MAZE

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

CONTENTS

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 "./t 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:

int	*while*	*null*
char	*bool*	*true*
return	*print*	*false*
if	*else*	*string*
void	*class*	

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.

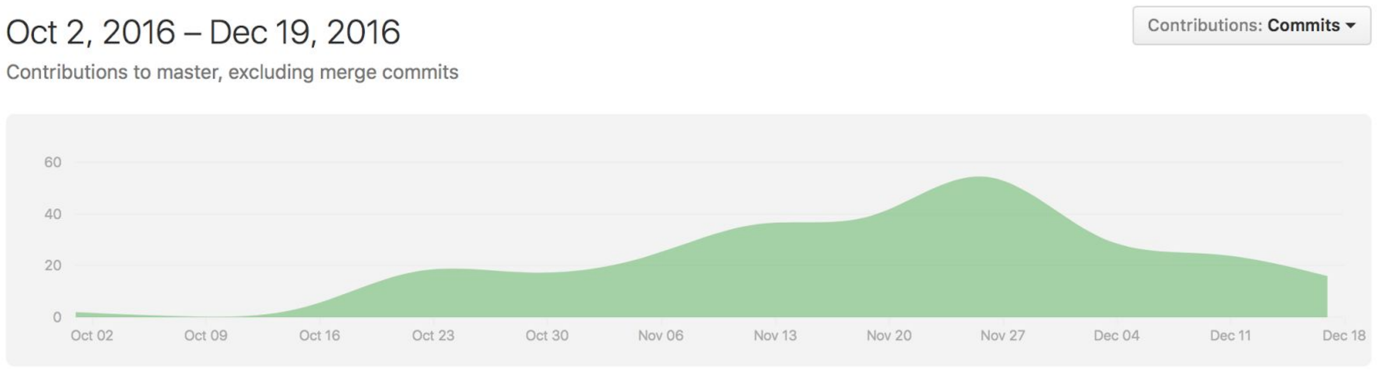
Team Roles

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



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.

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

```

```
    added recursive fib function test

commit d2c0cad321b249333acd8f9a78017bb53fc7c3cd
Author: AFreemantle <asf2161@columbia.edu>
Date:   Mon Dec 19 12:22:22 2016 -0500

    final update to README

commit eec85dfdc221e014fd358488a869c32abaa67eae
Author: Alexander Brown <aab2212@columbia.edu>
Date:   Mon Dec 19 05:30:05 2016 -0500

    final commit.

commit 411a5163717bac7bb996c8b026af89be92575d5e
Author: Alexander Brown <aab2212@columbia.edu>
Date:   Mon Dec 19 05:02:50 2016 -0500

    cruft n comments

commit 559a3e725a03197a32df590717ad0d900b2a0f9f
Merge: bce23f7 4d0ab45
Author: Michelle Navarro <mn2614@columbia.edu>
Date:   Mon Dec 19 00:53:13 2016 -0800

    Merge pull request #50 from AFreemantle/analyzer

    forgot to commit this change last time

commit bce23f7a49bfe7b9200aae1123000580ee93fa00
Merge: 514c21b 52fb001
Author: Michelle Navarro <mn2614@columbia.edu>
Date:   Mon Dec 19 00:48:56 2016 -0800

    Merge pull request #49 from AFreemantle/analyzer

    Analyzer

```

```
commit 4d0ab45a2183bffbb85ff787df18e0b0a1ae4629
Author: Alexander Brown <aab2212@columbia.edu>
Date:   Mon Dec 19 03:33:58 2016 -0500

    forgot to commit this change last time

commit 52fb001d279b0814a7ea0865e91ae633b37402bf
Author: Alexander Brown <aab2212@columbia.edu>
Date:   Mon Dec 19 03:28:05 2016 -0500

    check for main actually works now

commit 60940d16816c1444f4f6756f2f0859a27200a646
Author: Alexander Brown <aab2212@columbia.edu>
Date:   Mon Dec 19 02:47:59 2016 -0500

    checks for exactly 1 main fcn

commit 7787a7cdbda21bff214a44021e1ca14e7dfc01c7
Author: Alexander Brown <aab2212@columbia.edu>
Date:   Mon Dec 19 02:10:09 2016 -0500

    finished all type checking, tests to come

commit 4ae237f08e5e98c5a9a2e5fd7ceb66fbde0efe2d
Author: Alexander Brown <aab2212@columbia.edu>
Date:   Sun Dec 18 22:24:33 2016 -0500

    saving changes before branch switch

commit 05c6e8174a9c32f6aa7af3edc7d58a762631e51e
Author: Alexander Brown <aab2212@columbia.edu>
Date:   Sun Dec 18 21:26:12 2016 -0500

    now checks assignment exprs to ensure left side same type as right side

commit 99fbf9a4d278e3543ca1aff9605de2a4edc292eb
```

```
Author: Alexander Brown <aab2212@columbia.edu>
Date:   Sun Dec 18 21:20:24 2016 -0500

    check assign fcn for comparing rt and lt

commit 0370ace8eb3f4fc001005623a24d6cfc3b13294c
Author: Alexander Brown <aab2212@columbia.edu>
Date:   Sun Dec 18 21:16:27 2016 -0500

    clarified some binop type checking

commit da074a1d659fa4e75df51cd2864f6c910946b87a
Author: Alexander Brown <aab2212@columbia.edu>
Date:   Sun Dec 18 20:53:23 2016 -0500

    test binop with diff operand types

commit 453441d42cc0806a71f0d295c6b5943e63aa1eda
Author: Alexander Brown <aab2212@columbia.edu>
Date:   Sun Dec 18 20:19:01 2016 -0500

    Better binop checking, error message

commit 9f68c92dc99f042f08e1146376bd51580912ed54
Author: Alexander Brown <aab2212@columbia.edu>
Date:   Sun Dec 18 19:48:07 2016 -0500

    undeclared ID test

commit aa4b06f2b1871abeea758d2cbec7815c1d483115
Author: Alexander Brown <aab2212@columbia.edu>
Date:   Sun Dec 18 19:42:12 2016 -0500

    fixed uop expr checking, passes all tests

commit 6b6b46f44fdf7f8993e5687e39f07b6998ae8f84
Author: Alexander Brown <aab2212@columbia.edu>
Date:   Sun Dec 18 03:58:36 2016 -0500
```

```

    small cruft

commit 6ed6f2a9d19bb0c3851564c1b038d1caf723982f
Author: Alexander Brown <aab2212@columbia.edu>
Date:   Sun Dec 18 03:56:54 2016 -0500

    checks almost everything, look at BINOP, UNOP and FIX

commit fbb61eb4156afb850b4cd5bc839f9c962df8ed80
Author: Alexander Brown <aab2212@columbia.edu>
Date:   Sun Dec 18 00:06:47 2016 -0500

    type of identifier fcn

commit 943552bb2783084f9cff68e1e8d91f5a5940b6cb
Author: Alexander Brown <aab2212@columbia.edu>
Date:   Sun Dec 18 00:01:44 2016 -0500

    Full symbol map per class, helper fcns

commit 7acd7e359e54467095e7af9f82affd6f67fc7599
Author: Alexander Brown <aab2212@columbia.edu>
Date:   Sat Dec 17 23:53:42 2016 -0500

    beginning symbol checking, helper fcns

commit 2a839594dc366c0ce4db175a0be967ef7203264d
Author: Alexander Brown <aab2212@columbia.edu>
Date:   Sat Dec 17 22:26:46 2016 -0500

    duplicate formal test

commit 317059f1300a12680d69a9815bd7b9db89b14ace
Author: Alexander Brown <aab2212@columbia.edu>
Date:   Sat Dec 17 22:24:14 2016 -0500

    checks for duplicate formals
```

```

commit 2a5fc2eea8259a9ea2dd8394bbf9c1510fd1f737
Author: Alexander Brown <aab2212@columbia.edu>
Date:   Sat Dec 17 22:07:24 2016 -0500

    void formal test

commit b44b6c06b36067b067a8a9e02866134b5458402b
Author: Alexander Brown <aab2212@columbia.edu>
Date:   Sat Dec 17 22:01:35 2016 -0500

    Checks for void formals

commit 69d565c710eed94fb09d333c3ef1598f797d3d47
Author: Alexander Brown <aab2212@columbia.edu>
Date:   Sat Dec 17 19:37:37 2016 -0500

    started checkin func formals

commit 86933358cf63654397a1acd4ad6200cc5c925788
Author: Alexander Brown <aab2212@columbia.edu>
Date:   Sat Dec 17 19:20:24 2016 -0500

    cruft

commit 44dc464d4807575b8e5120a272e40583f3912e1a
Author: Alexander Brown <aab2212@columbia.edu>
Date:   Sat Dec 17 19:18:12 2016 -0500

    print_fname, function_decl working

commit f14dc08d0866f2163d0ab605fd099e84892cfe71
Author: Alexander Brown <aab2212@columbia.edu>
Date:   Sat Dec 17 19:03:08 2016 -0500

    fixed function map stuff

commit 514c21b0d00851952a024fadbae7de1318571507
```

```
Author: Michelle Navarro <mn2614@columbia.edu>
Date:   Sat Dec 17 18:49:45 2016 -0500

    shift/reduce err fix

commit 5da31dcc0d8f196692e49946e251dac8d898ceb5
Merge: 2b2a6b7 f41580b
Author: Alexander Brown <aab2212@columbia.edu>
Date:   Sat Dec 17 17:50:35 2016 -0500

    Merge branch 'master' of https://github.com/AFreemantle/maze into analyzer

commit f41580b813c91af3ad8239bbb992d9427dbd8255
Author: AFreemantle <asf2161@columbia.edu>
Date:   Sat Dec 17 17:35:11 2016 -0500

    fixed float printing test

commit 713b0a993e7098383ad172c1de79db9f79466737
Author: AFreemantle <asf2161@columbia.edu>
Date:   Sat Dec 17 17:22:17 2016 -0500

    removed extra set of brackets

commit 2d8900f9715897c6c00b420d5149bcc5eff8787f
Author: AFreemantle <asf2161@columbia.edu>
Date:   Sat Dec 17 17:06:15 2016 -0500

    floats print now!

commit 415e2a9cef2e176d7bd7a5aa0b8dda2738322103
Author: Lindsay <ls3245@columbia.edu>
Date:   Fri Dec 16 01:57:25 2016 -0500

    can create multiple classes

commit 3e091d1b4d9550bbcbaed8fb9ecb9bcb613fa25c
```

```
Author: Lindsay <ls3245@columbia.edu>
Date:   Fri Dec 16 01:57:02 2016 -0500

    added Object Create

commit 0215fe2e68657619257bdccab6e6eac5a7dcf384
Author: Lindsay <ls3245@columbia.edu>
Date:   Thu Dec 15 20:52:16 2016 -0500

    test for return works

commit 8df05b018277b21d15d239a6678160dc4d6c687b
Author: Lindsay <ls3245@columbia.edu>
Date:   Thu Dec 15 20:51:04 2016 -0500

    added objecttype, needed for multiple classes and new

commit 9ec0745fd525c5b9a489f5cef3d8b564e4f16286
Author: Lindsay <ls3245@columbia.edu>
Date:   Thu Dec 15 19:02:45 2016 -0500

    tested nested if stmts

commit 464c59d335e56dd95fc1928b820eb3b53713ab18
Author: Lindsay <ls3245@columbia.edu>
Date:   Thu Dec 15 19:02:20 2016 -0500

    tested empty if stmt

commit c318fd13043569d55329868666e4376c62840050
Author: Lindsay <ls3245@columbia.edu>
Date:   Thu Dec 15 19:01:45 2016 -0500

    multiple else if's

commit 2a26bb9581f464edab9eba2e5c40bfda41c91450
Author: AFreemantle <asf2161@columbia.edu>
Date:   Thu Dec 15 18:01:57 2016 -0500
```

```

    function calling working now too

commit 8e03c747a3683d0c10b3707cffb49c2668996bc7
Author: AFreemantle <asf2161@columbia.edu>
Date:   Thu Dec 15 17:56:42 2016 -0500

    removed gcd from top dir

commit 9e760b70f5d23fecb692ada9394a681274c785b5
Author: AFreemantle <asf2161@columbia.edu>
Date:   Thu Dec 15 17:54:31 2016 -0500

    gcd works

commit 5e43cb5918ade19e7460a41f7e789176448cc653
Author: Lindsay <ls3245@columbia.edu>
Date:   Thu Dec 15 17:00:02 2016 -0500

    added tests for float add, sub, mult, div

commit 0c085b2bd4d83520a96285691b5744e3aa182e6e
Author: Lindsay <ls3245@columbia.edu>
Date:   Thu Dec 15 16:11:04 2016 -0500

    tests addition of floats

commit 85f83f5c17f810c00500ade1c4884042c48d28a2
Author: Lindsay <ls3245@columbia.edu>
Date:   Thu Dec 15 16:10:43 2016 -0500

    added binop for floats

commit 8601efc1a4b77c7bdbfc6fe7150b302d33efb5a8
Author: Lindsay <ls3245@columbia.edu>
Date:   Thu Dec 15 16:10:21 2016 -0500

    added float to scanner
```

```

commit 71b31bda120efceb467501eaed90acfff30907c1
Author: Michelle Navarro <mn2614@columbia.edu>
Date:   Thu Dec 8 02:44:42 2016 -0500

    Giving Ids to print actually works now

commit 35bc4df4c4f561019c9591d6de10d052e556513f
Author: Michelle Navarro <mn2614@columbia.edu>
Date:   Thu Dec 8 02:32:51 2016 -0500

    figured out what it thinks the types are weeeeeee

commit f9a95fe6e78aab2e3b093ae8d3d2f82090feed28
Author: Michelle Navarro <mn2614@columbia.edu>
Date:   Thu Dec 8 00:56:45 2016 -0500

    HAHAHAHA CLOSER TO THE SOLUTION

commit 0c6f919e1d9e8ecbf60a440e10d27e5c445338a0
Merge: f02cff0 17e617e
Author: Alexander Freemantle <asf2161@columbia.edu>
Date:   Wed Dec 7 22:46:29 2016 -0500

    Merge pull request #46 from AFreemantle/print

    test for declaring then printing a bool

commit 17e617e6cb71c864eb0b3469cbe8bb0275ad6514
Author: AFreemantle <asf2161@columbia.edu>
Date:   Wed Dec 7 22:42:24 2016 -0500

    test for declaring then printing a bool

commit f02cff0f13970fbad7bf0d4e1de1735be1010466
Merge: bcc8595 60de00d
Author: Alexander Freemantle <asf2161@columbia.edu>
Date:   Wed Dec 7 22:37:30 2016 -0500
```

```

    Merge pull request #45 from AFreemantle/functions

    added tests for assignment then print for int and char

commit 60de00dbad8a376f20731a2ac8b83ee61d44ce13
Author: AFreemantle <asf2161@columbia.edu>
Date:   Wed Dec 7 22:37:07 2016 -0500

    added tests for assignment then print for int and char

commit bcc85950085fc519532903243221d4b855bc89e7
Author: Michelle Navarro <mn2614@columbia.edu>
Date:   Wed Dec 7 22:34:25 2016 -0500

    cruft

commit 98a4e3ad9cf9fb064b13385cf58819fe7088efe2
Author: AFreemantle <asf2161@columbia.edu>
Date:   Wed Dec 7 22:13:49 2016 -0500

    test files for printing strings after assignment

commit 35359c697cebc7d9fcec8e5a68c1706822ab1ae5
Merge: f7d4c25 ffcd6d3
Author: Alexander Freemantle <asf2161@columbia.edu>
Date:   Wed Dec 7 00:08:32 2016 -0500

    Merge pull request #44 from AFreemantle/functions

    missed one string args in test-binop

commit ffcd6d3e5be638ae2f8836a2cffa8627dab5ae77
Author: AFreemantle <asf2161@columbia.edu>
Date:   Wed Dec 7 00:06:45 2016 -0500

    missed one string args in test-binop

```

```
commit f7d4c259c3dae6ca3688a10dec741dff2acdcb5a
Merge: d17a44e 133e87b
Author: Michelle Navarro <mn2614@columbia.edu>
Date:   Tue Dec 6 23:57:27 2016 -0500

    :Merge branch 'master' of github.com:AFreemantle/maze

commit d17a44ef115dbf90dfc43196de03b4e0ee159906
Author: Michelle Navarro <mn2614@columbia.edu>
Date:   Tue Dec 6 23:57:11 2016 -0500

    took string useless string args out

commit 133e87bdcef4517d01af76bfd5ccd9d62eb9dc74
Merge: 01d5648 9efe3f2
Author: Alexander Freemantle <asf2161@columbia.edu>
Date:   Tue Dec 6 23:50:16 2016 -0500

    Merge pull request #43 from AFreemantle/print

    cruft from failed string printing

commit 9efe3f2212e987810e3c3d20a78968626817ef19
Author: AFreemantle <asf2161@columbia.edu>
Date:   Tue Dec 6 23:49:40 2016 -0500

    cruft from failed string printing

commit 01d5648066cbded07e021b85aaa6a2149dab85ca
Merge: 9d50677 9fbb3d8
Author: Alexander Freemantle <asf2161@columbia.edu>
Date:   Tue Dec 6 23:43:04 2016 -0500

    Merge pull request #42 from AFreemantle/print

    new test for assignment

commit 9fbb3d8ef318f6564768c7b07bd1873293c990ab
```

```
Author: AFreemantle <asf2161@columbia.edu>
Date:   Tue Dec 6 23:42:12 2016 -0500

    new test for assignment

commit 9d50677ce7feaa50abd091ef3309d7ad65d396bd
Merge: 4df4da3 c13d496
Author: Alexander Freemantle <asf2161@columbia.edu>
Date:   Tue Dec 6 23:40:02 2016 -0500

    Merge pull request #41 from AFreemantle/print

    jokes, assignment was the issue, working now (strings)

commit c13d4968c78db4e06690e7f91be88b2c7112c4b2
Author: AFreemantle <asf2161@columbia.edu>
Date:   Tue Dec 6 23:39:16 2016 -0500

    jokes, assignment was the issue, working now (strings)

commit 4df4da342a76531579547cac0c1b07f584ce2dbd
Merge: 311d528 33e4f78
Author: Alexander Freemantle <asf2161@columbia.edu>
Date:   Tue Dec 6 23:16:30 2016 -0500

    Merge pull request #40 from AFreemantle/print

    Print

commit 33e4f7834ccfd6f9ad3ff4e75f1def0e5a3ea443
Author: AFreemantle <asf2161@columbia.edu>
Date:   Tue Dec 6 23:15:40 2016 -0500

    finally got printing ids to work!

commit 290daae688a6b4858accdd055dd305569e1cc8ba
Author: AFreemantle <asf2161@columbia.edu>
Date:   Tue Dec 6 22:58:23 2016 -0500
```

```

    small test function for printing ids

commit 311d52812a2aeb2b6b6c63d36ec3b8472f72fcf1
Merge: 562097a ffd60b9
Author: Alexander Freemantle <asf2161@columbia.edu>
Date:   Tue Dec 6 22:31:01 2016 -0500

    Merge pull request #39 from AFreemantle/print

    Print

commit ffd60b904213c5cc1d35e375248eb3539ce5ca16
Author: AFreemantle <asf2161@columbia.edu>
Date:   Tue Dec 6 22:28:14 2016 -0500

    while loops work, debugged gcd

commit 972bf217b956d4202195c5911891cb059d534424
Merge: dd23f4b 562097a
Author: AFreemantle <asf2161@columbia.edu>
Date:   Tue Dec 6 22:05:33 2016 -0500

    Merge branch 'master' of https://github.com/AFreemantle/maze into print

commit 562097a9b46a0434dc02dbbcc6468ec2ec25f8da
Merge: 1b743bd 73affab
Author: Michelle Navarro <mn2614@columbia.edu>
Date:   Tue Dec 6 22:05:21 2016 -0500

    Merge pull request #38 from AFreemantle/gcd

    Gcd

commit 2b2a6b74b164ae5f49802e6ba544162f4fa62f64
Author: Alexander Brown <aab2212@columbia.edu>
Date:   Sun Dec 4 18:46:12 2016 -0500
```

```

    idk

commit 73affab0087c44215ab975654b59dba906bdc1b6
Author: Michelle Navarro <mn2614@columbia.edu>
Date:   Sun Dec 4 18:39:05 2016 -0500

    Still wrong

commit 773c70af33e2c7c2742af9010ae16aa7589b530e
Author: Michelle Navarro <mn2614@columbia.edu>
Date:   Sun Dec 4 17:21:01 2016 -0500

    wrong

commit 20b16ec28ad0a6e7cca29205b544d85b98944941
Author: Michelle Navarro <mn2614@columbia.edu>
Date:   Sun Dec 4 16:13:59 2016 -0500

    Renamed test to gcd.maze

commit 1b743bdbfa7c2a41dd1edb7e13d220590a60e27b
Merge: d12690f 7d5c164
Author: Michelle Navarro <mn2614@columbia.edu>
Date:   Sun Dec 4 16:11:36 2016 -0500

    Merge branch 'master' of github.com:AFreemantle/maze

commit 7d5c16439262ade2e15cc849b43590d49192e27c
Merge: 15f08d3 f805ca4
Author: Michelle Navarro <mn2614@columbia.edu>
Date:   Sun Dec 4 16:12:13 2016 -0500

    Merge pull request #37 from AFreemantle/analyzer

    Analyzer

commit d12690f096bf0102cfe1d510f7691935468ea30c
```

```
Author: Michelle Navarro <mn2614@columbia.edu>
Date:   Sun Dec 4 16:11:24 2016 -0500

    Fields test is actually gcd test now

commit 24910c67737a104796983d76127ea29ba5c22262
Merge: f805ca4 15f08d3
Author: Alexander Brown <aab2212@columbia.edu>
Date:   Sun Dec 4 15:57:16 2016 -0500

    Merge branch 'master' of https://github.com/AFreemantle/maze into analyzer

commit f805ca4c701c5de9e4e3e9fb36203e3ad77800e5
Author: Alexander Brown <aab2212@columbia.edu>
Date:   Sun Dec 4 15:55:29 2016 -0500

    removes .ll files in /tests

commit 6dc9800a6b5760efeb3f2d77bb5d54cf055b4605
Author: Alexander Brown <aab2212@columbia.edu>
Date:   Sun Dec 4 15:40:21 2016 -0500

    cruft

commit dd23f4b4bac549dd4d77c5375c6624e4ad29af91
Author: AFreemantle <asf2161@columbia.edu>
Date:   Sun Dec 4 14:39:33 2016 -0500

    subtraction tests working now

commit 15f08d3a54e9d5a1beb4a283e18478dd66edd3c0
Merge: 16b1496 9538ce6
Author: Michelle Navarro <mn2614@columbia.edu>
Date:   Sat Dec 3 23:39:22 2016 -0500

    Merge pull request #36 from AFreemantle/analyzer

    Analyzer

commit 9538ce6654cf2a6693414db03bd6c346bc7f2cc0
Merge: e7215f6 16b1496
Author: Alexander Brown <aab2212@columbia.edu>
Date:   Sat Dec 3 23:37:01 2016 -0500

    analyzer2 -> analyzer

commit e7215f644f99e1a3c49ade71dd80106f3ed5b1ff
Author: Alexander Brown <aab2212@columbia.edu>
Date:   Sat Dec 3 23:32:25 2016 -0500

    this will allow me to pull

commit 16b1496096427e083cf67254a47e28d2b4c637d5
Author: Michelle Navarro <mn2614@columbia.edu>
Date:   Sat Dec 3 23:28:58 2016 -0500

    Renamed analyzer2 to just analyzer

commit 3705743823b095865fc92fcdfd74f111e53ef37b
Author: Michelle Navarro <mn2614@columbia.edu>
Date:   Sat Dec 3 23:23:59 2016 -0500

    got rid of the other analyzer

commit 691718ca693e80a618da62bd02926aca1fbf7838
Merge: 226a581 7c54253
Author: Alexander Brown <aab2212@columbia.edu>
Date:   Sat Dec 3 23:15:51 2016 -0500

    Merge branch 'master' of https://github.com/AFreemantle/maze into analyzer

commit 226a5811491c8cc32755fde6457f526b87d7901f
Author: Alexander Brown <aab2212@columbia.edu>
Date:   Sat Dec 3 23:13:22 2016 -0500
```

```

    duplicate function test

commit 7c542531aa490b00616aa531981a1ed673f9e041
Merge: c58e56e e52dd6c
Author: Michelle Navarro <mn2614@columbia.edu>
Date:   Sat Dec 3 23:10:32 2016 -0500

    Merge pull request #35 from AFreemantle/exc

    Exc

commit e52dd6c9489e474a98231c4ea3f51cc994ec3f7e
Author: Michelle Navarro <mn2614@columbia.edu>
Date:   Sat Dec 3 23:09:42 2016 -0500

    small && addition

commit 42a49d73934bfafc07a7aba1b06e9ecca312f188
Author: Michelle Navarro <mn2614@columbia.edu>
Date:   Sat Dec 3 23:05:20 2016 -0500

    cruft

commit 0e6e20cb58313945485c379ec3626d3020b98a04
Author: Alexander Brown <aab2212@columbia.edu>
Date:   Sat Dec 3 22:59:14 2016 -0500

    test for void class var

commit 3e1380f15885bb634bac1aa656bf75ff4d3a985f
Author: Alexander Brown <aab2212@columbia.edu>
Date:   Sat Dec 3 22:51:58 2016 -0500

    test for duplicate class variables

commit 098b15c8b67c542522441d2d48fb1c15fc56f131
Author: Alexander Brown <aab2212@columbia.edu>
```

```
Date:   Sat Dec 3 22:47:11 2016 -0500

    Checks for duplicate/void class variables

commit 1593a899c66a12bbfa8a0b3692a912647106c850
Author: Michelle Navarro <mn2614@columbia.edu>
Date:   Sat Dec 3 22:44:18 2016 -0500

    Fixing more tests & bash script stuff

commit efea7a430cef1cb480ed91c9ff41b264434828ed
Author: Alexander Brown <aab2212@columbia.edu>
Date:   Sat Dec 3 22:29:45 2016 -0500

    Test for defining print

commit c58e56e877fdf0f4f6197037707674c93378791c
Merge: d96b814 00c9f5e
Author: Michelle Navarro <mn2614@columbia.edu>
Date:   Sat Dec 3 22:18:05 2016 -0500

    Merge pull request #34 from AFreemantle/analyzer

    Analyzer

commit 00c9f5e651559acf2edb2cbbb470b3072ac37db3
Author: Alexander Brown <aab2212@columbia.edu>
Date:   Sat Dec 3 22:17:27 2016 -0500

    Duplicate local variables test

commit 40dfa8a250cef68e55958faf1d2d54d3d0cc0f03
Author: Michelle Navarro <mn2614@columbia.edu>
Date:   Sat Dec 3 21:35:25 2016 -0500

    removed extra semant.ml file

commit bf1267c117fa13e220eb4225f4f89eb832838f40
```

```
Author: Michelle Navarro <mn2614@columbia.edu>
Date:   Sat Dec 3 21:32:58 2016 -0500

    Fixed script for failures

commit 71a5751bb63a70efbbd8c019d501067c73ae1972
Author: Alexander Brown <aab2212@columbia.edu>
Date:   Sat Dec 3 21:29:35 2016 -0500

    Duplicate class name test

commit be59bb2a99b2959e5d1e2e0008678b394244efdc
Author: Alexander Brown <aab2212@columbia.edu>
Date:   Sat Dec 3 21:24:22 2016 -0500

    Fixed stuff I accidently broke

commit 117f0968a2c939030f0733ff3612528dfb175f3b
Merge: 34e39b4 d96b814
Author: Alexander Brown <aab2212@columbia.edu>
Date:   Sat Dec 3 20:55:21 2016 -0500

    Fixed merge conflicts

commit 34e39b408c825a8fdb5cdb61a77975672b7f67ee
Author: Alexander Brown <aab2212@columbia.edu>
Date:   Sat Dec 3 20:52:42 2016 -0500

    Checks for duplicate classes

commit 8746d48dcb516b2246640f715af6712919d7239d
Author: Alexander Brown <aab2212@columbia.edu>
Date:   Sat Dec 3 20:48:09 2016 -0500

    Void locals fail test

commit d96b814fb71ddf96de20299cb9e7b9d2e92144a2
Author: Michelle Navarro <mn2614@columbia.edu>
```

```
Date:   Sat Dec 3 20:46:23 2016 -0500

    going to start dealing with fails

commit b5ce4ca0a001db73795e8b15972d804fa4c37813
Author: Alexander Brown <aab2212@columbia.edu>
Date:   Sat Dec 3 20:40:31 2016 -0500

    Void locals analyzer test

commit 06c66da5ae75032f8cc0aa606f4ab3e11f5141c2
Author: Michelle Navarro <mn2614@columbia.edu>
Date:   Sat Dec 3 20:28:15 2016 -0500

    found stray print stmt

commit 04a8c353a43699c5279683efdfd23d938aeff6c8
Merge: 18e047b f820347
Author: Alexander Brown <aab2212@columbia.edu>
Date:   Sat Dec 3 19:52:19 2016 -0500

    Merge branch 'master' of https://github.com/AFreemantle/maze into analyzer

commit f8203472148294e80b5b7f0d62acc7a3967dd5d9
Merge: 49790c9 340b752
Author: Michelle Navarro <mn2614@columbia.edu>
Date:   Sat Dec 3 19:49:16 2016 -0500

    Merge pull request #33 from AFreemantle/contrflow

    Contrflow

commit 340b752b7d19c08fb29232e584dd83c0947d0ab1
Author: Lindsay <ls3245@columbia.edu>
Date:   Sat Dec 3 19:45:09 2016 -0500

    tests AND and OR
```

```

commit 01fef156e7dd8d8a45b948199f6a72d59f0948aa
Author: Lindsay <ls3245@columbia.edu>
Date:   Sat Dec 3 19:44:46 2016 -0500

    tests neg and not

commit 49790c9cd324997497aad96825cca9ba28d3da86
Author: Lindsay <ls3245@columbia.edu>
Date:   Sat Dec 3 19:32:12 2016 -0500

    tested not unop

commit 5e24c9fb172f3185642bdab736f6ec5ade97d8bc
Merge: bfed1b2 37f306d
Author: Alexander Freemantle <asf2161@columbia.edu>
Date:   Sat Dec 3 19:01:43 2016 -0500

    Merge pull request #32 from AFreemantle/print

    Print

commit 37f306d0058ee6bdce4e36e4362dc03a2ac2fb34
Author: AFreemantle <asf2161@columbia.edu>
Date:   Sat Dec 3 18:34:18 2016 -0500

    printing booleans correctly now

commit 075b3d576d73967248798f018b61c09baf4009dd
Author: AFreemantle <asf2161@columbia.edu>
Date:   Sat Dec 3 17:51:53 2016 -0500

    added tests for printing multiplication of two ints

commit 27c4e4c70e012f036a19b30169fadb37c663ea57
Author: AFreemantle <asf2161@columbia.edu>
Date:   Sat Dec 3 17:47:54 2016 -0500

```

```
    Binops printing correctly now

commit 34c40c3f1f4f9b8db6ed2889d8952e65ca660dd6
Author: AFreemantle <asf2161@columbia.edu>
Date:   Sat Dec 3 17:41:07 2016 -0500

    trying to implement Binop printing

commit 6cbffa4669986eb17eb9db9b6b09297e5ff0758d
Author: Lindsay <ls3245@columbia.edu>
Date:   Fri Dec 2 22:11:44 2016 -0500

    tests add, sub, mult, and div

commit 20330b3dae24bf80ed8f9884eac6940a4abe4409
Author: Lindsay <ls3245@columbia.edu>
Date:   Fri Dec 2 22:01:46 2016 -0500

    while loop works

commit de812735b1c78d0440c3d09e5d96df3b551e502c
Author: Lindsay <ls3245@columbia.edu>
Date:   Fri Dec 2 22:01:24 2016 -0500

    addition works

commit 65fc54f9daad9aeaa99a7b7dba52deaae53285a7
Author: Lindsay <ls3245@columbia.edu>
Date:   Fri Dec 2 20:59:33 2016 -0500

    added equality tests, lt, gt, leq, geq, eq, neq

commit 255ddc7f2ee9227737a7c1df3bffab0a9d4ff017
Author: Lindsay <ls3245@columbia.edu>
Date:   Fri Dec 2 20:58:31 2016 -0500

    still doesnt work

```

```
commit 0385b3c8bfc7a522bda3eb572cb765d8816dc64d
Merge: 3645e3b bd9cce7
Author: Michelle Navarro <mn2614@columbia.edu>
Date:   Thu Dec 1 19:19:00 2016 -0500

    Merge pull request #31 from AFreemantle/binop

    added binop expr handling straight from microc

commit 3645e3b380ca753931ca2d6cfad04a97fe6d9571
Author: Lindsay <ls3245@columbia.edu>
Date:   Thu Dec 1 19:12:48 2016 -0500

    tests

commit 738af72a7b019063be819ee62f0742aaa8c9f071
Author: Lindsay <ls3245@columbia.edu>
Date:   Thu Dec 1 19:11:07 2016 -0500

    added float/char printing

commit bfed1b2c4b1736b9dd78054480b0715e86e322e8
Merge: 899a9d4 9efdcdc
Author: Michelle Navarro <mn2614@columbia.edu>
Date:   Thu Dec 1 00:06:39 2016 -0500

    Merge pull request #28 from AFreemantle/inttest

    Added print int test

commit 9efdcdc9ce04082c0eaa2e3fcb10216395c1064b
Author: Michelle Navarro <mn2614@columbia.edu>
Date:   Wed Nov 30 23:56:00 2016 -0500

    Added print int test

commit bd9cce7d225d2503ea2e4b79f9bd2636ac710069
Author: AFreemantle <asf2161@columbia.edu>
```

```
Date:   Wed Nov 30 23:43:59 2016 -0500

    added binop expr handling straight from microc

commit 899a9d441d5b9393abdf6b9bdee9bad29d6d6eb2
Author: AFreemantle <asf2161@columbia.edu>
Date:   Wed Nov 30 23:30:16 2016 -0500

    eliminated first warning

commit 1fb8559d6830955ccfb64936f0295e4a5bd1e1f0
Author: Lindsay <ls3245@columbia.edu>
Date:   Wed Nov 30 23:26:53 2016 -0500

    added Float,Char,Null pattern matching

commit a0a61789d104f8d18efca3385d22441537574be6
Merge: 6e4e4af dc1bce2
Author: Michelle Navarro <mn2614@columbia.edu>
Date:   Wed Nov 30 22:34:32 2016 -0500

    Merge pull request #27 from AFreemantle/print

    printing ints and strings now

commit dc1bce2ab43cd900dea5b6339ea1d5ba39d63d57
Author: AFreemantle <asf2161@columbia.edu>
Date:   Wed Nov 30 22:30:37 2016 -0500

    printing ints and strings now

commit 6e4e4afcb0a1b2b4294425c60c847189281f6de7
Author: Michelle Navarro <mn2614@columbia.edu>
Date:   Wed Nov 30 22:06:06 2016 -0500

    starting excpetions

commit d112208c59d56527946b6f8bf071fd6c876a3d77
```

```
Author: Alexander Brown <aab2212@columbia.edu>
Date:   Wed Nov 30 21:52:31 2016 -0500

    raises invalid arg string error

commit 8477f21c957293032794b6ba8292b096cb55f305
Author: Michelle Navarro <mn2614@columbia.edu>
Date:   Wed Nov 30 20:30:06 2016 -0500

    makefile removes .ll files in tests dir now

commit fa7fca0cf4fe5e664cd69506d1e0767eaf99874f
Merge: 7bded1b 3b5da29
Author: Michelle Navarro <mn2614@columbia.edu>
Date:   Wed Nov 30 20:17:45 2016 -0500

    Merge pull request #26 from AFreemantle/cleantests

    Cleaned test output

commit 3b5da290feb4d1f43187b1bdd102ddbebe90a6d6
Author: Michelle Navarro <mn2614@columbia.edu>
Date:   Wed Nov 30 20:16:23 2016 -0500

    Cleaned test output

commit 7bded1b1ca3326b7ead34bb6129b4ffa35d93fdc
Author: Michelle Navarro <mn2614@columbia.edu>
Date:   Wed Nov 30 16:22:14 2016 -0500

    Passes Hello World but output is ugly

commit 06db327c3d5b06ba230134c17833e2c38ed2f1db
Merge: 8c1314a 3254f5b
Author: Michelle Navarro <mn2614@columbia.edu>
Date:   Wed Nov 30 02:57:38 2016 -0500

    Merge pull request #25 from AFreemantle/codegen

    Codegen

commit 3254f5b4456c399738a99b9ce4e2a815a0ce52b4
Author: Michelle Navarro <mn2614@columbia.edu>
Date:   Wed Nov 30 02:56:23 2016 -0500

    cruft

commit f3b9370c29aef2efa1a7717acb45cee115f9cbef
Author: Michelle Navarro <mn2614@columbia.edu>
Date:   Wed Nov 30 02:43:29 2016 -0500

    updated ext

commit 96a51e85254a1300ab7e74ebf6958cc48ea079df
Merge: 20340a3 4d64af6
Author: Michelle Navarro <mn2614@columbia.edu>
Date:   Wed Nov 30 02:27:15 2016 -0500

    Merge pull request #24 from AFreemantle/ct

    Ct

commit 4d64af6d4dbfb3e21e2a6a24400261bf844da9af
Merge: 27e4cb5 20340a3
Author: Michelle Navarro <mn2614@columbia.edu>
Date:   Wed Nov 30 02:27:03 2016 -0500

    Merge Conflicts fixed

commit 27e4cb56901d9e9a0baebb46289c187cb551e78e
Author: Michelle Navarro <mn2614@columbia.edu>
Date:   Wed Nov 30 02:24:53 2016 -0500

    Fixing test bash script

commit 20340a349f8fec5e07368061b627f66b43643826
```

```
Author: Alexander Brown <aab2212@columbia.edu>
Date:   Wed Nov 30 00:29:33 2016 -0500

    misc improvements

commit 9c24d277e0d4eae6d1ec086c50b66d9f8566b1de
Author: AFreemantle <asf2161@columbia.edu>
Date:   Tue Nov 29 23:24:47 2016 -0500

    added .ll files to clean target

commit 3047f5795611e88ddf047552bc6a4d938275bf2d
Author: Alexander Brown <aab2212@columbia.edu>
Date:   Tue Nov 29 16:46:43 2016 -0500

    Added string_lit in builder, modified format str

commit 8c1314a9d663d7dda36a175ec62761e940440101
Merge: 0bf56b1 ba4fa17
Author: Michelle Navarro <mn2614@columbia.edu>
Date:   Tue Nov 29 15:40:29 2016 -0500

    Merge pull request #23 from AFreemantle/codegen

    cruft

commit 0bf56b1f98a39d2bdbc3a33ed9ea01693621a0d9
Merge: 9e4d7f4 38a4694
Author: Michelle Navarro <mn2614@columbia.edu>
Date:   Tue Nov 29 15:38:22 2016 -0500

    Merge pull request #20 from AFreemantle/codegen

    Codegen

commit ba4fa177556b677e8982979be57278cfb1fd3913
Author: Alexander Brown <aab2212@columbia.edu>
Date:   Tue Nov 29 15:37:37 2016 -0500
```

```

    cruft

commit 38a4694af643e5b4cc444dfd30b9f6d0eed6f7e5
Author: Alexander Brown <aab2212@columbia.edu>
Date:   Tue Nov 29 15:35:04 2016 -0500

    Fixed small print decl bug

commit 7e9463c32632323f937084b5411151d5b3e82586
Author: Alexander Brown <aab2212@columbia.edu>
Date:   Tue Nov 29 15:33:12 2016 -0500

    -c flag now writes bytecode to a.bc

commit b0d667ca3be31851541df8857f0a6bc08cc43617
Author: Alexander Brown <aab2212@columbia.edu>
Date:   Tue Nov 29 04:25:18 2016 -0500

    closer to working now...

commit 5022781df00f09563f6a2faa85ae5be10b8e4a53
Author: Alexander Brown <aab2212@columbia.edu>
Date:   Tue Nov 29 01:49:31 2016 -0500

    fixed errors, trying to loop thru classes

commit cc3d3ce3d6d1cca50301f16f291633876e281a41
Author: Alexander Brown <aab2212@columbia.edu>
Date:   Tue Nov 29 01:37:14 2016 -0500

    codegen wrapper fcns

commit 279b2268262b3774bc9d8051fc2826458937607f
Author: Michelle Navarro <mn2614@columbia.edu>
Date:   Tue Nov 29 00:50:23 2016 -0500

    ast err fic
```

```

commit b7430f8a05b22ca6dc5ff975a0a99b7bd0128e9a
Author: Alexander Brown <aab2212@columbia.edu>
Date:   Tue Nov 29 01:06:40 2016 -0500

    Fixed codegen compile errors

commit bc9a2c4808dc11d7cf143035a07999b18e869b09
Author: Alexander Brown <aab2212@columbia.edu>
Date:   Tue Nov 29 00:54:21 2016 -0500

    locals -> typeKey fcn

commit d31454509b21f4afdaebbfb2a24a1e18ba6e4693
Author: Alexander Brown <aab2212@columbia.edu>
Date:   Tue Nov 29 00:41:49 2016 -0500

    fixed what I thought I'd solved

commit 2bf388da5ed4d98b021e36ae7951ce7aa6716986
Author: Alexander Brown <aab2212@columbia.edu>
Date:   Tue Nov 29 00:37:47 2016 -0500

    removed 'Many' from formal, formal to map fcn

commit 6a964de8c34eb7cfb6c2ffe68a3ddf81a2f2728f
Author: Alexander Brown <aab2212@columbia.edu>
Date:   Tue Nov 29 00:28:09 2016 -0500

    string_of_FName and convert fname to string for 'name'

commit a46b10bfe3a7a1a6a314667b0afdbecef80872bc
Author: Alexander Brown <aab2212@columbia.edu>
Date:   Tue Nov 29 00:21:00 2016 -0500

    null and pattern matching warnings fixed

commit 9c5847bcb8c5b5af74aebfc606abbd43afdc9b95
```

```
Author: Alexander Brown <aab2212@columbia.edu>
Date:   Tue Nov 29 00:09:50 2016 -0500

    in

commit 93e83750af29c0e8bb79f724b83c05d3f4568a91
Author: Alexander Brown <aab2212@columbia.edu>
Date:   Tue Nov 29 00:07:18 2016 -0500

    ltype_of_formal function

commit 43b19dcb786951a9d1d29e4b660a3d94cbc1d967
Author: Alexander Brown <aab2212@columbia.edu>
Date:   Mon Nov 28 23:40:07 2016 -0500

    codegen helper functions

commit d6e3c4082a2b9d96667fd8061de22123253232fb
Author: Michelle Navarro <mn2614@columbia.edu>
Date:   Mon Nov 28 23:02:52 2016 -0500

    MAkefile llvm

commit 5ef138a735a1d4315d13e64463156b72ac9266c2
Merge: 383afff 9e4d7f4
Author: Michelle Navarro <mn2614@columbia.edu>
Date:   Mon Nov 28 22:49:49 2016 -0500

    Fixed merge conf

commit 383afff0462478d376b0505d4f1cbccf95607ddd
Author: AFreemantle <asf2161@columbia.edu>
Date:   Sat Nov 26 16:50:11 2016 -0500

    fixing some naming inconsistencies b/w ast and codegen

commit b3854d08db43eb34f3f374fa8d38b755909babcf
Author: AFreemantle <asf2161@columbia.edu>
```

```
Date:   Sat Nov 26 14:03:47 2016 -0500

    cruft

commit 9e4d7f4f22ad0d8a5b09bf28442b31c4724bb33f
Author: Michelle Navarro <mn2614@columbia.edu>
Date:   Fri Nov 25 04:29:54 2016 -0500

    Added one more test that def going to fail for now

commit 18e047bd71b021c95e5290fae59544b09d9be6bb
Author: Alexander Brown <aab2212@columbia.edu>
Date:   Sun Nov 20 23:05:55 2016 -0500

    moved some stuff around

commit e89dd4416d6e8b956a8847f134718750e217fa69
Author: Alexander Brown <aab2212@columbia.edu>
Date:   Sun Nov 20 21:52:22 2016 -0500

    cruft

commit e79befaf0811d8eab59b2f260d3c3bc0380154fb
Author: Alexander Brown <aab2212@columbia.edu>
Date:   Sun Nov 20 21:50:29 2016 -0500

    added built in print to flist

commit 02438170dc235e1dab6bc5a46bf483a5e94a547d
Author: Alexander Brown <aab2212@columbia.edu>
Date:   Sun Nov 20 20:39:06 2016 -0500

    checks for print fcn

commit 5799c0e343159cba55dba24deb40618a34dc9bc1
Merge: 25eeb49 b7c7a28
Author: Michelle Navarro <mn2614@columbia.edu>
Date:   Sun Nov 20 20:30:18 2016 -0500
```

```

    Merge pull request #19 from AFreemantle/tests

    Test Suite

commit b7c7a283335a71d0325313ce418ea3c54a41635d
Author: Michelle Navarro <mn2614@columbia.edu>
Date:   Sun Nov 20 20:12:21 2016 -0500

    Added test files

commit f249244e1964d9795b3bc6e07c289df559ae8360
Author: Michelle Navarro <mn2614@columbia.edu>
Date:   Sun Nov 20 19:16:13 2016 -0500

    Got rid of sast file

commit 36c877a08e34acdb00b114ec7ead2c8a552aa211
Author: Michelle Navarro <mn2614@columbia.edu>
Date:   Sun Nov 20 19:07:35 2016 -0500

    Tests continue

commit cac1f679a4eeb57b3909b3a0533cc418fd2617e8
Author: Alexander Brown <aab2212@columbia.edu>
Date:   Sun Nov 20 18:40:21 2016 -0500

    Organized, commented variable verifier

commit 5a08843b0d4d7b87e9ec9950d56315b8cac8d066
Author: Michelle Navarro <mn2614@columbia.edu>
Date:   Sun Nov 20 18:22:31 2016 -0500

    putting things where they belong

commit 5128a633e7770e7e778881a2530c115a06ffe121
Author: Michelle Navarro <mn2614@columbia.edu>
Date:   Sun Nov 20 15:33:17 2016 -0500
```

```

    moving tests

commit 339a2831baa471a57d91b83f1663854f33a07ce5
Author: Michelle Navarro <mn2614@columbia.edu>
Date:   Sun Nov 20 15:18:31 2016 -0500

    bash testing begins

commit 25eeb49404328df0238c02a541bd402e70e7dfab
Merge: b662e21 d362dbb
Author: Michelle Navarro <mn2614@columbia.edu>
Date:   Sun Nov 20 18:14:33 2016 -0500

    Merge pull request #22 from AFreemantle/analyzer

    Analyzer

commit d362dbb070090b7699ff6431450e4b6606b71086
Author: Alexander Brown <aab2212@columbia.edu>
Date:   Sun Nov 20 18:10:59 2016 -0500

    cruft for real

commit 8d8c9e05cd488f637585ae3c303adc5c6fda38e8
Author: Alexander Brown <aab2212@columbia.edu>
Date:   Sun Nov 20 18:04:39 2016 -0500

    now breaks on void variables

commit b662e216f7f7e2c770a2a777a0ad2fb248776070
Merge: 1740550 6e0304b
Author: abro9 <aab2212@columbia.edu>
Date:   Sun Nov 20 17:07:15 2016 -0500

    Merge pull request #21 from AFreemantle/analyzer

    main addition: check for duplicate variables
```

```

commit 65f10f1066af2ac9e36ff30d5ad3f4929bf7c240
Author: AFreemantle <asf2161@columbia.edu>
Date:   Sun Nov 20 17:00:16 2016 -0500

    added codegen to build statements

commit 6e0304b8991a6aaccd204bc4ff782dda63aafc8b
Author: Alexander Brown <aab2212@columbia.edu>
Date:   Sun Nov 20 16:59:12 2016 -0500

    cleaned up, still doesn't catch void

commit 29a4edbd535265cbdb91534d88635e613d58ac3c
Author: Alexander Brown <aab2212@columbia.edu>
Date:   Sun Nov 20 16:48:22 2016 -0500

    yells at duplicate variables

commit e7ae619c670855f2328c8f7a3f7792bb396d8020
Author: AFreemantle <asf2161@columbia.edu>
Date:   Sun Nov 20 16:48:15 2016 -0500

    added more skeleton, will call Analyzer soon

commit 79fdc68cc971a3c7f03252bf0cebfd9ecfc373b8
Author: AFreemantle <asf2161@columbia.edu>
Date:   Sun Nov 20 16:27:27 2016 -0500

    added framework for Compile and LLVM_IR options

commit 27a405261aeb1757dfe15065c0ba36b875974593
Author: AFreemantle <asf2161@columbia.edu>
Date:   Sun Nov 20 16:13:01 2016 -0500

    added add_terminal helper from MicroC

commit ca0634acfa578fb9eb2c17efca01945c6ab74cdd
```

```
Author: AFreemantle <asf2161@columbia.edu>
Date:   Sun Nov 20 15:44:01 2016 -0500

    started to define statements

commit 14412f6458f1f85d8b7aff71cc1293b1451c191f
Author: Alexander Brown <aab2212@columbia.edu>
Date:   Sun Nov 20 15:38:29 2016 -0500

    Now runs thru function variables

commit fb46aa5f309a85736de395f1ecee978c6406375a
Author: AFreemantle <asf2161@columbia.edu>
Date:   Sun Nov 20 15:34:28 2016 -0500

    still working on print function

commit 4725db18876d7ea6cb4e86b8ba2abd9ad94d5899
Author: AFreemantle <asf2161@columbia.edu>
Date:   Sun Nov 20 15:25:06 2016 -0500

    local variables and function bodies, moving onto expressions

commit 324961b5e134beaf5829c2d9c752c7def40a7ab4
Author: Alexander Brown <aab2212@columbia.edu>
Date:   Sun Nov 20 14:33:14 2016 -0500

    Added local variable printing to pprinter

commit 17405503d73de716d69026c6a5e5a7c1e8374f5e
Merge: 707ec6a ec1c518
Author: Michelle Navarro <mn2614@columbia.edu>
Date:   Sun Nov 20 14:22:19 2016 -0500

    Merge pull request #17 from AFreemantle/analyzer

    Analyzer

```

```
commit ec1c518d8bb0bff392c8a2618f946115986109f1
Merge: 4cb98cd 707ec6a
Author: Michelle Navarro <mn2614@columbia.edu>
Date:   Sun Nov 20 14:20:15 2016 -0500

    fixing merge conf

commit 4cb98cd5dacbca5ca1830936628c7f681beaf14c
Author: Alexander Brown <aab2212@columbia.edu>
Date:   Sun Nov 20 14:08:41 2016 -0500

    Added local vars to fdecls in parser

commit 3e4aff9b2bb90671900e43cdb478e876ad065a03
Author: AFreemantle <asf2161@columbia.edu>
Date:   Sun Nov 20 13:56:26 2016 -0500

    Llvm boiler plate + print function decleration

commit 88c5a24d3da7eab88c4ad0956f3276976dd2e8f7
Author: Alexander Brown <aab2212@columbia.edu>
Date:   Sun Nov 20 12:08:55 2016 -0500

    test-pretty -> tp

commit 89e7042df0fb8d777244a5648d6a1f5588d7d7b4
Author: Alexander Brown <aab2212@columbia.edu>
Date:   Sun Nov 20 00:44:27 2016 -0500

    sloppy void

commit 707ec6a60dfe94e39eb7f50590a5d154e5fc0884
Merge: 8fdbd33 c5841c5
Author: Michelle Navarro <mn2614@columbia.edu>
Date:   Sun Nov 20 00:17:48 2016 -0500

    Merge pull request #18 from AFreemantle/tests

    Closer to what hello world should look like, string works

commit c5841c50490567477cda2c386fd659762899771f
Author: Michelle Navarro <mn2614@columbia.edu>
Date:   Sun Nov 20 00:15:49 2016 -0500

    Closer to what hello world should look like, string works

commit 8e8a3039e109fed1203b8d14f341200a26e4f63e
Author: Alexander Brown <aab2212@columbia.edu>
Date:   Sat Nov 19 23:08:17 2016 -0500

    duplicates and not void pt. 2

commit 6fdc92d7e826e894a889e8c16c75d32b04e5133f
Author: Alexander Brown <aab2212@columbia.edu>
Date:   Sat Nov 19 22:28:55 2016 -0500

    buuuuuurp

commit 6f2f68b93115cde7384a7e46cfefe3ca97f97659
Author: Alexander Brown <aab2212@columbia.edu>
Date:   Sat Nov 19 21:48:38 2016 -0500

    analyzer2 beginnings

commit 8fdbd33c6703d8a6c5182154da282df4944b2613
Author: Alexander Brown <aab2212@columbia.edu>
Date:   Sat Nov 19 21:30:12 2016 -0500

    string[] to char pt. 2

commit 741c3578a123ea2f0dc40fff64a38d332d31e96e
Merge: 10b36a6 d4cbeab
Author: Michelle Navarro <mn2614@columbia.edu>
Date:   Sat Nov 19 21:39:35 2016 -0500

    Merge pull request #16 from AFreemantle/tests

    Tests

commit d4cbeab31fec1bde914976055bb27d8b97dfcaf4
Author: Michelle Navarro <mn2614@columbia.edu>
Date:   Sat Nov 19 21:37:43 2016 -0500

    cruft

commit e7bcfbd77683358f125d7cb865788a59ebe1bbfb
Author: Michelle Navarro <mn2614@columbia.edu>
Date:   Sat Nov 19 21:37:04 2016 -0500

    Lots of changes

commit 2234035caef46dc857f4fe659b7428628e7088f9
Author: Michelle Navarro <mn2614@columbia.edu>
Date:   Sat Nov 19 21:08:44 2016 -0500

    hello world test

commit 4adc2d89e46ca285eeade36044606c96b63f2c90
Author: Michelle Navarro <mn2614@columbia.edu>
Date:   Sat Nov 19 20:46:37 2016 -0500

    Makefile updates

commit c073d0c37374ce66a11a8d00ef615114ea4b6cc6
Author: Michelle Navarro <mn2614@columbia.edu>
Date:   Sat Nov 19 20:38:42 2016 -0500

    String changes

commit 62f3074040f7082618db65809b910fdfa40228ec
Author: Michelle Navarro <mn2614@columbia.edu>
Date:   Sat Nov 19 19:06:17 2016 -0500

    We had forgotten stringsgit add .git add .
```

```

commit 7bec8b1336e94de76eb61d0cc1e4bcea39bf48c9
Author: Michelle Navarro <mn2614@columbia.edu>
Date:   Sat Nov 19 17:29:48 2016 -0500

    Fixed Makefile

    Now it makes maze automatically when you typw 'make', rather
than clean

commit 577452fd40e2eaec27f77284d254e998519df91f
Author: Michelle Navarro <mn2614@columbia.edu>
Date:   Sat Nov 19 17:26:07 2016 -0500

    cruft

commit 0bece8f4b30b4be3965c954b401362c72ceaecd9
Author: Michelle Navarro <mn2614@columbia.edu>
Date:   Sat Nov 19 17:20:25 2016 -0500

    Working on what Hello World should look like

commit 10b36a6b31a507f6b7d6f45c8c6cda8d7ecae9d3
Merge: e1af195 8ac99ed
Author: Michelle Navarro <mn2614@columbia.edu>
Date:   Sat Nov 19 15:00:56 2016 -0500

    Merge pull request #15 from AFreemantle/pretty-print

    Pretty printer actually working

commit 8ac99ed44b3e0fe4c747f46d79797aabb8b3b8fa
Author: Michelle Navarro <mn2614@columbia.edu>
Date:   Sat Nov 19 14:58:40 2016 -0500

    Pretty printer actually working

    This is a small victory
    But a victory nonetheless
    maze.ml still freaks out when a file doesnt exists though

commit e1af195b316de981a39c7e02f4b44c09e3190b15
Merge: 7e94a00 ca5dc71
Author: Michelle Navarro <mn2614@columbia.edu>
Date:   Sat Nov 19 14:07:09 2016 -0500

    Merge pull request #14 from AFreemantle/pretty-print

    .

commit ca5dc71d0ad8edef99d532453de42521da05efac
Author: Michelle Navarro <mn2614@columbia.edu>
Date:   Sat Nov 19 14:03:56 2016 -0500

    need to deal with file notfound errors now

commit 5304f6b835c8ef28cc5d3373d31e2ba2dcfee30e
Author: Michelle Navarro <mn2614@columbia.edu>
Date:   Sat Nov 19 11:31:28 2016 -0500

    struggle continues

commit 770ac700ac68a26d04b950ce36db46590a0fd9c5
Author: Michelle Navarro <mn2614@columbia.edu>
Date:   Sat Nov 19 11:00:33 2016 -0500

    maze.ml progress but still not done

commit e6735b210571c3a9fb3c0a2d2358433cddfa674f
Author: Michelle Navarro <mn2614@columbia.edu>
Date:   Fri Nov 18 23:41:18 2016 -0500

    File i/o not working

commit 03595b6252cb57fbb12ad7e03f99fcad1fd2879b
Author: Michelle Navarro <mn2614@columbia.edu>
```

```
Date:   Fri Nov 18 22:54:46 2016 -0500

    removed some fluff

commit 7e94a009e16af04680141e613f679289e7709967
Author: Lindsay <ls3245@columbia.edu>
Date:   Sat Nov 19 10:18:43 2016 -0500

    added string expression

commit a3aeea983cd7aaddda04db3f6cb5a09ea4c14fed
Author: Lindsay <ls3245@columbia.edu>
Date:   Sat Nov 19 10:18:25 2016 -0500

    added string

commit 5d306436240e226dc3c5bd2e924caa00569b3091
Merge: d091669 7e0d85d
Author: Michelle Navarro <mn2614@columbia.edu>
Date:   Fri Nov 18 22:28:47 2016 -0500

    Merge pull request #13 from AFreemantle/pretty-print

    a better maze.ml

commit 7e0d85d0c68313ad00588a6c49da0df9916f7e79
Author: Michelle Navarro <mn2614@columbia.edu>
Date:   Fri Nov 18 22:25:45 2016 -0500

    It compiles now

commit 84b4f028c657cdaee554df2d5b1e5cc150f55d7a
Author: Michelle Navarro <mn2614@columbia.edu>
Date:   Fri Nov 18 22:04:07 2016 -0500

    file i/o start'
    '

```

```
commit 70e34d0c3e01323eb35aacef1e5a667217c79c81
Author: Michelle Navarro <mn2614@columbia.edu>
Date:   Thu Nov 17 18:56:42 2016 -0500

    Adding option checking to maze.ml

commit d091669ebea44d7810c6e821d504e8e8e177ba49
Author: Lindsay <ls3245@columbia.edu>
Date:   Fri Nov 18 17:09:22 2016 -0500

    added pretty printing for arrays

commit f46c262a843c3053adbb6865babb871637477517
Author: Lindsay <ls3245@columbia.edu>
Date:   Thu Nov 17 18:48:43 2016 -0500

    added Array datatype

commit e67828f59f875aabe054c522d4ff85be7e4387f2
Author: Michelle Navarro <mn2614@columbia.edu>
Date:   Thu Nov 17 14:37:15 2016 -0500

    maze.ml changes

commit f07299a571c83c8407f6468df44f23bd16d4aa38
Merge: 11c402a 64820d5
Author: Michelle Navarro <mn2614@columbia.edu>
Date:   Mon Nov 14 00:24:29 2016 -0500

    Merge pull request #11 from AFreemantle/pretty-print

    Make maze makes maze

commit 64820d5f3e47588644ee0539e3f4cc53fcfcf758
Author: Michelle Navarro <mn2614@columbia.edu>
Date:   Mon Nov 14 00:20:54 2016 -0500

    Fleshed out Makefile
```

```

commit c0ae58c09ae79002c6ce534233f96969047db24f
Author: Michelle Navarro <mn2614@columbia.edu>
Date:   Sun Nov 13 19:05:36 2016 -0500

    Fixed minor error in maze.ml

commit 11c402ad1ca0c9dee7e29046f040db528e55a855
Author: Lindsay <ls3245@columbia.edu>
Date:   Sun Nov 13 18:53:30 2016 -0500

    fixed incorrect function call

commit 8c611340fffea9dd1d6f0f785289b0aff3af74a9
Merge: b73d37b a011069
Author: Michelle Navarro <mn2614@columbia.edu>
Date:   Sun Nov 13 18:30:32 2016 -0500

    Merge pull request #10 from AFreemantle/pretty-print

    Scanner & Parser updates

commit a01106979743381f1ff4dcf50cec79c7d07b86ba
Author: Michelle Navarro <mn2614@columbia.edu>
Date:   Sun Nov 13 18:26:24 2016 -0500

    Debugged scanner

commit 6ecd1d1ee9c31e46094f7c67e3ea1cff0f1ed0ef
Author: Michelle Navarro <mn2614@columbia.edu>
Date:   Sun Nov 13 17:00:28 2016 -0500

    Got rid of object creation just for now

commit f10036cc795497a05b65caccd4d1ca4d0dd8f9c5
Author: Michelle Navarro <mn2614@columbia.edu>
Date:   Sun Nov 13 16:24:25 2016 -0500

```

```
    AST & Parser inconsistency fixes

commit b73d37bb64601e5406bab854b8739c8e5bb04084
Author: Lindsay <ls3245@columbia.edu>
Date:   Sun Nov 13 17:22:09 2016 -0500

    fixed two typo errors

commit 222db60c053b2850a05f62cbc9ad1347f55bf826
Author: Lindsay <ls3245@columbia.edu>
Date:   Sun Nov 13 17:15:38 2016 -0500

    fixed stash changes

commit 26fd5913066fd2399c79e2d0799bff5b9291f06c
Author: Lindsay <ls3245@columbia.edu>
Date:   Sun Nov 13 17:11:01 2016 -0500

    fixed issues with sast

commit 3354bb5fabd5992dbf91879a63c56877fa97e64b
Author: Lindsay <ls3245@columbia.edu>
Date:   Sun Nov 13 17:10:43 2016 -0500

    added a few more lines

commit 230605f2f7ff5b1e708cd0e0737120362a0e5017
Author: Lindsay <ls3245@columbia.edu>
Date:   Sun Nov 13 15:30:19 2016 -0500

    fixed error with let...and

commit deb8164d00f3eca0d6a0534d9c1b409ae0196950
Author: Lindsay <ls3245@columbia.edu>
Date:   Sun Nov 13 14:41:17 2016 -0500

    return type of an expression

```

```
commit 67cc45fb7576005bf0837d5c486b4194e1358827
Merge: bed3090 e1fef58
Author: Lindsay <ls3245@columbia.edu>
Date:   Sun Nov 13 14:36:02 2016 -0500

    Merge branch 'master' of https://github.com/AFreemantle/maze

commit bed30903d63b62e3227916400f814f758f51b96d
Author: Lindsay <ls3245@columbia.edu>
Date:   Sun Nov 13 14:34:47 2016 -0500

    created minimal sast

commit e1fef584512327293523f114ca778326750e019e
Merge: 53650f4 1f1066e
Author: Michelle Navarro <mn2614@columbia.edu>
Date:   Sun Nov 13 03:04:12 2016 -0500

    Merge pull request #9 from AFreemantle/pretty-print

    bug fixes

commit 1f1066e0ed57b7c3a89716204c5750b794efedd9
Author: Michelle Navarro <mn2614@columbia.edu>
Date:   Sun Nov 13 03:02:03 2016 -0500

    Fixed some ast bugs

commit 650cb19039fd1d08978de9b37e5011313dbbd295
Author: Michelle Navarro <mn2614@columbia.edu>
Date:   Sat Nov 12 19:03:24 2016 -0500

    Parser compilation fix

commit 54624a61099c46036c893059f9a7de42733e74db
Author: Michelle Navarro <mn2614@columbia.edu>
Date:   Sat Nov 12 18:45:46 2016 -0500

```

```
    small update to makefile & parser

commit 53650f44c6f53ff71109b36b7b69a98ee19da2f2
Merge: 31aa2b0 c8b6ee6
Author: Michelle Navarro <mn2614@columbia.edu>
Date:   Sat Nov 12 18:00:01 2016 -0500

    Merge pull request #8 from AFreemantle/pretty-print

    Pretty print

commit c8b6ee66ed0ea7d7f86d68c6ecfb80c4d93f85c4
Author: Michelle Navarro <mn2614@columbia.edu>
Date:   Sat Nov 12 17:58:38 2016 -0500

    Trying to add .native file for maze.ml complilation error

commit 74b38fff4b26ab9a6ad1db307b2f1401bf220c47
Author: Michelle Navarro <mn2614@columbia.edu>
Date:   Sat Nov 12 17:32:24 2016 -0500

    maze.ml changes

commit e7f542cd11930ad2b7d138a4efed6455e2c0b4f4
Author: Michelle Navarro <mn2614@columbia.edu>
Date:   Sat Nov 12 14:01:23 2016 -0500

    starting

commit 99b271b64b22eac6e249d98c0e4178e964c804e6
Author: Michelle Navarro <mn2614@columbia.edu>
Date:   Wed Nov 9 23:56:55 2016 -0500

    STarting with micro C test suite

commit 31aa2b08963e6ddf592326cf3715e0c02ab6b794
Author: AFreemantle <asf2161@columbia.edu>
Date:   Sat Nov 12 15:15:51 2016 -0500
```

```

    updating with each type from ast

commit e831bfab21ec1e367a3aeca027884c19af0a83d2
Author: AFreemantle <asf2161@columbia.edu>
Date:   Sat Nov 12 14:52:49 2016 -0500

    updated README

commit 2932760519536239a62cf3102951061769c95a49
Author: AFreemantle <asf2161@columbia.edu>
Date:   Sat Nov 12 14:45:36 2016 -0500

    initial stages of codegen, started it again

commit 73a1031ddfeb674a29aec6e4158716b77b2c92e7
Author: Lindsay <ls3245@columbia.edu>
Date:   Sat Nov 12 14:22:50 2016 -0500

    fixed fname in parser

commit 808403a243d78634ddcb7a25dac9b03db55fb21f
Author: Lindsay <ls3245@columbia.edu>
Date:   Sat Nov 12 14:22:19 2016 -0500

    fixed fname is ast

commit da465e13bcce210cdf70f1cc39fc8a6c3522b429
Merge: 42c0b26 96596c9
Author: Michelle Navarro <mn2614@columbia.edu>
Date:   Sat Nov 12 14:08:39 2016 -0500

    Merge branch 'master' of github.com:AFreemantle/maze

commit 42c0b261c65164c7d15a6c9df37e9dc358fd92d3
Author: Michelle Navarro <mn2614@columbia.edu>
Date:   Sat Nov 12 14:07:57 2016 -0500

```

```
    constructor fixes

commit 96596c943dd0205dfb2070a251a5491c8882f0c7
Author: AFreemantle <asf2161@columbia.edu>
Date:   Thu Nov 10 11:53:24 2016 -0500

    fixed llvm api calls

commit 8ceab9fc305734f2af87020f36095a447eccc02c
Merge: 33e13e8 50f516f
Author: Michelle Navarro <mn2614@columbia.edu>
Date:   Wed Nov 9 23:59:44 2016 -0500

    Merge branch 'master' of github.com:AFreemantle/maze

commit 33e13e81430475249c79b7f3a198bcd51dc98425
Author: Michelle Navarro <mn2614@columbia.edu>
Date:   Wed Nov 9 23:59:14 2016 -0500

    Fixed very annoying typo

commit 50f516fc4e694bda837bb600e83689a1a9594960
Merge: 5adc718 1b138a6
Author: Lindsay <ls3245@columbia.edu>
Date:   Wed Nov 9 23:38:46 2016 -0500

    Merge branch 'master' of https://github.com/AFreemantle/maze

commit 5adc7185afa7a7db387037ff83f9544fab91c34a
Author: Lindsay <ls3245@columbia.edu>
Date:   Wed Nov 9 23:37:34 2016 -0500

    pretty printer compiles

commit 1b138a67de4a15905377d1e033be0777ac70a211
Author: AFreemantle <asf2161@columbia.edu>
Date:   Wed Nov 9 23:25:37 2016 -0500

```

```
    codegen started, types adddded from ast

commit e8c2b14c07b5eb0b2d18d43aa9fc246ee35a5cde
Author: Lindsay <ls3245@columbia.edu>
Date:   Wed Nov 9 23:01:29 2016 -0500

    fixed half of it to work with our language

commit 5ac337b539db2a69b80a625c23f25a141ed1f1fd
Author: Lindsay <ls3245@columbia.edu>
Date:   Wed Nov 9 23:00:51 2016 -0500

    added datatype function

commit bcf2f434a1af48208cbff57119bb6c658de71ba0
Author: AFreemantle <asf2161@columbia.edu>
Date:   Wed Nov 9 21:39:18 2016 -0500

    skeleton for semantic checker

commit 099b36d54cb9b65bad2284467a7d4adad97a160b
Author: Lindsay <ls3245@columbia.edu>
Date:   Wed Nov 9 21:24:51 2016 -0500

    nothing new

commit 1e6c24749a8b9ccf22ce120d6e77a15d06b2803d
Author: Lindsay <ls3245@columbia.edu>
Date:   Wed Nov 9 21:24:39 2016 -0500

    added string def

commit 2cf0fbe10a902e37415f57de743bf9f37e68b2fd
Author: Lindsay <ls3245@columbia.edu>
Date:   Sat Nov 5 20:02:36 2016 -0400

    only contains clean function

```

```
commit 62fb91010bfaa7e59c982047019f82b5625ffc43
Author: Lindsay <ls3245@columbia.edu>
Date:   Sat Nov 5 20:01:44 2016 -0400

    fixed error with brackets in decls

commit 75b148efc04125e9fcdc84de45faf8fd4c80e3ad
Author: Lindsay <ls3245@columbia.edu>
Date:   Sat Nov 5 20:00:58 2016 -0400

    no change

commit 518a15d34e4a7dbd8ef915a3e21a1942306325d1
Author: Lindsay <ls3245@columbia.edu>
Date:   Wed Nov 2 16:23:04 2016 -0400

    added char

commit 13c022a9028a114a5fdb651db44faed0f6aa2378
Author: Lindsay <ls3245@columbia.edu>
Date:   Wed Nov 2 15:20:13 2016 -0400

    no change

commit 81eb684b921cb418c31ca9def23ed925526379a6
Author: Lindsay <ls3245@columbia.edu>
Date:   Wed Nov 2 15:19:41 2016 -0400

    fixed syntax error with noexpr, added pattern matching for float, string, char and null to pretty printer

commit bcd949fb965b213dfbef33febfcd745d3743d33e
Author: Lindsay <ls3245@columbia.edu>
Date:   Wed Nov 2 13:58:23 2016 -0400

    added extends and new definitions

commit 616291a0d4e3ba6ebc4fb2c13824a6a49e1e2549
```

```
Author: Lindsay <ls3245@columbia.edu>
Date:   Wed Nov 2 13:57:39 2016 -0400

    added type extends

commit 68416769022ccacd84b4cbea7710b38ceab18bdd
Author: Michelle Navarro <mn2614@columbia.edu>
Date:   Wed Nov 2 01:42:02 2016 -0400

    Added class printing

commit 37a1cfe9c3a092959e7dda6c93d42c4cea2dc5be
Author: Michelle Navarro <mn2614@columbia.edu>
Date:   Tue Nov 1 21:40:45 2016 -0400

    printing fucntions

commit da9eda74d57a16411e0852be4844ae3d20176058
Author: Michelle Navarro <mn2614@columbia.edu>
Date:   Tue Nov 1 20:44:36 2016 -0400

    Working on pretty print

commit 39a81e090b967bac385ec6f088be107fc8eee78b
Author: AFreemantle <asf2161@columbia.edu>
Date:   Mon Oct 31 20:23:07 2016 -0400

    added pretty print functioncs from microc example

commit f3d177b5b7bf2a0a694fe1747e5d78153e570136
Author: Michelle Navarro <mn2614@columbia.edu>
Date:   Mon Oct 31 20:15:24 2016 -0400

    Removed some stuff we didnt need from parser

commit c7a72f4f6b36832d5701cbeb8817248a2b39d7c1
Author: AFreemantle <asf2161@columbia.edu>
Date:   Mon Oct 31 19:37:56 2016 -0400
```

```

    grouped functions more intelligently

commit a4727712055eea490d8f9d488237f95044c41995
Author: AFreemantle <asf2161@columbia.edu>
Date:   Mon Oct 31 19:17:41 2016 -0400

    vdecl now reachable

commit 8c256bb257f8383fba767dc6505160ae818f6b5e
Author: AFreemantle <asf2161@columbia.edu>
Date:   Fri Oct 28 20:08:33 2016 -0400

    hopefully not generating empty lang anymore

commit 352e3ba7944a40befe52e5d40327009b159544cf
Author: Michelle Navarro <mn2614@columbia.edu>
Date:   Fri Oct 28 19:57:33 2016 -0400

    Func decl changes

commit 6bcf6bed69a39d028eb79add03bdec7961400db8
Merge: 4632b99 ebbe100
Author: Michelle Navarro <mn2614@columbia.edu>
Date:   Fri Oct 28 19:38:58 2016 -0400

    Merge pull request #7 from AFreemantle/ast

    Added some class stuff to AST

commit ebbe10006381cc2082872fb1303bb5be7f16517f
Author: Michelle Navarro <mn2614@columbia.edu>
Date:   Fri Oct 28 19:34:06 2016 -0400

    Added some class stuff to AST

commit 4632b991ff1e02ab67f0e976302325fb27157bf7
Author: AFreemantle <asf2161@columbia.edu>
```

```
Date:   Fri Oct 28 19:31:54 2016 -0400

    added CONSTRUCTOR keyword, defining class decl now

commit d1589a3b842fdac42a711a118c036f3566e958a6
Author: Michelle Navarro <mn2614@columbia.edu>
Date:   Fri Oct 28 16:26:23 2016 -0400

    Fixed BOOLS and ID in parser

commit 1aee131e7ed9559a0d237a5eb06b32ab7d562e22
Author: AFreemantle <asf2161@columbia.edu>
Date:   Thu Oct 27 23:24:58 2016 -0400

    Fixed error with literals, slowly going from c --> java like structure

commit 52bcdaadf446b7d2cf4cb04f2921b8735ad26a2a
Author: AFreemantle <asf2161@columbia.edu>
Date:   Thu Oct 27 22:07:03 2016 -0400

    updated function declaration

commit dd9a48e94284350b14468c5471ef71bdd767cbf2
Author: Lindsay <ls3245@columbia.edu>
Date:   Thu Oct 27 20:31:38 2016 -0400

    changed misnamed types

commit 8dba2a0fcacb5eaa7f4ac15033798bb127433334
Author: Lindsay <ls3245@columbia.edu>
Date:   Thu Oct 27 20:31:10 2016 -0400

    added let commands

commit 4084d86ee06e8605578713694e3ac6dd3726cafb
Author: Lindsay <ls3245@columbia.edu>
Date:   Thu Oct 27 20:30:17 2016 -0400
```

```

    changed type_Lit

commit e63cb647c4a9cd71fac3e7b908a5c8f8166b5219
Author: Lindsay <ls3245@columbia.edu>
Date:   Thu Oct 27 19:19:03 2016 -0400

    added some declarations

commit 759fabe8a5d5b428aa8f04b452e32c8ff35f3395
Author: Lindsay <ls3245@columbia.edu>
Date:   Thu Oct 27 18:44:26 2016 -0400

    removed FOR, added statements and expressions

commit 31d8248187d612defcc834e2ac39a70a59afd025
Author: Lindsay <ls3245@columbia.edu>
Date:   Thu Oct 27 17:56:16 2016 -0400

    same

commit 310b77f4d71c270c1a488db52993cfc3d4b6dcd5
Author: Lindsay <ls3245@columbia.edu>
Date:   Thu Oct 27 17:55:31 2016 -0400

    fixed char definition

commit a5b35a54854a5d21bbf95581df4bdd5b52f25732
Author: Alexander Brown <aab2212@columbia.edu>
Date:   Mon Oct 24 22:41:11 2016 -0400

    Added float scanning

commit a58204f580cc524b86301a43df608c5408fdada6
Author: Alexander Brown <aab2212@columbia.edu>
Date:   Mon Oct 24 21:55:21 2016 -0400

    fixed merge conflict
```

```

commit dca9ee023d47f1bfeaaaec35ca9d8d152689e297
Author: Lindsay <ls3245@columbia.edu>
Date:   Mon Oct 24 21:48:12 2016 -0400

    added keywords to ast

commit d3791e51e5feb91ba375f5f25a347a95d40b2ee1
Author: Lindsay <ls3245@columbia.edu>
Date:   Mon Oct 24 21:43:45 2016 -0400

    added keywords from scanner

commit a49f19b1fbe23dff126d4366a6288d9fad2a5136
Merge: 87abb1a a3522e1
Author: Lindsay <ls3245@columbia.edu>
Date:   Mon Oct 24 21:36:06 2016 -0400

    added remainder of our keywords

commit 87abb1a166027f13eb08f990078206c127316f25
Author: Lindsay <ls3245@columbia.edu>
Date:   Mon Oct 24 21:34:11 2016 -0400

    added remainder of our keywords

commit 9921eca0808913dfcc36980459eedab82dd0ac7b
Author: Lindsay <ls3245@columbia.edu>
Date:   Mon Oct 24 21:29:56 2016 -0400

    added the rest of our keywords

commit a3522e1a7014e2561f1aedfd89e863fbd009ad4d
Merge: 6d82d68 4c03b70
Author: Alexander Brown <aab2212@columbia.edu>
Date:   Mon Oct 24 21:22:36 2016 -0400

    Merge branch 'master' of https://github.com/AFreemantle/maze
```

```

commit 6d82d687b166e107b26d09f9120140030ae8ed5e
Author: Alexander Brown <aab2212@columbia.edu>
Date:   Mon Oct 24 21:22:05 2016 -0400

    added void, extends, class keywords

commit 4c03b70948cd605fce22aa2fb8b4857d090c9b16
Author: Lindsay <ls3245@columbia.edu>
Date:   Mon Oct 24 21:18:34 2016 -0400

    skeleton ast

commit 530474070d1a3b10da56750c0d022eae23603bab
Author: Alex <aab2212@columbia.edu>
Date:   Mon Oct 24 20:58:21 2016 -0400

    added 'null' keyword

commit 9bdc4bd7ae4b8387017ad965284836b2ada3cee5
Author: Lindsay <linskeaub@yahoo.com>
Date:   Mon Oct 24 20:32:56 2016 -0400

    skeleton parser

commit 1b590a58663278ac3337d38d20a5e7da7fd6eb51
Author: Lindsay <linskeaub@yahoo.com>
Date:   Mon Oct 24 20:12:22 2016 -0400

    updated scanner

commit 03a5dde002e5ba55d5ff511b6dd84103ba4911f7
Author: Lindsay <linskeaub@yahoo.com>
Date:   Sat Oct 22 17:57:28 2016 -0400

    scanner

commit a4ed27d845d9a14571a32c6d28faa793ad7246de
```

```
Author: AFreemantle <asf2161@columbia.edu>
Date:   Sat Oct 8 13:26:13 2016 -0400

    Updated the Readme, time to get started!

commit 8ecc506940be00a7dfd98abc1608bcc2ed054bf4
Author: Alexander Freemantle <asf2161@columbia.edu>
Date:   Sat Oct 8 12:52:23 2016 -0400

    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

```
open Ast
open Codegen

module StringMap = Map.Make(String)

let check classes =

  (* -------------------- Verify variable section -------------------- *)

  (* Raise an exception if the given list has a duplicate *)
  let report_duplicate exceptf list =
    let rec helper = function
        n1 :: n2 :: _ when n1 = n2 -> raise (Failure (exceptf n1))
      | _ :: t -> helper t
      | [] -> ()
    in helper (List.sort compare list)
  in

  (* Check for duplicate class names *)
  report_duplicate (fun n -> "Duplicate class name " ^ n) (List.map (fun n -> n.dname) classes);

  let check_assign lvaluet rvaluet err =
      if lvaluet == rvaluet then lvaluet else raise err
  in

  (* Helper function for check_not_void *)
  let typ_of_datatype = function
      Arraytype(p, i) -> p
    | Datatype(p) -> p
    | Any -> Null                (* <- this will cause problems eventually *)
  in

  (* Raise an exception if a given binding is to a void type *)
  let check_not_void exceptf = function
      Field(t, n) -> if typ_of_datatype t == Void then raise (Failure (exceptf n)) else ()
  in

  let check_not_voidf exceptf = function
      Formal(t, n) -> if typ_of_datatype t == Void then raise (Failure (exceptf n)) else ()
  in

  (* Grabs second element of Field (vdecl) *)
  let get_second = function
      Field(t, n) -> n
  in

  let get_second_formal = function
      Formal(t, n) -> n
```

```
in

let get_typ_formal = function
    Formal(t, n) -> typ_of_datatype t
in

(* Checks that local method variables are neither null nor duplicates *)
let check_locals_lists someMethod =
    List.iter (check_not_void (fun n -> "illegal void variable " ^ n)) someMethod.locals;
    report_duplicate (fun n -> "duplicate variable " ^ n) (List.map get_second someMethod.locals);
in

(* Checks local variables for each method of a class argument *)
let check_locals_class someClass =
    List.iter check_locals_lists someClass.dbody.methods;
in

let check_class_vars someClass =
    List.iter (check_not_void (fun n -> "illegal void variable " ^ n)) someClass.dbody.vdecls;
    report_duplicate (fun n -> "duplicate variable " ^ n) (List.map get_second someClass.dbody.vdecls);
in

List.iter check_locals_class classes;
List.iter check_class_vars classes;


(* -------------------- Verify function section -------------------- *)
(*let check_methods_class someClass =*)
let string_of_fname = function
    FName(s) -> s
  | Constructor -> ""
in

(* checks to ensure that user doesn't define print *)
let check_for_print funcList =
    if List.mem "print" (List.map (fun f -> string_of_fname f.fname) funcList)
    then raise (Failure ("function print is already defined")) else ();
in

let m = StringMap.empty in

(* adds print to Map *)
let built_in_decls = StringMap.add "print"
   { returnType = Datatype(Void); fname = FName("print"); formals = [Formal(Datatype(String), "")];
locals = []; body = [] } m
in

(* Adds funclist to Map *)
let build_f_decls funcList =
        List.fold_left (fun m f -> StringMap.add (string_of_fname f.fname) f m) built_in_decls funcList
    in

    (* Tries to find function s in a given Map *)
    let function_decl s someMap = try StringMap.find s someMap
         with Not_found -> raise (Failure ("unrecognized function " ^ s))
    in

    let map_add_vdecl m v =
        match v with
      | Field(t, n) -> StringMap.add n (typ_of_datatype t) m
    in

    let map_add_formal m v =
        match v with
      | Formal(t, n) -> StringMap.add n (typ_of_datatype t) m
    in

    (* returns a list of all locals in a class *)
    let rec grab_func_locals = function
        [] -> []
      | [x] -> let y = x.locals in y
      | head :: tail -> let r = head.locals in r @ grab_func_locals tail
    in

    (* returns a list of all formals in a class *)
    let rec grab_func_formals = function
        [] -> []
      | [x] -> let y = x.formals in y
      | head :: tail -> let r = head.formals in r @ grab_func_formals tail
    in

    (* verifies each class for correct type use, duplicates, incorrect voids *)
    let check_methods_class someClass =
        let methods = someClass.dbody.methods in
          let function_decls = build_f_decls methods in
          check_for_print methods;
          report_duplicate (fun n -> "duplicate function " ^ n)
                  (List.map (fun f -> string_of_fname f.fname) methods);

          let symbols_classVars = List.fold_left map_add_vdecl StringMap.empty (someClass.dbody.vdecls @
(grab_func_locals methods))
          in

          (* Map of class vars, locals, and formals for fcns in the class *)
          let symbols = List.fold_left map_add_formal symbols_classVars (grab_func_formals methods)
          in

          (* gets the type of some ID *)
          let type_of_identifier s =
              try StringMap.find s symbols
              with Not_found -> raise (Failure ("undeclared identifier " ^ s))
          in

          (* ensures valid expressions, statements *)
          let check_func func =
              let rec expr = function
                  Id s -> type_of_identifier s
                | Int_Lit _ -> Int
                | Bool_Lit _ -> Bool
                | Float_Lit _ -> Float
                | Char_Lit _ -> Char
                | String_Lit _ -> String
                | Noexpr -> Void
                | Null -> Null
                | Unop(op, e) as ex -> let t = expr e
                  in (match op with
                    Neg when t = Int -> Int
                  | Not when t = Bool -> Bool
                  | _ -> raise (Failure ("illegal unary operator " ^
                    string_of_uop op ^ string_of_typ t ^ " in " ^
                    string_of_expr ex)))
                | Binop(e1, op, e2) as e -> let t1 = expr e1 and t2 = expr e2
                  in (match op with
                    Add | Sub | Mult | Div -> if ((t1 = Int || t1 = Float)
                                          && (t1 = t2))
                          then (t1)
                          else raise (Failure ("operator " ^
                                      string_of_op op ^ " requires " ^
                                      "two ints or two floats"))
                  | Equal | Neq when t1 = t2 -> Bool
                           (*add error message, test *)
                  | Less | Leq | Greater | Geq ->
                          if ((t1 = Int || t1 = Float) &&
                              (t1 = t2))
                          then (Bool)
                          else raise (Failure ("operator " ^ string_of_op op ^
                                      " requires int or float operands"))

                  | And | Or when t1 = Bool && t2 = Bool -> Bool
                  | _ -> raise (Failure ("illegal binary operator " ^
                        string_of_typ t1 ^ " " ^ string_of_op op ^ " " ^
                        string_of_typ t2 ^ " in " ^ string_of_expr e))
                  )
                | Assign(var, e) as ex -> let lt = type_of_identifier var
                                          and rt = expr e in
                  check_assign lt rt (Failure ("illegal assignment " ^
                                  string_of_typ lt ^ " = " ^ string_of_typ rt
```

```
                                ^ " in " ^ string_of_expr ex))
        (*   | ObjCreate(oname, actuals) -> Void
            | ObjAccess(e1, e2) -> Void  *)
             | Call(fname, actuals) as call ->
                   if (fname = "print") then Void else
               let fd = function_decl fname function_decls in
               if List.length actuals != List.length fd.formals then
                   raise (Failure ("expecting " ^ string_of_int
                     (List.length fd.formals) ^ " arguments in " ^
                     string_of_expr call))
               else List.iter2 (fun f e -> let et = expr e in
                  ignore (check_assign (get_typ_formal f) et
                    (Failure ("illegal actual argument found " ^
                    string_of_typ et ^ " expected " ^
                    string_of_typ (get_typ_formal f)
                    ^ " in " ^ string_of_expr e )))) fd.formals actuals;
                  typ_of_datatype fd.returnType

          in


          let check_bool_expr e = if expr e != Bool
            then raise (Failure ("expected Boolean expression in " ^ string_of_expr e)) else () in

          let rec stmt = function
              Block sl -> let rec check_block = function
                 [Return _ as s] -> stmt s
               | Return _ :: _ -> raise (Failure "nothing may follow a return")
               | Block sl :: ss -> check_block (sl @ ss)
               | s :: ss -> stmt s ; check_block ss
               | [] -> ()
              in check_block sl
            | Expr e -> ignore (expr e)
            | Return e -> let t = expr e in if t = (typ_of_datatype func.returnType) then () else
                raise (Failure ("return gives " ^ string_of_typ t ^
                      " expected " ^ (string_of_datatype func.returnType)
                      ^ " in " ^ string_of_expr e))
            | If(p, b1, b2) -> check_bool_expr p; stmt b1; stmt b2
            | While(p, s) -> check_bool_expr p; stmt s
           in

           stmt (Block func.body);


          List.iter (check_not_voidf (fun n -> "illegal void formal " ^ n ^ " in " ^ string_of_fname
func.fname)) func.formals;
          report_duplicate (fun n -> "duplicate formal " ^ n ^ " in " ^ string_of_fname func.fname)
(List.map get_second_formal func.formals) in
```

```
        List.iter check_func methods
  in

    (* collects all the functions in the program *)
    let rec grab_class_fcns = function
        [] -> []
      | [x] -> let y = x.dbody.methods in y
      | head :: tail -> let r = head.dbody.methods in r @ grab_class_fcns tail
    in

    let string_of_method f = string_of_fname f.fname in

    (* ensures that there is exactly 1 main method in the program *)
    let check_for_main fl =
      let rec helper i flist =
        match flist with
      | [] -> raise (Failure ("Must have exactly 1 'main' method"))
      | [x] -> if (i = 0 && ((string_of_method x) = "main")) then ()
              else if ( i != 1 || (i = 1 && ((string_of_method x) = "main")))
              then raise (Failure ("Must have exactly 1 'main' method"))
              else ()
      | h :: t -> if (string_of_method h) = "main"
                 then helper (i+1) t
                 else helper i t
      in helper 0 fl
    in

    let all_methods = grab_class_fcns classes in
    check_for_main all_methods;




  List.iter check_methods_class classes;
```

ast.ml

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

```
type op = Add | Sub | Mult | Div | Equal | Neq |
    Less | Leq | Greater | Geq | And | Or

type uop = Neg | Not

type typ = Int | Bool | Char | String | Float | Void | (* Objecttype of string  | *) Null

type extends = NoParent | Parent of string

type datatype = Datatype of typ | Arraytype of typ * int | Any
type fname = Constructor | FName of string

type formal = Formal of datatype * string

type bind = typ * string

type expr = Int_Lit of int
      | Id of string
      | Binop of expr * op * expr
      | Assign of string * expr
      | Bool_Lit of bool
      | Float_Lit of float
      | Char_Lit of char
      | String_Lit of string
   (* | ObjCreate of string * expr list
     | ObjAccess of expr * expr *)
      | Noexpr
      | Unop of uop *expr
      | Call of string * expr list
       | Null

type vdecl = Field of datatype * string

type stmt = Block of stmt list
      | If of expr * stmt * stmt
      | While of expr * stmt
      | Expr of expr
      | Return of expr

(* type vdecl = Field of datatype * string *)

type func_decl = {
      fname   :  fname;
            returnType : datatype;
      formals :  formal list;
      locals  :  vdecl list;
      body    :  stmt list;
}

type dbody = {
        vdecls : vdecl list;
        constructors : func_decl list;
        methods : func_decl list;
}

type class_decl = {
        dname : string;
        dbody : dbody;
```

```
  extends : extends;
}

type program = class_decl list

(* Pretty-printing functions *)

let string_of_op = function
    Add -> "+"
  | Sub -> "-"
  | Mult -> "*"
  | Div -> "/"
  | Equal -> "=="
  | Neq -> "!="
  | Less -> "<"
  | Leq -> "<="
  | Greater -> ">"
  | Geq -> ">="
  | And -> "&&"
  | Or -> "||"

let string_of_uop = function
    Neg -> "-"
  | Not -> "!"

let string_of_typ = function
    Int -> "int"
  | Bool -> "bool"
  | Void -> "void"
  | Char -> "char"
  | String -> "string"
  | Float -> "float"
  | Null -> "null"
(*| Objecttype(s) -> "class" ^ s

let string_of_object = function
 Datatype(Objecttype(s)) -> s
| _ -> "" *)


let string_of_datatype = function
  Arraytype(p, i) -> (string_of_typ p)
| Datatype(p) -> (string_of_typ p)
| Any -> "Any"
(*| Void -> "Void"*)


let rec string_of_bracket = function
  [] -> ""
```

```ocaml
| head :: tail -> "[" ^ (string_of_expr head) ^ "]" ^ (string_of_bracket tail)

and string_of_array_typ = function
  [] -> ""
| [last] -> string_of_expr last
| head :: tail -> string_of_expr head ^ ", " ^ string_of_array_typ tail



and string_of_expr = function
    Int_Lit(l) -> string_of_int l
  | Bool_Lit(true) -> "true"
  | Bool_Lit(false) -> "false"
  | Float_Lit(m) -> string_of_float m
  | Char_Lit(c) -> String.make 1 c
  | String_Lit(s) -> s
  | Null -> ""
  | Id(s) -> s
  | Binop(e1, o, e2) ->
      string_of_expr e1 ^ " " ^ string_of_op o ^ " " ^ string_of_expr e2
  | Unop(o, e) -> string_of_uop o ^ string_of_expr e
  | Assign(v, e) -> v ^ " = " ^ string_of_expr e
  | Call(f, el) ->
      f ^ "(" ^ String.concat ", " (List.map string_of_expr el) ^ ")"
  | Noexpr -> ""
(*| ObjAccess(e1, e2) -> string_of_expr e1 ^ "." ^ string_of_expr e2
 | ObjCreate(s, el) -> "new" ^ s ^ "(" ^ String.concat ", " (List.map string_of_expr el) ^ ")" *)


let string_of_vdecl = function
Field(t, id) -> (string_of_datatype t) ^ " " ^ id ^ ";\n"

let rec string_of_stmt = function
    Block(stmts) ->
      "{\n" ^ String.concat "" (List.map string_of_stmt stmts) ^ "}\n"
  | Expr(expr) -> string_of_expr expr ^ ";\n";
  | Return(expr) -> "return " ^ string_of_expr expr ^ ";\n";
  | If(e, s, Block([])) -> "if (" ^ string_of_expr e ^ ")\n" ^ string_of_stmt s
  | If(e, s1, s2) ->  "if (" ^ string_of_expr e ^ ")\n" ^
      string_of_stmt s1 ^ "else\n" ^ string_of_stmt s2
  | While(e, s) -> "while (" ^ string_of_expr e ^ ") " ^ string_of_stmt s
(*  | Vdecl(vdecl) -> string_of_vdecl vdecl ^ ";\n" *)


let string_of_fname = function
     Constructor -> "constructor"
|    FName(s) -> s


```

```
(*let string_of_vdecl = function
Field(t, id) -> (string_of_datatype t) ^ " " ^ id ^ ";\n" *)

let string_of_formal = function
Formal(d, s) -> (string_of_datatype d) ^ " " ^ s


let string_of_func_decl fdecl = (string_of_datatype fdecl.returnType) ^ " " ^ (string_of_fname
fdecl.fname) ^ " " ^
    "\n{\n" ^
     String.concat "," (List.map string_of_formal fdecl.formals) ^
     String.concat "" (List.map string_of_vdecl fdecl.locals) ^ "\n" ^
     String.concat "" (List.map string_of_stmt fdecl.body) ^ "}\n"

let string_of_dbody dbody =
  String.concat "" (List.map string_of_vdecl dbody.vdecls) ^
  String.concat "" (List.map string_of_func_decl dbody.constructors) ^
  String.concat "" (List.map string_of_func_decl dbody.methods)

let string_of_class decl =
    "class" ^ decl.dname ^ " {\n" ^ (string_of_dbody decl.dbody) ^" }\n"

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

```
(* Codegen for MAZE - based off of MicroC *)

module L = Llvm
module A = Ast

module StringMap = Map.Make(String)


let translate (classes) =

    let grab_dbody someClass =
        someClass.A.dbody
    in

    let rec grab_fcn_lists = function
        [] -> []
      | [x] -> let y = x.A.methods in y
      | head :: tail -> let r = head.A.methods in r @ grab_fcn_lists tail
    in

    let dbodies = List.map grab_dbody classes in
```

```
  let functions = grab_fcn_lists dbodies in

  let context = L.global_context () in
  let the_module = L.create_module context "maze"

  and i32_t = L.i32_type context      (* int *)
  and i8_t = L.i8_type context        (* printf *)
  and i1_t = L.i1_type context        (* bool *)
  and f_t = L.double_type context     (* float *)
  and void_t = L.void_type context    (* void *)
  and str_t = L.pointer_type (L.i8_type context) in
  (* add our other types here *)

  let typ_of_datatype = function
      A.Arraytype(p, i) -> p
    | A.Datatype(p) -> p
    | A.Any -> A.Null
  in

  let ltype_of_typ = function
      A.Int -> i32_t
    | A.Bool -> i1_t
    | A.Void -> void_t
    | A.String -> str_t
    | A.Float -> f_t
    | A.Char -> i8_t
    | A.Null -> i32_t in

  let ltype_of_formal = function
      A.Formal(t, n) -> ltype_of_typ(typ_of_datatype t)
  in

 let typeKey_of_formal = function
      A.Formal(t, n) -> (typ_of_datatype t, n)
  in

  let typeKey_of_local = function
      A.Field(t, n) -> (typ_of_datatype t, n)
  in

  let string_of_FName = function
      A.FName(f) -> f
    | A.Constructor -> ""
  in

  (* This is where global var func would go *)

  (* Lets make a function that does some of this boilerplate stuff
  * below so that we dont have to keep writing it out over and over
```

```
    * again for each function *)
  let printf_t =
      L.var_arg_function_type i32_t [| L.pointer_type i8_t |] in
  let printf_func =
      L.declare_function "printf" printf_t the_module in

  (* Define functions *)

  let function_decls =
      let function_decl m fdecl =
          let name = (string_of_FName fdecl.A.fname)
          and formal_types =
              Array.of_list (List.map ltype_of_formal fdecl.A.formals)
          in let ftype =
              L.function_type (ltype_of_typ (typ_of_datatype fdecl.A.returnType)) formal_types in
          StringMap.add name (L.define_function name ftype the_module,
                              fdecl) m in
      List.fold_left function_decl StringMap.empty functions in

  (* Here we fill in the body of each of the functions *)

  let build_function_body fdecl =

      let (the_function, _) =
          StringMap.find (string_of_FName fdecl.A.fname) function_decls in

      let builder =
          L.builder_at_end context (L.entry_block the_function) in

      (* print *)
      let int_format_str =
          L.build_global_stringptr "%d\n" "fmt" builder in
      let str_format_str =
          L.build_global_stringptr "%s\n" "fmt" builder in
      let char_format_str =
    L.build_global_stringptr "%c\n" "fmt" builder in
      let float_format_str =
    L.build_global_stringptr "%f\n" "fmt" builder in (* adds zeros to end of number *)


      (* For the cool TADS like feature I am going to
      * need to do string formatting *)

  (* local variables - Print takes an argument,need this for hello world*)
  let local_vars =
      let add_formal m (t, n) p = L.set_value_name n p;
        let local = L.build_alloca (ltype_of_typ t) n builder in
        ignore (L.build_store p local builder);
        StringMap.add n local m in

      let add_local m (t, n) =
          let local_var = L.build_alloca (ltype_of_typ t) n builder
          in StringMap.add n local_var m in

      let formals = List.fold_left2 add_formal StringMap.empty
          (List.map typeKey_of_formal fdecl.A.formals) (Array.to_list (L.params the_function)) in
      List.fold_left add_local formals (List.map typeKey_of_local fdecl.A.locals) in

  let lookup n = StringMap.find n local_vars in

  let type_of_val = function
      "i32*" -> int_format_str (*int*)
    | "i8**" -> str_format_str (*string*)
    | "i8*" -> char_format_str (*char*)
    | "i1*" -> int_format_str (*bool*)
    | "double*" -> float_format_str (*float*)
    | _ -> str_format_str
  in

  let check_print_input = function
      A.Int_Lit e -> int_format_str
    | A.String_Lit e -> str_format_str
    | A.Char_Lit c -> char_format_str
    | A.Float_Lit f -> float_format_str
    | A.Binop (e1, op, e2) -> int_format_str
    | A.Bool_Lit b -> int_format_str
    | A.Id s -> type_of_val(L.string_of_lltype(L.type_of (lookup s)))
    | A.Assign (s, e) -> int_format_str
    | A.Noexpr -> int_format_str
    | A.Unop (op, e) -> int_format_str
    | A.Call (s, actuals) -> int_format_str
    | A.Null -> int_format_str
  in

  (* Generate code for an expression *)

  let rec expr builder = function
      A.Int_Lit i -> L.const_int i32_t i
    | A.Bool_Lit b -> L.const_int i1_t (if b then 1 else 0)
    | A.String_Lit s -> L.build_global_stringptr s "str" builder
    | A.Noexpr -> L.const_int i32_t 0
    | A.Id s -> L.build_load (lookup s) s builder
    | A.Binop (e1, op, e2) ->
  let e1' = expr builder e1
  and e2' = expr builder e2
  and float_ops = (match op with
    A.Add     -> L.build_fadd
  | A.Sub     -> L.build_fsub
```

```
   | A.Mult    -> L.build_fmul
         | A.Div     -> L.build_fdiv
   | A.And     -> L.build_and
   | A.Or      -> L.build_or
   | A.Equal   -> L.build_fcmp L.Fcmp.Oeq
   | A.Neq     -> L.build_fcmp L.Fcmp.One
   | A.Less    -> L.build_fcmp L.Fcmp.Olt
   | A.Leq     -> L.build_fcmp L.Fcmp.Ole
   | A.Greater -> L.build_fcmp L.Fcmp.Ogt
   | A.Geq     -> L.build_fcmp L.Fcmp.Oge
   )

         and int_ops = match op with
     A.Add     -> L.build_add
   | A.Sub     -> L.build_sub
   | A.Mult    -> L.build_mul
         | A.Div     -> L.build_sdiv
   | A.And     -> L.build_and
   | A.Or      -> L.build_or
   | A.Equal   -> L.build_icmp L.Icmp.Eq
   | A.Neq     -> L.build_icmp L.Icmp.Ne
   | A.Less    -> L.build_icmp L.Icmp.Slt
   | A.Leq     -> L.build_icmp L.Icmp.Sle
   | A.Greater -> L.build_icmp L.Icmp.Sgt
   | A.Geq     -> L.build_icmp L.Icmp.Sge in
         if(L.type_of e1' = f_t || L.type_of e2' = f_t) then float_ops e1' e2' "tmp" builder
   else int_ops e1' e2' "tmp" builder

  (* | A.ObjCreate(id, e1) ->
  let f = func_lookup id in
  let params = List.map (expr builder) e1 in
  let obj = L.build_call f (Array.of_list params) "tmp" builder in
  obj *)


     | A.Unop(op, e) ->
   let e' = expr builder e in
   (match op with
     A.Neg     -> L.build_neg
         | A.Not     -> L.build_not) e' "tmp" builder
     | A.Assign (s, e) -> let e' = expr builder e in
              ignore (L.build_store e' (lookup s) builder); e'
     | A.Float_Lit f -> L.const_float f_t f
     | A.Char_Lit c -> L.const_int i8_t (Char.code c)
     | A.Null -> L.const_null i32_t
     | A.Call ("print", [e]) -> L.build_call printf_func
               [| check_print_input e; (expr builder e) |]
               "printf" builder
     (* This evaluates arguments backwards *)
```

```
      | A.Call (f, act) ->
         let (fdef, fdecl) = StringMap.find f function_decls in
         let actuals =
             List.rev (List.map (expr builder) (List.rev act)) in
         let result = (match (typ_of_datatype fdecl.A.returnType) with A.Void -> ""
                                             | _ -> f ^ "_result") in
         L.build_call fdef (Array.of_list actuals) result builder in

      (* MicroC has a helper here: add_terminal *)
      let add_terminal builder f =
          match L.block_terminator (L.insertion_block builder) with
          Some _ -> ()
        | None -> ignore (f builder) in


      (* define statements here *)
      let rec stmt builder = function
          A.Block sl -> List.fold_left stmt builder sl

        | A.Expr e -> ignore (expr builder e); builder

        | A.Return e -> ignore (match (typ_of_datatype fdecl.A.returnType) with
          A.Void -> L.build_ret_void builder
        | _ -> L.build_ret (expr builder e) builder); builder

        | A.If (pred, then_stmt, else_stmt) ->
      let bool_val = expr builder pred in
      let merge_bb = L.append_block context "merge" the_function in
      let then_bb = L.append_block context "then" the_function in
      add_terminal (stmt (L.builder_at_end context then_bb) then_stmt) (L.build_br merge_bb);
      let else_bb = L.append_block context "else" the_function in
      add_terminal (stmt (L.builder_at_end context else_bb) else_stmt) (L.build_br merge_bb);

      ignore (L.build_cond_br bool_val then_bb else_bb builder);
      L.builder_at_end context merge_bb

        | A.While (pred, body) ->
     let pred_bb = L.append_block context "while" the_function in
     ignore (L.build_br pred_bb builder);
     let body_bb = L.append_block context "while_body" the_function in
     add_terminal (stmt (L.builder_at_end context body_bb) body) (L.build_br pred_bb);
     let pred_builder = L.builder_at_end context pred_bb in
     let bool_val = expr pred_builder pred in
     let merge_bb = L.append_block context "merge" the_function in
     ignore (L.build_cond_br bool_val body_bb merge_bb pred_builder);
     L.builder_at_end context merge_bb

in

```

```
    (* Code for each statement in the function *)
    let builder = stmt builder (A.Block fdecl.A.body) in

    (*MicroC behavior here is to have program return void
    * if control falls off the end of the program *)
    add_terminal builder (match (typ_of_datatype fdecl.A.returnType) with
        A.Void -> L.build_ret_void
      | t -> L.build_ret (L.const_int (ltype_of_typ t) 0))
    in


    List.iter build_function_body functions;
    the_module
```

Makefile

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

```
OBJS = ast.cmx codegen.cmx parser.cmx scanner.cmx analyzer.cmx maze.cmx

maze : $(OBJS)
    ocamlfind ocamlopt -linkpkg -package llvm str.cmxa -package llvm.analysis $(OBJS) -o maze

scanner.ml : scanner.mll
    ocamllex scanner.mll

parser.ml parser.mli : parser.mly
    ocamlyacc parser.mly

%.cmo : %.ml
    ocamlc -c $<

%.cmi : %.mli
    ocamlc -c $<

%.cmx : %.ml
    ocamlfind ocamlopt -c -package llvm str.cmxa $<

###Generated by "ocamldep *.ml *.mli" after building scanner.ml and parser.ml
ast.cmo :
ast.cmx :
codegen.cmo : ast.cmo
codegen.cmx : ast.cmx
maze.cmo : analyzer.cmo scanner.cmo parser.cmi codegen.cmo ast.cmo
maze.cmx : analyzer.cmx scanner.cmx parser.cmx codegen.cmx ast.cmx
parser.cmo : ast.cmo parser.cmi
parser.cmx : ast.cmx parser.cmi
scanner.cmo : parser.cmi
scanner.cmx : parser.cmx
```

```
analyzer.cmo: ast.cmo
analyzer.cmx: ast.cmx
parser.cmi : ast.cmo

.PHONY : clean
clean:
    ocamlbuild -clean
    rm -rf ./tests/*.ll
    rm -rf testall.log *.diff maze scanner.ml parser.ml parser.mli
    rm -rf *.cmx *.cmi *.cmo *.o *.out *.ll
```

maze.ml
Authors: Alexander Freemantle, Alex Brown, Michelle Navarro, Lindsay Schiminske

```
open Printf
open Str

type action = Ast | Help | Error | LLVM_IR | Compile

let help_string = (
    "Usage: ./maze [option] <source file>\n" ^
    "\n Valid Options: \n"^
    "\t -h: Help \n" ^
    "\t -c: Compile \n" ^
    "\t -l: Generate LLVM without checking \n" ^
    "\t -a: Print Abstract Syntax tree \n"
)

let invalid_arg_string = ("Invalid Arguments")

let ast_holder = ("Ast would be printing\n")

let check_option = function
     "-h" -> Help, "."
    | _ -> Error, "."

let check_action = function
      "-h" -> Help
    | "-a" -> Ast
    | "-l" -> LLVM_IR
    | "-c" -> Compile
    | _ -> Error

let _ =
    let action, filename =
        if Array.length Sys.argv = 1 then Help, "."
        else if Array.length Sys.argv = 2 then check_option (Sys.argv.(1))
        else if Array.length Sys.argv = 3 then check_action Sys.argv.(1), (Sys.argv.(2))
        else Error, "." in

    let check_filename =
        if filename = "." then raise (Failure (invalid_arg_string)) else () in
    check_filename;

let in_channel = open_in filename in
let lexbuf = Lexing.from_channel in_channel in
let program = Parser.program Scanner.token lexbuf in

let filename_list = Str.split (regexp "[.]") filename in
let basename = List.hd filename_list in

Analyzer.check program;

    match action with
        Help -> print_string help_string
      | Ast -> print_string (Ast.string_of_program program)
      | LLVM_IR -> print_string (Llvm.string_of_llmodule
                                          (Codegen.translate program))
      | Compile -> let m = Codegen.translate program in
        Llvm_analysis.assert_valid_module m; (*Built in check*)
        (*print_string (Llvm.string_of_llmodule m);*)
        (*Llvm.dump_module m;*)
        let oc = open_out (basename ^ ".ll") in fprintf oc "%s\n" (Llvm.string_of_llmodule m); close_out
oc;
      | Error -> print_string (invalid_arg_string ^ "\n")
```

parser.mly

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

```
%{ open Ast %}

%token SEMI LPAREN RPAREN LBRACE RBRACE DOT COMMA
%token PLUS MINUS TIMES DIVIDE ASSIGN NOT
%token EQ NEQ LT LEQ GT GEQ TRUE FALSE AND OR
%token RETURN IF ELSE WHILE
%token INT BOOL CHAR STRING FLOAT VOID NULL
%token CLASS CONSTRUCTOR EXTENDS NEW

%token <int> INT_LITERAL
```

```
%token <char> CHAR_LITERAL
%token <float> FLOAT_LITERAL
%token <string> ID
%token <string> STRING_LITERAL
%token EOF

%nonassoc NOELSE
%nonassoc ELSE
%right ASSIGN
%left OR
%left AND
%left EQ NEQ
%left LT GT LEQ GEQ
%left PLUS MINUS
%left TIMES DIVIDE
%right NOT NEG
%right DOT

%start program
%type <Ast.program> program

%%


program: decls EOF { $1 }

/* Classes */

decls:
    class_decl_list   { List.rev $1}

class_decl_list:
    class_decl           { [$1] }
  | class_decl_list class_decl { $2::$1 }

class_decl:
        CLASS ID LBRACE dbody RBRACE { {
            dname = $2;
            extends = NoParent;
            dbody = $4;
        } }

       | CLASS ID EXTENDS ID LBRACE dbody RBRACE { {
    dname = $2;
    extends = Parent($4);
    dbody = $6;
       } }
dbody:

        /* nothing here */ { {
            vdecls = [];
            constructors = [];
            methods = [];
        } }

     |   dbody vdecl { {
            vdecls =$2 :: $1.vdecls;
            constructors = $1.constructors;
            methods = $1.methods;
        } }


     |   dbody constructor { {
            vdecls = $1.vdecls;
            constructors = $2 :: $1.constructors;
            methods = $1.methods;
        } }

     |   dbody fdecl { {
            vdecls = $1.vdecls;
            constructors = $1.constructors;
            methods = $2 :: $1.methods;
        } }

/* Constructors */

constructor:
    CONSTRUCTOR LPAREN formals_opt RPAREN LBRACE stmt_list RBRACE {
        {
            fname = Constructor;
            formals = $3;
            body = List.rev $6;
            locals = [];
            returnType = Any
        }
    }



/* Methods */

fname:
    ID { $1 }

fdecl:
     typ fname LPAREN formals_opt RPAREN LBRACE vdecl_list stmt_list RBRACE
     {
         {
              fname = FName($2);
              returnType = $1;
              formals = $4;
              locals = List.rev $7;
              body = List.rev $8; (* stmt_list; *)
          }
      }


/* Variables */

 vdecl_list:
    /* nothing */  { [] }
 | vdecl_list vdecl { $2 :: $1 }

vdecl: typ ID SEMI { Field($1, $2) }


/* Formals */

formals_opt:
   /* nothing */ { [] }
 | formal_list  {List.rev $1 }

formal_list:
   typ ID   { [Formal($1, $2)] }
 | formal_list COMMA typ ID { Formal($3, $4) :: $1 }


actuals_opt:
  /* nothing */  { [] }
 | actuals_list { List.rev $1}

actuals_list:
   expr    { [$1] }
 | actuals_list COMMA expr { $3 :: $1 }


/* TYPES */

typ:
   INT   { Datatype(Int) }
 | FLOAT { Datatype(Float) }
 | CHAR  { Datatype(Char) }
 | STRING {Datatype(String)}
 | BOOL  { Datatype(Bool) }
 | VOID  { Datatype(Void) }


```

```
/* Expressions */

stmt_list:
   /* nothing */{ [] }
  | stmt_list stmt  { $2 :: $1 }

stmt:
   expr SEMI     { Expr $1 }
 | RETURN SEMI    { Return Noexpr }
 | RETURN expr SEMI  { Return $2 }
 | LBRACE stmt_list RBRACE   { Block(List.rev $2) }
 | IF LPAREN expr RPAREN stmt %prec NOELSE   { If($3, $5, Block([])) }
 | IF LPAREN expr RPAREN stmt ELSE stmt   { If($3, $5, $7) }
 | WHILE LPAREN expr RPAREN stmt  { While($3, $5) }
/* | typ ID SEMI { Vdecl($1, $2) } */


expr:
   literals     { $1 }
 | expr PLUS expr  { Binop($1, Add, $3) }
 | expr MINUS expr { Binop($1, Sub, $3) }
 | expr TIMES expr { Binop($1, Mult, $3) }
 | expr DIVIDE expr { Binop($1, Div, $3) }
 | expr EQ expr  { Binop($1, Equal, $3) }
 | expr NEQ expr { Binop($1, Neq, $3) }
 | expr LT expr { Binop($1, Less, $3) }
 | expr GT expr { Binop($1, Greater, $3) }
 | expr LEQ expr { Binop($1, Leq, $3) }
 | expr GEQ expr { Binop($1, Geq, $3) }
 | expr AND expr { Binop($1, And, $3) }
 | expr OR expr { Binop($1, Or, $3) }
/* | expr DOT expr { ObjAccess($1, $3) } */
 | MINUS expr %prec NEG { Unop(Neg, $2) }
 | NOT expr  { Unop(Not, $2) }
 | ID ASSIGN expr  { Assign($1, $3) }
 | LPAREN expr RPAREN  { $2 }
 | ID LPAREN actuals_opt RPAREN { Call($1, $3) }
/* | NEW ID LPAREN actuals_opt RPAREN { ObjCreate($2, $4) } */



literals:
  INT_LITERAL     { Int_Lit($1) }
| FLOAT_LITERAL   { Float_Lit($1) }
| CHAR_LITERAL    { Char_Lit($1) }
| STRING_LITERAL  { String_Lit($1) }
| TRUE      { Bool_Lit(true) }
| FALSE           { Bool_Lit(false) }
| ID       { Id($1) }
| NULL            { Null }
```

scanner.mll

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

```
{ open Parser

  let unescape s = Scanf.sscanf ("\"" ^ s ^ "\"") "%S%!" (fun x->x)
}

let alpha = ['a' - 'z' 'A' - 'Z']
let digit = ['0' - '9']
let escape = '\\' ['\\' ''' '"' 'n' 'r' 't']
let ascii = ([' '-'!' '#'-'[' ']'-'~'])
let float = (digit+) ['.'] digit+
let int = digit+
let char = ''' ( ascii | digit ) '''
let string = '"' ( (ascii | escape)* as s) '"'
let id = alpha(alpha | digit | '_')*

rule token = parse
    [ ' ' '\t' '\r' '\n' ] {token lexbuf} (* Whitespace *)
| "(*" {comment lexbuf} (*Comments*)

| '(' {LPAREN}
| ')' {RPAREN}
| '{' {LBRACE}
| '}' {RBRACE}
| ';' {SEMI}
| ',' {COMMA}
| '.' {DOT}

| '+' {PLUS}
| '-' {MINUS}
| '*' {TIMES}
| '/' {DIVIDE}
| '=' {ASSIGN}
| "==" {EQ}
| "!=" {NEQ}
| '<' {LT}
| '>' {GT}
| "<=" {LEQ}
| ">=" {GEQ}
| "&&" {AND}
| "||" {OR}
| "!" {NOT}

| "if" {IF}
| "else" {ELSE}
| "while" {WHILE}
| "return" {RETURN}
| "int" {INT}
| "bool" {BOOL}
| "char" {CHAR}
| "float" {FLOAT}
| "string" {STRING}
| "void" {VOID}
| "true" {TRUE}
| "false" {FALSE}
| "null" {NULL}
| "void" {VOID}
| "extends" {EXTENDS}
| "class" {CLASS}
| "new" {NEW}

(*| ['0' - '9']+ as lxm { LITERAL(int_of_string lxm) } *)
(*| ['0' - '9']+['.']['0' - '9']+ as lxm { FLOAT_LITERAL(float_of_string lxm) }
| ['a' - 'z' 'A' - 'Z']['a' - 'z' 'A' - 'Z' '0' - '9' '_']* as lxm { ID(lxm) }
| eof {EOF}
| _ as char { raise (Failure("illegal character " ^ Char.escaped char)) }
*)

| int as lxm  { INT_LITERAL(int_of_string lxm) }
| float as lxm   { FLOAT_LITERAL(float_of_string lxm) }
| char as lxm  { CHAR_LITERAL(String.get lxm 1) }
| string   { STRING_LITERAL(unescape s) }
| id as lxm  { ID(lxm) }
| eof   { EOF }

and comment = parse
    "*)" {token lexbuf}
| _ {comment lexbuf}
```

testall.sh

Authors: Michelle Navarro (adapted code from MicroC)

```
#!/bin/sh

# Regression testing script for Maze
# Step through a list of files
#  Compile, run, and check the output of each expected-to-work test
#  Compile and check the error of each expected-to-fail test

# Path to the LLVM interpreter
LLI="lli"
#LLI="/usr/local/opt/llvm/bin/lli"

# Path to the maze compiler.
MAZE="./maze"
#MAZE="./maze"

# Set time limit for all operations
ulimit -t 30

globallog=testall.log
rm -f $globallog
error=0
globalerror=0

keep=0

Usage() {
    echo "Usage: testall.sh [options] [.maze files]"
    echo "-k    Keep intermediate files"
    echo "-h    Print this help"
    exit 1
}

SignalError() {
    if [ $error -eq 0 ] ; then
   echo "FAILED"
    error=1
    fi
   echo "  $1"
}

# Compare <outfile> <reffile> <difffile>
# Compares the outfile with reffile.  Differences, if any, written to difffile
Compare() {
    generatedfiles="$generatedfiles $3"
    echo diff -b $1 $2 ">" $3 1>&2
    diff -b "$1" "$2" > "$3" 2>&1 || {
    SignalError "$1 differs"
    echo "FAILED $1 differs from $2" 1>&2
    }
}

# Run <args>
# Report the command, run it, and report any errors
Run() {
    echo $* 1>&2
    eval $* || {
    SignalError "$1 failed on $*"
    return 1
    }
}

# RunFail <args>
# Report the command, run it, and expect an error
RunFail() {
    echo $* 1>&2
    eval $* && {
    SignalError "failed: $* did not report an error"
    return 1
    }
    return 0
}

Check() {
    error=0
    basename=`echo $1 | sed 's/.*\\///
                             s/.maze//'`
    reffile=`echo $1 | sed 's/.maze$//'`
    basedir="`echo $1 | sed 's/\/[^\/]*$//'`/."

    echo -n "$basename..."

    echo 1>&2
    echo "###### Testing $basename" 1>&2

    generatedfiles=""


    generatedfiles="$generatedfiles ${basename}.ll ${basename}.out" &&
    Run "$MAZE -c ${reffile}.maze" &&
    $LLI "${reffile}.ll" > "${basename}.out"
    Compare ${basename}.out ${reffile}.out ${basename}.diff

    # Report the status and clean up the generated files

    if [ $error -eq 0 ] ; then
   if [ $keep -eq 0 ] ; then
       rm -f $generatedfiles
    fi
   echo "OK"
    echo "###### SUCCESS" 1>&2
    else
   echo "###### FAILED" 1>&2
    globalerror=$error
    fi
}

CheckFail() {
    error=0
    basename=`echo $1 | sed 's/.*\\///
                            s/.maze//'`
    reffile=`echo $1 | sed 's/.maze$//'`
    basedir="`echo $1 | sed 's/\/[^\/]*$//'`/."

    echo -n "$basename..."

    echo 1>&2
    echo "###### Testing $basename" 1>&2

    generatedfiles=""

    generatedfiles="$generatedfiles ${basename}.err ${basename}.diff"&&
    RunFail "$MAZE -c  ${reffile}.maze" "2>" "${basename}.err" ">>" $globallog &&
    Compare ${basename}.err ${reffile}.err ${basename}.diff

    # Report the status and clean up the generated files

    if [ $error -eq 0 ] ; then
   if [ $keep -eq 0 ] ; then
       rm -f $generatedfiles
    fi
   echo "OK"
    echo "###### SUCCESS" 1>&2
    else
   echo "###### FAILED" 1>&2
    globalerror=$error
    fi
}

while getopts kdpsh c; do
   case $c in
    k) # Keep intermediate files
        keep=1
        ;;
    h) # Help
```

```
        Usage
        ;;
    esac
done

shift `expr $OPTIND - 1`

LLIFail() {
  echo "Could not find the LLVM interpreter \"$LLI\"."
  echo "Check your LLVM installation and/or modify the LLI variable in testall.sh"
  exit 1
}

which "$LLI" >> $globallog || LLIFail


if [ $# -ge 1 ]
then
   files=$@
else
   files="tests/test-*.maze tests/fail-*.maze"
fi

for file in $files
do
   case $file in
    *test-*)
        Check $file 2>> $globallog
        ;;
    *fail-*)
        CheckFail $file 2>> $globallog
        ;;
    *)
        echo "unknown file type $file"
        globalerror=1
        ;;
    esac
done

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

```
correct
correct
```

./test-bool.out

```
1
0
1
0
```

./test-if2.maze

```
class test {
   void main () {

       if(false){
         print("if");
     }

     else if(false){
        print("elseif1");
       }

     else if(false){
         print("elseif2");
     }

     else{
         print("else");
     }
   }
}
```

./test-unop.maze

```
class test {
   void main () {

       if(!false){
         print("correct");
     }

     if(3 != -3){
         print("correct");
     }
   }
}
```

./test-addition.maze

```
class test {
  void main () {

    int x;
    x = 3 + 7;

    if(x==10){
        print("correct");
    }
  }
}
```

./test-printid-bool.out

```
0
```

./test-equality.maze

```
class test {
  void main () {

    if(42<50){
          print("less");
      }

      if(60>3){
        print("greater");
    }

    if(100==100){
        print("equal");
    }

    if(30!=53){
        print("not equal");
    }

    if(10<=40){
        print("leq");
    }

    if(50>=50){
        print("geq");
    }

  }
}
```

./fail-binopOperands.maze

```
class test {
   void main () {
       int y;
       float z;

       y = 42;
       z = 1.0;
       y + z;

   }
}
```

./fail-defprint.err

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

./test-recursion.out

```
34
```

./test-printid-int.maze

```
class test {
   void main() {
       int x;
       x = 42;
       print(x);
   }
}
```

./fail-voidClassvar.err

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

./fail-voidvar.err

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

./test-whilestmt.maze

```
class test {
   void main () {

       int i;
       i = 0;

     while(i < 5){
         print("while loop");
           i = i + 1;
       }
   }
}
```

./fail-dupvar.maze

```
class test {
   void main () {
       int x;
       int x;
   }
}
```

./test-subtraction.out

```
correct
```

./test-ifEmpty.maze

```
class test {
   void main () {

       if(false){}
     else{}

     print("empty");
   }
}
```

./test-ifEmpty.out

```
empty
```

./test-printid-float.maze

```
class test {
   void main() {
       float x;
       x = 42.00000;
       print(x);
   }
}
```

./test-gcd.out

```
5
```

./test-binop.maze

```
class test {
   void main(){
       print(4114 + 1);
   }
}
```

./test-hello.out

```
Hello World
```

./test-ifNested.maze

```
class test {
   void main () {

     int x;
     int y;

     x=3;
     y=4;

       if(x == 3){
         if(y == 4){
             print("nested");
         }
     }
   }
}
```

./test-printid-simple.maze

```
class test {

   void main(){
       string x;
       x = "Hello";
   }

}
```

./fail-voidClassvar.maze

```
class test {
   void x;
   void main () {
       int y;
   }
}
```

./fail-undeclaredID.err

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

./test-return.maze

```
class test {

   int computeValue(int x, int y){
       return x + y;
```

```
  }

  void main(){

      int a;
    a = computeValue(10,5);
    print(a);

  }
}
```

./test-recursion.maze

```
class test {

  int fib(int n){

    if(n <= 1){
        return n;
    }
    return fib(n-1) + fib(n-2);
  }

  void main() {
      int n;
      int answer;

      n = 9;
      answer = fib(n);
      print(answer);
  }

}
```

./fail-dupClassvar.err

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

./test-printfloat.maze

```
class test {
  void main () {
      print(3.14);
  }
}
```

./test-printid-bool.maze

```
class test {
  void main() {
       bool x;
       x = false;
       print(x);
   }
}
```

./test-printid-string.maze

```
class test {

   void main(){
       string x;
       x = "Hello";
       print(x);
   }

}
```

./test-subtraction.maze

```
class test {
   void main () {

     int x;
     x = 3 - 7;

     if(x==-4){
         print("correct");
     }
   }
}
```

./test-printid-simple.out

./test-classes.out

```
ok
```

./fail-binopOperands.err

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

./test-binop.out

```
4115
```

./test-ifstmt.out

```
correct
```

./fail-dupClassvar.maze

```
class test {
   int x;
   int x;
   void main () {
       int y;
   }
}
```

./fail-dupFun.maze

```
class test {

   void main (){}
   void test () {
       int y;
   }

   void test (){}
}
```

./fail-undeclaredID.maze

```
class test {
   void main () {

       if (y){
       print("works");
       }
   }
}
```

./test-return.out

```
15
```

./test-classes.maze

```
class shape {

}


class test {
   void main(){

       print("ok");

   }

}
```

./test-printid-char.maze

```
class test {
   void main() {
       char x;
       x = 'c';
       print(x);
   }
}
```

./test-binop2.out

```
correct
correct
```

./test-binopmult.out

```
25
```

./fail-dupvar.err

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

./test-fbinop.out

```
add
sub
mult
div
```

./test-ops.maze

```
class test {
   void main () {

     int x;
     int y;
     int z;
     int a;

     x = 3 + 7;
       y = 10 - 2;
     z = 3 * 2;
     a = 50 / 10;

     if(x==10){
         print("add");
     }

     if(y==8){
         print("sub");
```

```
        }

        if(z==6){
            print("mult");
        }

        if(a==5){
            print("div");
        }

    }
}
```

./test-printchar.maze

```
class test {
    void main () {
        print('a');
    }
}
```

./test-ifstmt.maze

```
class test {
    void main () {

        if(true){
          print("correct");
      }
      else{
         print("incorrect");
        }
    }
}
```

./test-printint.out

```
42
```

./test-printint.maze

```
class test {
    void main () {
        print(42);
    }
}
```

./fail-dupClass.maze

```
class test {
    void main () {
        int x;
    }
}

class test {
    void wag(){}
}
```

./test-printid-float.out

```
42.000000
```

./fail-voidvar.maze

```
class test {
    void main () {
        void x;
    }
}
```

./test-printid-char.out

```
c
```

./test-func-call.maze

```
class test {

    void dummy(){
        int x;
        x = 1;
        print(x);
    }

    int main(){
        dummy();
    }
}
```

./test-printfloat.out

```
3.140000
```

./test-ifNested.out

```
nested
```

./test-binopmult.maze

```
class test {
    void main(){
        print(5 * 5);
    }
}
```

./test-binop2.maze

```
class test {
   void main(){

       if(true && true){
         print("correct");
     }

     if(false || true){
         print("correct");
     }

   }
}
```

./fail-defprint.maze

```
class test {
   void main () {
       int x;
   }

   void print (string s){}
}
```

./test-func-call.out

```
1
```

./test-printchar.out

```
a
```

./test-printid-int.out

```
42
```

./fail-dupFormal.err

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

./test-hello.maze

```
class test {
   void main () {
       print("Hello World");
   }
}
```

./test-if2.out

```
else
```

./fail-dupFun.err

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

./test-bool.maze

```
class test {
   void main () {
         print(true);
           print(false);
           print(1==1);
           print(2==3);
   }
}
```

./fail-voidFormal.maze

```
class test {
   void main (void x) {
       print("Hello World");
   }
}
```

./test-fbinop.maze

```
class test {
   void main (){

       float x;
     x = 3.2 + 4.5;

     if(x == 7.700000){
           print("add");
     }

       x = 5.4 - 2.2;
     if(x == 3.200000){
         print("sub");
     }

     x = 4.2 * 6.8;
     if(x == 28.560000){
        print("mult");
     }

     x = 4.6 / 2.3;
     if(x == 2.000000){
         print("div");
        }
    }
}
```

./test-addition.out

```
correct
```

./fail-voidFormal.err

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

./fail-dupFormal.maze

```
class test {
   void main (int x, int x) {
       print("Hello World");
   }
}
```

./test-equality.out

```
less
greater
equal
not equal
leq
geq
```

./test-ops.out

```
add
sub
mult
div
```

./test-whilestmt.out

```
while loop
while loop
while loop
while loop
while loop
```

./fail-dupClass.err

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

./test-gcd.maze

```
class test {
   void main () {
       int x;
       int y;
       x = 15;
       y = 20;

       if (x == 0) {
          print(x);
       }

       while(x != y)
       {
           if ( x > y) {
               x= x-y;
           } else {
               y= y-x;
           }
       }
       print(x);
   }
}
```

./test-printid-string.out

```
Hello
```