alexander Brown <aab2212@columbia.edu>
177 Date: Sat Dec 17 19:20:24 2016 -0500
178
179 cruft
180
181 commit 44dc464d4807575b8e5120a272e40583f3912e1a
182 Author: Alexander Brown <aab2212@columbia.edu>
183 Date: Sat Dec 17 19:18:12 2016 -0500
184
185 print_fname, function_decl working
186
187 commit f14dc08d0866f2163d0ab605fd099e84892cfe71
188 Author: Alexander Brown <aab2212@columbia.edu>
189 Date: Sat Dec 17 19:03:08 2016 -0500
190
191 fixed function map stuff
192
193 commit 514c21b0d00851952a024fadbae7de1318571507

194 Author: Michelle Navarro <mn2614@columbia.edu>
195 Date: Sat Dec 17 18:49:45 2016 -0500
196
197 shift/reduce err fix
198
199 commit 5da31dcc0d8f196692e49946e251dac8d898ceb5
200 Merge: 2b2a6b7 f41580b
201 Author: Alexander Brown <aab2212@columbia.edu>
202 Date: Sat Dec 17 17:50:35 2016 -0500
203
204 Merge branch 'master' of <https://github.com/AFreemantle/maze>
into analyzer
205
206 commit f41580b813c91af3ad8239bbb992d9427dbd8255
207 Author: AFreemantle <asf2161@columbia.edu>
208 Date: Sat Dec 17 17:35:11 2016 -0500
209
210 fixed float printing test
211
212 commit 713b0a993e7098383ad172c1de79db9f79466737
213 Author: AFreemantle <asf2161@columbia.edu>
214 Date: Sat Dec 17 17:22:17 2016 -0500
215
216 removed extra set of brackets
217
218 commit 2d8900f9715897c6c00b420d5149bcc5eff8787f
219 Author: AFreemantle <asf2161@columbia.edu>
220 Date: Sat Dec 17 17:06:15 2016 -0500
221
222 floats print now!
223
224 commit 415e2a9cef2e176d7bd7a5aa0b8dda2738322103
225 Author: Lindsay <ls3245@columbia.edu>
226 Date: Fri Dec 16 01:57:25 2016 -0500
227
228 can create multiple classes
229
230 commit 3e091d1b4d9550bbcbaed8fb9ecb9bcb613fa25c

231 Author: Lindsay <ls3245@columbia.edu>
232 Date: Fri Dec 16 01:57:02 2016 -0500
233
234 added Object Create
235
236 commit 0215fe2e68657619257bdccab6e6eac5a7dcf384
237 Author: Lindsay <ls3245@columbia.edu>
238 Date: Thu Dec 15 20:52:16 2016 -0500
239
240 test for return works
241
242 commit 8df05b018277b21d15d239a6678160dc4d6c687b
243 Author: Lindsay <ls3245@columbia.edu>
244 Date: Thu Dec 15 20:51:04 2016 -0500
245
246 added objecttype, needed for multiple classes and new
247
248 commit 9ec0745fd525c5b9a489f5cef3d8b564e4f16286
249 Author: Lindsay <ls3245@columbia.edu>
250 Date: Thu Dec 15 19:02:45 2016 -0500
251
252 tested nested if stmts
253
254 commit 464c59d335e56dd95fc1928b820eb3b53713ab18
255 Author: Lindsay <ls3245@columbia.edu>
256 Date: Thu Dec 15 19:02:20 2016 -0500
257
258 tested empty if stmt
259
260 commit c318fd13043569d55329868666e4376c62840050
261 Author: Lindsay <ls3245@columbia.edu>
262 Date: Thu Dec 15 19:01:45 2016 -0500
263
264 multiple else if's
265
266 commit 2a26bb9581f464edab9eba2e5c40bfda41c91450
267 Author: AFreemantle <asf2161@columbia.edu>
268 Date: Thu Dec 15 18:01:57 2016 -0500

269
270 function calling working now too
271
272 commit 8e03c747a3683d0c10b3707cffb49c2668996bc7
273 Author: AFree mantle <asf2161@columbia.edu>
274 Date: Thu Dec 15 17:56:42 2016 -0500
275
276 removed gcd from top dir
277
278 commit 9e760b70f5d23fecb692ada9394a681274c785b5
279 Author: AFree mantle <asf2161@columbia.edu>
280 Date: Thu Dec 15 17:54:31 2016 -0500
281
282 gcd works
283
284 commit 5e43cb5918ade19e7460a41f7e789176448cc653
285 Author: Lindsay <ls3245@columbia.edu>
286 Date: Thu Dec 15 17:00:02 2016 -0500
287
288 added tests for float add, sub, mult, div
289
290 commit 0c085b2bd4d83520a96285691b5744e3aa182e6e
291 Author: Lindsay <ls3245@columbia.edu>
292 Date: Thu Dec 15 16:11:04 2016 -0500
293
294 tests addition of floats
295
296 commit 85f83f5c17f810c00500ade1c4884042c48d28a2
297 Author: Lindsay <ls3245@columbia.edu>
298 Date: Thu Dec 15 16:10:43 2016 -0500
299
300 added binop for floats
301
302 commit 8601efc1a4b77c7bdbfc6fe7150b302d33efb5a8
303 Author: Lindsay <ls3245@columbia.edu>
304 Date: Thu Dec 15 16:10:21 2016 -0500
305
306 added float to scanner

307
308 commit 71b31bda120efceb467501eaed90acfff30907c1
309 Author: Michelle Navarro <mn2614@columbia.edu>
310 Date: Thu Dec 8 02:44:42 2016 -0500
311
312 Giving Ids to print actually works now
313
314 commit 35bc4df4c4f561019c9591d6de10d052e556513f
315 Author: Michelle Navarro <mn2614@columbia.edu>
316 Date: Thu Dec 8 02:32:51 2016 -0500
317
318 figured out what it thinks the types are weeeeeeee
319
320 commit f9a95fe6e78aab2e3b093ae8d3d2f82090feed28
321 Author: Michelle Navarro <mn2614@columbia.edu>
322 Date: Thu Dec 8 00:56:45 2016 -0500
323
324 HAHahaha CLOSER TO THE SOLUTION
325
326 commit 0c6f919e1d9e8ecbf60a440e10d27e5c445338a0
327 Merge: f02cff0 17e617e
328 Author: Alexander Freemantle <asf2161@columbia.edu>
329 Date: Wed Dec 7 22:46:29 2016 -0500
330
331 Merge pull request #46 from AFreemantle/print
332
333 test for declaring then printing a bool
334
335 commit 17e617e6cb71c864eb0b3469cbe8bb0275ad6514
336 Author: AFreemantle <asf2161@columbia.edu>
337 Date: Wed Dec 7 22:42:24 2016 -0500
338
339 test for declaring then printing a bool
340
341 commit f02cff0f13970fbad7bf0d4e1de1735be1010466
342 Merge: bcc8595 60de00d
343 Author: Alexander Freemantle <asf2161@columbia.edu>
344 Date: Wed Dec 7 22:37:30 2016 -0500

345
346 Merge pull request #45 from AFree mantle/functions
347
348 added tests for assignment then print for int and char
349
350 commit 60de00dbad8a376f20731a2ac8b83ee61d44ce13
351 Author: AFree mantle <asf2161@columbia.edu>
352 Date: Wed Dec 7 22:37:07 2016 -0500
353
354 added tests for assignment then print for int and char
355
356 commit bcc85950085fc519532903243221d4b855bc89e7
357 Author: Michelle Navarro <mn2614@columbia.edu>
358 Date: Wed Dec 7 22:34:25 2016 -0500
359
360 cruft
361
362 commit 98a4e3ad9cf9fb064b13385cf58819fe7088efe2
363 Author: AFree mantle <asf2161@columbia.edu>
364 Date: Wed Dec 7 22:13:49 2016 -0500
365
366 test files for printing strings after assignment
367
368 commit 35359c697cebc7d9fcec8e5a68c1706822ab1ae5
369 Merge: f7d4c25 ffcd6d3
370 Author: Alexander Free mantle <asf2161@columbia.edu>
371 Date: Wed Dec 7 00:08:32 2016 -0500
372
373 Merge pull request #44 from AFree mantle/functions
374
375 missed one string args in test-binop
376
377 commit ffcd6d3e5be638ae2f8836a2cffa8627dab5ae77
378 Author: AFree mantle <asf2161@columbia.edu>
379 Date: Wed Dec 7 00:06:45 2016 -0500
380
381 missed one string args in test-binop
382

383 commit f7d4c259c3dae6ca3688a10dec741dff2acdcb5a
384 Merge: d17a44e 133e87b
385 Author: Michelle Navarro <mn2614@columbia.edu>
386 Date: Tue Dec 6 23:57:27 2016 -0500
387
388 :Merge branch 'master' of github.com:AFreemantle/maze
389
390 commit d17a44ef115dbf90dfc43196de03b4e0ee159906
391 Author: Michelle Navarro <mn2614@columbia.edu>
392 Date: Tue Dec 6 23:57:11 2016 -0500
393
394 took string useless string args out
395
396 commit 133e87bdcef4517d01af76bfd5ccd9d62eb9dc74
397 Merge: 01d5648 9efe3f2
398 Author: Alexander Freemantle <asf2161@columbia.edu>
399 Date: Tue Dec 6 23:50:16 2016 -0500
400
401 Merge pull request #43 from AFreemantle/print
402
403 cruft from failed string printing
404
405 commit 9efe3f2212e987810e3c3d20a78968626817ef19
406 Author: AFreemantle <asf2161@columbia.edu>
407 Date: Tue Dec 6 23:49:40 2016 -0500
408
409 cruft from failed string printing
410
411 commit 01d5648066cbded07e021b85aaa6a2149dab85ca
412 Merge: 9d50677 9fbb3d8
413 Author: Alexander Freemantle <asf2161@columbia.edu>
414 Date: Tue Dec 6 23:43:04 2016 -0500
415
416 Merge pull request #42 from AFreemantle/print
417
418 new test for assignment
419
420 commit 9fbb3d8ef318f6564768c7b07bd1873293c990ab

421 Author: AFree mantle <asf2161@columbia.edu>
422 Date: Tue Dec 6 23:42:12 2016 -0500
423
424 new test for assignment
425
426 commit 9d50677ce7feaa50abd091ef3309d7ad65d396bd
427 Merge: 4df4da3 c13d496
428 Author: Alexander Freemantle <asf2161@columbia.edu>
429 Date: Tue Dec 6 23:40:02 2016 -0500
430
431 Merge pull request #41 from AFree mantle/print
432
433 jokes, assignment was the issue, working now (strings)
434
435 commit c13d4968c78db4e06690e7f91be88b2c7112c4b2
436 Author: AFree mantle <asf2161@columbia.edu>
437 Date: Tue Dec 6 23:39:16 2016 -0500
438
439 jokes, assignment was the issue, working now (strings)
440
441 commit 4df4da342a76531579547cac0c1b07f584ce2dbd
442 Merge: 311d528 33e4f78
443 Author: Alexander Freemantle <asf2161@columbia.edu>
444 Date: Tue Dec 6 23:16:30 2016 -0500
445
446 Merge pull request #40 from AFree mantle/print
447
448 Print
449
450 commit 33e4f7834ccfd6f9ad3ff4e75f1def0e5a3ea443
451 Author: AFree mantle <asf2161@columbia.edu>
452 Date: Tue Dec 6 23:15:40 2016 -0500
453
454 finally got printing ids to work!
455
456 commit 290daae688a6b4858accdd055dd305569e1cc8ba
457 Author: AFree mantle <asf2161@columbia.edu>
458 Date: Tue Dec 6 22:58:23 2016 -0500

459
460 small test function for printing ids
461
462 commit 311d52812a2aeb2b6b6c63d36ec3b8472f72fcf1
463 Merge: 562097a ffd60b9
464 Author: Alexander Freemantle <asf2161@columbia.edu>
465 Date: Tue Dec 6 22:31:01 2016 -0500
466
467 Merge pull request #39 from AFreemantle/print
468
469 Print
470
471 commit ffd60b904213c5cc1d35e375248eb3539ce5ca16
472 Author: AFreemantle <asf2161@columbia.edu>
473 Date: Tue Dec 6 22:28:14 2016 -0500
474
475 while loops work, debugged gcd
476
477 commit 972bf217b956d4202195c5911891cb059d534424
478 Merge: dd23f4b 562097a
479 Author: AFreemantle <asf2161@columbia.edu>
480 Date: Tue Dec 6 22:05:33 2016 -0500
481
482 Merge branch 'master' of <https://github.com/AFreemantle/maze>
into print
483
484 commit 562097a9b46a0434dc02dbbcc6468ec2ec25f8da
485 Merge: 1b743bd 73affab
486 Author: Michelle Navarro <mn2614@columbia.edu>
487 Date: Tue Dec 6 22:05:21 2016 -0500
488
489 Merge pull request #38 from AFreemantle/gcd
490
491 Gcd
492
493 commit 2b2a6b74b164ae5f49802e6ba544162f4fa62f64
494 Author: Alexander Brown <aab2212@columbia.edu>
495 Date: Sun Dec 4 18:46:12 2016 -0500

496
497 idk
498
499 commit 73affab0087c44215ab975654b59dba906bdc1b6
500 Author: Michelle Navarro <mn2614@columbia.edu>
501 Date: Sun Dec 4 18:39:05 2016 -0500
502
503 Still wrong
504
505 commit 773c70af33e2c7c2742af9010ae16aa7589b530e
506 Author: Michelle Navarro <mn2614@columbia.edu>
507 Date: Sun Dec 4 17:21:01 2016 -0500
508
509 wrong
510
511 commit 20b16ec28ad0a6e7cca29205b544d85b98944941
512 Author: Michelle Navarro <mn2614@columbia.edu>
513 Date: Sun Dec 4 16:13:59 2016 -0500
514
515 Renamed test to gcd.maze
516
517 commit 1b743bdbfa7c2a41dd1edb7e13d220590a60e27b
518 Merge: d12690f 7d5c164
519 Author: Michelle Navarro <mn2614@columbia.edu>
520 Date: Sun Dec 4 16:11:36 2016 -0500
521
522 Merge branch 'master' of github.com:AFreemantle/maze
523
524 commit 7d5c16439262ade2e15cc849b43590d49192e27c
525 Merge: 15f08d3 f805ca4
526 Author: Michelle Navarro <mn2614@columbia.edu>
527 Date: Sun Dec 4 16:12:13 2016 -0500
528
529 Merge pull request #37 from AFreemantle/analyzer
530
531 Analyzer
532
533 commit d12690f096bf0102cfe1d510f7691935468ea30c

534 Author: Michelle Navarro <mn2614@columbia.edu>
535 Date: Sun Dec 4 16:11:24 2016 -0500
536
537 Fields test is actually gcd test now
538
539 commit 24910c67737a104796983d76127ea29ba5c22262
540 Merge: f805ca4 15f08d3
541 Author: Alexander Brown <aab2212@columbia.edu>
542 Date: Sun Dec 4 15:57:16 2016 -0500
543
544 Merge branch 'master' of <https://github.com/AFreemantle/maze>
into analyzer
545
546 commit f805ca4c701c5de9e4e3e9fb36203e3ad77800e5
547 Author: Alexander Brown <aab2212@columbia.edu>
548 Date: Sun Dec 4 15:55:29 2016 -0500
549
550 removes .ll files in /tests
551
552 commit 6dc9800a6b5760efeb3f2d77bb5d54cf055b4605
553 Author: Alexander Brown <aab2212@columbia.edu>
554 Date: Sun Dec 4 15:40:21 2016 -0500
555
556 cruft
557
558 commit dd23f4b4bac549dd4d77c5375c6624e4ad29af91
559 Author: AFreemantle <asf2161@columbia.edu>
560 Date: Sun Dec 4 14:39:33 2016 -0500
561
562 subtraction tests working now
563
564 commit 15f08d3a54e9d5a1beb4a283e18478dd66edd3c0
565 Merge: 16b1496 9538ce6
566 Author: Michelle Navarro <mn2614@columbia.edu>
567 Date: Sat Dec 3 23:39:22 2016 -0500
568
569 Merge pull request #36 from AFreemantle/analyzer
570

571 Analyzer
572
573 commit 9538ce6654cf2a6693414db03bd6c346bc7f2cc0
574 Merge: e7215f6 16b1496
575 Author: Alexander Brown <aab2212@columbia.edu>
576 Date: Sat Dec 3 23:37:01 2016 -0500
577
578 analyzer2 -> analyzer
579
580 commit e7215f644f99e1a3c49ade71dd80106f3ed5b1ff
581 Author: Alexander Brown <aab2212@columbia.edu>
582 Date: Sat Dec 3 23:32:25 2016 -0500
583
584 this will allow me to pull
585
586 commit 16b1496096427e083cf67254a47e28d2b4c637d5
587 Author: Michelle Navarro <mn2614@columbia.edu>
588 Date: Sat Dec 3 23:28:58 2016 -0500
589
590 Renamed analyzer2 to just analyzer
591
592 commit 3705743823b095865fc92fcd74f111e53ef37b
593 Author: Michelle Navarro <mn2614@columbia.edu>
594 Date: Sat Dec 3 23:23:59 2016 -0500
595
596 got rid of the other analyzer
597
598 commit 691718ca693e80a618da62bd02926aca1fbf7838
599 Merge: 226a581 7c54253
600 Author: Alexander Brown <aab2212@columbia.edu>
601 Date: Sat Dec 3 23:15:51 2016 -0500
602
603 Merge branch 'master' of <https://github.com/AFreemantle/maze>
into analyzer
604
605 commit 226a5811491c8cc32755fde6457f526b87d7901f
606 Author: Alexander Brown <aab2212@columbia.edu>
607 Date: Sat Dec 3 23:13:22 2016 -0500

608
609 duplicate function test
610
611 commit 7c542531aa490b00616aa531981aled673f9e041
612 Merge: c58e56e e52dd6c
613 Author: Michelle Navarro <mn2614@columbia.edu>
614 Date: Sat Dec 3 23:10:32 2016 -0500
615
616 Merge pull request #35 from AFree mantle/exc
617
618 Exc
619
620 commit e52dd6c9489e474a98231c4ea3f51cc994ec3f7e
621 Author: Michelle Navarro <mn2614@columbia.edu>
622 Date: Sat Dec 3 23:09:42 2016 -0500
623
624 small && addition
625
626 commit 42a49d73934bfafc07a7aba1b06e9ecca312f188
627 Author: Michelle Navarro <mn2614@columbia.edu>
628 Date: Sat Dec 3 23:05:20 2016 -0500
629
630 cruft
631
632 commit 0e6e20cb58313945485c379ec3626d3020b98a04
633 Author: Alexander Brown <aab2212@columbia.edu>
634 Date: Sat Dec 3 22:59:14 2016 -0500
635
636 test for void class var
637
638 commit 3e1380f15885bb634baclaa656bf75ff4d3a985f
639 Author: Alexander Brown <aab2212@columbia.edu>
640 Date: Sat Dec 3 22:51:58 2016 -0500
641
642 test for duplicate class variables
643
644 commit 098b15c8b67c542522441d2d48fb1c15fc56f131
645 Author: Alexander Brown <aab2212@columbia.edu>

646 Date: Sat Dec 3 22:47:11 2016 -0500
647
648 Checks for duplicate/void class variables
649
650 commit 1593a899c66a12bbfa8a0b3692a912647106c850
651 Author: Michelle Navarro <mn2614@columbia.edu>
652 Date: Sat Dec 3 22:44:18 2016 -0500
653
654 Fixing more tests & bash script stuff
655
656 commit efea7a430cef1cb480ed91c9ff41b264434828ed
657 Author: Alexander Brown <aab2212@columbia.edu>
658 Date: Sat Dec 3 22:29:45 2016 -0500
659
660 Test for defining print
661
662 commit c58e56e877fdf0f4f6197037707674c93378791c
663 Merge: d96b814 00c9f5e
664 Author: Michelle Navarro <mn2614@columbia.edu>
665 Date: Sat Dec 3 22:18:05 2016 -0500
666
667 Merge pull request #34 from AFree mantle/analyzer
668
669 Analyzer
670
671 commit 00c9f5e651559acf2edb2cbbb470b3072ac37db3
672 Author: Alexander Brown <aab2212@columbia.edu>
673 Date: Sat Dec 3 22:17:27 2016 -0500
674
675 Duplicate local variables test
676
677 commit 40dfa8a250cef68e55958faf1d2d54d3d0cc0f03
678 Author: Michelle Navarro <mn2614@columbia.edu>
679 Date: Sat Dec 3 21:35:25 2016 -0500
680
681 removed extra semant.ml file
682
683 commit bf1267c117fa13e220eb4225f4f89eb832838f40

684 Author: Michelle Navarro <mn2614@columbia.edu>
685 Date: Sat Dec 3 21:32:58 2016 -0500
686
687 Fixed script for failures
688
689 commit 71a5751bb63a70efbbd8c019d501067c73ae1972
690 Author: Alexander Brown <aab2212@columbia.edu>
691 Date: Sat Dec 3 21:29:35 2016 -0500
692
693 Duplicate class name test
694
695 commit be59bb2a99b2959e5d1e2e0008678b394244efdc
696 Author: Alexander Brown <aab2212@columbia.edu>
697 Date: Sat Dec 3 21:24:22 2016 -0500
698
699 Fixed stuff I accidentally broke
700
701 commit 117f0968a2c939030f0733ff3612528dfb175f3b
702 Merge: 34e39b4 d96b814
703 Author: Alexander Brown <aab2212@columbia.edu>
704 Date: Sat Dec 3 20:55:21 2016 -0500
705
706 Fixed merge conflicts
707
708 commit 34e39b408c825a8fdb5cdb61a77975672b7f67ee
709 Author: Alexander Brown <aab2212@columbia.edu>
710 Date: Sat Dec 3 20:52:42 2016 -0500
711
712 Checks for duplicate classes
713
714 commit 8746d48dcb516b2246640f715af6712919d7239d
715 Author: Alexander Brown <aab2212@columbia.edu>
716 Date: Sat Dec 3 20:48:09 2016 -0500
717
718 Void locals fail test
719
720 commit d96b814fb71ddf96de20299cb9e7b9d2e92144a2
721 Author: Michelle Navarro <mn2614@columbia.edu>

722 Date: Sat Dec 3 20:46:23 2016 -0500
723
724 going to start dealing with fails
725
726 commit b5ce4ca0a001db73795e8b15972d804fa4c37813
727 Author: Alexander Brown <aab2212@columbia.edu>
728 Date: Sat Dec 3 20:40:31 2016 -0500
729
730 Void locals analyzer test
731
732 commit 06c66da5ae75032f8cc0aa606f4ab3e11f5141c2
733 Author: Michelle Navarro <mn2614@columbia.edu>
734 Date: Sat Dec 3 20:28:15 2016 -0500
735
736 found stray print stmt
737
738 commit 04a8c353a43699c5279683efdfd23d938aeff6c8
739 Merge: 18e047b f820347
740 Author: Alexander Brown <aab2212@columbia.edu>
741 Date: Sat Dec 3 19:52:19 2016 -0500
742
743 Merge branch 'master' of <https://github.com/AFreemantle/maze>
into analyzer
744
745 commit f8203472148294e80b5b7f0d62acc7a3967dd5d9
746 Merge: 49790c9 340b752
747 Author: Michelle Navarro <mn2614@columbia.edu>
748 Date: Sat Dec 3 19:49:16 2016 -0500
749
750 Merge pull request #33 from AFreemantle/contrflow
751
752 Contrflow
753
754 commit 340b752b7d19c08fb29232e584dd83c0947d0ab1
755 Author: Lindsay <ls3245@columbia.edu>
756 Date: Sat Dec 3 19:45:09 2016 -0500
757
758 tests AND and OR

759
760 commit 01fef156e7dd8d8a45b948199f6a72d59f0948aa
761 Author: Lindsay <ls3245@columbia.edu>
762 Date: Sat Dec 3 19:44:46 2016 -0500
763
764 tests neg and not
765
766 commit 49790c9cd324997497aad96825cca9ba28d3da86
767 Author: Lindsay <ls3245@columbia.edu>
768 Date: Sat Dec 3 19:32:12 2016 -0500
769
770 tested not unop
771
772 commit 5e24c9fb172f3185642bdab736f6ec5ade97d8bc
773 Merge: bfed1b2 37f306d
774 Author: Alexander Freemantle <asf2161@columbia.edu>
775 Date: Sat Dec 3 19:01:43 2016 -0500
776
777 Merge pull request #32 from AFreemantle/print
778
779 Print
780
781 commit 37f306d0058ee6bdce4e36e4362dc03a2ac2fb34
782 Author: AFreemantle <asf2161@columbia.edu>
783 Date: Sat Dec 3 18:34:18 2016 -0500
784
785 printing booleans correctly now
786
787 commit 075b3d576d73967248798f018b61c09baf4009dd
788 Author: AFreemantle <asf2161@columbia.edu>
789 Date: Sat Dec 3 17:51:53 2016 -0500
790
791 added tests for printing multiplication of two ints
792
793 commit 27c4e4c70e012f036a19b30169fadb37c663ea57
794 Author: AFreemantle <asf2161@columbia.edu>
795 Date: Sat Dec 3 17:47:54 2016 -0500
796

797 Binops printing correctly now
798
799 commit 34c40c3f1f4f9b8db6ed2889d8952e65ca660dd6
800 Author: AFree mantle <asf2161@columbia.edu>
801 Date: Sat Dec 3 17:41:07 2016 -0500
802
803 trying to implement Binop printing
804
805 commit 6cbffa4669986eb17eb9db9b6b09297e5ff0758d
806 Author: Lindsay <ls3245@columbia.edu>
807 Date: Fri Dec 2 22:11:44 2016 -0500
808
809 tests add, sub, mult, and div
810
811 commit 20330b3dae24bf80ed8f9884eac6940a4abe4409
812 Author: Lindsay <ls3245@columbia.edu>
813 Date: Fri Dec 2 22:01:46 2016 -0500
814
815 while loop works
816
817 commit de812735b1c78d0440c3d09e5d96df3b551e502c
818 Author: Lindsay <ls3245@columbia.edu>
819 Date: Fri Dec 2 22:01:24 2016 -0500
820
821 addition works
822
823 commit 65fc54f9daad9aeaa99a7b7dba52deaae53285a7
824 Author: Lindsay <ls3245@columbia.edu>
825 Date: Fri Dec 2 20:59:33 2016 -0500
826
827 added equality tests, lt, gt, leq, geq, eq, neq
828
829 commit 255ddc7f2ee9227737a7c1df3bfffab0a9d4ff017
830 Author: Lindsay <ls3245@columbia.edu>
831 Date: Fri Dec 2 20:58:31 2016 -0500
832
833 still doesnt work
834

835 commit 0385b3c8bfc7a522bda3eb572cb765d8816dc64d
836 Merge: 3645e3b bd9cce7
837 Author: Michelle Navarro <mn2614@columbia.edu>
838 Date: Thu Dec 1 19:19:00 2016 -0500
839
840 Merge pull request #31 from AFree mantle/binop
841
842 added binop expr handling straight from microc
843
844 commit 3645e3b380ca753931ca2d6cfad04a97fe6d9571
845 Author: Lindsay <ls3245@columbia.edu>
846 Date: Thu Dec 1 19:12:48 2016 -0500
847
848 tests
849
850 commit 738af72a7b019063be819ee62f0742aaa8c9f071
851 Author: Lindsay <ls3245@columbia.edu>
852 Date: Thu Dec 1 19:11:07 2016 -0500
853
854 added float/char printing
855
856 commit bfed1b2c4b1736b9dd78054480b0715e86e322e8
857 Merge: 899a9d4 9efdc dc
858 Author: Michelle Navarro <mn2614@columbia.edu>
859 Date: Thu Dec 1 00:06:39 2016 -0500
860
861 Merge pull request #28 from AFree mantle/inttest
862
863 Added print int test
864
865 commit 9efdc dc9ce04082c0eaa2e3fcb10216395c1064b
866 Author: Michelle Navarro <mn2614@columbia.edu>
867 Date: Wed Nov 30 23:56:00 2016 -0500
868
869 Added print int test
870
871 commit bd9cce7d225d2503ea2e4b79f9bd2636ac710069
872 Author: AFree mantle <asf2161@columbia.edu>

873 Date: Wed Nov 30 23:43:59 2016 -0500
874
875 added binop expr handling straight from microc
876
877 commit 899a9d441d5b9393abdf6b9bdee9bad29d6d6eb2
878 Author: AFree mantle <asf2161@columbia.edu>
879 Date: Wed Nov 30 23:30:16 2016 -0500
880
881 eliminated first warning
882
883 commit 1fb8559d6830955ccfb64936f0295e4a5bd1e1f0
884 Author: Lindsay <ls3245@columbia.edu>
885 Date: Wed Nov 30 23:26:53 2016 -0500
886
887 added Float,Char,Null pattern matching
888
889 commit a0a61789d104f8d18efca3385d22441537574be6
890 Merge: 6e4e4af dc1bce2
891 Author: Michelle Navarro <mn2614@columbia.edu>
892 Date: Wed Nov 30 22:34:32 2016 -0500
893
894 Merge pull request #27 from AFree mantle/print
895
896 printing ints and strings now
897
898 commit dc1bce2ab43cd900dea5b6339ea1d5ba39d63d57
899 Author: AFree mantle <asf2161@columbia.edu>
900 Date: Wed Nov 30 22:30:37 2016 -0500
901
902 printing ints and strings now
903
904 commit 6e4e4afcb0a1b2b4294425c60c847189281f6de7
905 Author: Michelle Navarro <mn2614@columbia.edu>
906 Date: Wed Nov 30 22:06:06 2016 -0500
907
908 starting excpetions
909
910 commit d112208c59d56527946b6f8bf071fd6c876a3d77

911 Author: Alexander Brown <aab2212@columbia.edu>
912 Date: Wed Nov 30 21:52:31 2016 -0500
913
914 raises invalid arg string error
915
916 commit 8477f21c957293032794b6ba8292b096cb55f305
917 Author: Michelle Navarro <mn2614@columbia.edu>
918 Date: Wed Nov 30 20:30:06 2016 -0500
919
920 makefile removes .ll files in tests dir now
921
922 commit fa7fca0cf4fe5e664cd69506d1e0767eaf99874f
923 Merge: 7bded1b 3b5da29
924 Author: Michelle Navarro <mn2614@columbia.edu>
925 Date: Wed Nov 30 20:17:45 2016 -0500
926
927 Merge pull request #26 from AFree mantle/cleantests
928
929 Cleaned test output
930
931 commit 3b5da290feb4d1f43187b1bdd102ddbebe90a6d6
932 Author: Michelle Navarro <mn2614@columbia.edu>
933 Date: Wed Nov 30 20:16:23 2016 -0500
934
935 Cleaned test output
936
937 commit 7bded1b1ca3326b7ead34bb6129b4ffa35d93fdc
938 Author: Michelle Navarro <mn2614@columbia.edu>
939 Date: Wed Nov 30 16:22:14 2016 -0500
940
941 Passes Hello World but output is ugly
942
943 commit 06db327c3d5b06ba230134c17833e2c38ed2f1db
944 Merge: 8c1314a 3254f5b
945 Author: Michelle Navarro <mn2614@columbia.edu>
946 Date: Wed Nov 30 02:57:38 2016 -0500
947
948 Merge pull request #25 from AFree mantle/codegen

949
950 Codegen
951
952 commit 3254f5b4456c399738a99b9ce4e2a815a0ce52b4
953 Author: Michelle Navarro <mn2614@columbia.edu>
954 Date: Wed Nov 30 02:56:23 2016 -0500
955
956 cruft
957
958 commit f3b9370c29aef2efa1a7717acb45cee115f9cbef
959 Author: Michelle Navarro <mn2614@columbia.edu>
960 Date: Wed Nov 30 02:43:29 2016 -0500
961
962 updated ext
963
964 commit 96a51e85254a1300ab7e74ebf6958cc48ea079df
965 Merge: 20340a3 4d64af6
966 Author: Michelle Navarro <mn2614@columbia.edu>
967 Date: Wed Nov 30 02:27:15 2016 -0500
968
969 Merge pull request #24 from AFree mantle/ct
970
971 Ct
972
973 commit 4d64af6d4dbfb3e21e2a6a24400261bf844da9af
974 Merge: 27e4cb5 20340a3
975 Author: Michelle Navarro <mn2614@columbia.edu>
976 Date: Wed Nov 30 02:27:03 2016 -0500
977
978 Merge Conflicts fixed
979
980 commit 27e4cb56901d9e9a0baebb46289c187cb551e78e
981 Author: Michelle Navarro <mn2614@columbia.edu>
982 Date: Wed Nov 30 02:24:53 2016 -0500
983
984 Fixing test bash script
985
986 commit 20340a349f8fec5e07368061b627f66b43643826

987 Author: Alexander Brown <aab2212@columbia.edu>
988 Date: Wed Nov 30 00:29:33 2016 -0500
989
990 misc improvements
991
992 commit 9c24d277e0d4eae6d1ec086c50b66d9f8566b1de
993 Author: AFree mantle <asf2161@columbia.edu>
994 Date: Tue Nov 29 23:24:47 2016 -0500
995
996 added .ll files to clean target
997
998 commit 3047f5795611e88ddf047552bc6a4d938275bf2d
999 Author: Alexander Brown <aab2212@columbia.edu>
1000 Date: Tue Nov 29 16:46:43 2016 -0500
1001
1002 Added string_lit in builder, modified format str
1003
1004 commit 8c1314a9d663d7dda36a175ec62761e940440101
1005 Merge: 0bf56b1 ba4fa17
1006 Author: Michelle Navarro <mn2614@columbia.edu>
1007 Date: Tue Nov 29 15:40:29 2016 -0500
1008
1009 Merge pull request #23 from AFree mantle/codegen
1010
1011 cruft
1012
1013 commit 0bf56b1f98a39d2bdbc3a33ed9ea01693621a0d9
1014 Merge: 9e4d7f4 38a4694
1015 Author: Michelle Navarro <mn2614@columbia.edu>
1016 Date: Tue Nov 29 15:38:22 2016 -0500
1017
1018 Merge pull request #20 from AFree mantle/codegen
1019
1020 Codegen
1021
1022 commit ba4fa177556b677e8982979be57278cfb1fd3913
1023 Author: Alexander Brown <aab2212@columbia.edu>
1024 Date: Tue Nov 29 15:37:37 2016 -0500

1025
1026 cruft
1027
1028 commit 38a4694af643e5b4cc444dfd30b9f6d0eed6f7e5
1029 Author: Alexander Brown <aab2212@columbia.edu>
1030 Date: Tue Nov 29 15:35:04 2016 -0500
1031
1032 Fixed small print decl bug
1033
1034 commit 7e9463c32632323f937084b5411151d5b3e82586
1035 Author: Alexander Brown <aab2212@columbia.edu>
1036 Date: Tue Nov 29 15:33:12 2016 -0500
1037
1038 -c flag now writes bytecode to a.bc
1039
1040 commit b0d667ca3be31851541df8857f0a6bc08cc43617
1041 Author: Alexander Brown <aab2212@columbia.edu>
1042 Date: Tue Nov 29 04:25:18 2016 -0500
1043
1044 closer to working now...
1045
1046 commit 5022781df00f09563f6a2faa85ae5be10b8e4a53
1047 Author: Alexander Brown <aab2212@columbia.edu>
1048 Date: Tue Nov 29 01:49:31 2016 -0500
1049
1050 fixed errors, trying to loop thru classes
1051
1052 commit cc3d3ce3d6d1cca50301f16f291633876e281a41
1053 Author: Alexander Brown <aab2212@columbia.edu>
1054 Date: Tue Nov 29 01:37:14 2016 -0500
1055
1056 codegen wrapper fcns
1057
1058 commit 279b2268262b3774bc9d8051fc2826458937607f
1059 Author: Michelle Navarro <mn2614@columbia.edu>
1060 Date: Tue Nov 29 00:50:23 2016 -0500
1061
1062 ast err fic

1063
1064 commit b7430f8a05b22ca6dc5ff975a0a99b7bd0128e9a
1065 Author: Alexander Brown <aab2212@columbia.edu>
1066 Date: Tue Nov 29 01:06:40 2016 -0500
1067
1068 Fixed codegen compile errors
1069
1070 commit bc9a2c4808dc11d7cf143035a07999b18e869b09
1071 Author: Alexander Brown <aab2212@columbia.edu>
1072 Date: Tue Nov 29 00:54:21 2016 -0500
1073
1074 locals -> typeKey fcn
1075
1076 commit d31454509b21f4afdaebbf2a24a1e18ba6e4693
1077 Author: Alexander Brown <aab2212@columbia.edu>
1078 Date: Tue Nov 29 00:41:49 2016 -0500
1079
1080 fixed what I thought I'd solved
1081
1082 commit 2bf388da5ed4d98b021e36ae7951ce7aa6716986
1083 Author: Alexander Brown <aab2212@columbia.edu>
1084 Date: Tue Nov 29 00:37:47 2016 -0500
1085
1086 removed 'Many' from formal, formal to map fcn
1087
1088 commit 6a964de8c34eb7cfb6c2ffe68a3ddf81a2f2728f
1089 Author: Alexander Brown <aab2212@columbia.edu>
1090 Date: Tue Nov 29 00:28:09 2016 -0500
1091
1092 string_of_FName and convert fname to string for 'name'
1093
1094 commit a46b10bfe3a7a1a6a314667b0afdbecef80872bc
1095 Author: Alexander Brown <aab2212@columbia.edu>
1096 Date: Tue Nov 29 00:21:00 2016 -0500
1097
1098 null and pattern matching warnings fixed
1099
1100 commit 9c5847bcb8c5b5af74aebfc606abbd43afdc9b95

1101 Author: Alexander Brown <aab2212@columbia.edu>
1102 Date: Tue Nov 29 00:09:50 2016 -0500
1103
1104 in
1105
1106 commit 93e83750af29c0e8bb79f724b83c05d3f4568a91
1107 Author: Alexander Brown <aab2212@columbia.edu>
1108 Date: Tue Nov 29 00:07:18 2016 -0500
1109
1110 ltype_of_formal function
1111
1112 commit 43b19dcb786951a9d1d29e4b660a3d94cbcd967
1113 Author: Alexander Brown <aab2212@columbia.edu>
1114 Date: Mon Nov 28 23:40:07 2016 -0500
1115
1116 codegen helper functions
1117
1118 commit d6e3c4082a2b9d96667fd8061de22123253232fb
1119 Author: Michelle Navarro <mn2614@columbia.edu>
1120 Date: Mon Nov 28 23:02:52 2016 -0500
1121
1122 MAkefile llvm
1123
1124 commit 5ef138a735a1d4315d13e64463156b72ac9266c2
1125 Merge: 383afff 9e4d7f4
1126 Author: Michelle Navarro <mn2614@columbia.edu>
1127 Date: Mon Nov 28 22:49:49 2016 -0500
1128
1129 Fixed merge conf
1130
1131 commit 383afff0462478d376b0505d4f1cbccf95607ddd
1132 Author: AFree mantle <asf2161@columbia.edu>
1133 Date: Sat Nov 26 16:50:11 2016 -0500
1134
1135 fixing some naming inconsistencies b/w ast and codegen
1136
1137 commit b3854d08db43eb34f3f374fa8d38b755909babcf
1138 Author: AFree mantle <asf2161@columbia.edu>

1139 Date: Sat Nov 26 14:03:47 2016 -0500
1140
1141 cruft
1142
1143 commit 9e4d7f4f22ad0d8a5b09bf28442b31c4724bb33f
1144 Author: Michelle Navarro <mn2614@columbia.edu>
1145 Date: Fri Nov 25 04:29:54 2016 -0500
1146
1147 Added one more test that def going to fail for now
1148
1149 commit 18e047bd71b021c95e5290fae59544b09d9be6bb
1150 Author: Alexander Brown <aab2212@columbia.edu>
1151 Date: Sun Nov 20 23:05:55 2016 -0500
1152
1153 moved some stuff around
1154
1155 commit e89dd4416d6e8b956a8847f134718750e217fa69
1156 Author: Alexander Brown <aab2212@columbia.edu>
1157 Date: Sun Nov 20 21:52:22 2016 -0500
1158
1159 cruft
1160
1161 commit e79befaf0811d8eab59b2f260d3c3bc0380154fb
1162 Author: Alexander Brown <aab2212@columbia.edu>
1163 Date: Sun Nov 20 21:50:29 2016 -0500
1164
1165 added built in print to flist
1166
1167 commit 02438170dc235e1dab6bc5a46bf483a5e94a547d
1168 Author: Alexander Brown <aab2212@columbia.edu>
1169 Date: Sun Nov 20 20:39:06 2016 -0500
1170
1171 checks for print fcn
1172
1173 commit 5799c0e343159cba55dba24deb40618a34dc9bc1
1174 Merge: 25eeb49 b7c7a28
1175 Author: Michelle Navarro <mn2614@columbia.edu>
1176 Date: Sun Nov 20 20:30:18 2016 -0500

1177
1178 Merge pull request #19 from AFree mantle/tests
1179
1180 Test Suite
1181
1182 commit b7c7a283335a71d0325313ce418ea3c54a41635d
1183 Author: Michelle Navarro <mn2614@columbia.edu>
1184 Date: Sun Nov 20 20:12:21 2016 -0500
1185
1186 Added test files
1187
1188 commit f249244e1964d9795b3bc6e07c289df559ae8360
1189 Author: Michelle Navarro <mn2614@columbia.edu>
1190 Date: Sun Nov 20 19:16:13 2016 -0500
1191
1192 Got rid of sast file
1193
1194 commit 36c877a08e34acdb00b114ec7ead2c8a552aa211
1195 Author: Michelle Navarro <mn2614@columbia.edu>
1196 Date: Sun Nov 20 19:07:35 2016 -0500
1197
1198 Tests continue
1199
1200 commit cac1f679a4eeb57b3909b3a0533cc418fd2617e8
1201 Author: Alexander Brown <aab2212@columbia.edu>
1202 Date: Sun Nov 20 18:40:21 2016 -0500
1203
1204 Organized, commented variable verifier
1205
1206 commit 5a08843b0d4d7b87e9ec9950d56315b8cac8d066
1207 Author: Michelle Navarro <mn2614@columbia.edu>
1208 Date: Sun Nov 20 18:22:31 2016 -0500
1209
1210 putting things where they belong
1211
1212 commit 5128a633e7770e7e778881a2530c115a06ffe121
1213 Author: Michelle Navarro <mn2614@columbia.edu>
1214 Date: Sun Nov 20 15:33:17 2016 -0500

1215
1216 moving tests
1217
1218 commit 339a2831baa471a57d91b83f1663854f33a07ce5
1219 Author: Michelle Navarro <mn2614@columbia.edu>
1220 Date: Sun Nov 20 15:18:31 2016 -0500
1221
1222 bash testing begins
1223
1224 commit 25eeb49404328df0238c02a541bd402e70e7dfab
1225 Merge: b662e21 d362dbb
1226 Author: Michelle Navarro <mn2614@columbia.edu>
1227 Date: Sun Nov 20 18:14:33 2016 -0500
1228
1229 Merge pull request #22 from AFree mantle/analyzer
1230
1231 Analyzer
1232
1233 commit d362dbb070090b7699ff6431450e4b6606b71086
1234 Author: Alexander Brown <aab2212@columbia.edu>
1235 Date: Sun Nov 20 18:10:59 2016 -0500
1236
1237 cruft for real
1238
1239 commit 8d8c9e05cd488f637585ae3c303adc5c6fda38e8
1240 Author: Alexander Brown <aab2212@columbia.edu>
1241 Date: Sun Nov 20 18:04:39 2016 -0500
1242
1243 now breaks on void variables
1244
1245 commit b662e216f7f7e2c770a2a777a0ad2fb248776070
1246 Merge: 1740550 6e0304b
1247 Author: abro9 <aab2212@columbia.edu>
1248 Date: Sun Nov 20 17:07:15 2016 -0500
1249
1250 Merge pull request #21 from AFree mantle/analyzer
1251
1252 main addition: check for duplicate variables

1253
1254 commit 65f10f1066af2ac9e36ff30d5ad3f4929bf7c240
1255 Author: AFree mantle <asf2161@columbia.edu>
1256 Date: Sun Nov 20 17:00:16 2016 -0500
1257
1258 added codegen to build statements
1259
1260 commit 6e0304b8991a6aaccd204bc4ff782dda63aafc8b
1261 Author: Alexander Brown <aab2212@columbia.edu>
1262 Date: Sun Nov 20 16:59:12 2016 -0500
1263
1264 cleaned up, still doesn't catch void
1265
1266 commit 29a4edbd535265cbdb91534d88635e613d58ac3c
1267 Author: Alexander Brown <aab2212@columbia.edu>
1268 Date: Sun Nov 20 16:48:22 2016 -0500
1269
1270 yells at duplicate variables
1271
1272 commit e7ae619c670855f2328c8f7a3f7792bb396d8020
1273 Author: AFree mantle <asf2161@columbia.edu>
1274 Date: Sun Nov 20 16:48:15 2016 -0500
1275
1276 added more skeleton, will call Analyzer soon
1277
1278 commit 79fdc68cc971a3c7f03252bf0cebfd9ecfc373b8
1279 Author: AFree mantle <asf2161@columbia.edu>
1280 Date: Sun Nov 20 16:27:27 2016 -0500
1281
1282 added framework for Compile and LLVM_IR options
1283
1284 commit 27a405261aeb1757dfe15065c0ba36b875974593
1285 Author: AFree mantle <asf2161@columbia.edu>
1286 Date: Sun Nov 20 16:13:01 2016 -0500
1287
1288 added add_terminal helper from MicroC
1289
1290 commit ca0634acfa578fb9eb2c17efca01945c6ab74cdd

1291 Author: AFree mantle <asf2161@columbia.edu>
1292 Date: Sun Nov 20 15:44:01 2016 -0500
1293
1294 started to define statements
1295
1296 commit 14412f6458f1f85d8b7aff71cc1293b1451c191f
1297 Author: Alexander Brown <aab2212@columbia.edu>
1298 Date: Sun Nov 20 15:38:29 2016 -0500
1299
1300 Now runs thru function variables
1301
1302 commit fb46aa5f309a85736de395f1ecee978c6406375a
1303 Author: AFree mantle <asf2161@columbia.edu>
1304 Date: Sun Nov 20 15:34:28 2016 -0500
1305
1306 still working on print function
1307
1308 commit 4725db18876d7ea6cb4e86b8ba2abd9ad94d5899
1309 Author: AFree mantle <asf2161@columbia.edu>
1310 Date: Sun Nov 20 15:25:06 2016 -0500
1311
1312 local variables and function bodies, moving onto expressions
1313
1314 commit 324961b5e134beaf5829c2d9c752c7def40a7ab4
1315 Author: Alexander Brown <aab2212@columbia.edu>
1316 Date: Sun Nov 20 14:33:14 2016 -0500
1317
1318 Added local variable printing to pprinter
1319
1320 commit 17405503d73de716d69026c6a5e5a7c1e8374f5e
1321 Merge: 707ec6a eclc518
1322 Author: Michelle Navarro <mn2614@columbia.edu>
1323 Date: Sun Nov 20 14:22:19 2016 -0500
1324
1325 Merge pull request #17 from AFree mantle/analyzer
1326
1327 Analyzer
1328

1329 commit ec1c518d8bb0bfff392c8a2618f946115986109f1
1330 Merge: 4cb98cd 707ec6a
1331 Author: Michelle Navarro <mn2614@columbia.edu>
1332 Date: Sun Nov 20 14:20:15 2016 -0500
1333
1334 fixing merge conf
1335
1336 commit 4cb98cd5dacbca5ca1830936628c7f681beaf14c
1337 Author: Alexander Brown <aab2212@columbia.edu>
1338 Date: Sun Nov 20 14:08:41 2016 -0500
1339
1340 Added local vars to fdecls in parser
1341
1342 commit 3e4aff9b2bb90671900e43cdb478e876ad065a03
1343 Author: AFree mantle <asf2161@columbia.edu>
1344 Date: Sun Nov 20 13:56:26 2016 -0500
1345
1346 Lllvm boiler plate + print function decleration
1347
1348 commit 88c5a24d3da7eab88c4ad0956f3276976dd2e8f7
1349 Author: Alexander Brown <aab2212@columbia.edu>
1350 Date: Sun Nov 20 12:08:55 2016 -0500
1351
1352 test-pretty -> tp
1353
1354 commit 89e7042df0fb8d777244a5648d6a1f5588d7d7b4
1355 Author: Alexander Brown <aab2212@columbia.edu>
1356 Date: Sun Nov 20 00:44:27 2016 -0500
1357
1358 sloppy void
1359
1360 commit 707ec6a60dfe94e39eb7f50590a5d154e5fc0884
1361 Merge: 8fdbd33 c5841c5
1362 Author: Michelle Navarro <mn2614@columbia.edu>
1363 Date: Sun Nov 20 00:17:48 2016 -0500
1364
1365 Merge pull request #18 from AFree mantle/tests
1366

1367 Closer to what hello world should look like, string works
1368
1369 commit c5841c50490567477cda2c386fd659762899771f
1370 Author: Michelle Navarro <mn2614@columbia.edu>
1371 Date: Sun Nov 20 00:15:49 2016 -0500
1372
1373 Closer to what hello world should look like, string works
1374
1375 commit 8e8a3039e109fed1203b8d14f341200a26e4f63e
1376 Author: Alexander Brown <aab2212@columbia.edu>
1377 Date: Sat Nov 19 23:08:17 2016 -0500
1378
1379 duplicates and not void pt. 2
1380
1381 commit 6fdc92d7e826e894a889e8c16c75d32b04e5133f
1382 Author: Alexander Brown <aab2212@columbia.edu>
1383 Date: Sat Nov 19 22:28:55 2016 -0500
1384
1385 buuuuuurp
1386
1387 commit 6f2f68b93115cde7384a7e46cfefe3ca97f97659
1388 Author: Alexander Brown <aab2212@columbia.edu>
1389 Date: Sat Nov 19 21:48:38 2016 -0500
1390
1391 analyzer2 beginnings
1392
1393 commit 8fdbd33c6703d8a6c5182154da282df4944b2613
1394 Author: Alexander Brown <aab2212@columbia.edu>
1395 Date: Sat Nov 19 21:30:12 2016 -0500
1396
1397 string[] to char pt. 2
1398
1399 commit 741c3578a123ea2f0dc40fff64a38d332d31e96e
1400 Merge: 10b36a6 d4cbeab
1401 Author: Michelle Navarro <mn2614@columbia.edu>
1402 Date: Sat Nov 19 21:39:35 2016 -0500
1403
1404 Merge pull request #16 from AFree mantle/tests

1405
1406 Tests
1407
1408 commit d4cbeab31fec1bde914976055bb27d8b97dfcaf4
1409 Author: Michelle Navarro <mn2614@columbia.edu>
1410 Date: Sat Nov 19 21:37:43 2016 -0500
1411
1412 cruft
1413
1414 commit e7bcfbd77683358f125d7cb865788a59ebelbbfb
1415 Author: Michelle Navarro <mn2614@columbia.edu>
1416 Date: Sat Nov 19 21:37:04 2016 -0500
1417
1418 Lots of changes
1419
1420 commit 2234035caef46dc857f4fe659b7428628e7088f9
1421 Author: Michelle Navarro <mn2614@columbia.edu>
1422 Date: Sat Nov 19 21:08:44 2016 -0500
1423
1424 hello world test
1425
1426 commit 4adc2d89e46ca285eeade36044606c96b63f2c90
1427 Author: Michelle Navarro <mn2614@columbia.edu>
1428 Date: Sat Nov 19 20:46:37 2016 -0500
1429
1430 Makefile updates
1431
1432 commit c073d0c37374ce66a11a8d00ef615114ea4b6cc6
1433 Author: Michelle Navarro <mn2614@columbia.edu>
1434 Date: Sat Nov 19 20:38:42 2016 -0500
1435
1436 String changes
1437
1438 commit 62f3074040f7082618db65809b910fdfa40228ec
1439 Author: Michelle Navarro <mn2614@columbia.edu>
1440 Date: Sat Nov 19 19:06:17 2016 -0500
1441
1442 We had forgotten stringsgit add .git add .

1443
1444 commit 7bec8b1336e94de76eb61d0cc1e4bcea39bf48c9
1445 Author: Michelle Navarro <mn2614@columbia.edu>
1446 Date: Sat Nov 19 17:29:48 2016 -0500
1447
1448 Fixed Makefile
1449
1450 Now it makes maze automatically when you typw 'make', rather
than clean
1451
1452 commit 577452fd40e2eaec27f77284d254e998519df91f
1453 Author: Michelle Navarro <mn2614@columbia.edu>
1454 Date: Sat Nov 19 17:26:07 2016 -0500
1455
1456 cruft
1457
1458 commit 0bece8f4b30b4be3965c954b401362c72ceaecd9
1459 Author: Michelle Navarro <mn2614@columbia.edu>
1460 Date: Sat Nov 19 17:20:25 2016 -0500
1461
1462 Working on what Hello World should look like
1463
1464 commit 10b36a6b31a507f6b7d6f45c8c6cda8d7ecae9d3
1465 Merge: e1af195 8ac99ed
1466 Author: Michelle Navarro <mn2614@columbia.edu>
1467 Date: Sat Nov 19 15:00:56 2016 -0500
1468
1469 Merge pull request #15 from AFree mantle/pretty-print
1470
1471 Pretty printer actually working
1472
1473 commit 8ac99ed44b3e0fe4c747f46d79797aabb8b3b8fa
1474 Author: Michelle Navarro <mn2614@columbia.edu>
1475 Date: Sat Nov 19 14:58:40 2016 -0500
1476
1477 Pretty printer actually working
1478
1479 This is a small victory

1480 But a victory nonetheless
1481 maze.ml still freaks out when a file doesnt exists though
1482
1483 commit e1af195b316de981a39c7e02f4b44c09e3190b15
1484 Merge: 7e94a00 ca5dc71
1485 Author: Michelle Navarro <mn2614@columbia.edu>
1486 Date: Sat Nov 19 14:07:09 2016 -0500
1487
1488 Merge pull request #14 from AFree mantle/pretty-print
1489
1490 .
1491
1492 commit ca5dc71d0ad8edef99d532453de42521da05efac
1493 Author: Michelle Navarro <mn2614@columbia.edu>
1494 Date: Sat Nov 19 14:03:56 2016 -0500
1495
1496 need to deal with file notfound errors now
1497
1498 commit 5304f6b835c8ef28cc5d3373d31e2ba2dcfee30e
1499 Author: Michelle Navarro <mn2614@columbia.edu>
1500 Date: Sat Nov 19 11:31:28 2016 -0500
1501
1502 struggle continues
1503
1504 commit 770ac700ac68a26d04b950ce36db46590a0fd9c5
1505 Author: Michelle Navarro <mn2614@columbia.edu>
1506 Date: Sat Nov 19 11:00:33 2016 -0500
1507
1508 maze.ml progress but still not done
1509
1510 commit e6735b210571c3a9fb3c0a2d2358433cddfa674f
1511 Author: Michelle Navarro <mn2614@columbia.edu>
1512 Date: Fri Nov 18 23:41:18 2016 -0500
1513
1514 File i/o not working
1515
1516 commit 03595b6252cb57fbb12ad7e03f99fcad1fd2879b
1517 Author: Michelle Navarro <mn2614@columbia.edu>

1518 Date: Fri Nov 18 22:54:46 2016 -0500
1519
1520 removed some fluff
1521
1522 commit 7e94a009e16af04680141e613f679289e7709967
1523 Author: Lindsay <ls3245@columbia.edu>
1524 Date: Sat Nov 19 10:18:43 2016 -0500
1525
1526 added string expression
1527
1528 commit a3aeaa983cd7aaddda04db3f6cb5a09ea4c14fed
1529 Author: Lindsay <ls3245@columbia.edu>
1530 Date: Sat Nov 19 10:18:25 2016 -0500
1531
1532 added string
1533
1534 commit 5d306436240e226dc3c5bd2e924caa00569b3091
1535 Merge: d091669 7e0d85d
1536 Author: Michelle Navarro <mn2614@columbia.edu>
1537 Date: Fri Nov 18 22:28:47 2016 -0500
1538
1539 Merge pull request #13 from AFree mantle/pretty-print
1540
1541 a better maze.ml
1542
1543 commit 7e0d85d0c68313ad00588a6c49da0df9916f7e79
1544 Author: Michelle Navarro <mn2614@columbia.edu>
1545 Date: Fri Nov 18 22:25:45 2016 -0500
1546
1547 It compiles now
1548
1549 commit 84b4f028c657cdaee554df2d5b1e5cc150f55d7a
1550 Author: Michelle Navarro <mn2614@columbia.edu>
1551 Date: Fri Nov 18 22:04:07 2016 -0500
1552
1553 file i/o start'
1554 '
1555

1556 commit 70e34d0c3e01323eb35aacef1e5a667217c79c81
1557 Author: Michelle Navarro <mn2614@columbia.edu>
1558 Date: Thu Nov 17 18:56:42 2016 -0500
1559
1560 Adding option checking to maze.ml
1561
1562 commit d091669ebea44d7810c6e821d504e8e8e177ba49
1563 Author: Lindsay <ls3245@columbia.edu>
1564 Date: Fri Nov 18 17:09:22 2016 -0500
1565
1566 added pretty printing for arrays
1567
1568 commit f46c262a843c3053adbb6865babb871637477517
1569 Author: Lindsay <ls3245@columbia.edu>
1570 Date: Thu Nov 17 18:48:43 2016 -0500
1571
1572 added Array datatype
1573
1574 commit e67828f59f875aabe054c522d4ff85be7e4387f2
1575 Author: Michelle Navarro <mn2614@columbia.edu>
1576 Date: Thu Nov 17 14:37:15 2016 -0500
1577
1578 maze.ml changes
1579
1580 commit f07299a571c83c8407f6468df44f23bd16d4aa38
1581 Merge: 11c402a 64820d5
1582 Author: Michelle Navarro <mn2614@columbia.edu>
1583 Date: Mon Nov 14 00:24:29 2016 -0500
1584
1585 Merge pull request #11 from AFree mantle/pretty-print
1586
1587 Make maze makes maze
1588
1589 commit 64820d5f3e47588644ee0539e3f4cc53fcfcf758
1590 Author: Michelle Navarro <mn2614@columbia.edu>
1591 Date: Mon Nov 14 00:20:54 2016 -0500
1592
1593 Fleshed out Makefile

1594
1595 commit c0ae58c09ae79002c6ce534233f96969047db24f
1596 Author: Michelle Navarro <mn2614@columbia.edu>
1597 Date: Sun Nov 13 19:05:36 2016 -0500
1598
1599 Fixed minor error in maze.ml
1600
1601 commit 11c402ad1ca0c9dee7e29046f040db528e55a855
1602 Author: Lindsay <ls3245@columbia.edu>
1603 Date: Sun Nov 13 18:53:30 2016 -0500
1604
1605 fixed incorrect function call
1606
1607 commit 8c611340fffea9dd1d6f0f785289b0aff3af74a9
1608 Merge: b73d37b a011069
1609 Author: Michelle Navarro <mn2614@columbia.edu>
1610 Date: Sun Nov 13 18:30:32 2016 -0500
1611
1612 Merge pull request #10 from AFree mantle/pretty-print
1613
1614 Scanner & Parser updates
1615
1616 commit a01106979743381f1ff4dcf50cec79c7d07b86ba
1617 Author: Michelle Navarro <mn2614@columbia.edu>
1618 Date: Sun Nov 13 18:26:24 2016 -0500
1619
1620 Debugged scanner
1621
1622 commit 6ecd1d1ee9c31e46094f7c67e3ea1cff0f1ed0ef
1623 Author: Michelle Navarro <mn2614@columbia.edu>
1624 Date: Sun Nov 13 17:00:28 2016 -0500
1625
1626 Got rid of object creation just for now
1627
1628 commit f10036cc795497a05b65caccd4d1ca4d0dd8f9c5
1629 Author: Michelle Navarro <mn2614@columbia.edu>
1630 Date: Sun Nov 13 16:24:25 2016 -0500
1631

1632 AST & Parser inconsistency fixes
1633
1634 commit b73d37bb64601e5406bab854b8739c8e5bb04084
1635 Author: Lindsay <ls3245@columbia.edu>
1636 Date: Sun Nov 13 17:22:09 2016 -0500
1637
1638 fixed two typo errors
1639
1640 commit 222db60c053b2850a05f62cbc9ad1347f55bf826
1641 Author: Lindsay <ls3245@columbia.edu>
1642 Date: Sun Nov 13 17:15:38 2016 -0500
1643
1644 fixed stash changes
1645
1646 commit 26fd5913066fd2399c79e2d0799bff5b9291f06c
1647 Author: Lindsay <ls3245@columbia.edu>
1648 Date: Sun Nov 13 17:11:01 2016 -0500
1649
1650 fixed issues with sast
1651
1652 commit 3354bb5fabd5992dbf91879a63c56877fa97e64b
1653 Author: Lindsay <ls3245@columbia.edu>
1654 Date: Sun Nov 13 17:10:43 2016 -0500
1655
1656 added a few more lines
1657
1658 commit 230605f2f7ff5b1e708cd0e0737120362a0e5017
1659 Author: Lindsay <ls3245@columbia.edu>
1660 Date: Sun Nov 13 15:30:19 2016 -0500
1661
1662 fixed error with let...and
1663
1664 commit deb8164d00f3eca0d6a0534d9c1b409ae0196950
1665 Author: Lindsay <ls3245@columbia.edu>
1666 Date: Sun Nov 13 14:41:17 2016 -0500
1667
1668 return type of an expression
1669

1670 commit 67cc45fb7576005bf0837d5c486b4194e1358827
1671 Merge: bed3090 elfef58
1672 Author: Lindsay <ls3245@columbia.edu>
1673 Date: Sun Nov 13 14:36:02 2016 -0500
1674
1675 Merge branch 'master' of <https://github.com/AFreemantle/maze>
1676
1677 commit bed30903d63b62e3227916400f814f758f51b96d
1678 Author: Lindsay <ls3245@columbia.edu>
1679 Date: Sun Nov 13 14:34:47 2016 -0500
1680
1681 created minimal sast
1682
1683 commit elfef584512327293523f114ca778326750e019e
1684 Merge: 53650f4 1f1066e
1685 Author: Michelle Navarro <mn2614@columbia.edu>
1686 Date: Sun Nov 13 03:04:12 2016 -0500
1687
1688 Merge pull request #9 from AFreemantle/pretty-print
1689
1690 bug fixes
1691
1692 commit 1f1066e0ed57b7c3a89716204c5750b794efedd9
1693 Author: Michelle Navarro <mn2614@columbia.edu>
1694 Date: Sun Nov 13 03:02:03 2016 -0500
1695
1696 Fixed some ast bugs
1697
1698 commit 650cb19039fd1d08978de9b37e5011313dbbd295
1699 Author: Michelle Navarro <mn2614@columbia.edu>
1700 Date: Sat Nov 12 19:03:24 2016 -0500
1701
1702 Parser compilation fix
1703
1704 commit 54624a61099c46036c893059f9a7de42733e74db
1705 Author: Michelle Navarro <mn2614@columbia.edu>
1706 Date: Sat Nov 12 18:45:46 2016 -0500
1707

1708 small update to makefile & parser
1709
1710 commit 53650f44c6f53ff71109b36b7b69a98ee19da2f2
1711 Merge: 31aa2b0 c8b6ee6
1712 Author: Michelle Navarro <mn2614@columbia.edu>
1713 Date: Sat Nov 12 18:00:01 2016 -0500
1714
1715 Merge pull request #8 from AFree mantle/pretty-print
1716
1717 Pretty print
1718
1719 commit c8b6ee66ed0ea7d7f86d68c6ecfb80c4d93f85c4
1720 Author: Michelle Navarro <mn2614@columbia.edu>
1721 Date: Sat Nov 12 17:58:38 2016 -0500
1722
1723 Trying to add .native file for maze.ml compilation error
1724
1725 commit 74b38fff4b26ab9a6ad1db307b2f1401bf220c47
1726 Author: Michelle Navarro <mn2614@columbia.edu>
1727 Date: Sat Nov 12 17:32:24 2016 -0500
1728
1729 maze.ml changes
1730
1731 commit e7f542cd11930ad2b7d138a4efed6455e2c0b4f4
1732 Author: Michelle Navarro <mn2614@columbia.edu>
1733 Date: Sat Nov 12 14:01:23 2016 -0500
1734
1735 starting
1736
1737 commit 99b271b64b22eac6e249d98c0e4178e964c804e6
1738 Author: Michelle Navarro <mn2614@columbia.edu>
1739 Date: Wed Nov 9 23:56:55 2016 -0500
1740
1741 STarting with micro C test suite
1742
1743 commit 31aa2b08963e6ddf592326cf3715e0c02ab6b794
1744 Author: AFree mantle <asf2161@columbia.edu>
1745 Date: Sat Nov 12 15:15:51 2016 -0500

1746
1747 updating with each type from ast
1748
1749 commit e831bfab21ec1e367a3aeca027884c19af0a83d2
1750 Author: AFree mantle <asf2161@columbia.edu>
1751 Date: Sat Nov 12 14:52:49 2016 -0500
1752
1753 updated README
1754
1755 commit 2932760519536239a62cf3102951061769c95a49
1756 Author: AFree mantle <asf2161@columbia.edu>
1757 Date: Sat Nov 12 14:45:36 2016 -0500
1758
1759 initial stages of codegen, started it again
1760
1761 commit 73a1031ddfeb674a29aec6e4158716b77b2c92e7
1762 Author: Lindsay <ls3245@columbia.edu>
1763 Date: Sat Nov 12 14:22:50 2016 -0500
1764
1765 fixed fname in parser
1766
1767 commit 808403a243d78634ddcb7a25dac9b03db55fb21f
1768 Author: Lindsay <ls3245@columbia.edu>
1769 Date: Sat Nov 12 14:22:19 2016 -0500
1770
1771 fixed fname is ast
1772
1773 commit da465e13bcce210cdf70f1cc39fc8a6c3522b429
1774 Merge: 42c0b26 96596c9
1775 Author: Michelle Navarro <mn2614@columbia.edu>
1776 Date: Sat Nov 12 14:08:39 2016 -0500
1777
1778 Merge branch 'master' of github.com:AFree mantle/maze
1779
1780 commit 42c0b261c65164c7d15a6c9df37e9dc358fd92d3
1781 Author: Michelle Navarro <mn2614@columbia.edu>
1782 Date: Sat Nov 12 14:07:57 2016 -0500
1783

1784 constructor fixes
1785
1786 commit 96596c943dd0205dfb2070a251a5491c8882f0c7
1787 Author: AFree mantle <asf2161@columbia.edu>
1788 Date: Thu Nov 10 11:53:24 2016 -0500
1789
1790 fixed llvm api calls
1791
1792 commit 8ceab9fc305734f2af87020f36095a447eccc02c
1793 Merge: 33e13e8 50f516f
1794 Author: Michelle Navarro <mn2614@columbia.edu>
1795 Date: Wed Nov 9 23:59:44 2016 -0500
1796
1797 Merge branch 'master' of github.com:AFree mantle/maze
1798
1799 commit 33e13e81430475249c79b7f3a198bcd51dc98425
1800 Author: Michelle Navarro <mn2614@columbia.edu>
1801 Date: Wed Nov 9 23:59:14 2016 -0500
1802
1803 Fixed very annoying typo
1804
1805 commit 50f516fc4e694bda837bb600e83689a1a9594960
1806 Merge: 5adc718 1b138a6
1807 Author: Lindsay <ls3245@columbia.edu>
1808 Date: Wed Nov 9 23:38:46 2016 -0500
1809
1810 Merge branch 'master' of https://github.com/AFree mantle/maze
1811
1812 commit 5adc7185afa7a7db387037ff83f9544fab91c34a
1813 Author: Lindsay <ls3245@columbia.edu>
1814 Date: Wed Nov 9 23:37:34 2016 -0500
1815
1816 pretty printer compiles
1817
1818 commit 1b138a67de4a15905377d1e033be0777ac70a211
1819 Author: AFree mantle <asf2161@columbia.edu>
1820 Date: Wed Nov 9 23:25:37 2016 -0500
1821

1822 codegen started, types added from ast
1823
1824 commit e8c2b14c07b5eb0b2d18d43aa9fc246ee35a5cde
1825 Author: Lindsay <ls3245@columbia.edu>
1826 Date: Wed Nov 9 23:01:29 2016 -0500
1827
1828 fixed half of it to work with our language
1829
1830 commit 5ac337b539db2a69b80a625c23f25a141ed1f1fd
1831 Author: Lindsay <ls3245@columbia.edu>
1832 Date: Wed Nov 9 23:00:51 2016 -0500
1833
1834 added datatype function
1835
1836 commit bcf2f434a1af48208cbff57119bb6c658de71ba0
1837 Author: AFree mantle <asf2161@columbia.edu>
1838 Date: Wed Nov 9 21:39:18 2016 -0500
1839
1840 skeleton for semantic checker
1841
1842 commit 099b36d54cb9b65bad2284467a7d4adad97a160b
1843 Author: Lindsay <ls3245@columbia.edu>
1844 Date: Wed Nov 9 21:24:51 2016 -0500
1845
1846 nothing new
1847
1848 commit 1e6c24749a8b9ccf22ce120d6e77a15d06b2803d
1849 Author: Lindsay <ls3245@columbia.edu>
1850 Date: Wed Nov 9 21:24:39 2016 -0500
1851
1852 added string def
1853
1854 commit 2cf0fbel0a902e37415f57de743bf9f37e68b2fd
1855 Author: Lindsay <ls3245@columbia.edu>
1856 Date: Sat Nov 5 20:02:36 2016 -0400
1857
1858 only contains clean function
1859

1860 commit 62fb91010bfaa7e59c982047019f82b5625ffc43
1861 Author: Lindsay <ls3245@columbia.edu>
1862 Date: Sat Nov 5 20:01:44 2016 -0400
1863
1864 fixed error with brackets in decls
1865
1866 commit 75b148efc04125e9fcdc84de45faf8fd4c80e3ad
1867 Author: Lindsay <ls3245@columbia.edu>
1868 Date: Sat Nov 5 20:00:58 2016 -0400
1869
1870 no change
1871
1872 commit 518a15d34e4a7dbd8ef915a3e21a1942306325d1
1873 Author: Lindsay <ls3245@columbia.edu>
1874 Date: Wed Nov 2 16:23:04 2016 -0400
1875
1876 added char
1877
1878 commit 13c022a9028a114a5fdb651db44faed0f6aa2378
1879 Author: Lindsay <ls3245@columbia.edu>
1880 Date: Wed Nov 2 15:20:13 2016 -0400
1881
1882 no change
1883
1884 commit 81eb684b921cb418c31ca9def23ed925526379a6
1885 Author: Lindsay <ls3245@columbia.edu>
1886 Date: Wed Nov 2 15:19:41 2016 -0400
1887
1888 fixed syntax error with noexpr, added pattern matching for
float, string, char and null to pretty printer
1889
1890 commit bcd949fb965b213dfbef33febfcd745d3743d33e
1891 Author: Lindsay <ls3245@columbia.edu>
1892 Date: Wed Nov 2 13:58:23 2016 -0400
1893
1894 added extends and new definitions
1895
1896 commit 616291a0d4e3ba6ebc4fb2c13824a6a49e1e2549

1897 Author: Lindsay <ls3245@columbia.edu>
1898 Date: Wed Nov 2 13:57:39 2016 -0400
1899
1900 added type extends
1901
1902 commit 68416769022ccacd84b4cbea7710b38ceab18bdd
1903 Author: Michelle Navarro <mn2614@columbia.edu>
1904 Date: Wed Nov 2 01:42:02 2016 -0400
1905
1906 Added class printing
1907
1908 commit 37a1cfe9c3a092959e7dda6c93d42c4cea2dc5be
1909 Author: Michelle Navarro <mn2614@columbia.edu>
1910 Date: Tue Nov 1 21:40:45 2016 -0400
1911
1912 printing fucntions
1913
1914 commit da9eda74d57a16411e0852be4844ae3d20176058
1915 Author: Michelle Navarro <mn2614@columbia.edu>
1916 Date: Tue Nov 1 20:44:36 2016 -0400
1917
1918 Working on pretty print
1919
1920 commit 39a81e090b967bac385ec6f088be107fc8eee78b
1921 Author: AFree mantle <asf2161@columbia.edu>
1922 Date: Mon Oct 31 20:23:07 2016 -0400
1923
1924 added pretty print functioncs from microc example
1925
1926 commit f3d177b5b7bf2a0a694fe1747e5d78153e570136
1927 Author: Michelle Navarro <mn2614@columbia.edu>
1928 Date: Mon Oct 31 20:15:24 2016 -0400
1929
1930 Removed some stuff we didnt need from parser
1931
1932 commit c7a72f4f6b36832d5701cbeb8817248a2b39d7c1
1933 Author: AFree mantle <asf2161@columbia.edu>
1934 Date: Mon Oct 31 19:37:56 2016 -0400

1935
1936 grouped functions more intelligently
1937
1938 commit a4727712055eea490d8f9d488237f95044c41995
1939 Author: AFree mantle <asf2161@columbia.edu>
1940 Date: Mon Oct 31 19:17:41 2016 -0400
1941
1942 vdecl now reachable
1943
1944 commit 8c256bb257f8383fba767dc6505160ae818f6b5e
1945 Author: AFree mantle <asf2161@columbia.edu>
1946 Date: Fri Oct 28 20:08:33 2016 -0400
1947
1948 hopefully not generating empty lang anymore
1949
1950 commit 352e3ba7944a40befe52e5d40327009b159544cf
1951 Author: Michelle Navarro <mn2614@columbia.edu>
1952 Date: Fri Oct 28 19:57:33 2016 -0400
1953
1954 Func decl changes
1955
1956 commit 6bcf6bed69a39d028eb79add03bdec7961400db8
1957 Merge: 4632b99 ebbe100
1958 Author: Michelle Navarro <mn2614@columbia.edu>
1959 Date: Fri Oct 28 19:38:58 2016 -0400
1960
1961 Merge pull request #7 from AFree mantle/ast
1962
1963 Added some class stuff to AST
1964
1965 commit ebbe10006381cc2082872fb1303bb5be7f16517f
1966 Author: Michelle Navarro <mn2614@columbia.edu>
1967 Date: Fri Oct 28 19:34:06 2016 -0400
1968
1969 Added some class stuff to AST
1970
1971 commit 4632b991ff1e02ab67f0e976302325fb27157bf7
1972 Author: AFree mantle <asf2161@columbia.edu>

1973 Date: Fri Oct 28 19:31:54 2016 -0400
1974
1975 added CONSTRUCTOR keyword, defining class decl now
1976
1977 commit d1589a3b842fdac42a711a118c036f3566e958a6
1978 Author: Michelle Navarro <mn2614@columbia.edu>
1979 Date: Fri Oct 28 16:26:23 2016 -0400
1980
1981 Fixed BOOLS and ID in parser
1982
1983 commit laeel31e7ed9559a0d237a5eb06b32ab7d562e22
1984 Author: AFree mantle <asf2161@columbia.edu>
1985 Date: Thu Oct 27 23:24:58 2016 -0400
1986
1987 Fixed error with literals, slowly going from c --> java like
structure
1988
1989 commit 52bcdaadf446b7d2cf4cb04f2921b8735ad26a2a
1990 Author: AFree mantle <asf2161@columbia.edu>
1991 Date: Thu Oct 27 22:07:03 2016 -0400
1992
1993 updated function declaration
1994
1995 commit dd9a48e94284350b14468c5471ef71bdd767cbf2
1996 Author: Lindsay <ls3245@columbia.edu>
1997 Date: Thu Oct 27 20:31:38 2016 -0400
1998
1999 changed misnamed types
2000
2001 commit 8dba2a0fcacb5eaa7f4ac15033798bb127433334
2002 Author: Lindsay <ls3245@columbia.edu>
2003 Date: Thu Oct 27 20:31:10 2016 -0400
2004
2005 added let commands
2006
2007 commit 4084d86ee06e8605578713694e3ac6dd3726cafb
2008 Author: Lindsay <ls3245@columbia.edu>
2009 Date: Thu Oct 27 20:30:17 2016 -0400

2010
2011 changed type_Lit
2012
2013 commit e63cb647c4a9cd71fac3e7b908a5c8f8166b5219
2014 Author: Lindsay <ls3245@columbia.edu>
2015 Date: Thu Oct 27 19:19:03 2016 -0400
2016
2017 added some declarations
2018
2019 commit 759fabe8a5d5b428aa8f04b452e32c8ff35f3395
2020 Author: Lindsay <ls3245@columbia.edu>
2021 Date: Thu Oct 27 18:44:26 2016 -0400
2022
2023 removed FOR, added statements and expressions
2024
2025 commit 31d8248187d612defcc834e2ac39a70a59afd025
2026 Author: Lindsay <ls3245@columbia.edu>
2027 Date: Thu Oct 27 17:56:16 2016 -0400
2028
2029 same
2030
2031 commit 310b77f4d71c270c1a488db52993cfc3d4b6dcd5
2032 Author: Lindsay <ls3245@columbia.edu>
2033 Date: Thu Oct 27 17:55:31 2016 -0400
2034
2035 fixed char definition
2036
2037 commit a5b35a54854a5d21bbf95581df4bdd5b52f25732
2038 Author: Alexander Brown <aab2212@columbia.edu>
2039 Date: Mon Oct 24 22:41:11 2016 -0400
2040
2041 Added float scanning
2042
2043 commit a58204f580cc524b86301a43df608c5408fdada6
2044 Author: Alexander Brown <aab2212@columbia.edu>
2045 Date: Mon Oct 24 21:55:21 2016 -0400
2046
2047 fixed merge conflict

2048
2049 commit dca9ee023d47f1bfeaaaec35ca9d8d152689e297
2050 Author: Lindsay <ls3245@columbia.edu>
2051 Date: Mon Oct 24 21:48:12 2016 -0400
2052
2053 added keywords to ast
2054
2055 commit d3791e51e5feb91ba375f5f25a347a95d40b2ee1
2056 Author: Lindsay <ls3245@columbia.edu>
2057 Date: Mon Oct 24 21:43:45 2016 -0400
2058
2059 added keywords from scanner
2060
2061 commit a49f19b1fbe23dff126d4366a6288d9fad2a5136
2062 Merge: 87abb1a a3522e1
2063 Author: Lindsay <ls3245@columbia.edu>
2064 Date: Mon Oct 24 21:36:06 2016 -0400
2065
2066 added remainder of our keywords
2067
2068 commit 87abb1a166027f13eb08f990078206c127316f25
2069 Author: Lindsay <ls3245@columbia.edu>
2070 Date: Mon Oct 24 21:34:11 2016 -0400
2071
2072 added remainder of our keywords
2073
2074 commit 9921eca0808913dfcc36980459eedab82dd0ac7b
2075 Author: Lindsay <ls3245@columbia.edu>
2076 Date: Mon Oct 24 21:29:56 2016 -0400
2077
2078 added the rest of our keywords
2079
2080 commit a3522e1a7014e2561f1aedfd89e863fbd009ad4d
2081 Merge: 6d82d68 4c03b70
2082 Author: Alexander Brown <aab2212@columbia.edu>
2083 Date: Mon Oct 24 21:22:36 2016 -0400
2084
2085 Merge branch 'master' of <https://github.com/AFreemantle/maze>

2086
2087 commit 6d82d687b166e107b26d09f9120140030ae8ed5e
2088 Author: Alexander Brown <aab2212@columbia.edu>
2089 Date: Mon Oct 24 21:22:05 2016 -0400
2090
2091 added void, extends, class keywords
2092
2093 commit 4c03b70948cd605fce22aa2fb8b4857d090c9b16
2094 Author: Lindsay <ls3245@columbia.edu>
2095 Date: Mon Oct 24 21:18:34 2016 -0400
2096
2097 skeleton ast
2098
2099 commit 530474070d1a3b10da56750c0d022eae23603bab
2100 Author: Alex <aab2212@columbia.edu>
2101 Date: Mon Oct 24 20:58:21 2016 -0400
2102
2103 added 'null' keyword
2104
2105 commit 9bdc4bd7ae4b8387017ad965284836b2ada3cee5
2106 Author: Lindsay <linskeaub@yahoo.com>
2107 Date: Mon Oct 24 20:32:56 2016 -0400
2108
2109 skeleton parser
2110
2111 commit 1b590a58663278ac3337d38d20a5e7da7fd6eb51
2112 Author: Lindsay <linskeaub@yahoo.com>
2113 Date: Mon Oct 24 20:12:22 2016 -0400
2114
2115 updated scanner
2116
2117 commit 03a5dde002e5ba55d5ff511b6dd84103ba4911f7
2118 Author: Lindsay <linskeaub@yahoo.com>
2119 Date: Sat Oct 22 17:57:28 2016 -0400
2120
2121 scanner
2122
2123 commit a4ed27d845d9a14571a32c6d28faa793ad7246de

2124 Author: AFree mantle <asf2161@columbia.edu>
2125 Date: Sat Oct 8 13:26:13 2016 -0400
2126
2127 Updated the Readme, time to get started!
2128
2129 commit 8ecc506940be00a7dfd98abc1608bcc2ed054bf4
2130 Author: Alexander Freemantle <asf2161@columbia.edu>
2131 Date: Sat Oct 8 12:52:23 2016 -0400
2132
2133 Initial commit

5. Language Evolution

Original Idea

We originally meant to design a language specifically for text-based games. We thought we could incorporate features such as built-in "World" and "Cell" classes to make text-based games easier to write. However, this was more of an idea for a library rather than a language. The feedback we received from our proposal indicated that we should try to build a general purpose object oriented programming language instead.

Actual Implementation

We implemented an imperative language with Java-like syntax. We planned to make an object-oriented language, but we were not able to make object creation happen. However, maze supports the basic data types, methods, and control flow statements. This allows the programmer to create interesting programs.

6. Architecture

The Compiler

Here is the breakdown of our compiler:

Modules:

- **analyzer.ml** : This module checks a source program for its adherence to various rules.
- **codegen.ml** : This module uses the abstract syntax tree that was passed through the analyzer to construct the LLVM IR file.
- **maze.ml** : This is the main module that calls on the other modules.
- **parser.mly** : This module produces an abstract syntax tree from the tokens generated by the scanner.
- **scanner.mll** : Generates tokens and removes whitespace and comments.
- **ast.ml** : This module represents the program after the parser.

Compilation and Testing:

- **Makefile** : builds the maze executable
- **testall.sh** : this is the script that runs the tests in our regressive test suite

The Scanner

The Scanner scans through the input and tokenizes the input, such as keywords, identifiers, operators, and symbols. It discards characters that are no longer needed such as whitespace and comments.

The Parser

The Parser creates an abstract syntax tree (Ast) from the tokens created by the Scanner. The top level contains all classes and structures, then constructors and methods, and the bottom level contains statements and expressions.

The Analyzer

The Analyzer checks the Ast representation of the source program for its adherence to a set of rules. This module checks the source program for rules such as duplicate variable names in a function or duplicate class names. Our analyzer also provides some informative error messages when a rule is found to be broken in the source code. For example, when there is code following a return statement, our analyzer will raise an error and print the error message "nothing may follow a return".

The Code Generator

The codegen module handles building the LLVM IR instructions into a file. Codegen.ml uses the abstract syntax tree passed to it by analyzer.

7. Test Plan

We followed MicroC's example by having a testall.sh script and a "tests" directory containing all of our test cases. Every time we wrote a new component of the language, we would write a test for it. This allowed us to work out the bugs immediately, one component at a time. By adding tests as we added new features, we made sure that the addition of different components would not break pre-existing tests that were known to work. The bash script would ensure that the output of our test code matched the expected output.

Testing Phases

Unit Testing

We tested the parser individually by using Menhir and its --interpret flag to see whether the parser would accept given

input tokens. Other than this, we focused on integration testing as Professor Edwards advised.

Integration Testing

To make sure that our scanner, parser, and ast were working together properly, we wrote a pretty printer. We modeled ours off of MicroC's pretty printer. We made sure that the output tokens were the same as the input tokens.

After we knew that the front-end of the compiler worked, we moved on to testing by adding test files in the tests directory that the testall.sh script would execute. Any member of the group could create a test case if they had a new test case idea or if they created a new feature in codegen or the analyzer. In addition to checking that certain programs worked, we also created tests to identify invalid code. For example, we wrote a test to make sure that attempting to declare a duplicate variable resulted in an error with the appropriate error message.

Automation

We automated our testing by creating a bash script ./testall.sh. This runs through all of our tests and relays whether the test passed or failed. It is a basically a slightly modified version of the test script in MicroC.

Test Suite

The "tests" directory contains all of our test cases. We tested each feature that was added to our language.

Contribution

By the middle of the semester we realized that it would be tough for each of us to work on only what our role described and we all ended up touching all parts of the code. Everyone contributed to the project.

8. Conclusions: Lessons Learned

Lindsay

I ended this project having a much better understanding of the intricacies of a compiler. Through the process of struggling through this project, I have a much clearer picture of how all of the components work together to create a working compiler, even a small, basic one such as the one we created.

Additionally, never be afraid to utilize the members of your group. There will be moments when you will get stuck on a section of code and be making no progress. Sometimes all you need is a fresh set of eyes to take a look at the problem. Remember, you are all in this together, so don't get caught up in only worrying about your assigned role.

Michelle

This project was tough not only because building a compiler in Ocaml is difficult, but also because we had to manage multiple moving parts the entire time. So, having good teammates and good communication with them was key to completing this project. At any given moment, we all knew what everyone else was working on. This helped us keep each other accountable for completing our respective tasks and made it easier to notice when a group member was struggling. We were also not afraid of asking for help from one another, which made us work more efficiently since we spent less time being "stuck" on our own.

Like many previous groups have already said, it was also important to start early. We were glad that we heeded this advice because we tended to move pretty slowly. We met several times a week starting in September, worked up to the very end, and we still felt that we could have used more time.

Alex F.

All of the typical advice for this project is true. Start early and try to touch the code everyday you can. Starting early allowed my group to begin the project at a comfortable pace. I think that these early developments on each of our modules helped us prevent errors later down the road. Despite our best efforts, we still worked up to the wire on this assignment, but it was not nearly as stressful as it could have been had we not started as early as we did.

Additional advice I have to give is to touch the code often. We had a little bit of a break in our development of the project towards the end of the semester and this made for a slow start when we picked up the coding again. Work often so that you don't lose familiarity with the issues you are facing or get confused about where you left off.

Alex B.

An extremely helpful lesson I learned this semester was about how to effectively use git. It's necessary to have some kind of version control, and helpful to be able to break the code while allowing teammates to continue working. I hadn't had much experience working with a team on a large project like this, but my teammates helped get me up to speed.

Also, start early. Overestimate how long various features will take. Accept that you'll need to get semi-comfortable with OCaml. Pretty much everyone hates it at the beginning so don't worry if you do too.

9. Code Listing

analyzer.ml

Authors: Alexander Freemantle, Alex Brown, Michelle Navarro, Lindsay Schiminske

```

1 open Ast
2 open Codegen
3
4 module StringMap = Map.Make(String)
5
6 let check_classes =
7
8   (* ----- Verify variable section ----- *)
9
10  (* Raise an exception if the given list has a duplicate *)
11  let report_duplicate exceptf list =
12    let rec helper = function
13      | n1 :: n2 :: _ when n1 = n2 -> raise (Failure (exceptf n1))
14      | _ :: t -> helper t
15      | [] -> ()
16    in helper (List.sort compare list)
17  in
18
19  (* Check for duplicate class names *)
20  report_duplicate (fun n -> "Duplicate class name " ^ n) (List.map (fun n -> n.dname) classes);
21
22  let check_assign lvaluet rvaluet err =
23    if lvaluet == rvaluet then lvaluet else raise err
24  in
25
26  (* Helper function for check_not_void *)
27  let typ_of_datatype = function
28    | Arraytype(p, i) -> p
29    | Datatype(p) -> p
30    | Any -> Null (* <- this will cause problems eventually *)
31  in
32
33  (* Raise an exception if a given binding is to a void type *)
34  let check_not_void exceptf = function
35    | Field(t, n) -> if typ_of_datatype t == Void then raise (Failure (exceptf n)) else ()
36  in
37
38  let check_not_voidf exceptf = function
39    | Formal(t, n) -> if typ_of_datatype t == Void then raise (Failure (exceptf n)) else ()
40  in
41
42  (* Grabs second element of Field (vdecl) *)
43  let get_second = function
44    | Field(t, n) -> n
45  in
46
47  let get_second_formal = function
48    | Formal(t, n) -> n

```

```

49 in
50
51 let get_typ_formal = function
52     Formal(t, n) -> typ_of_datatype t
53 in
54
55 (* Checks that local method variables are neither null nor duplicates *)
56 let check_locals_lists someMethod =
57     List.iter (check_not_void (fun n -> "illegal void variable " ^ n)) someMethod.locals;
58     report_duplicate (fun n -> "duplicate variable " ^ n) (List.map get_second someMethod.locals);
59 in
60
61 (* Checks local variables for each method of a class argument *)
62 let check_locals_class someClass =
63     List.iter check_locals_lists someClass.dbody.methods;
64 in
65
66 let check_class_vars someClass =
67     List.iter (check_not_void (fun n -> "illegal void variable " ^ n)) someClass.dbody.vdecls;
68     report_duplicate (fun n -> "duplicate variable " ^ n) (List.map get_second someClass.dbody.vdecls);
69 in
70
71 List.iter check_locals_class classes;
72 List.iter check_class_vars classes;
73
74
75 (* ----- Verify function section ----- *)
76 (*let check_methods_class someClass =*)
77 let string_of_fname = function
78     FName(s) -> s
79     | Constructor -> ""
80 in
81
82 (* checks to ensure that user doesn't define print *)
83 let check_for_print funcList =
84     if List.mem "print" (List.map (fun f -> string_of_fname f.fname) funcList)
85     then raise (Failure ("function print is already defined")) else ();
86 in
87
88 let m = StringMap.empty in
89
90 (* adds print to Map *)
91 let built_in_decls = StringMap.add "print"
92     { returnType = Datatype(Void); fname = FName("print"); formals = [Formal(Datatype(String), "")] };
locals = []; body = [] } m
93 in
94
95 (* Adds funclist to Map *)
96 let build_f_decls funcList =

```



```

97     List.fold_left (fun m f -> StringMap.add (string_of_fname f.fname) f m) built_in_decls funcList
98 in
99
100 (* Tries to find function s in a given Map *)
101 let function_decl s someMap = try StringMap.find s someMap
102     with Not_found -> raise (Failure ("unrecognized function " ^ s))
103 in
104
105 let map_add_vdecl m v =
106     match v with
107     | Field(t, n) -> StringMap.add n (typ_of_datatype t) m
108 in
109
110 let map_add_formal m v =
111     match v with
112     | Formal(t, n) -> StringMap.add n (typ_of_datatype t) m
113 in
114
115 (* returns a list of all locals in a class *)
116 let rec grab_func_locals = function
117     [] -> []
118   | [x] -> let y = x.locals in y
119   | head :: tail -> let r = head.locals in r @ grab_func_locals tail
120 in
121
122 (* returns a list of all formals in a class *)
123 let rec grab_func_formals = function
124     [] -> []
125   | [x] -> let y = x.formals in y
126   | head :: tail -> let r = head.formals in r @ grab_func_formals tail
127 in
128
129 (* verifies each class for correct type use, duplicates, incorrect voids *)
130 let check_methods_class someClass =
131     let methods = someClass.dbody.methods in
132     let function_decls = build_f_decls methods in
133     check_for_print methods;
134     report_duplicate (fun n -> "duplicate function " ^ n)
135         (List.map (fun f -> string_of_fname f.fname) methods);
136
137     let symbols_classVars = List.fold_left map_add_vdecl StringMap.empty (someClass.dbody.vdecls @
138 (grab_func_locals methods))
139 in
140
141     (* Map of class vars, locals, and formals for fcns in the class *)
142     let symbols = List.fold_left map_add_formal symbols_classVars (grab_func_formals methods)
143 in
144
145     (* gets the type of some ID *)

```

```

145 let type_of_identifier s =
146     try StringMap.find s symbols
147     with Not_found -> raise (Failure ("undeclared identifier " ^ s))
148 in
149
150 (* ensures valid expressions, statements *)
151 let check_func func =
152     let rec expr = function
153         Id s -> type_of_identifier s
154         | Int_Lit _ -> Int
155         | Bool_Lit _ -> Bool
156         | Float_Lit _ -> Float
157         | Char_Lit _ -> Char
158         | String_Lit _ -> String
159         | Noexpr -> Void
160         | Null -> Null
161         | Unop(op, e) as ex -> let t = expr e
162         in (match op with
163             Neg when t = Int -> Int
164             | Not when t = Bool -> Bool
165             | _ -> raise (Failure ("illegal unary operator " ^
166                 string_of_uop op ^ string_of_typ t ^ " in " ^
167                 string_of_expr ex))
168         | Binop(e1, op, e2) as e -> let t1 = expr e1 and t2 = expr e2
169         in (match op with
170             Add | Sub | Mult | Div -> if ((t1 = Int || t1 = Float)
171                 && (t1 = t2))
172             then (t1)
173             else raise (Failure ("operator " ^
174                 string_of_op op ^ " requires " ^
175                 "two ints or two floats"))
176         | Equal | Neq when t1 = t2 -> Bool
177             (*add error message, test *)
178         | Less | Leq | Greater | Geq ->
179             if ((t1 = Int || t1 = Float) &&
180                 (t1 = t2))
181             then (Bool)
182             else raise (Failure ("operator " ^ string_of_op op ^
183                 " requires int or float operands"))
184
185         | And | Or when t1 = Bool && t2 = Bool -> Bool
186         | _ -> raise (Failure ("illegal binary operator " ^
187             string_of_typ t1 ^ " " ^ string_of_op op ^ " " ^
188             string_of_typ t2 ^ " in " ^ string_of_expr e))
189         )
190     | Assign(var, e) as ex -> let lt = type_of_identifier var
191         and rt = expr e in
192         check_assign lt rt (Failure ("illegal assignment " ^
193             string_of_typ lt ^ " = " ^ string_of_typ rt

```

```

194         ^ " in " ^ string_of_expr ex))
195 (* | ObjCreate(oname, actuals) -> Void
196 | ObjAccess(e1, e2) -> Void *)
197 | Call(fname, actuals) as call ->
198     if (fname = "print") then Void else
199     let fd = function_decl fname function_decls in
200     if List.length actuals != List.length fd.formals then
201         raise (Failure ("expecting " ^ string_of_int
202             (List.length fd.formals) ^ " arguments in " ^
203             string_of_expr call))
204     else List.iter2 (fun f e -> let et = expr e in
205         ignore (check_assign (get_typ_formal f) et
206             (Failure ("illegal actual argument found " ^
207                 string_of_typ et ^ " expected " ^
208                 string_of_typ (get_typ_formal f)
209                 ^ " in " ^ string_of_expr e)))) fd.formals actuals;
210         typ_of_datatype fd.returnType
211
212 in
213
214
215 let check_bool_expr e = if expr e != Bool
216     then raise (Failure ("expected Boolean expression in " ^ string_of_expr e)) else () in
217
218 let rec stmt = function
219     Block s1 -> let rec check_block = function
220         [Return _ as s] -> stmt s
221         | Return _ :: _ -> raise (Failure "nothing may follow a return")
222         | Block s1 :: ss -> check_block (s1 @ ss)
223         | s :: ss -> stmt s ; check_block ss
224         | [] -> ()
225     in check_block s1
226 | Expr e -> ignore (expr e)
227 | Return e -> let t = expr e in if t = (typ_of_datatype func.returnType) then () else
228     raise (Failure ("return gives " ^ string_of_typ t ^
229         " expected " ^ (string_of_datatype func.returnType)
230         ^ " in " ^ string_of_expr e))
231 | If(p, b1, b2) -> check_bool_expr p; stmt b1; stmt b2
232 | While(p, s) -> check_bool_expr p; stmt s
233 in
234
235 stmt (Block func.body);
236
237
238 List.iter (check_not_voidf (fun n -> "illegal void formal " ^ n ^ " in " ^ string_of_fname
func.fname)) func.formals;
239 report_duplicate (fun n -> "duplicate formal " ^ n ^ " in " ^ string_of_fname func.fname)
(List.map get_second_formal func.formals) in
240

```

```

241     List.iter check_func methods
242 in
243
244     (* collects all the functions in the program *)
245     let rec grab_class_fcns = function
246         [] -> []
247         | [x] -> let y = x.dbody.methods in y
248         | head :: tail -> let r = head.dbody.methods in r @ grab_class_fcns tail
249     in
250
251     let string_of_method f = string_of_fname f.fname in
252
253     (* ensures that there is exactly 1 main method in the program *)
254     let check_for_main fl =
255         let rec helper i flist =
256             match flist with
257             | [] -> raise (Failure ("Must have exactly 1 'main' method"))
258             | [x] -> if (i = 0 && ((string_of_method x) = "main")) then ()
259                     else if (i != 1 || (i = 1 && ((string_of_method x) = "main")))
260                         then raise (Failure ("Must have exactly 1 'main' method"))
261                         else ()
262             | h :: t -> if (string_of_method h) = "main"
263                         then helper (i+1) t
264                         else helper i t
265         in helper 0 fl
266     in
267
268     let all_methods = grab_class_fcns classes in
269     check_for_main all_methods;
270
271
272
273
274     List.iter check_methods_class classes;

```

ast.ml

Authors: Alexander Freemantle, Alex Brown, Michelle Navarro, Lindsay Schiminske

```

1 type op = Add | Sub | Mult | Div | Equal | Neq |
2     Less | Leq | Greater | Geq | And | Or
3
4 type uop = Neg | Not
5
6 type typ = Int | Bool | Char | String | Float | Void | (* Objecttype of string | *) Null
7
8 type extends = NoParent | Parent of string
9
10 type datatype = Datatype of typ | Arraytype of typ * int | Any

```

```

11
12 type fname = Constructor | FName of string
13
14 type formal = Formal of datatype * string
15
16 type bind = typ * string
17
18 type expr = Int_Lit of int
19           | Id of string
20           | Binop of expr * op * expr
21           | Assign of string * expr
22           | Bool_Lit of bool
23           | Float_Lit of float
24           | Char_Lit of char
25           | String_Lit of string
26           (* | ObjCreate of string * expr list
27            | ObjAccess of expr * expr *)
28           | Noexpr
29           | Unop of uop *expr
30           | Call of string * expr list
31           | Null
32
33 type vdecl = Field of datatype * string
34
35 type stmt = Block of stmt list
36           | If of expr * stmt * stmt
37           | While of expr * stmt
38           | Expr of expr
39           | Return of expr
40
41 (* type vdecl = Field of datatype * string *)
42
43 type func_decl = {
44     fname      : fname;
45     returnType : datatype;
46     formals    : formal list;
47     locals     : vdecl list;
48     body       : stmt list;
49 }
50
51 type dbody = {
52     vdecls      : vdecl list;
53     constructors : func_decl list;
54     methods     : func_decl list;
55 }
56
57 type class_decl = {
58     dname : string;
59     dbody : dbody;

```

```

60 extends : extends;
61 }
62
63 type program = class_decl list
64
65 (* Pretty-printing functions *)
66
67 let string_of_op = function
68   Add -> "+"
69   | Sub -> "-"
70   | Mult -> "*"
71   | Div -> "/"
72   | Equal -> "=="
73   | Neq -> "!="
74   | Less -> "<"
75   | Leq -> "<="
76   | Greater -> ">"
77   | Geq -> ">="
78   | And -> "&&"
79   | Or -> "||"
80
81 let string_of_uop = function
82   Neg -> "-"
83   | Not -> "!"
84
85 let string_of_ttyp = function
86   Int -> "int"
87   | Bool -> "bool"
88   | Void -> "void"
89   | Char -> "char"
90   | String -> "string"
91   | Float -> "float"
92   | Null -> "null"
93   (*| Objecttype(s) -> "class" ^ s
94
95 let string_of_object = function
96   Datatype(Objecttype(s)) -> s
97   | _ -> "" *)
98
99
100 let string_of_datatype = function
101   Arraytype(p, i) -> (string_of_ttyp p)
102   | Datatype(p) -> (string_of_ttyp p)
103   | Any -> "Any"
104   (*| Void -> "Void"*)
105
106
107 let rec string_of_bracket = function
108   [] -> ""

```

```

109 | head :: tail -> "[" ^ (string_of_expr head) ^ "]" ^ (string_of_bracket tail)
110
111 and string_of_array_typ = function
112   [] -> ""
113 | [last] -> string_of_expr last
114 | head :: tail -> string_of_expr head ^ ", " ^ string_of_array_typ tail
115
116
117
118 and string_of_expr = function
119   Int_Lit(l) -> string_of_int l
120 | Bool_Lit(true) -> "true"
121 | Bool_Lit(false) -> "false"
122 | Float_Lit(m) -> string_of_float m
123 | Char_Lit(c) -> String.make 1 c
124 | String_Lit(s) -> s
125 | Null -> ""
126 | Id(s) -> s
127 | Binop(e1, o, e2) ->
128   string_of_expr e1 ^ " " ^ string_of_op o ^ " " ^ string_of_expr e2
129 | Unop(o, e) -> string_of_uop o ^ string_of_expr e
130 | Assign(v, e) -> v ^ " = " ^ string_of_expr e
131 | Call(f, el) ->
132   f ^ "(" ^ String.concat ", " (List.map string_of_expr el) ^ ")"
133 | Noexpr -> ""
134 (*| ObjAccess(e1, e2) -> string_of_expr e1 ^ "." ^ string_of_expr e2
135 | ObjCreate(s, el) -> "new" ^ s ^ "(" ^ String.concat ", " (List.map string_of_expr el) ^ ")" *)
136
137
138 let string_of_vdecl = function
139 Field(t, id) -> (string_of_datatype t) ^ " " ^ id ^ ";\n"
140
141 let rec string_of_stmt = function
142   Block(stmts) ->
143     "{\n" ^ String.concat "" (List.map string_of_stmt stmts) ^ "}\n"
144 | Expr(expr) -> string_of_expr expr ^ ";\n";
145 | Return(expr) -> "return " ^ string_of_expr expr ^ ";\n";
146 | If(e, s, Block([])) -> "if (" ^ string_of_expr e ^ ")\n" ^ string_of_stmt s
147 | If(e, s1, s2) -> "if (" ^ string_of_expr e ^ ")\n" ^
148   string_of_stmt s1 ^ "else\n" ^ string_of_stmt s2
149 | While(e, s) -> "while (" ^ string_of_expr e ^ ") " ^ string_of_stmt s
150 (* | Vdecl(vdecl) -> string_of_vdecl vdecl ^ ";\n" *)
151
152
153 let string_of_fname = function
154   Constructor -> "constructor"
155 | FName(s) -> s
156
157

```

```

158 (*let string_of_vdecl = function
159 Field(t, id) -> (string_of_datatype t) ^ " " ^ id ^ ";\n" *)
160
161 let string_of_formal = function
162 Formal(d, s) -> (string_of_datatype d) ^ " " ^ s
163
164
165 let string_of_func_decl fdecl = (string_of_datatype fdecl.returnType) ^ " " ^ (string_of_fname
fdecl.fname) ^ " " ^
166     "\n{\n" ^
167     String.concat "," (List.map string_of_formal fdecl.formals) ^
168     String.concat "" (List.map string_of_vdecl fdecl.locals) ^ "\n" ^
169     String.concat "" (List.map string_of_stmt fdecl.body) ^ "}\n"
170
171 let string_of_dbody dbody =
172 String.concat "" (List.map string_of_vdecl dbody.vdecls) ^
173 String.concat "" (List.map string_of_func_decl dbody.constructors) ^
174 String.concat "" (List.map string_of_func_decl dbody.methods)
175
176 let string_of_class decl =
177     "class" ^ decl.dname ^ " {\n" ^ (string_of_dbody decl.dbody) ^ " }\n"
178
179 let string_of_program(decls) = String.concat "\n" (List.map string_of_class decls) ^ ""

```

codegen.ml

Authors: Alexander Freemantle, Alex Brown, Michelle Navarro, Lindsay Schiminske

```

1 (* Codegen for MAZE - based off of MicroC *)
2
3 module L = Llvm
4 module A = Ast
5
6 module StringMap = Map.Make(String)
7
8
9 let translate (classes) =
10
11     let grab_dbody someClass =
12         someClass.A.dbody
13     in
14
15     let rec grab_fcn_lists = function
16         [] -> []
17         | [x] -> let y = x.A.methods in y
18         | head :: tail -> let r = head.A.methods in r @ grab_fcn_lists tail
19     in
20
21     let dbodies = List.map grab_dbody classes in

```



```

22  let functions = grab_fcn_lists dbodies in
23
24  let context = L.global_context () in
25  let the_module = L.create_module context "maze"
26
27  and i32_t = L.i32_type context      (* int *)
28  and i8_t = L.i8_type context       (* printf *)
29  and i1_t = L.i1_type context       (* bool *)
30  and f_t = L.double_type context    (* float *)
31  and void_t = L.void_type context   (* void *)
32  and str_t = L.pointer_type (L.i8_type context) in
33  (* add our other types here *)
34
35  let typ_of_datatype = function
36    | A.Arraytype(p, i) -> p
37    | A.Datatype(p) -> p
38    | A.Any -> A.Null
39  in
40
41  let ltype_of_typ = function
42    | A.Int -> i32_t
43    | A.Bool -> i1_t
44    | A.Void -> void_t
45    | A.String -> str_t
46    | A.Float -> f_t
47    | A.Char -> i8_t
48    | A.Null -> i32_t in
49
50  let ltype_of_formal = function
51    | A.Formal(t, n) -> ltype_of_typ(typ_of_datatype t)
52  in
53
54  let typeKey_of_formal = function
55    | A.Formal(t, n) -> (typ_of_datatype t, n)
56  in
57
58  let typeKey_of_local = function
59    | A.Field(t, n) -> (typ_of_datatype t, n)
60  in
61
62  let string_of_FName = function
63    | A.FName(f) -> f
64    | A.Constructor -> ""
65  in
66
67  (* This is where global var func would go *)
68
69  (* Lets make a function that does some of this boilerplate stuff
70  * below so that we dont have to keep writing it out over and over

```

```

71  * again for each function *)
72  let printf_t =
73      L.var_arg_function_type i32_t [| L.pointer_type i8_t |] in
74  let printf_func =
75      L.declare_function "printf" printf_t the_module in
76
77  (* Define functions *)
78
79  let function_decls =
80      let function_decl m fdecl =
81          let name = (string_of_FName fdecl.A.fname)
82          and formal_types =
83              Array.of_list (List.map ltype_of_formal fdecl.A.formals)
84          in let ftype =
85              L.function_type (ltype_of_ttyp (typ_of_datatype fdecl.A.returnType)) formal_types in
86              StringMap.add name (L.define_function name ftype the_module,
87                  fdecl) m in
88          List.fold_left function_decl StringMap.empty functions in
89
90  (* Here we fill in the body of each of the functions *)
91
92  let build_function_body fdecl =
93
94      let (the_function, _) =
95          StringMap.find (string_of_FName fdecl.A.fname) function_decls in
96
97      let builder =
98          L.builder_at_end context (L.entry_block the_function) in
99
100     (* print *)
101     let int_format_str =
102         L.build_global_stringptr "%d\n" "fmt" builder in
103     let str_format_str =
104         L.build_global_stringptr "%s\n" "fmt" builder in
105     let char_format_str =
106         L.build_global_stringptr "%c\n" "fmt" builder in
107     let float_format_str =
108         L.build_global_stringptr "%f\n" "fmt" builder in (* adds zeros to end of number *)
109
110
111     (* For the cool TADS like feature I am going to
112     * need to do string formatting *)
113
114     (* local variables - Print takes an argument, need this for hello world*)
115     let local_vars =
116         let add_formal m (t, n) p = L.set_value_name n p;
117         let local = L.build_alloca (ltype_of_ttyp t) n builder in
118         ignore (L.build_store p local builder);
119         StringMap.add n local m in

```

```

120
121     let add_local m (t, n) =
122         let local_var = L.build_alloca (ltype_of_typ t) n builder
123         in StringMap.add n local_var m in
124
125     let formals = List.fold_left2 add_formal StringMap.empty
126         (List.map typeKey_of_formal fdecl.A.formals) (Array.to_list (L.params the_function)) in
127     List.fold_left add_local formals (List.map typeKey_of_local fdecl.A.locals) in
128
129 let lookup n = StringMap.find n local_vars in
130
131 let type_of_val = function
132     "i32*" -> int_format_str (*int*)
133 | "i8*" -> str_format_str (*string*)
134 | "i8*" -> char_format_str (*char*)
135 | "i1*" -> int_format_str (*bool*)
136 | "double*" -> float_format_str (*float*)
137 | _ -> str_format_str
138 in
139
140 let check_print_input = function
141     A.Int_Lit e -> int_format_str
142 | A.String_Lit e -> str_format_str
143 | A.Char_Lit c -> char_format_str
144 | A.Float_Lit f -> float_format_str
145 | A.Binop (e1, op, e2) -> int_format_str
146 | A.Bool_Lit b -> int_format_str
147 | A.Id s -> type_of_val (L.string_of_lltype (L.type_of (lookup s)))
148 | A.Assign (s, e) -> int_format_str
149 | A.Noexpr -> int_format_str
150 | A.Unop (op, e) -> int_format_str
151 | A.Call (s, actuals) -> int_format_str
152 | A.Null -> int_format_str
153 in
154
155 (* Generate code for an expression *)
156
157 let rec expr builder = function
158     A.Int_Lit i -> L.const_int i32_t i
159 | A.Bool_Lit b -> L.const_int i1_t (if b then 1 else 0)
160 | A.String_Lit s -> L.build_global_stringptr s "str" builder
161 | A.Noexpr -> L.const_int i32_t 0
162 | A.Id s -> L.build_load (lookup s) s builder
163 | A.Binop (e1, op, e2) ->
164 let e1' = expr builder e1
165 and e2' = expr builder e2
166 and float_ops = (match op with
167     A.Add -> L.build_fadd
168 | A.Sub -> L.build_fsub

```

```

169 | A.Mult    -> L.build_fmuls
170 |   | A.Div    -> L.build_fdiv
171 | A.And     -> L.build_and
172 | A.Or      -> L.build_or
173 | A.Equal   -> L.build_fcmp L.Fcmp.Oeq
174 | A.Neq     -> L.build_fcmp L.Fcmp.One
175 | A.Less    -> L.build_fcmp L.Fcmp.Olt
176 | A.Leq     -> L.build_fcmp L.Fcmp.Ole
177 | A.Greater -> L.build_fcmp L.Fcmp.Ogt
178 | A.Geq     -> L.build_fcmp L.Fcmp.Oge
179 )
180
181     and int_ops = match op with
182     A.Add     -> L.build_add
183 | A.Sub      -> L.build_sub
184 | A.Mult     -> L.build_mul
185 |   | A.Div    -> L.build_sdiv
186 | A.And      -> L.build_and
187 | A.Or       -> L.build_or
188 | A.Equal    -> L.build_icmp L.Icmp.Eq
189 | A.Neq      -> L.build_icmp L.Icmp.Ne
190 | A.Less     -> L.build_icmp L.Icmp.Slt
191 | A.Leq      -> L.build_icmp L.Icmp.Sle
192 | A.Greater  -> L.build_icmp L.Icmp.Sgt
193 | A.Geq      -> L.build_icmp L.Icmp.Sge in
194     if(L.type_of e1' = f_t || L.type_of e2' = f_t) then float_ops e1' e2' "tmp" builder
195 else int_ops e1' e2' "tmp" builder
196
197 (* | A.ObjCreate(id, e1) ->
198 let f = func_lookup id in
199 let params = List.map (expr builder) e1 in
200 let obj = L.build_call f (Array.of_list params) "tmp" builder in
201 obj *)
202
203
204 | A.Unop(op, e) ->
205 let e' = expr builder e in
206 (match op with
207   A.Neg     -> L.build_neg
208 |   | A.Not    -> L.build_not) e' "tmp" builder
209 | A.Assign (s, e) -> let e' = expr builder e in
210     ignore (L.build_store e' (lookup s) builder); e'
211 | A.Float_Lit f -> L.const_float f_t f
212 | A.Char_Lit c -> L.const_int i8_t (Char.code c)
213 | A.Null -> L.const_null i32_t
214 | A.Call ("printf", [e]) -> L.build_call printf_func
215     [| check_print_input e; (expr builder e) |]
216     "printf" builder
217 (* This evaluates arguments backwards *)

```

```

218 | A.Call (f, act) ->
219     let (fdef, fdecl) = StringMap.find f function_decls in
220     let actuals =
221         List.rev (List.map (expr builder) (List.rev act)) in
222     let result = (match (typ_of_datatype fdecl.A.returnType) with A.Void -> ""
223                    | _ -> f ^ "_result") in
224     L.build_call fdef (Array.of_list actuals) result builder in
225
226 (* MicroC has a helper here: add_terminal *)
227 let add_terminal builder f =
228     match L.block_terminator (L.insertion_block builder) with
229     Some _ -> ()
230 | None -> ignore (f builder) in
231
232
233 (* define statements here *)
234 let rec stmt builder = function
235     A.Block sl -> List.fold_left stmt builder sl
236
237 | A.Expr e -> ignore (expr builder e); builder
238
239 | A.Return e -> ignore (match (typ_of_datatype fdecl.A.returnType) with
240     A.Void -> L.build_ret_void builder
241 | _ -> L.build_ret (expr builder e) builder); builder
242
243 | A.If (pred, then_stmt, else_stmt) ->
244 let bool_val = expr builder pred in
245 let merge_bb = L.append_block context "merge" the_function in
246 let then_bb = L.append_block context "then" the_function in
247 add_terminal (stmt (L.builder_at_end context then_bb) then_stmt) (L.build_br merge_bb);
248 let else_bb = L.append_block context "else" the_function in
249 add_terminal (stmt (L.builder_at_end context else_bb) else_stmt) (L.build_br merge_bb);
250
251 ignore (L.build_cond_br bool_val then_bb else_bb builder);
252 L.builder_at_end context merge_bb
253
254 | A.While (pred, body) ->
255 let pred_bb = L.append_block context "while" the_function in
256 ignore (L.build_br pred_bb builder);
257 let body_bb = L.append_block context "while_body" the_function in
258 add_terminal (stmt (L.builder_at_end context body_bb) body) (L.build_br pred_bb);
259 let pred_builder = L.builder_at_end context pred_bb in
260 let bool_val = expr pred_builder pred in
261 let merge_bb = L.append_block context "merge" the_function in
262 ignore (L.build_cond_br bool_val body_bb merge_bb pred_builder);
263 L.builder_at_end context merge_bb
264
265 in
266

```

```
267 (* Code for each statement in the function *)
268 let builder = stmt builder (A.Block fdecl.A.body) in
269
270 (*MicroC behavior here is to have program return void
271 * if control falls off the end of the program *)
272 add_terminal builder (match (typ_of_datatype fdecl.A.returnType) with
273     A.Void -> L.build_ret_void
274     | t -> L.build_ret (L.const_int (ltype_of_typ t) 0))
275 in
276
277
278 List.iter build_function_body functions;
279 the_module
```

Makefile

Authors: Alexander Freemantle, Alex Brown, Michelle Navarro, Lindsay Schiminske

```
1 OBJS = ast.cmx codegen.cmx parser.cmx scanner.cmx analyzer.cmx maze.cmx
2
3 maze : $(OBJS)
4     ocamlfind ocamlopt -linkpkg -package llvm str.cmxa -package llvm.analysis $(OBJS) -o maze
5
6 scanner.ml : scanner.mll
7     ocamllex scanner.mll
8
9 parser.ml parser.mli : parser.mly
10    ocamlyacc parser.mly
11
12 %.cmo : %.ml
13    ocamlc -c $<
14
15 %.cmi : %.mli
16    ocamlc -c $<
17
18 %.cmx : %.ml
19    ocamlfind ocamlopt -c -package llvm str.cmxa $<
20
21 ###Generated by "ocamldep *.ml *.mli" after building scanner.ml and parser.ml
22 ast.cmo :
23 ast.cmx :
24 codegen.cmo : ast.cmo
25 codegen.cmx : ast.cmx
26 maze.cmo : analyzer.cmo scanner.cmo parser.cmi codegen.cmo ast.cmo
27 maze.cmx : analyzer.cmx scanner.cmx parser.cmx codegen.cmx ast.cmx
28 parser.cmo : ast.cmo parser.cmi
29 parser.cmx : ast.cmx parser.cmi
30 scanner.cmo : parser.cmi
31 scanner.cmx : parser.cmx
```

```
32 analyzer.cmo: ast.cmo
33 analyzer.cmx: ast.cmx
34 parser.cmi : ast.cmo
35
36 .PHONY : clean
37 clean:
38     ocamlbuild -clean
39     rm -rf ./tests/*.ll
40     rm -rf testall.log *.diff maze scanner.ml parser.ml parser.mli
41     rm -rf *.cmx *.cmi *.cmo *.o *.out *.ll
```

maze.ml

Authors: Alexander Freemantle, Alex Brown, Michelle Navarro, Lindsay Schiminske

```
1 open Printf
2 open Str
3
4 type action = Ast | Help | Error | LLVM_IR | Compile
5
6 let help_string = (
7     "Usage: ./maze [option] <source file>\n" ^
8     "\n Valid Options: \n"^
9     "\t -h: Help \n" ^
10    "\t -c: Compile \n" ^
11    "\t -l: Generate LLVM without checking \n" ^
12    "\t -a: Print Abstract Syntax tree \n"
13 )
14
15 let invalid_arg_string = ("Invalid Arguments")
16
17 let ast_holder = ("Ast would be printing\n")
18
19 let check_option = function
20     "-h" -> Help, "."
21     | _ -> Error, "."
22
23 let check_action = function
24     "-h" -> Help
25     | "-a" -> Ast
26     | "-l" -> LLVM_IR
27     | "-c" -> Compile
28     | _ -> Error
29
30 let _ =
31     let action, filename =
32         if Array.length Sys.argv = 1 then Help, "."
33         else if Array.length Sys.argv = 2 then check_option (Sys.argv.(1))
```

```

34     else if Array.length Sys.argv = 3 then check_action Sys.argv.(1), (Sys.argv.(2))
35     else Error, "." in
36
37 let check_filename =
38     if filename = "." then raise (Failure (invalid_arg_string)) else () in
39 check_filename;
40
41 let in_channel = open_in filename in
42 let lexbuf = Lexing.from_channel in_channel in
43 let program = Parser.program Scanner.token lexbuf in
44
45 let filename_list = Str.split (regexp "[.]") filename in
46 let basename = List.hd filename_list in
47
48 Analyzer.check program;
49
50 match action with
51 | Help -> print_string help_string
52 | Ast -> print_string (Ast.string_of_program program)
53 | LLVM_IR -> print_string (Llvm.string_of_llmodule
54                         (Codegen.translate program))
55 | Compile -> let m = Codegen.translate program in
56 | Llvm_analysis.assert_valid_module m; (*Built in check*)
57 | (*print_string (Llvm.string_of_llmodule m);*)
58 | (*Llvm.dump_module m;*)
59 | let oc = open_out (basename ^ ".ll") in fprintf oc "%s\n" (Llvm.string_of_llmodule m); close_out
60 | Error -> print_string (invalid_arg_string ^ "\n")
61
62
63
64
65
66

```

parser.mly

Authors: Alexander Freemantle, Alex Brown, Michelle Navarro, Lindsay Schiminske

```

1 %{ open Ast %}
2
3 %token SEMI LPAREN RPAREN LBRACE RBRACE DOT COMMA
4 %token PLUS MINUS TIMES DIVIDE ASSIGN NOT
5 %token EQ NEQ LT LEQ GT GEQ TRUE FALSE AND OR
6 %token RETURN IF ELSE WHILE
7 %token INT BOOL CHAR STRING FLOAT VOID NULL
8 %token CLASS CONSTRUCTOR EXTENDS NEW
9
10 %token <int> INT_LITERAL

```



```

11 %token <char> CHAR_LITERAL
12 %token <float> FLOAT_LITERAL
13 %token <string> ID
14 %token <string> STRING_LITERAL
15 %token EOF
16
17 %nonassoc NOELSE
18 %nonassoc ELSE
19 %right ASSIGN
20 %left OR
21 %left AND
22 %left EQ NEQ
23 %left LT GT LEQ GEQ
24 %left PLUS MINUS
25 %left TIMES DIVIDE
26 %right NOT NEG
27 %right DOT
28
29 %start program
30 %type <Ast.program> program
31
32 %%
33
34
35 program: decls EOF { $1 }
36
37 /* Classes */
38
39 decls:
40     class_decl_list { List.rev $1}
41
42 class_decl_list:
43     class_decl { [$1] }
44 | class_decl_list class_decl { $2::$1 }
45
46 class_decl:
47     CLASS ID LBRACE dbody RBRACE { {
48         dname = $2;
49         extends = NoParent;
50         dbody = $4;
51     } }
52
53 | CLASS ID EXTENDS ID LBRACE dbody RBRACE { {
54     dname = $2;
55     extends = Parent($4);
56     dbody = $6;
57     } }
58 dbody:
59

```

```

60     /* nothing here */ { {
61         vdecls = [];
62         constructors = [];
63         methods = [];
64     } }
65
66 |   dbody vdecl { {
67     vdecls = $2 :: $1.vdecls;
68     constructors = $1.constructors;
69     methods = $1.methods;
70   } }
71
72
73 |   dbody constructor { {
74     vdecls = $1.vdecls;
75     constructors = $2 :: $1.constructors;
76     methods = $1.methods;
77   } }
78
79 |   dbody fdecl { {
80     vdecls = $1.vdecls;
81     constructors = $1.constructors;
82     methods = $2 :: $1.methods;
83   } }
84
85 /* Constructors */
86
87 constructor:
88     CONSTRUCTOR LPAREN formals_opt RPAREN LBRACE stmt_list RBRACE {
89     {
90         fname = Constructor;
91         formals = $3;
92         body = List.rev $6;
93         locals = [];
94         returnType = Any
95     }
96 }
97
98
99
100 /* Methods */
101
102 fname:
103     ID { $1 }
104
105 fdecl:
106     typ fname LPAREN formals_opt RPAREN LBRACE vdecl_list stmt_list RBRACE
107     {
108     {

```

```

109         fname = FName($2);
110         returnType = $1;
111         formals = $4;
112         locals = List.rev $7;
113         body = List.rev $8; (* stmt_list; *)
114     }
115 }
116
117
118 /* Variables */
119
120 vdecl_list:
121     /* nothing */ { [] }
122 | vdecl_list vdecl { $2 :: $1 }
123
124 vdecl: typ ID SEMI { Field($1, $2) }
125
126
127 /* Formals */
128
129 formals_opt:
130     /* nothing */ { [] }
131 | formal_list {List.rev $1 }
132
133 formal_list:
134     typ ID { [Formal($1, $2)] }
135 | formal_list COMMA typ ID { Formal($3, $4) :: $1 }
136
137
138 actuals_opt:
139     /* nothing */ { [] }
140 | actuals_list { List.rev $1 }
141
142 actuals_list:
143     expr { [$1] }
144 | actuals_list COMMA expr { $3 :: $1 }
145
146
147 /* TYPES */
148
149 typ:
150     INT { Datatype(Int) }
151 | FLOAT { Datatype(Float) }
152 | CHAR { Datatype(Char) }
153 | STRING {Datatype(String)}
154 | BOOL { Datatype(Bool) }
155 | VOID { Datatype(Void) }
156
157

```

```

158 /* Expressions */
159
160 stmt_list:
161     /* nothing */{ [] }
162     | stmt_list stmt { $2 :: $1 }
163
164 stmt:
165     expr SEMI      { Expr $1 }
166     | RETURN SEMI { Return Noexpr }
167     | RETURN expr SEMI { Return $2 }
168     | LBRACE stmt_list RBRACE { Block(List.rev $2) }
169     | IF LPAREN expr RPAREN stmt %prec NOELSE { If($3, $5, Block([])) }
170     | IF LPAREN expr RPAREN stmt ELSE stmt { If($3, $5, $7) }
171     | WHILE LPAREN expr RPAREN stmt { While($3, $5) }
172 /* | typ ID SEMI { Vdecl($1, $2) } */
173
174
175 expr:
176     literals      { $1 }
177     | expr PLUS expr { Binop($1, Add, $3) }
178     | expr MINUS expr { Binop($1, Sub, $3) }
179     | expr TIMES expr { Binop($1, Mult, $3) }
180     | expr DIVIDE expr { Binop($1, Div, $3) }
181     | expr EQ expr { Binop($1, Equal, $3) }
182     | expr NEQ expr { Binop($1, Neq, $3) }
183     | expr LT expr { Binop($1, Less, $3) }
184     | expr GT expr { Binop($1, Greater, $3) }
185     | expr LEQ expr { Binop($1, Leq, $3) }
186     | expr GEQ expr { Binop($1, Geq, $3) }
187     | expr AND expr { Binop($1, And, $3) }
188     | expr OR expr { Binop($1, Or, $3) }
189 /* | expr DOT expr { ObjAccess($1, $3) } */
190     | MINUS expr %prec NEG { Unop(Neg, $2) }
191     | NOT expr { Unop(Not, $2) }
192     | ID ASSIGN expr { Assign($1, $3) }
193     | LPAREN expr RPAREN { $2 }
194     | ID LPAREN actuals_opt RPAREN { Call($1, $3) }
195 /* | NEW ID LPAREN actuals_opt RPAREN { ObjCreate($2, $4) } */
196
197
198
199 literals:
200     INT_LITERAL    { Int_Lit($1) }
201     | FLOAT_LITERAL { Float_Lit($1) }
202     | CHAR_LITERAL  { Char_Lit($1) }
203     | STRING_LITERAL { String_Lit($1) }
204     | TRUE          { Bool_Lit(true) }
205     | FALSE         { Bool_Lit(false) }

```

```
206 | ID          { Id($1) }
207 | NULL        { Null }
```

scanner.mll

Authors: Alexander Freemantle, Alex Brown, Michelle Navarro, Lindsay Schiminske

```
1 { open Parser
2
3 let unescape s = Scanf.sscanf ("\\" ^ s ^ "\\") "%S%!" (fun x->x)
4 }
5
6 let alpha = ['a' - 'z' 'A' - 'Z']
7 let digit = ['0' - '9']
8 let escape = '\\\ ['\\' '\n' '\r' '\t']
9 let ascii = ([' '-!' '#'-[' ' ]'-~'])
10 let float = (digit+ ['.' ] digit+)
11 let int = digit+
12 let char = ' ' ( ascii | digit ) ' '
13 let string = ' ' ( (ascii | escape)* as s) ' '
14 let id = alpha(alpha | digit | '_' )*
15
16 rule token = parse
17   [ ' ' '\t' '\r' '\n' ] {token lexbuf} (* Whitespace *)
18 | "(*" {comment lexbuf} (*Comments*)
19
20 | '(' {LPAREN}
21 | ')' {RPAREN}
22 | '{' {LBRACE}
23 | '}' {RBRACE}
24 | ';' {SEMI}
25 | ',' {COMMA}
26 | '.' {DOT}
27
28 | '+' {PLUS}
29 | '-' {MINUS}
30 | '*' {TIMES}
31 | '/' {DIVIDE}
32 | '=' {ASSIGN}
33 | "==" {EQ}
34 | "!=" {NEQ}
35 | '<' {LT}
36 | '>' {GT}
37 | "<=" {LEQ}
38 | ">=" {GEQ}
39 | "&&" {AND}
40 | "||" {OR}
```

```

41 | "!" {NOT}
42
43 | "if" {IF}
44 | "else" {ELSE}
45 | "while" {WHILE}
46 | "return" {RETURN}
47 | "int" {INT}
48 | "bool" {BOOL}
49 | "char" {CHAR}
50 | "float" {FLOAT}
51 | "string" {STRING}
52 | "void" {VOID}
53 | "true" {TRUE}
54 | "false" {FALSE}
55 | "null" {NULL}
56 | "void" {VOID}
57 | "extends" {EXTENDS}
58 | "class" {CLASS}
59 | "new" {NEW}
60
61 (*| ['0' - '9']+ as lxm { LITERAL(int_of_string lxm) } *)
62 (*| ['0' - '9']+['.']['0' - '9']+ as lxm { FLOAT_LITERAL(float_of_string lxm) }
63 | ['a' - 'z' 'A' - 'Z']['a' - 'z' 'A' - 'Z' '0' - '9' '_']* as lxm { ID(lxm) }
64 | eof {EOF}
65 | _ as char { raise (Failure("illegal character " ^ Char.escaped char)) }
66 *)
67
68 | int as lxm { INT_LITERAL(int_of_string lxm) }
69 | float as lxm { FLOAT_LITERAL(float_of_string lxm) }
70 | char as lxm { CHAR_LITERAL(String.get lxm 1) }
71 | string { STRING_LITERAL(unescape s) }
72 | id as lxm { ID(lxm) }
73 | eof { EOF }
74
75 and comment = parse
76     "*)" {token lexbuf}
77 | _ {comment lexbuf}

```

testall.sh

Authors: Michelle Navarro (adapted code from MicroC)

```

1 #!/bin/sh
2
3 # Regression testing script for Maze
4 # Step through a list of files
5 # Compile, run, and check the output of each expected-to-work test
6 # Compile and check the error of each expected-to-fail test

```

```
7
8 # Path to the LLVM interpreter
9 LLI="lli"
10 #LLI="/usr/local/opt/llvm/bin/lli"
11
12 # Path to the maze compiler.
13 MAZE="./maze"
14 #MAZE="./maze"
15
16 # Set time limit for all operations
17 ulimit -t 30
18
19 globallog=testall.log
20 rm -f $globallog
21 error=0
22 globalerror=0
23
24 keep=0
25
26 Usage() {
27     echo "Usage: testall.sh [options] [.maze files]"
28     echo "-k    Keep intermediate files"
29     echo "-h    Print this help"
30     exit 1
31 }
32
33 SignalError() {
34     if [ $error -eq 0 ] ; then
35         echo "FAILED"
36         error=1
37     fi
38     echo " $1"
39 }
40
41 # Compare <outfile> <reffile> <difffile>
42 # Compares the outfile with reffile. Differences, if any, written to difffile
43 Compare() {
44     generatedfiles="$generatedfiles $3"
45     echo diff -b $1 $2 ">" $3 1>&2
46     diff -b "$1" "$2" > "$3" 2>&1 || {
47         SignalError "$1 differs"
48         echo "FAILED $1 differs from $2" 1>&2
49     }
50 }
51
52 # Run <args>
```

```

53 # Report the command, run it, and report any errors
54 Run() {
55     echo $* 1>&2
56     eval $* || {
57         SignalError "$1 failed on $*"
58         return 1
59     }
60 }
61
62 # RunFail <args>
63 # Report the command, run it, and expect an error
64 RunFail() {
65     echo $* 1>&2
66     eval $* && {
67         SignalError "failed: $* did not report an error"
68         return 1
69     }
70     return 0
71 }
72
73 Check() {
74     error=0
75     basename=`echo $1 | sed 's/.*\\\/\\\/\\\/
76                 s/.maze//'\`
77     reffile=`echo $1 | sed 's/.maze$//'\`
78     basedir=`echo $1 | sed 's/\\/[^\\\/]*$//'\`/."
79
80     echo -n "$basename..."
81
82     echo 1>&2
83     echo "##### Testing $basename" 1>&2
84
85     generatedfiles=""
86
87
88     generatedfiles="$generatedfiles ${basename}.ll ${basename}.out"&&
89     Run "$MAZE -c ${reffile}.maze" &&
90     $LLI "${reffile}.ll" > "${basename}.out"
91     Compare ${basename}.out ${reffile}.out ${basename}.diff
92
93     # Report the status and clean up the generated files
94
95     if [ $error -eq 0 ] ; then
96 if [ $keep -eq 0 ] ; then
97     rm -f $generatedfiles
98 fi

```



```

99     echo "OK"
100     echo "##### SUCCESS" 1>&2
101     else
102     echo "##### FAILED" 1>&2
103     globalerror=$error
104     fi
105 }
106
107 CheckFail() {
108     error=0
109     basename=`echo $1 | sed 's/.*\///
110                 s/.maze//'\`
111     reffile=`echo $1 | sed 's/.maze$//'\`
112     basedir=`echo $1 | sed 's/\[^\/]*$//'\`/."
113
114     echo -n "$basename..."
115
116     echo 1>&2
117     echo "##### Testing $basename" 1>&2
118
119     generatedfiles=""
120
121     generatedfiles="$generatedfiles ${basename}.err ${basename}.diff"&&
122     RunFail "$MAZE -c ${reffile}.maze" "2>" "${basename}.err" ">>" $globallog &&
123     Compare ${basename}.err ${reffile}.err ${basename}.diff
124
125     # Report the status and clean up the generated files
126
127     if [ $error -eq 0 ] ; then
128     if [ $keep -eq 0 ] ; then
129         rm -f $generatedfiles
130     fi
131     echo "OK"
132     echo "##### SUCCESS" 1>&2
133     else
134     echo "##### FAILED" 1>&2
135     globalerror=$error
136     fi
137 }
138
139 while getopts kdpsh c; do
140     case $c in
141         k) # Keep intermediate files
142             keep=1
143             ;;
144         h) # Help

```

```

145         Usage
146         ;;
147     esac
148 done
149
150 shift `expr $OPTIND - 1`
151
152 LLIFail() {
153     echo "Could not find the LLVM interpreter \"\$LLI\"."
154     echo "Check your LLVM installation and/or modify the LLI variable in testall.sh"
155     exit 1
156 }
157
158 which "\$LLI" >> $globallog || LLIFail
159
160
161 if [ $# -ge 1 ]
162 then
163     files=$@
164 else
165     files="tests/test-*.maze tests/fail-*.maze"
166 fi
167
168 for file in $files
169 do
170     case $file in
171         *test-*)
172             Check $file 2>> $globallog
173             ;;
174         *fail-*)
175             CheckFail $file 2>> $globallog
176             ;;
177         *)
178             echo "unknown file type $file"
179             globalerror=1
180             ;;
181     esac
182 done
183
184 exit $globalerror

```

Below are all of our test files and the corresponding check files.

Authors: Alexander Freemantle, Alex Brown, Michelle Navarro, Lindsay Schiminske

./test-unop.out

```
1 correct
```

```
2 correct
```

```
./test-bool.out
```

```
1 1
```

```
2 0
```

```
3 1
```

```
4 0
```

```
./test-if2.maze
```

```
1 class test {
2     void main () {
3
4         if(false){
5             print("if");
6         }
7
8         else if(false){
9             print("elseif1");
10        }
11
12        else if(false){
13            print("elseif2");
14        }
15
16        else{
17            print("else");
18        }
19    }
20 }
```

```
./test-unop.maze
```

```
1 class test {
2     void main () {
3
4         if(!false){
5             print("correct");
6         }
7
8         if(3 != -3){
9             print("correct");
10        }
11    }
12 }
```

```
./test-addition.maze
```

```
1 class test {
```

```
2 void main () {
3
4     int x;
5     x = 3 + 7;
6
7     if(x==10){
8         print("correct");
9     }
10 }
11 }
```

./test-printid-bool.out

1 0

./test-equality.maze

```
1 class test {
2     void main () {
3
4         if(42<50){
5             print("less");
6         }
7
8         if(60>3){
9             print("greater");
10        }
11
12        if(100==100){
13            print("equal");
14        }
15
16        if(30!=53){
17            print("not equal");
18        }
19
20        if(10<=40){
21            print("leq");
22        }
23
24        if(50>=50){
25            print("geq");
26        }
27
28    }
29 }
```

./fail-binopOperands.maze

```
1 class test {
2     void main () {
3         int y;
4         float z;
5
6         y = 42;
7         z = 1.0;
8         y + z;
9
10    }
11 }
```

./fail-defprint.err

```
1 Fatal error: exception Failure("function print is already defined")
```

./test-recursion.out

```
1 34
```

./test-printid-int.maze

```
1 class test {
2     void main() {
3         int x;
4         x = 42;
5         print(x);
6     }
7 }
```

./fail-voidClassvar.err

```
1 Fatal error: exception Failure("illegal void variable x")
```

./fail-voidvar.err

```
1 Fatal error: exception Failure("illegal void variable x")
```

./test-whilestmt.maze

```
1 class test {
2     void main () {
3
4         int i;
5         i = 0;
6
7         while(i < 5){
8             print("while loop");
9             i = i + 1;
10        }
11    }
12 }
```

./fail-dupvar.maze

```
1 class test {
2     void main () {
3         int x;
4         int x;
5     }
6 }
```

./test-subtraction.out

1 correct

./test-ifEmpty.maze

```
1 class test {
2     void main () {
3
4         if(false){}
5         else{}
6
7         print("empty");
8     }
9 }
```

./test-ifEmpty.out

1 empty

./test-printid-float.maze

```
1 class test {
2     void main() {
3         float x;
4         x = 42.00000;
5         print(x);
6     }
7 }
```

./test-gcd.out

1 5

./test-binop.maze

```
1 class test {
2     void main(){
3         print(4114 + 1);
4     }
5 }
```

./test-hello.out

```
1 Hello World
```

./test-ifNested.maze

```
1 class test {
2     void main () {
3
4         int x;
5         int y;
6
7         x=3;
8         y=4;
9
10        if(x == 3){
11            if(y == 4){
12                print("nested");
13            }
14        }
15    }
16 }
```

./test-printid-simple.maze

```
1 class test {
2
3     void main(){
4         string x;
5         x = "Hello";
6     }
7
8 }
```

./fail-voidClassvar.maze

```
1 class test {
2     void x;
3     void main () {
4         int y;
5     }
6 }
```

./fail-undeclaredID.err

```
1 Fatal error: exception Failure("undeclared identifier y")
```

./test-return.maze

```
1 class test {
2
3     int computeValue(int x, int y){
4         return x + y;
5     }
6 }
```

```
5     }
6
7     void main(){
8
9         int a;
10        a = computeValue(10,5);
11        print(a);
12
13    }
14 }
```

./test-recursion.maze

```
1 class test {
2
3     int fib(int n){
4
5         if(n <= 1){
6             return n;
7         }
8         return fib(n-1) + fib(n-2);
9     }
10
11    void main() {
12        int n;
13        int answer;
14
15        n = 9;
16        answer = fib(n);
17        print(answer);
18    }
19
20 }
```

./fail-dupClassvar.err

```
1 Fatal error: exception Failure("duplicate variable x")
```

./test-printfloat.maze

```
1 class test {
2     void main () {
3         print(3.14);
4     }
5 }
```

./test-printid-bool.maze

```
1 class test {
2     void main() {
```



```
3     bool x;
4     x = false;
5     print(x);
6 }
7 }
```

./test-printid-string.maze

```
1 class test {
2
3     void main(){
4         string x;
5         x = "Hello";
6         print(x);
7     }
8
9 }
```

./test-subtraction.maze

```
1 class test {
2     void main () {
3
4         int x;
5         x = 3 - 7;
6
7         if(x==4){
8             print("correct");
9         }
10    }
11 }
```

./test-printid-simple.out

./test-classes.out

```
1 ok
```

./fail-binopOperands.err

```
1 Fatal error: exception Failure("operator + requires two ints or two floats")
```

./test-binop.out

```
1 4115
```

./test-ifstmt.out

```
1 correct
```

./fail-dupClassvar.maze

```
1 class test {
2     int x;
3     int x;
4     void main () {
5         int y;
6     }
7 }
```

./fail-dupFun.maze

```
1 class test {
2
3     void main () {}
4     void test () {
5         int y;
6     }
7
8     void test () {}
9 }
```

./fail-undeclaredID.maze

```
1 class test {
2     void main () {
3
4         if (y){
5             print("works");
6         }
7     }
8 }
```

./test-return.out

```
1 15
```

./test-classes.maze

```
1 class shape {
2
3 }
4
5
6 class test {
7     void main(){
8
9         print("ok");
10
11     }
12
13 }
```

./test-printid-char.maze

```
1 class test {
2     void main() {
3         char x;
4         x = 'c';
5         print(x);
6     }
7 }
```

./test-binop2.out

```
1 correct
2 correct
```

./test-binopmult.out

```
1 25
```

./fail-dupvar.err

```
1 Fatal error: exception Failure("duplicate variable x")
```

./test-fbinop.out

```
1 add
2 sub
3 mult
4 div
```

./test-ops.maze

```
1 class test {
2     void main () {
3
4         int x;
5         int y;
6         int z;
7         int a;
8
9         x = 3 + 7;
10        y = 10 - 2;
11        z = 3 * 2;
12        a = 50 / 10;
13
14        if(x==10){
15            print("add");
16        }
17
18        if(y==8){
19            print("sub");
```

```
20     }
21
22     if(z==6){
23         print("mult");
24     }
25
26     if(a==5){
27         print("div");
28     }
29
30 }
31 }
```

./test-printchar.maze

```
1 class test {
2     void main () {
3         print('a');
4     }
5 }
```

./test-ifstmt.maze

```
1 class test {
2     void main () {
3
4         if(true){
5             print("correct");
6         }
7         else{
8             print("incorrect");
9         }
10    }
11 }
```

./test-printint.out

```
1 42
```

./test-printint.maze

```
1 class test {
2     void main () {
3         print(42);
4     }
5 }
```

./fail-dupClass.maze

```
1 class test {
2     void main () {
```

```
3     int x;
4   }
5 }
6
7 class test {
8   void wag(){}
9 }
```

./test-printid-float.out

```
1 42.000000
```

./fail-voidvar.maze

```
1 class test {
2   void main () {
3     void x;
4   }
5 }
```

./test-printid-char.out

```
1 c
```

./test-func-call.maze

```
1 class test {
2
3   void dummy(){
4     int x;
5     x = 1;
6     print(x);
7   }
8
9   int main(){
10    dummy();
11  }
12 }
```

./test-printfloat.out

```
1 3.140000
```

./test-ifNested.out

```
1 nested
```

./test-binopmult.maze

```
1 class test {
2   void main(){
3     print(5 * 5);
4   }
```

```
5 }
```

./test-binop2.maze

```
1 class test {
2     void main(){
3
4         if(true && true){
5             print("correct");
6         }
7
8         if(false || true){
9             print("correct");
10        }
11
12    }
13 }
```

./fail-defprint.maze

```
1 class test {
2     void main () {
3         int x;
4     }
5
6     void print (string s){}
7 }
```

./test-func-call.out

```
1 1
```

./test-printchar.out

```
1 a
```

./test-printid-int.out

```
1 42
```

./fail-dupFormal.err

```
1 Fatal error: exception Failure("duplicate formal x in main")
```

./test-hello.maze

```
1 class test {
2     void main () {
3         print("Hello World");
4     }
5 }
```

./test-if2.out

```
1 else
```

```
./fail-dupFun.err
```

```
1 Fatal error: exception Failure("duplicate function test")
```

```
./test-bool.maze
```

```
1 class test {
2   void main () {
3       print(true);
4       print(false);
5       print(1==1);
6       print(2==3);
7   }
8 }
```

```
./fail-voidFormal.maze
```

```
1 class test {
2   void main (void x) {
3       print("Hello World");
4   }
5 }
```

```
./test-fbinop.maze
```

```
1 class test {
2   void main () {
3
4       float x;
5       x = 3.2 + 4.5;
6
7       if(x == 7.700000){
8           print("add");
9       }
10
11      x = 5.4 - 2.2;
12      if(x == 3.200000){
13          print("sub");
14      }
15
16      x = 4.2 * 6.8;
17      if(x == 28.560000){
18          print("mult");
19      }
20
21      x = 4.6 / 2.3;
22      if(x == 2.000000){
23          print("div");
```

```
24     }
25     }
26 }
```

./test-addition.out

```
1 correct
```

./fail-voidFormal.err

```
1 Fatal error: exception Failure("illegal void formal x in main")
```

./fail-dupFormal.maze

```
1 class test {
2     void main (int x, int x) {
3         print("Hello World");
4     }
5 }
```

./test-equality.out

```
1 less
2 greater
3 equal
4 not equal
5 leq
6 geq
```

./test-ops.out

```
1 add
2 sub
3 mult
4 div
```

./test-whilestmt.out

```
1 while loop
2 while loop
3 while loop
4 while loop
5 while loop
```

./fail-dupClass.err

```
1 Fatal error: exception Failure("Duplicate class name test")
```

./test-gcd.maze

```
1 class test {
2     void main () {
3         int x;
4         int y;
```



```
5     x = 15;
6     y = 20;
7
8     if (x == 0) {
9         print(x);
10    }
11
12    while(x != y)
13    {
14        if ( x > y) {
15            x= x-y;
16        } else {
17            y= y-x;
18        }
19    }
20    print(x);
21 }
22 }
```

./test-printid-string.out

1 Hello