

Dice Project Report



"Java, but worse"

Project Manager:	David Watkins	djw2146
Language Guru:	Emily Chen	ec2805
System Architect:	Philip Schiffrin	pjs2186
Tester & Verifier:	Khaled Atef	kaa2168

CONTENTS

1. INTRODUCTION

The Dice programming language is an object-oriented, general purpose programming language. It is designed to let programmers who are more familiar with object oriented programming languages to feel comfortable with common design patterns to build useful applications. The syntax of Dice resembles the Java programming language. Dice compiles down to LLVM IR which is a cross-platform runtime environment. This allows Dice code to work on any system as long as there is an LLVM port for it, which includes Windows, Mac OS X, and Linux or various processor architectures such as x86, MIPS, and ARM¹.

Dice lays programs out the same way a Java program would. Variables and methods of a class can be declared with private scope. There is a simple to use inheritance that allows for multiple children inheriting the fields and methods of its parent. Dice also allows for convenient use of functions that exist in C, such as malloc, open, and write. This allows the user to construct objects and call c functions using those objects.

Background

Object-oriented programming (OOP) is a programming paradigm based on the concept of "objects". These objects are data structures that contain data, in the form of fields, often known as attributes. The code itself are contained within methods in the code which are compiled to varying subroutines. The most useful aspect of OOP is that these methods and fields can modify one another allowing for a rich and varied use case.

Class based OOP specifically creates instances of classes, referred to as objects, which have their values modified at runtime. There are many languages that implement their language this way including Java and C#.

Inheritance is when an object or class is based on another class using the same implementation. This allows for a class to serve as a blueprint for subclasses. Polymorphism allows an object to take on many forms. This may include an object being assigned to a type that is a class it inherits from, or being used in place of a class it inherits from.

We want to leverage these capabilities using LLVM code to produce a syntactically Java-like language but offer a cross platform solution that is simple and easy to use. Implementing inheritance and objects in a c-like context like LLVM allows for fine control over the code.

Related Work

Object-oriented programming languages have existed since the late 20th century. Java, C#, C++, Objective-C, Python, and many more languages have facilities for defining custom user classes and manipulating them at runtime.

¹<http://llvm.org/>

Implementing an object-oriented paradigm using C is a well-known solution, but compiling object-oriented code down to LLVM is not publicly available. We want to contribute to the LLVM community by adding additional information regarding the creation of a compiler using OCaml that compiles to LLVM code.

Goals

Cross-Platform

Utilizing the LLVM IR we are able to compile the source once and have it work on multiple architectures without fail.

Flexibility

Allowing the user to define their own classes and offering them the ability to inherit functionality from other user defined types offer a wide range of possibilities for their programs and also saves the user time when implementing large programs.

Transparency

Using the LLVM IR allows the user to see exactly what the program is doing after the compiler is done. For a more optimal result it can then be compiled to bitcode representation using the LLVM compiler.

Familiarity

Incorporate familiar primitive data types most commonly found in languages such as C, C++, and Java such as int, char, float, and bool.

2. LANGUAGE TUTORIAL

Environment Setup

The compiler has been built and tested using an Ubuntu 15.10 virtual machine. The ISO for downloading Ubuntu 15.10 can be found here¹. This is followed by downloading virtualbox and following the corresponding tutorial for setting up a custom Ubuntu VM here². Once inside the VM there are a series of packages that need to be installed before you can compile the compiler. Run the following commands to install the corresponding packages:

```
>sudo apt-get install m4 clang-3.7 clang-3.7-doc libclang-common-3.7-dev libclang-3.7-dev
↪ libclang1-3.7 libclang1-3.7-dbg libllvm-3.7-ocaml-dev libllvm3.7 libllvm3.7-dbg
↪ lldb-3.7 llvm-3.7 llvm-3.7-dev llvm-3.7-doc llvm-3.7-examples llvm-3.7-runtime
↪ clang-modernize-3.7 clang-format-3.7 python-clang-3.7 lldb-3.7-dev liblldb-3.7-dbg
↪ opam llvm-runtime
```

Then initialize OCaml's package manager (OPAM) in your home directory:

```
>opam init
>opam switch 4.02.1
>eval $(opam config env)
>opam install core batteries llvm yojson
```

After OPAM is initialized, go to the directory where you want Dice installed and clone the git repository:

```
>git clone https://github.com/DavidWatkins/Dice.git
```

Using the Compiler

Inside the directory 'Dice' type **make**. This creates the dice compiler that takes in '.dice' files and compiles them to corresponding '.ll' files corresponding to LLVM IR. The syntax for running the dice executable is: **dice [optional-option] <source file>**. There are also additional flags with respect to the compiler that allow for additional options.

- **-h** - Print help text
- **-tendl** - Prints tokens with newlines intact
- **-t** - Prints token stream
- **-p** - Pretty prints Ast as a program
- **-ast** - Prints abstract syntax tree as json
- **-sast** - Prints semantically checked syntax tree as json

¹<http://www.ubuntu.com/>

²<http://www.wikihow.com/Install-Ubuntu-on-VirtualBox>

- **-c** - Compiles source and prints result to stdout
- **-f** - Compiles source to file ($\langle\text{filename}\rangle.\langle\text{ext}\rangle \rightarrow \langle\text{filename}\rangle.ll$)

The following sample dice code demonstrates the following features:

- The mandatory main function that exists within **only** one class. The syntax for a main declaration is **public void main(char[] args)**
- Calling the built-in print function, which takes an arbitrary list of primitive values, including char[].
- A string literal with escape characters
- Defining a base class with one or more fields.

```
1 class example1 {  
2     public void main(char[] args) {  
3         print("This is example 1\n");  
4     }  
5 }
```

To compile the sample code above, type:

```
> ./dice example1.dice
```

The output will be a file named **example1.ll** which will run using the **lli** command:

```
>lli example1.ll  
This is example 1  
>
```

If you get an error: "error: expected value token" from lli, that means your version of lli is probably set incorrectly. Run the following command to verify the version:

```
>lli --version
```

If it's anything other than version **3.7** change it with the following commands:

```
>sudo rm /usr/bin/lli  
>ln -s /usr/lib/llvm-3.7/bin/lli /usr/bin/lli
```

The basics

Primitives

All primitives are declared starting with their type followed by an identification. Dice supports the following primitives:

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

```
1 class example2 {
2     public void main(char[] [] args) {
3
4         (* This is a comment (* with a nested one inside *) *)
5         int a; (* Declaring an integer primitive variable *)
6         a = 1; (* Assigning the number one to variable a *)
7
8         float b = 1.5; (* Combined declaration and assignment is okay *)
9
10        (* Characters and booleans are primitives as well *)
11        char c = 'c';    (* ASCII or digits only within single quotes*)
12        bool d = true;   (* or 'false' *)
13    }
14 }
```

Arrays

Arrays are indexed collections of values of a datatype (primitive or object). Dice allows for single dimension arrays only. The elements within the arrays created default to null which, like C, are implemented with zeros.

```
1 class example3 {
2     public void main(char[] [] args) {
3
4         int[] a = new int[10]; (* int array with 10 elements set to zero *)
5
6         a[0] = 1; (* Access the first element and assign the integer 1 to it *)
7
8         int[] b = {0,1,2,3,4,5}; (* int array with 6 int elements *)
9
10        print(b.length); (* prints 6 *)
11
12        char[] c = {'h','i', 0}; (* ints are allowed to be stored in char
↪ elements *)
13
14    }
15 }
```

Operators

Dice supports the following binary operators:

- Arithmetic (+ , - , * , / , %)
- Relational (== , != , > , < , >= , <=)
- Logical (and, or)

Unary operators:

- Logical negation (not)
- Negative number (-)

```
1 class example4 {
2     public void main(char[] [] args) {
3
4         int a = 1 + 2;      (* a is now 3 *)
5         float b = 2.5 - 2;  (* 2 is promoted to float, b is now 0.5 *)
6         int c = 5 + 2 * 4;  (* c is 13 due to operator precedence *)
7         int d = 10 / 5 + 3; (* d is now 5 *)
8         int e = 5 % 3;      (* e is now 2 *)
9
10        bool f = true; bool g = false;
11        f == f; f != g; 5 > 2; 3 >= 3; f or g; (* all expressions evaluate to
    ↪ true *)
12        f and g; not f; (* evaluate to false *)
13
14        c = -a;          (* c is now -3 *)
15    }
16 }
```

Control Flow

The statements inside source files are generally executed from top to bottom, in the order that they appear. Control flow statements, however, break up the flow of execution by employing decision making, looping, and branching, enabling your program to conditionally execute particular blocks of code. This section describes the decision-making statements (if-then, if-then-else), the looping statements (for, while), and the branching statements (break, continue, return) supported by Dice.

Branching

```
1 class example5 {
2     public void main(char[] [] args) {
3         int a;
4         if (true)
5             a = 1;
6         else
7             a = 0;
8         (* a is now 1 *)
9     }
```



```

9
10     int b;
11     if (false){
12         b = 2; a = 3;
13     }
14     else {
15         b = 0; a = 0;
16     }
17     (* b and a are now 0 *)
18
19     int c;
20     if(false){ a = 1; b = 1; c = 1;}
21     else if(true) { a = 5; b = 5; c = 5;}
22     else { a = 0; b = 0; c = 0;}
23     (* a,b,c are now set to 5 *)
24 }
25 }

```

Loops

The two types of loops that Dice supports are 'for' and 'while' loops. The for statement allows the programmer to iterate over a range of values. The while statement executes a user-defined block of statements as long as a particular conditional expression evaluates to true.

```

1  class example6 {
2      public void main(char[] [] args) {
3          int a = 0;
4          int i;                (* The loop counter must be declared outside the
↳   for loop *)
5          for (i = 0 ; i < 5 ; i = i + 1) {
6              a = a + 2;
7          }
8          (* a is now set to 10 *)
9
10         int b = 0;
11         int j;
12         for (j = 0 ; j < 5 ; j = j + 1) {
13             a = a + 2;
14             if(a >= 14){
15                 break;                (* will break out of the parent for
↳   loop *)
16             }
17             else { continue; } (* will skip the remaining code and start the
↳   next iteration *)
18             b = b + 10;
19         }
20         (* b is still zero, a is 14 *)
21
22         while(b<5){
23             b = b + 1;

```

```
24         }
25
26         (* b is now 4 *)
27     }
28 }
```

Defining methods

Dice supports methods that return a datatype after execution or simply execute without returning anything. Methods can accept arguments which are computed in an applicative order. Each method must also contain a scope (public/private) which determine access for outside classes. The following example will show two kinds of methods:

```
1  class example7 {
2      public int p(int i){
3          print(i);
4          return i;
5      }
6
7      public void q(int a, int b, int c){
8          int total = a ;
9          print(b);
10         total = total + c ;
11     }
12
13     public void main(char[][] args) {
14         this.q( this.p(1), 2, this.p(3));
15     }
16 }
```

The output of this program is:

132

Classes and Inheritance

Since Dice is an Object Oriented language, you can create custom classes that can serve as datatypes. A class contains three sections:

- Fields
- Constructors
- Methods

These sections may be written in any order desired. You may also mix them up if desired. For example, a constructor may be added inbetween field declarations if desired. If no constructors are defined, Dice will use a default constructor to instantiate objects. A parent class can also be assigned any class that is a descendant of it as shown below:

```
1  class shape {
2      public int xCoord;    (* Fields *)
```

```

3      public int yCoord;
4
5      constructor(){                                (* Constructor *)
6          this.xCoord = 0;
7          this.yCoord = 0;
8      }
9
10     (* Constructor with a different signature due to the two arguments *)
11     constructor(int x, int y){
12         this.xCoord = x;
13         this.yCoord = y;
14     }
15
16     public void myAction(){ (* Method *)
17         print("shape");
18     }
19 }
20
21 class circle extends shape {
22     public int radius;          (* Field unique to circle *)
23
24     constructor(){
25         this.xCoord = 0;        (* xCoord and yCoord from parent class 'shape'
↳ *)
26         this.yCoord = 0;
27         this.radius = 0;
28     }
29
30     constructor(int x, int y, int r){
31         this.xCoord = x;
32         this.yCoord = y;
33         this.radius = r;
34     }
35
36     public void myAction(){ (* This method overrides the one defined in parent
↳ class *)
37         print("circle\n");
38         print(this.radius);
39     }
40 }
41 }
42
43 class example8 {
44     public void main(char[][] args) {
45         class circle a = new circle(1, 2, 3);
46         class circle[] b = new class circle[10];
47         b[0] = a;
48         print(b[0].radius, "\n");
49     }

```

```
50         class shape c = new circle(4, 5, 6);  (* Inheritance in action! *)
51         c.myAction();
52         print("\n");
53     }
54 }
```

The output for example8 is:

```
3
circle
6
```

3. LANGUAGE REFERENCE MANUAL

Introduction

Dice is a general purpose, object-oriented programming language. The principal is simplicity, pulling many themes of the language from Java. Dice is a high level language that utilizes LLVM IR to abstract away hardware implementation of code. Utilizing the LLVM as a backend allows for automatic garbage collection of variables as well.

Dice is a strongly typed programming language, meaning that at compile time the language will be type-checked, thus preventing runtime errors of type.

This language reference manual is organized as follows:

- Section 2 Describes types, values, and variables, subdivided into primitive types and reference types
- Section 3 Describes the lexical structure of Dice, based on Java. The language is written in the ASCII character set
- Section 4 Describes the expressions and operators that are available to be used in the language
- Section 5 Describes different statements and how to invoke them
- Section 6 Describes the structure of a program and how to determine scope
- Section 7 Describes classes, how they are defined, fields of classes or their variables, and their methods

The syntax of the language is meant to be reminiscent of Java, thereby allowing ease of use for the programmer.

Types

There are two kinds of types in the Dice programming language: primitive types and non-primitive types. There are, correspondingly, two kinds of data values that can be stored in variables, passed as arguments, returned by methods, and operated on: primitive values and non-primitive values.

```
Type:
PrimitiveType
NonprimitiveType
```

There is also a special null type, the type of the expression *null*, which has no name. Because the null type has no name, it is impossible to declare a variable of the null type. The null reference is the only possible value of an expression of null type. The null reference can always undergo a widening reference conversion to any reference type. In practice, the programmer can ignore the null type and just pretend that *null* is merely a special literal that can be of any reference type.

Primitive Types and Values

A primitive type is predefined by the Dice programming language and named by its reserved keyword.

```
PrimitiveType:
NumericType
bool
NumericType:
IntegralType
float
IntegralType: one of
int char
```

int

A value of type *int* is stored as a 32-bit signed two's-complement integer. The *int* type can hold values ranging from -2,147,483,648 to 2,147,483,647, inclusive.

float

The float type stores the given value in 64 bits. The *float* type can hold values ranging from 1e-37 to 1e37. Since all values are represented in binary, certain floating point values must be approximated.

char

The *char* data type is a 8-bit ASCII character. A *char* value maps to an integral ASCII code. The decimal values 0 through 31, and 127, represent non-printable control characters. All other characters can be printed by the computer, i.e. displayed on the screen or printed on printers, and are called printable characters. The character 'A' has the code value of 65, 'B' has the value 66, and so on. The ASCII values of letters 'A' through 'Z' are in a contiguous increasing numeric sequence. The values of the lower case letters 'a' through 'z' are also in a contiguous increasing sequence starting at the code value 97. Similarly, the digit symbol characters '0' through '9' are also in an increasing contiguous sequence starting at the code value 48.

bool

A variable of type *bool* can take one of two values, *true* or *false*. A bool could also be *null*.

Non-Primitive Types

Non-primitive types include arrays and classes.

Arrays

An array stores one or more values of the same type contiguously in memory. The type of an array can be any primitive or an array type. This allows the creation of an n-dimensional array, the members of which can be accessed by first indexing to the desired element of the outermost array, which is of type *array*, and then accessing into the desired element of the immediately nested array, and continuing n-1 times.

Classes

Classes are user-defined types. See chapter 7 to learn about their usage.

Casting

Casting is not supported in this language. There are behaviors between ints and float defined in the section on operators that imitate casting, but there is no syntax to support casting between types directly.

Lexical Conventions

This chapter describes the lexical elements that make up Dice source code. These elements are called tokens. There are six types of tokens: identifiers, keywords, literals, separators, and operators. White space, sometimes required to separate tokens, is also described in this chapter.

Identifiers

Identifiers are sequences of characters used for naming variables, functions and new data types. Valid identifier characters include ASCII letters, decimal digits, and the underscore character '_'. The first character must be alphabetic.

An identifier cannot have the same spelling (character sequence) as a keyword, boolean or null literal, a compile-time error occurs. Lowercase letters and uppercase letters are distinct, such that foo and Foo are two different identifiers.

```
ID = "[ 'a'-'z' 'A'-'Z' ] ( [ 'a'-'z' 'A'-'Z' ] | [ '0'-'9' ] | '_' ) *"
```

Keywords

Keywords are special identifiers reserved for use as part of the programming language itself. You cannot use them for any other purpose. Dice recognizes the following keywords:

if	else	for	while	
break	continue	return		
int	float	bool	char	void
null	true	false	class	constructor
public	private	extends	include	this

Literals

A literal is the source code representation of a value of a primitive type or the null type.

Integer Literals

An integer literal is expressed in decimal (base 10). It is represented with either the single ASCII digit 0, representing the integer zero, or an ASCII digit from 1 to 9 optionally followed by one or more ASCII digits from 0 to 9.

```
INT = "[ '0'-'9' ] +"
```

Float Literals

A float literal has the following parts: an integer part, a decimal point (represented by an ASCII period character), and a fraction part. The integer and fraction parts are defined by a single digit 0 or one digit from 1-9 followed by more ASCII digits from 0 to 9.

```
FLOAT = "[ '0'-'9' ] + [ '.' ] [ '0'-'9' ] +"
```

Boolean Literals

The boolean type has two values, represented by the boolean literals `true` and `false`, formed from ASCII letters.

```
BOOL = "true|false"
```

Character Literals

A character literal is always of type `char`, and is formed by an ascii character appearing between two single quotes. The following characters are represented with an escape sequence, which consists of a backslash and another character:

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

It is a compile-time error for the character following the character literal to be other than a single-quote character `'`.

```
CHAR = "\' ( ([' \'-!' \'#\'-\'[\' \']\'-\'~\'] | \'\\\' [ \'\\\' \'\"\' \'n\' \'r\' \'t\' ]) )\' "
```

String Literals

A string literal is always of type `char[]` and is initialized with zero or more characters or escape sequences enclosed in double quotes.

```
char[] x = "abcdef\n";
```

```
STRING = "\"( ([' \'-!' \'#\'-\'[\' \']\'-\'~\'] | \'\\\' [ \'\\\' \'\"\' \'n\' \'r\' \'t\' ]) )*\\""
```

Separators

A separator separates tokens. White space is a separator but it is not a token. The other separators are all single-character tokens themselves: `() [] ; , .`

```
'('      { LPAREN }
')'      { RPAREN }
'{'      { LBRACE }
'}'      { RBRACE }
';'      { SEMI }
','      { COMMA }
'['      { LBRACKET }
']'      { RBRACKET }
'.'      { DOT }
```


Operators

The following operators are reserved lexical elements in the language. See the expression and operators section for more detail on their defined behavior.

+	-	*	/	%
=	==	!=		
<	<=	>	>=	
and	or	not		
new	delete			

White Space

White space refers to one or more of the following characters:

- the ASCII SP character, also known as "space"
- the ASCII HT character, also known as "horizontal tab"
- the ASCII FF character, also known as "form feed"
- LineTerminator

White space is ignored, except when it is used to separate tokens. Aside from its use in separating tokens, it is optional.

```
WHITESPACE = "[ ' ' '\t' '\r' '\n' ]"
```

Comments

The characters `(*` introduce a comment, which terminates with the characters `*)`. Comments may be nested within each other.

```
COMMENT = "( \* [ ^ \* ] \* \* )"
```

Expressions and Operators

Syntax Notation

In the syntax notation used in this manual, syntactic categories are indicated by *italic* type and literal words are indicated in **bold** type.

`{expression}` indicates a required expression in braces.

An optional terminal or non-terminal symbol has the subscript *opt* appended, so that `{expressionopt}` indicates an optional expression in braces.

Operator Precedence

The precedence of expression operators is the same as the order of the major subsections of this section (highest precedence first). Within each subsection, the operators have the same precedence. Left- or right-associativity is specified in each subsection for the operators discussed therein.

Primary Expressions

Primary expressions involving `.`, subscripting, and function calls group left to right.

identifier

An identifier is a primary expression, provided it has been suitably declared as discussed below. Its type is specified by its declaration.

constant

A constant of any of the primitive types discussed in Chapter 3 is a primary expression.

(expression)

A parenthesized expression is a primary expression whose type and value are identical to those of the unadorned expression. The presence of parentheses does not affect whether the expression is an lvalue.

Array Literal

|expression_{opt}|
| expression-list |

A string, which originally has the type "array of **char**", is a primary expression. An array literal storing another type is also a primary expression.

Array Access

primary-expression[expression]

A primary expression followed by an expression in square brackets is a primary expression. The intuitive meaning is that of a subscript. The primary expression has type array of . . . and the type of the result is The type of the subscript expression must be a type that is convertible to an integral type, or a compile-time error occurs.

Function Call

primary-expression (expression – list_{opt})

A function call is a primary expression followed by parentheses containing a possibly empty, comma-separated list of expressions which constitute the actual arguments to the function. The result of the function call is the function's return type. Recursive calls to any function are permissible.

Object Member Access

primary-lvalue . r-value

*primary-lvalue: identifier | **this** | (expression) | primary-expression[expression]*

primary-rvalue: identifier | primary-expression (expression – list_{opt})

An lvalue expression followed by a dot followed by the name of a class member is a primary expression. The object referred to by the lvalue is assumed to be an instance of the class defining the class member. The given lvalue can be an instance of any user-defined class.

Unary Operations

unary-operator expression

*unary-operator: **not** | -*

Expressions with unary operators group right-to-left.

Logical Not

not *expression*

The result of the logical negation operator **not** is **true** if the value of the expression is **false**, **false** if the value of the expression is **true**. The type of the result is **bool**. This operator is applicable only to operands that evaluate to **bool**.

Negation

-constant | *-(expression)*

The result is the negative of the expression, and has the same type. The type of the expression must be **char**, **int**, or **float**.

Dynamic Memory Management

The **new** operator is used to allocate dynamic memory in two scenarios: array creation and object creation.

Array Creation

new *type*[*expression*]

Object Creation

new *identifier*(*expression_{opt}*)
new *identifier*(*expression-list*)

Memory Deallocation

delete *r-value*

The **delete** operator is used to deallocate heap memory. The *r-value* can be either an l-value or r-value of either an array creation or object creation expression.

Multiplicative Operations

expression multiplicative-operator expression
multiplicative-operator: * | / | %

The multiplicative operators group left-to-right. They operate on numeric types (**int**, **char**, **float**). If both operands are of type **int**, the result is of type **int**. If either operand is of type **float**, then the result is of type **float**. If either operand is of type **char**, then the result is of type **char**.

Additive Operations

expression additive-operator expression
additive-operator: + | -

The additive operators + and - group left-to-right. They operate on numeric operands (**int**, **char**, **float**). The same type considerations as for multiplication apply. Overflow of a **char** type during an addition operation results in wraparound.

Relational Operations

expression relational-operator expression

relational-operator: < | > | <= | >=

The relational operators group left-to-right. They operate on numeric operands (**int**, **char**, **float**). The relational operators all yield **true** if the specified relation is true and **false** otherwise.

Equality Operations

expression equality-operator expression

equality-operator: == | !=

The == (equal to) and the != (not equal to) operators are exactly analogous to the relational operators except for their lower precedence.

Logical Operations

expression logical-operator expression

logical-operator: **and** | **or**

Both operands must evaluate to a value of type **bool**. The **and** operator returns **true** if both its operands evaluate to **true**, **false** otherwise. The second expression is not evaluated if the first evaluates to **false**. The **or** operator returns **true** if either of its operands evaluate to **true**, and **false** otherwise. The second operand is not evaluated if the value of the first operand evaluates to **true**.

Assignment Operation

lvalue = expression

primary-lvalue: *identifier* | {**this**|*identifier*} . *expression* | *primary-expression*[*expression*]

The value of the expression replaces that of the object referred to by the lvalue. Both operands must have the same type.

Statements

A statement forms a complete unit of execution. Most statements are expression statements and have the form

expression ;

So that several statements can be used where one is expected, the compound statement is provided:

compound-statement:

{*statement* – *list*}

statement-list:

statement *statement-list*

Control Flow Statements

The statements inside source files are generally executed from top to bottom, in the order that they appear. Control flow statements, however, break up the flow of execution by employing decision making, looping, and branching, enabling your program to conditionally execute particular blocks of code. This section describes the conditional statements (if-then, if-then-else), looping statements (for, while), and branching statements (break, continue, return) supported by the Dice programming language.

Conditional Statement

The forms of the conditional statement are:

if (*expression*) *statement*

if (*expression*) *statement* (**else if** *statement*)* **else** *statement*

The expression enclosed in balanced parentheses is evaluated and if it is **true**, the first substatement is executed. In the second case, if the expression evaluates to **false** and there is an **else-if** clause, then the substatement in the **else-if** clause is executed. If the expression evaluates to **false** and no **else-if** clause exists, then the substatement in the **else** clause is executed. As usual, the **else** ambiguity is resolved by connecting an else with the last encountered elseless if.

Looping

The while statement has the form

while (*expression*) *statement*

The substatement is executed repeatedly so long as the value of the expression remains non-zero. The test takes place before each execution of the statement.

The **for** statement has the form:

for (*expression_{opt}* ; *expression_{opt}* ; *expression_{opt}*) *expression*

This statement is equivalent to:

```
while (expression-2) {  
    statement  
    expression-3 ;  
}
```

Thus the first expression specifies initialization for the loop; the second specifies a test, made before each iteration, such that the loop is exited when the expression becomes **false**; the third expression typically specifies an incrementation which is performed after each iteration. Any or all of the expressions may be dropped. A missing expression-2 makes the implied while clause equivalent to while(**true**); other missing expressions are simply dropped from the expansion above.

Branching

The statement

break;

causes termination of the outermost enclosing **while** or **for** statement; control passes to the statement following the terminated statement.

The statement

continue;

causes control to pass to the loop-continuation portion of the outermost enclosing **while** or **for** statement; that is to the end of the loop.

A function returns to its caller by means of the return statement, which has one of the forms:

return;

return (*expression*);

In the first case no value is returned. In the second case, the value of the expression is returned to the caller of the function. If a function has no **return** statement, then it returns with no returned value.

File Inclusion

If a .dice file contains a statement of the following form:

include(*expression*);

where the expression is a string literal that specifies the path to another .dice file, then all classes defined in that file are available to be used in definitions of classes in the .dice file in which the include statement appears. Include statements must appear before other types of statements in a .dice file.

Declaration Statements

Instance Field Declaration

A field declaration statement declares an instance field of a class and has the following form:

scope type-specifier identifier ;

scope: **public** | **private**

type-specifier: *type* | **class** *identifier* | **class** *identifier*[] | *type*[]

type: any primitive type in Dice

Note that this is the only legal format of a field declaration statement; assignment statements are not a valid way to declare instance fields in Dice.

Local Variable Declaration

type-specifier identifier ;

type-specifier: *type* | **class** *identifier* | **class** *identifier*[] | *type*[]

type: any primitive type in Dice

Instance Method Declaration

A method declaration statement declares an instance method of a class and has the following form:

scope type name (formal-list_{opt}) {statement-list_{opt}}

scope: **public** | **private**

type-specifier: *type* | **class** *identifier* | **class** *identifier*[] — *type*[]

type: Any primitive or non-primitive type in Dice, or **void**. If the *type* is **void**, then the method being declared returns no value.

name: **main** | *identifier*

Only one method per program may be declared with the *name* **main**.

identifier: Any identifier, excluding the following, which are names of built-in functions in Dice:

print	input	malloc	open
close	read	write	lseek
exit	realloc	getchar	

formal: *type-specifier identifier*

statement: *local-variable-declaration* | *expression-statement*

expression-statement: *assignment-expression-statement* | *function-call-expression-statement*

Constructor Declaration

A constructor declaration statement has the following form:

constructor (*formal-list_{opt}*) {*statement-list_{opt}*}

formal: *type-specifier identifier*

type-specifier: *type* | **class** *identifier* | **class** *identifier*[] — *type*[]

type: any primitive type in Dice

statement: *local-variable-declaration* | *expression-statement*

expression-statement: *assignment-expression-statement* | *function-call-expression-statement*

Class Declaration

A class declaration statement has one of the following forms:

class *identifier* {*cbody*}

class *identifier* **extends** *identifier* {*cbody*}

identifier: The *identifier* that follows the keyword **extends** must be the name of another class declared in the same program. The *identifier* that follows the keyword **class** must not be identical to the name of any other class declared in the same program.

cbody: {*statement-list_{opt}*}

statement: *instance-field-declaration* | *instance-method-declaration* | *constructor-declaration*

Program Structure and Scope

Program structure and scope define what variables are accessible and where. When inside a class, there are many different cases of scope, however those are better defined in chapter 7.

Program Structure

A Dice program may exist either within one source file or spread among multiple files which can be linked at compile-time. An example of such a linked file is the standard library, or *stdlib.dice*. When an include statement is executed at compile time, it will compile all classes in the included file along with the classes in the file on which the compilation was run. Therefore at compilation, one only needs to compile with *dicecmaster.dice*. If an included module defines a class that has the same name as one of the classes defined in the including module, then the compiler throws an error. The compiler does not resolve recursive includes; if *foo.dice* includes *bar.dice* and *bar.dice* includes *foo.dice*, the compiler throws an error.

A program consists of zero or more include statements, followed by one or more class definitions. Each class defined in a module must have a distinct name. Classes cannot have two methods with the same name regardless of the method's signature. Only one class out of all classes may have a main method, defined with *public void main(char[][] args)* which designates the entry point for a program to begin executing code. All Dice files are expected to end with the file extension *.dice* and follow the following syntactic layout.

Scope refers to which variables, methods, and classes are available at any given time in the program. All classes are available to all other classes regardless of their relative position in a program or library. Variable scope falls into two categories: fields (instance variables), which are defined at the top of a class, and local variables, which are defined within a method. Fields and methods can be public or private. If a field or method is public then it is accessible whenever an instance of that class is instantiated. Private fields and methods are only accessible within the same class.

Local variables are variables that are declared inside of a method. Local variables are only accessible within the same method in which they are declared, and they may have the same name as fields within the same class since fields in a class are only accessible by calling the *this* keyword.

Classes

Classes are the constructs whereby a programmer defines their own types. All state changes in a Dice program must happen in the context of changes in state maintained by an object that is an instance of a user-defined class.

Class Declaration

The syntax for declaring a class is in the "Declarations" subsection of the "Statements" section. According to the class declaration syntax, fields, constructor and methods are optional for each class and may appear in any order in the class body.

Methods may not be overloaded: For any method name, only one method per class may be defined with that name.

If no constructors are defined, the compiler defines a default constructor. Unlike methods, they may be overloaded. When the programmer declares an instance of the class, either a user-defined constructor or the default constructor is automatically called. It is a compile-time error to declare two constructors with equivalent signatures in a class.

Inheritance

Dice supports multiple levels of inheritance. The syntax for declaring a class that inherits from another class via the **extends** keyword is in the "Declarations" subsection of the "Statements" section. A class inherits the public fields and methods of all its ancestors. Constructors are not inherited.

Overriding

A class can override any inherited method by defining its own method with the same method signature and a custom body. Two method signatures are considered to be the same if they match on their return type and name and have the same number of formal arguments, with the sequence of types of their formals matching. Constructor declarations are never inherited and therefore are not subject to overriding.

Access Modifiers

Fields and methods must have one of the following access modifiers: **public** | **private**. If a field or method has a public access modifier, then it may be accessed by the method of any class in the program. Private fields and methods are accessible from within the class in which they are declared, but not from any descendant classes.

Unlike fields and methods, access to constructors is not governed by access modifiers. Constructors are accessible from any class.

Referencing instances

When the keyword **this** is used that keyword is effectively replaced with an instance of the containing object at runtime.

Grammar

Below you will find an entire grammar listing for our language. You will see several tokens that were generated directly from our Scanner. The following are the list of tokens and their associated regexes:

```

1  let alpha = ['a'-'z' 'A'-'Z']
2  let escape = '\\\' ['\\\' \'\' \'\"\' 'n' 'r' 't']
3  let escape_char = \'\' (escape) \'\'
4  let ascii = ([\' \'-'!\' \'#\'-'[\' \']\'-'~\'])
5  let digit = ['0'-'9']
6  let id = alpha (alpha | digit | \'_\' ) *
7  let string = \'\' ( (ascii | escape)* as s) \'\'
8  let char = \'\' ( ascii | digit ) \'\'
9  let float = (digit+) [\'.\'] digit+
10 let int = digit+
11
12 | '('      { LPAREN }
13 | ')'      { RPAREN }
14 | '{'      { LBRACE }
15 | '}'      { RBRACE }
16 | ';'      { SEMI }
17 | ','      { COMMA }
18 | '+'      { PLUS }
19 | '-'      { MINUS }
20 | '*'      { TIMES }
21 | '/'      { DIVIDE }
22 | '%'      { MODULO }
23 | '='      { ASSIGN }
24 | '=='     { EQ }
25 | '!='     { NEQ }
26 | '<'      { LT }
27 | '<='     { LEQ }
28 | '>'      { GT }
29 | '>='     { GEQ }
30 | 'and'    { AND }
31 | 'or'     { OR }
32 | 'not'    { NOT }
33 | '.'      { DOT }
34 | '['      { LBRACKET }
35 | ']'      { RBRACKET }
36 | '|'      { BAR }
37 | 'if'     { IF }
38 | 'else'   { ELSE }
39 | 'for'    { FOR }
40 | 'while'  { WHILE }
41 | 'return' { RETURN }
42 | 'int'    { INT }
43 | 'float'  { FLOAT }
44 | 'bool'   { BOOL }
45 | 'char'   { CHAR }

```

```

46 | "void"    { VOID }
47 | "null"   { NULL }
48 | "true"   { TRUE }
49 | "false"  { FALSE }
50 | "class"   { CLASS }
51 | "constructor" { CONSTRUCTOR }
52 | "public"  { PUBLIC }
53 | "private" { PRIVATE }
54 | "extends" { EXTENDS }
55 | "include" { INCLUDE }
56 | "this"    { THIS }
57 | "break"   { BREAK }
58 | "continue" { CONTINUE }
59 | "new"     { NEW }
60 | "delete"  { DELETE }
61
62 | int as lxm          { INT_LITERAL(int_of_string lxm) }
63 | float as lxm       { FLOAT_LITERAL(float_of_string lxm) }
64 | char as lxm        { CHAR_LITERAL( String.get lxm 1 ) }
65 | escape_char as lxm { CHAR_LITERAL( String.get (unescape lxm) 1) }
66 | string            { STRING_LITERAL(unescape s) }
67 | id as lxm         { ID(lxm) }
68 | eof              { EOF }
69
70 | (* *) {COMMENT*}

```

It should be noted that comments were handled to allow for nested comments. Therefore this cannot be captured strictly using a grammar, and instead is better shown in the scanner.mll documentation at the end of this document. The following grammar is the same as the grammar shown in parser.mly at the end of this document except it does not have the rules it will turn into regarding OCaml code. This is very similar to the syntax for ocaml yacc.

```

1 program:
2     includes cdecls EOF
3
4 includes:
5     /* nothing */
6     | include_list
7
8 include_list:
9     include_decl
10    | include_list include_decl
11
12 include_decl:
13     INCLUDE LPAREN STRING_LITERAL RPAREN SEMI
14
15 cdecls:
16     cdecl_list
17
18 cdecl_list:

```

```
19         cdecl
20         |         cdecl_list cdecl
21
22 cdecl:
23         CLASS ID LBRACE cbody RBRACE
24         |         CLASS ID EXTENDS ID LBRACE cbody RBRACE
25
26 cbody:
27         /* nothing */
28         |         cbody field
29         |         cbody constructor
30         |         cbody fdecl
31
32 constructor:
33         CONSTRUCTOR LPAREN formals_opt RPAREN LBRACE stmt_list RBRACE
34
35 scope:
36         PRIVATE
37         |         PUBLIC
38
39 field:
40         scope datatype ID SEMI
41
42 fname:
43         ID
44
45 fdecl:
46         scope datatype fname LPAREN formals_opt RPAREN LBRACE stmt_list RBRACE
47
48 formals_opt:
49         /* nothing */
50         |         formal_list
51
52 formal_list:
53         formal
54         |         formal_list COMMA formal
55
56 formal:
57         datatype ID
58
59 actuals_opt:
60         /* nothing */
61         |         actuals_list
62
63 actuals_list:
64         expr
65         |         actuals_list COMMA expr
66
67 primitive:
```

```

68             INT
69         |             FLOAT
70         |             CHAR
71         |             BOOL
72         |             VOID
73
74 name:
75     CLASS ID
76
77 type_tag:
78     primitive
79     |
80     name
81
82 array_type:
83     type_tag LBRACKET brackets RBRACKET
84
85 datatype:
86     type_tag
87     |
88     array_type
89
90 brackets:
91     /* nothing */
92     |
93     brackets RBRACKET LBRACKET
94
95 stmt_list:
96     /* nothing */
97     |
98     stmt_list stmt
99
100 stmt:
101     expr SEMI
102     |
103     RETURN expr SEMI
104     |
105     RETURN SEMI
106     |
107     LBRACE stmt_list RBRACE
108     |
109     IF LPAREN expr RPAREN stmt
110     |
111     IF LPAREN expr RPAREN stmt ELSE stmt
112     |
113     FOR LPAREN expr_opt SEMI expr_opt SEMI expr_opt RPAREN stmt
114     |
115     WHILE LPAREN expr RPAREN stmt
116     |
117     BREAK SEMI
118     |
119     CONTINUE SEMI
120     |
121     datatype ID SEMI
122     |
123     datatype ID ASSIGN expr SEMI
124
125 expr_opt:
126     /* nothing */
127     |
128     expr
129
130 expr:
131     literals
132     |
133     expr PLUS expr

```

```
117      |      expr MINUS expr
118      |      expr TIMES expr
119      |      expr DIVIDE expr
120      |      expr EQ      expr
121      |      expr NEQ      expr
122      |      expr LT      expr
123      |      expr LEQ      expr
124      |      expr GT      expr
125      |      expr GEQ      expr
126      |      expr AND      expr
127      |      expr MODULO   expr
128      |      NOT expr
129      |      expr OR       expr
130      |      expr DOT      expr
131      |      expr ASSIGN   expr
132      |      DELETE expr
133      | MINUS expr
134      |      ID LPAREN actuals_opt RPAREN
135      |      NEW ID LPAREN actuals_opt RPAREN
136      |      NEW type_tag bracket_args RBRACKET
137      |      expr bracket_args RBRACKET
138      |      LPAREN expr RPAREN
139
140 bracket_args:
141     LBRACKET expr
142     |      bracket_args RBRACKET LBRACKET expr
143
144 literals:
145     INT_LITERAL
146     |      FLOAT_LITERAL
147     |      TRUE
148     |      FALSE
149     |      STRING_LITERAL
150     |      CHAR_LITERAL
151     |      THIS
152     |      ID
153     |      NULL
154     |      BAR array_prim BAR
155
156 array_prim:
157     expr
158     |      array_prim COMMA expr
```

4. PROJECT PLAN

Planning Process

Throughout the project we embodied the principles of agile development. At any point in time during our development we had working code on the master branch and every member of the team was brought up to speed with what has been completed and worked on. All goals for the project were put on Github and as they were resolved they were cleared. We created several milestones which captured our goals for completing the parser, scanner, analyzer, codegen, and final report milestones. We also worked closely with Professor Edwards at Columbia University to receive guidance on how best to implement this language. The following milestones were created and cleared over the course of the semester:

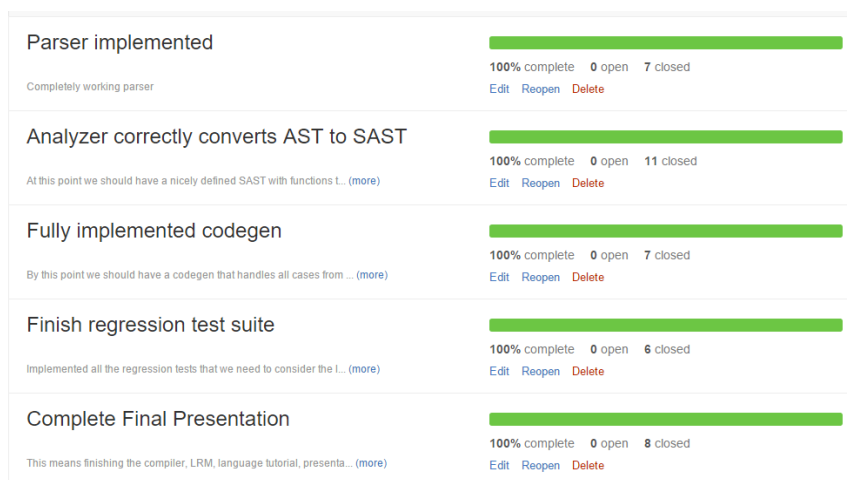


Figure 4.1: Milestoning on Github.

Specification Process

At the beginning of the semester we had originally intended our language to be a distributed software solution that would conveniently allow the developer to distribute tasks to various slave machines that had compiled the tasks to LLVM IR. After discussing this with professor Edwards we then decided to opt for an object oriented programming language that specifically compiled to LLVM IR. This way we as a team could learn more about making compilers and showing the power of LLVM.

Once we decided on the theme of Dice, we met to discuss the features we wanted most in our object oriented language. In our case we wanted arrays, inheritance, objects, and file IO to be some of the key highlights of our language. We then built up the scanner and parser to get a more solid idea as to what the language would look like, and by November 15th we had solidified our plans to implement the aforementioned features.

Development Process

Implementation was very dependent on the course deadlines. We started with the scanner and parser specifically so the language reference manual was better defined. This was completed by October 26th. We then iterated on the analyzer and codegen until it was capable of producing hello world. This was completed on November 15th. The month afterwards was spent implementing inheritance and arrays until they were finally completed on December 18th.

Testing Process

Throughout the development process we had numerous tests. The plan was to always have tests that were non-functional so a feature could then be implemented to get them working. If we encountered an error that we were unsure of how to fix, we added more error messages in our compiler until we could exactly pinpoint where the error was occurring. We also made a rule for our team to handle each and every exception that could occur as a custom error message to be printed out by the compiler.

Team Responsibilities

Team responsibilities were divided up and evenly distributed amongst the four group members. While we could not adhere to a strict division of labor based on group member titles, every member contributed to the codebase.

Team Member	Responsibility
David Watkins	Scanner, Parser, Analyzer, Codegen, Utils, LRM, Final Report, Latex, Code cleanup
Emily Chen	Inheritance in Analyzer, Expression types in Analyzer, LRM
Khaled Atef	Test Suite, Binary and unary expression evaluation in codegen
Phillip Schiffrin	Standard Library, Class map generation

Github Usernames

The following Github usernames correspond to the following group members:

- Emily Chen - six5532one, ec2805
- Khaled Atef - KhaledAtef
- David Watkins - DavidWatkins
- Philip Schiffrin - nethacker11

Project Log

To demonstrate our timeline we captured the number of git commits over time for our project.

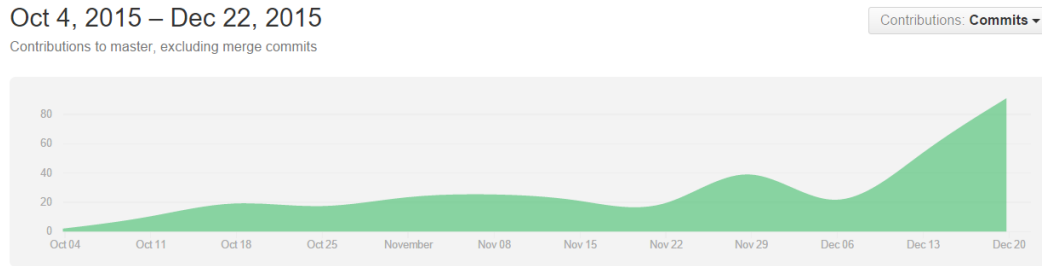


Figure 4.2: Commit timeline on Github.

The timeline shows that we have been diligent at constantly working on the project since the beginning of the semester. All group members have contributed to this project. The following issues are a list of git issues that were cleared as part of our project, as well as the person who closed the issue. We did not have a rule for who closed an issue so sometimes the person who completed the issue may not have been the one to close it.

- #71 Should not be able to access variables outside of scope
- #137 Awesome!
- #134 Subclass assignment [by @six5532one, @DavidWatkins]
- #133 string length tests
- #132 fix delete test, no multiple arrays
- #131 this should raise no exceptions
- #130 Expected stderr: "exception Exceptions.LocalAssignTypeMismatch("B", "C")"
- #129 passing in an inherited class for classes
- #128 E-test-privateFieldsAccess.dice
- #127 Create test for cyclical inheritance
- #126 Add error message for assigning parameters
- #125 test-gcd.dice Bug. You cannot assign values to parameters
- #124 test-constructor1.dice is written incorrectly
- #123 Maximum float is limited to 6 digits after the decimal
- #122 char[][] args does not work in main
- #121 Test max/min floats
- #120 Test default constructor
- #119 Test overloading std-lib functions
- #118 Exit not working in runtime
- #117 Test args

- #116 assign ints to floats
- #115 Integer toString generates string twice
- #114 concat adds an extra character to the string
- #113 Test exit
- #111 Errors.log from script output isn't working properly
- #110 add teststdlib .out
- #109 Add Test returning objects
- #108 add tests for empty blocks
- #107 For inheritance of functions we should have an id to determine which function to call
- #106 Includes should check with String_lit not ID
- #105 Odd invalid numbering of blocks bug
- #104 Fix parameters on library functions
- #103 Get Dice exec working so tests can run again.
- #102 "Get the t-shirts made"
- #101 Adapt codegen to changes in analyzer that add inherited fields to sprogram.classes
- #100 Need to test includes
- #99 add test for empty conditionals
- #98 add empty for loop test
- #97 Add nested comments
- #96 test order of fdecl,fields,constructor in classes
- #95 primitive type limit tests
- #94 test constructors
- #93 test private scope function
- #92 Help needed: env.env_class_maps seems correct but exception is raised when I try to access an inherited field
- #91 default constructors
- #90 Need to add an environment variable to point to the includes
- #89 Strings need to be initialized and accessed differently from normal arrays
- #88 This should raise "UndefinedClass: H"
- #87 Use of Delete
- #86 add static scoping test
- #85 Add applicative order test

- #84 Add delete command to free memory
- #82 Add exit call
- #81 return statements in branches aren't recognized
- #80 dice executable doesn't run without any args
- #79 Kappa [by @DavidWatkins]
- #78 Add tests for recursion
- #77 Obj access [by @DavidWatkins]
- #75 Test invalid functions
- #74 Test multiple classes
- #73 Parent cannot have fields of type of its children
- #72 Cannot call return inside of a constructor
- #135 check for overridden methods takes ret type into account [by @six5532one, @DavidWatkins]
- #69 Casting rules questions
- #68 Kappa [by @DavidWatkins]
- #67 Floats print with extra trailing zeros. Kinda ugly.
- #66 Emily [by @six5532one, @DavidWatkins]
- #65 local decl (primitives): stderr should be "DuplicateLocal: myc"
- #64 object creation: this should raise no exception
- #63 object creation: this should raise no exceptions
- #62 Compiler doesn't allow formal to be an object
- #61 object creation: This should throw no exceptions
- #60 Object creation: this should raise "ConstructorNotFound: Foo.constructor.int.bool.char.float"
- #59 object decl without assignment expr: This should throw no exceptions
- #58 This should throw exception "UndefinedClass: Baz"
- #57 incorrect check for duplicate constructors
- #56 Emily [by @six5532one, @DavidWatkins]
- #55 Create arith tests that have signed values
- #54 Parser issue with reading user-defined objects.
- #53 Emily [by @six5532one]
- #52 Decide whether to promote all ints to floats in binops
- #51 Consecutive print statements don't work. Compiler only outputs first print statement.

- #50 Epsilon [by @six5532one]
- #49 Reorganize object accesses for functions
- #46 Kreygasm [by @DavidWatkins]
- #45 Add shakespeare and stephen number to tester
- #44 Create symbol table for cdecls, fdecls, fields
- #39 static analysis checks for variable access
- #38 use 'new' keyword for object and array instantiation
- #37 support addition of chars and ints
- #36 Update LRM: support addition of chars and ints
- #35 Change parser array create type to type tag and not primitive
- #34 Evaluate whether to add new as a keyword to object initialization
- #33 Exceptions, try, catch?
- #32 Implement basic primitive expressions for codegen
- #31 Should we add continuous checking even when an illegal character/parser error occurs like java?
- #30 Add annotation for source code position to AST
- #29 We should evaluate whether we want to move variable declarations to stmts
- #28 Do we need to add an additional layer of abstraction from SAST to Codegen?
- #27 Complete pretty printing abstract syntax tree to Utils
- #26 How does LLVM handle allocating on the heap
- #24 Strings with escape characters are not being displayed properly
- #23 Create OCamlDoc Documentation
- #22 Should we switch the llvm package to ollvm?
- #21 Add file operator functions to Codegen
- #20 Write the File class
- #19 Write the String class
- #18 Write the Math class
- #17 Add support for utilizing line number and character number in Analyzer
- #16 Add class name and function name collision detection
- #15 Add testing for arrays
- #14 Evaluate the type of an expression in Analyzer.get_expr_type
- #13 Add testing for extends

- #12 Add mentioning of unary minus to LRM
- #11 Remove '-' symbol from regex in floats and ints of LRM
- #10 Convert AST.cdecl to SAST.cdecl
- #9 Convert AST.expr to SAST.expr in Analyzer.convert_expr
- #8 Analyzer.process_includes does not check absolute path
- #7 Delta [by @DavidWatkins]
- #6 Delta [by @DavidWatkins]
- #5 Special chars (tabs/newlines/etc) aren't getting tokenized properly
- #4 float limit
- #3 David fix [by @DavidWatkins]
- #2 Merge pull request #1 from DavidWatkins/DavidFix [by @DavidWatkins]
- #1 David fix [by @DavidWatkins]

Git Commit History

Here are all of the commits as performed by the team. Everyone contributed to the project.

```
1  commit dda28468a3b4742c94d7913c4bc6ff0b0a99bd90
2  Author: David Watkins <davidw@tkins.me>
3  Date:   Tue Dec 22 23:38:27 2015 -0500
4
5      works now
6
7  commit f3b2aa7577bc33ad86e4d0f9e6fdc7c006bd9bdc
8  Merge: 2a272c5 43b5f73
9  Author: David Watkins <davidw@tkins.me>
10 Date:   Tue Dec 22 23:36:10 2015 -0500
11
12      Merge branch 'master' of https://github.com/DavidWatkins/Dice
13
14  commit 2a272c53b0cd524509e328ef9cd6792212db959b
15  Author: David Watkins <davidw@tkins.me>
16  Date:   Tue Dec 22 23:35:52 2015 -0500
17
18      New tarball
19
20  commit 43b5f73cf52535d79b4aee7ab869e4422eedf7da
21  Merge: 8687402 7cec151
22  Author: David Watkins <djrival7@gmail.com>
23  Date:   Tue Dec 22 23:29:43 2015 -0500
24
25      Merge branch 'master' of https://github.com/DavidWatkins/DiceLanguage
26
```

```
27 commit 86874025d9fa49779ddce79a422fe48f81a0b324
28 Author: David Watkins <djrival7@gmail.com>
29 Date: Tue Dec 22 23:29:19 2015 -0500
```

```
30
31 Updated references
```

```
32
33 commit 7cec1517ad01b7373fb61978584885e74097b339
34 Author: David Watkins <davidw@tkins.me>
35 Date: Tue Dec 22 23:29:07 2015 -0500
```

```
36
37 tarball
```

```
38
39 commit f27267ea354a2cc3a0cf2fb32b68971d6e8e1064
40 Author: David Watkins <djrival7@gmail.com>
41 Date: Tue Dec 22 23:23:14 2015 -0500
```

```
42
43 Updated
```

```
44
45 commit 71474ab6741c314246a7e5d115573f66ceee279c
46 Author: David Watkins <djrival7@gmail.com>
47 Date: Tue Dec 22 23:14:10 2015 -0500
```

```
48
49 Final version
```

```
50
51 commit 4ea7bf79afd1e7158cda2693406534979162c0f3
52 Author: David Watkins <djrival7@gmail.com>
53 Date: Tue Dec 22 23:07:30 2015 -0500
```

```
54
55 Fixed LRM
```

```
56
57 commit 483c33d46296f05db29a66e8ed5f04ca7bc10253
58 Author: David Watkins <davidw@tkins.me>
59 Date: Tue Dec 22 23:05:35 2015 -0500
```

```
60
61 Fixed some tests
```

```
62
63 commit 0648fe1c5d2df25c899fa519db295f8509826962
64 Merge: fdee613 6191e9b
65 Author: Khaled Atef <kaa2168@columbia.edu>
66 Date: Tue Dec 22 22:51:41 2015 -0500
```

```
67
68 Merge branch 'master' of https://github.com/DavidWatkins/Dice
```

```
69
70 commit fdee61348f8641ac0c057a679c1790a027622ad8
71 Author: Khaled Atef <kaa2168@columbia.edu>
72 Date: Tue Dec 22 22:51:10 2015 -0500
```

```
73
74 Fixed tests to match new error messages. This should be the last time I touch these tests. I'm tired
```

```
75
```

```
76 commit 6191e9b1d1f37d1699dc1f2cc95d0bed8588ff8d
77 Merge: cc84574 4552c81
78 Author: David Watkins <djrival7@gmail.com>
79 Date: Tue Dec 22 22:40:32 2015 -0500
80
81 Merge branch 'master' of https://github.com/DavidWatkins/DiceLanguage
82
83 commit cc845741ba0de805ab5defdba8b9f83b48714b55
84 Author: David Watkins <djrival7@gmail.com>
85 Date: Tue Dec 22 22:40:08 2015 -0500
86
87 Finished Tutorial
88
89 commit 4552c811c8978550a7b910b665d1cfc5e700b3f1
90 Merge: cdb0e4f 5201688
91 Author: Emily Chen <ec2805@columbia.edu>
92 Date: Tue Dec 22 22:37:03 2015 -0500
93
94 Merge branch 'master' of https://github.com/DavidWatkins/Dice
95
96 commit cdb0e4f2e8ac1d288b4f5490edb4f4049f1aec63
97 Author: Emily Chen <ec2805@columbia.edu>
98 Date: Tue Dec 22 22:36:22 2015 -0500
99
100 fix type specifiers for non-primitive types
101
102 commit 5201688e6ceff8a3d48e7d912297b6ad52c2fc3c
103 Merge: 5b88cc0 275a5f0
104 Author: Khaled Atef <kaa2168@columbia.edu>
105 Date: Tue Dec 22 22:28:00 2015 -0500
106
107 Merge branch 'master' of https://github.com/DavidWatkins/Dice
108
109 commit 5b88cc01e1bf17b11db1fa57d990612867a8f618
110 Author: Khaled Atef <kaa2168@columbia.edu>
111 Date: Tue Dec 22 22:27:50 2015 -0500
112
113 Removed the 812981298129
114
115 commit 4bd7f0a61871fca70425d7d7b7cd82ec8d36204e
116 Author: Khaled Atef <kaa2168@columbia.edu>
117 Date: Tue Dec 22 22:27:06 2015 -0500
118
119 Added to Tutorial
120
121 commit 275a5f012f3a70dc3a2ff2dcb9e804cb54a29fa9
122 Author: David Watkins <djrival7@gmail.com>
123 Date: Tue Dec 22 22:25:54 2015 -0500
124
```

```
125     Done with Architecture
126
127     commit cfd9dee0069b201d9bed13089afb22becf973bca
128     Author: Emily Chen <ec2805@columbia.edu>
129     Date:    Tue Dec 22 22:20:39 2015 -0500
130
131     updated Classes
132
133     commit 7943a55f63ac9acb77e4e6eef73060146d319bf2
134     Author: David Watkins <djrival7@gmail.com>
135     Date:    Tue Dec 22 22:10:11 2015 -0500
136
137     Look at me
138
139     commit 46e4516edd6d72f4a4f3a451eae9de63d51b48e0
140     Merge: a97dddc 65f9ed5
141     Author: David Watkins <djrival7@gmail.com>
142     Date:    Tue Dec 22 21:49:57 2015 -0500
143
144     Merge branch 'master' of https://github.com/DavidWatkins/DiceLanguage
145
146     commit a97dddc5745f0db4badd426a61ef0aaa45c59d3
147     Author: David Watkins <djrival7@gmail.com>
148     Date:    Tue Dec 22 21:47:01 2015 -0500
149
150     Added new stuf
151
152     commit 65f9ed5e72520c2f856ada1ef21bfa84d9261676
153     Author: nethacker11 <philip.schiffrin@gmail.com>
154     Date:    Tue Dec 22 21:41:10 2015 -0500
155
156     updated Architecture.tex for code generation
157
158     commit 7452f7b962d3324283af73b5adb8a22fe435072b
159     Author: Emily Chen <ec2805@columbia.edu>
160     Date:    Tue Dec 22 21:27:16 2015 -0500
161
162     add restrictions to class name, method name
163
164     commit e1797fbd385bc059ab83150c5813abda31789912
165     Merge: fbf4ba9 738b055
166     Author: nethacker11 <philip.schiffrin@gmail.com>
167     Date:    Tue Dec 22 21:22:42 2015 -0500
168
169     Merge branch 'master' of https://github.com/DavidWatkins/Dice
170
171     commit fbf4ba9a85e6a17419bfdf5c93cdbf592ece31a
172     Author: nethacker11 <philip.schiffrin@gmail.com>
173     Date:    Tue Dec 22 21:22:26 2015 -0500
```

```
174
175     updated Architecture.tex
176
177 commit 55ca1d128472a5cb1451784d89be2e174d455ced
178 Author: Emily Chen <ec2805@columbia.edu>
179 Date:   Tue Dec 22 21:10:47 2015 -0500
180
181     mention built-in functions as reserved
182
183 commit 4120bbefa038a2a3f7fda5c682e286e7d71327aa
184 Author: Emily Chen <ec2805@columbia.edu>
185 Date:   Tue Dec 22 20:58:29 2015 -0500
186
187     rewrote statements, expressions sections
188
189 commit 738b0558ddb9fe894a7611be0f1f9f590f38094a
190 Merge: 700e197 df6915a
191 Author: David Watkins <djrival7@gmail.com>
192 Date:   Tue Dec 22 20:45:33 2015 -0500
193
194     Merge branch 'master' of https://github.com/DavidWatkins/DiceLanguage
195
196 commit 700e1979474d977ffb3496c7435f4f9dbace09e2
197 Author: David Watkins <djrival7@gmail.com>
198 Date:   Tue Dec 22 20:39:19 2015 -0500
199
200     Added changes to standard library description
201
202 commit df6915a7d7a802e673daca3a6b364060b024035b
203 Merge: 8ac36b8 dbf27d2
204 Author: nethacker11 <philip.schiffrin@gmail.com>
205 Date:   Tue Dec 22 20:34:58 2015 -0500
206
207     Merge branch 'master' of https://github.com/DavidWatkins/Dice
208
209 commit 8ac36b8a6d9714d1f096f8c8e990da9bd971afe7
210 Author: nethacker11 <philip.schiffrin@gmail.com>
211 Date:   Tue Dec 22 20:34:51 2015 -0500
212
213     CFuncs.tex added
214
215 commit dbf27d2d940c93f0de61a210f98167faefeae014
216 Author: David Watkins <djrival7@gmail.com>
217 Date:   Tue Dec 22 20:28:22 2015 -0500
218
219     Added grammar and small changes to lrm
220
221 commit 421588dcb8b30f42134c880143492e4822dbba2e
222 Author: nethacker11 <philip.schiffrin@gmail.com>
```



```
223 Date: Tue Dec 22 20:23:33 2015 -0500
224
225 added Builtin.tex
226
227 commit 0ec68907641d9be8a992b3dd7b023ec8e4f48afc
228 Merge: ef75162 ea3b98f
229 Author: David Watkins <djrival7@gmail.com>
230 Date: Tue Dec 22 19:46:19 2015 -0500
231
232 Merge branch 'master' of https://github.com/DavidWatkins/DiceLanguage
233
234 commit ef75162bd113e48a2ba794aa7ce002b613eeae3c
235 Author: David Watkins <djrival7@gmail.com>
236 Date: Tue Dec 22 19:45:53 2015 -0500
237
238 Added additional code for test plan in final report
239
240 commit ea3b98f6be0be4f8a66158b94d8391cd7b719948
241 Merge: 8524dfd 6378550
242 Author: nethacker11 <philip.schiffrin@gmail.com>
243 Date: Tue Dec 22 19:45:41 2015 -0500
244
245 Merge branch 'master' of https://github.com/DavidWatkins/Dice
246
247 commit 8524dfd397ddf421dcf7c7bd948649a825c355f5
248 Author: nethacker11 <philip.schiffrin@gmail.com>
249 Date: Tue Dec 22 19:45:33 2015 -0500
250
251 updated standard library in Library.tex
252
253 commit 63785501516be9117a65f4bc0908396b5496058c
254 Merge: 48d7e07 035c054
255 Author: David Watkins <djrival7@gmail.com>
256 Date: Tue Dec 22 19:12:36 2015 -0500
257
258 Merge branch 'master' of https://github.com/DavidWatkins/DiceLanguage
259
260 commit 48d7e0772b5cfe8e899efecb622041b198199497
261 Author: David Watkins <djrival7@gmail.com>
262 Date: Tue Dec 22 19:12:14 2015 -0500
263
264 Added additional stuff to proposal and tutorial
265
266 commit 035c054a00bf0ccc1b2b8d2dc1809f6fbab4dc08
267 Author: Khaled Atef <kaa2168@columbia.edu>
268 Date: Tue Dec 22 19:11:12 2015 -0500
269
270 Added Test Plan and Khal lessons learned to Final Report directory
271
```

```
272 commit 39a768eca63505299bfacc07eef0322753a4de64
273 Author: nethacker11 <philip.schiffrin@gmail.com>
274 Date: Tue Dec 22 18:59:25 2015 -0500
275
276     updated Syntax.tex for final report
277
278 commit 41e9106396bb0b2e693dd35bfd131151c7c1b641
279 Author: David Watkins <djw2146@columbia.edu>
280 Date: Tue Dec 22 18:28:54 2015 -0500
281
282     ADedd more stuf
283
284 commit c58d595f376df552bb65e1fdb33ec05a674eb8cd
285 Author: David Watkins <davidw@tkins.me>
286 Date: Tue Dec 22 18:23:32 2015 -0500
287
288     Added Demo_Animals to tex file
289
290 commit afa84191ecd40be39d14295c6c1f3fa25e7be6f6
291 Author: David Watkins <davidw@tkins.me>
292 Date: Tue Dec 22 18:19:54 2015 -0500
293
294     Fixed hello world demo breaking tests
295
296 commit 7e2a1b9e07040cb9929b5dc971a297c83b0a9fe1
297 Author: David Watkins <davidw@tkins.me>
298 Date: Tue Dec 22 18:14:31 2015 -0500
299
300     iejsiu
301
302 commit ab07735004b3f480677e269e65e9f009e9f10bdb
303 Author: David Watkins <davidw@tkins.me>
304 Date: Tue Dec 22 18:12:52 2015 -0500
305
306     ijij
307
308 commit b21f0885522047bf0a62afb4da5edb292958ade4
309 Author: David Watkins <davidw@tkins.me>
310 Date: Tue Dec 22 18:11:09 2015 -0500
311
312     Maybe this works?
313
314 commit 727a2a262837bcf87371f1d5313e065b5a855d36
315 Author: Emily Chen <ec2805@columbia.edu>
316 Date: Tue Dec 22 16:36:04 2015 -0500
317
318     finished updating expressions
319
320 commit 5dd98b2548b8718ebf8342ef3460bfa740a6ffad
```

```
321 Author: David Watkins <davidw@tkins.me>
322 Date: Tue Dec 22 15:20:54 2015 -0500
323
324 Fixed another bug
325
326 commit d6b49aae775f433bc4c733ef539f9d5b84605c6f
327 Author: David Watkins <djwt146@columbia.edu>
328 Date: Tue Dec 22 15:19:31 2015 -0500
329
330 updated code.texY
331
332 commit d92d49c30ebe2da66203d0eb28db41d78b0d9ec5
333 Author: David Watkins <davidw@tkins.me>
334 Date: Tue Dec 22 15:17:11 2015 -0500
335
336 Fixed tests
337
338 commit cfebb0d5104705df4e358b35879645c6f5190439
339 Author: David Watkins <davidw@tkins.me>
340 Date: Tue Dec 22 15:09:13 2015 -0500
341
342 Fixed section title on tests
343
344 commit b8e048f0a326c3fc4fccee0a99928aa0564f8233
345 Merge: e94920a 07ee0b6
346 Author: David Watkins <davidw@tkins.me>
347 Date: Tue Dec 22 15:06:33 2015 -0500
348
349 Merge branch 'master' of https://github.com/DavidWatkins/Dice
350
351 commit e94920ae5f16643a0f5ae85393d7cae7e8dc58f5
352 Author: David Watkins <davidw@tkins.me>
353 Date: Tue Dec 22 15:06:16 2015 -0500
354
355 Added code for adding tests to final report
356
357 commit 07ee0b6cc870f6a7c171d159a85f8c142807f6f7
358 Author: David Watkins <DavidWatkins@users.noreply.github.com>
359 Date: Tue Dec 22 13:59:47 2015 -0500
360
361 Update README.md
362
363 commit a16003fbdec97727c492c857335bc93478a50a70
364 Author: David Watkins <djwt146@columbia.edu>
365 Date: Tue Dec 22 05:15:01 2015 -0500
366
367 Added basis for final project report
368
369 commit f3e5fe83dae72565f2950c096c6ff0efecb1b567
```

```
370 Author: David Watkins <davidw@tkins.me>
371 Date: Tue Dec 22 04:46:38 2015 -0500
372
373     Need to fixed error tests
374
375 commit e16dc0448ac4444fe75f7fee46b10825fda2ba6d
376 Author: David Watkins <djrival7@gmail.com>
377 Date: Mon Dec 21 20:14:33 2015 -0500
378
379     Added presentation
380
381 commit 0bc2d56336f2bed25b1715a1b5c632a49147eea8
382 Author: Khaled Atef <kaa2168@columbia.edu>
383 Date: Mon Dec 21 15:25:43 2015 -0500
384
385     Logo modified
386
387 commit d39a5d9feb9ba50426b6caa3c32668ab57c410c5
388 Author: David Watkins <davidw@tkins.me>
389 Date: Mon Dec 21 14:23:01 2015 -0500
390
391     Finished demo code
392
393 commit c0ccf162f43b88ef2c732de15acd419250e5db5c
394 Author: David Watkins <davidw@tkins.me>
395 Date: Mon Dec 21 14:21:42 2015 -0500
396
397     Removed unnecessary files
398
399 commit a03afb187f7b93c8c05874e1357975d3edf69fac
400 Author: David Watkins <davidw@tkins.me>
401 Date: Mon Dec 21 13:55:16 2015 -0500
402
403     Fixed the demo
404
405 commit eec6e6f7989d4022ac261cc453bb7646e84e0a69
406 Author: Khaled Atef <kaa2168@columbia.edu>
407 Date: Mon Dec 21 07:31:49 2015 -0500
408
409     input/output coordinated
410
411 commit ca6abe8eeda764edfa1c2abd2bce730619ee53c9
412 Author: Khaled Atef <kaa2168@columbia.edu>
413 Date: Mon Dec 21 07:02:23 2015 -0500
414
415     basics implemented for demo
416
417 commit 8d2eda8d25c81a0294b3cc52c285c76314600870
418 Author: Khaled Atef <kaa2168@columbia.edu>
```

```
419 Date:   Mon Dec 21 06:23:25 2015 -0500
420
421     modified ascii art for demo
422
423 commit 0a3a0c3958e224b1883714e99bd317624dd5514b
424 Merge: 96d30dd 2437414
425 Author: Khaled Atef <kaa2168@columbia.edu>
426 Date:   Mon Dec 21 06:18:42 2015 -0500
427
428     Merge branch 'master' of https://github.com/DavidWatkins/Dice
429
430 commit 96d30ddc28576c7013902f157a5435315967ddd1
431 Author: Khaled Atef <kaa2168@columbia.edu>
432 Date:   Mon Dec 21 06:18:36 2015 -0500
433
434     file for demo
435
436 commit 24374142973e158c61ea3955ac8d963599a2b75d
437 Author: Khaled Atef <kaa2168@columbia.edu>
438 Date:   Mon Dec 21 05:57:03 2015 -0500
439
440     Othello still broken after many compiler errors
441
442 commit b5fba0a2101e0c18d6bc476f0e7dfc18539c356
443 Merge: bff1792 502eff9
444 Author: Emily Chen <emchennyc@gmail.com>
445 Date:   Mon Dec 21 02:39:56 2015 -0500
446
447     Merge branch 'master' of https://github.com/DavidWatkins/Dice
448
449 commit bff17927857bd562451279c9109ba57f01469829
450 Author: Emily Chen <emchennyc@gmail.com>
451 Date:   Mon Dec 21 02:39:00 2015 -0500
452
453     halfway through translating OthelloGame
454
455 commit 502eff9a39c369dcd131c4b36220018c0e16fbc4
456 Merge: 4da809d 79744e6
457 Author: nethacker11 <philip.schiffrin@gmail.com>
458 Date:   Mon Dec 21 02:35:23 2015 -0500
459
460     Merge branch 'master' of https://github.com/DavidWatkins/Dice
461
462 commit 4da809d3964870b705e10f8126e77e80c152474f
463 Author: nethacker11 <philip.schiffrin@gmail.com>
464 Date:   Mon Dec 21 02:34:47 2015 -0500
465
466     updated humanplayer, doesn't work
467
```

```
468 commit 79744e6e61a16d7e049323d5af621e6be2049bb6
469 Merge: 1086a20 76df32a
470 Author: Khaled Atef <kaa2168@columbia.edu>
471 Date: Mon Dec 21 02:10:20 2015 -0500
472
473 Merge branch 'master' of https://github.com/DavidWatkins/Dice
474
475 commit 1086a2003fcf4604b4b799b3c3e18cbb05901b48
476 Author: Khaled Atef <kaa2168@columbia.edu>
477 Date: Mon Dec 21 02:10:11 2015 -0500
478
479 First round of edits to parserScanner regex rules
480
481 commit 76df32ae8b70759eeddb134f57b8e3f6403e2e5f
482 Merge: 8a75b65 fb0a776
483 Author: Emily Chen <emchennyc@gmail.com>
484 Date: Mon Dec 21 02:08:18 2015 -0500
485
486 Merge branch 'master' of https://github.com/DavidWatkins/Dice
487
488 commit 8a75b65ddc464749d36e7998dcd243e8ef47b241
489 Author: Emily Chen <emchennyc@gmail.com>
490 Date: Mon Dec 21 02:07:45 2015 -0500
491
492 includes classes HumanPlayer, Player, LocationObj
493
494 commit fb0a7763290ca205303a36e595792cab8bda14b
495 Author: nethacker11 <philip.schiffrin@gmail.com>
496 Date: Mon Dec 21 02:04:24 2015 -0500
497
498 updated demo files
499
500 commit a7e0a84173eee4c06f0413a7b8bde8c3a3ee1844
501 Author: nethacker11 <philip.schiffrin@gmail.com>
502 Date: Mon Dec 21 01:10:57 2015 -0500
503
504 updated demo stuff
505
506 commit c5882be1259eee843e06004c347cc1d047c79851
507 Merge: e91324a 15fe681
508 Author: nethacker11 <philip.schiffrin@gmail.com>
509 Date: Sun Dec 20 23:38:05 2015 -0500
510
511 Merge branch 'master' of https://github.com/DavidWatkins/Dice
512
513 commit e91324aef67a7876f967e35b4f4a6ca323af95f7
514 Author: nethacker11 <philip.schiffrin@gmail.com>
515 Date: Sun Dec 20 23:35:45 2015 -0500
516
```

```
517     added toInteger in stdlib
518
519     commit 15fe681f3b48135f96cfcf0c191bd6989b76fad9
520     Author: Khaled Atef <kaa2168@columbia.edu>
521     Date:    Sun Dec 20 22:19:03 2015 -0500
522
523     125 tests working!
524
525     commit 9dc00916011d9c69d13ff247268e615c2b0ac122
526     Author: David Watkins <davidw@tkins.me>
527     Date:    Sun Dec 20 21:50:00 2015 -0500
528
529     OthelloRunner Basic working
530
531     commit 9451871b5f68a79f41c4c463894b0cb6cf802b1f
532     Merge: e6007de bed598a
533     Author: Khaled Atef <kaa2168@columbia.edu>
534     Date:    Sun Dec 20 21:21:19 2015 -0500
535
536     Merge branch 'master' of https://github.com/DavidWatkins/Dice
537
538     commit e6007de0f670b43d7ff183860c77b95e0d381b99
539     Author: Khaled Atef <kaa2168@columbia.edu>
540     Date:    Sun Dec 20 21:21:04 2015 -0500
541
542     first draft Othello
543
544     commit bed598a8d60c21c69228029a024e7a5c3526c77d
545     Author: David Watkins <davidw@tkins.me>
546     Date:    Sun Dec 20 21:09:58 2015 -0500
547
548     Got object access working
549
550     commit d82e1a593479bd9dd04454014feedfa7dab7f0b4
551     Author: Khaled Atef <kaa2168@columbia.edu>
552     Date:    Sun Dec 20 20:53:29 2015 -0500
553
554     fileio test output works!
555
556     commit f000aa8d545bb8450340105b070501e9c242bcf1
557     Author: Khaled Atef <kaa2168@columbia.edu>
558     Date:    Sun Dec 20 20:50:32 2015 -0500
559
560     removed delete test
561
562     commit 9a1f7cde27e9c688ec84ad76385e27ffd1e7dcb1
563     Merge: 4c82a21 41949c7
564     Author: David Watkins <davidw@tkins.me>
565     Date:    Sun Dec 20 20:45:44 2015 -0500
```

```
566
567     Merge branch 'master' of https://github.com/DavidWatkins/Dice
568
569 commit 41949c76776af134beb6de2a473e3e869403a2d5
570 Author: Khaled Atef <kaa2168@columbia.edu>
571 Date:    Sun Dec 20 20:45:29 2015 -0500
572
573     Modified output to match test
574
575 commit 4c82a21756ba8abf9aa149d16f9b949e4b3f80c4
576 Author: David Watkins <davidw@tkins.me>
577 Date:    Sun Dec 20 20:45:17 2015 -0500
578
579     test-fileio now prints and writes itself
580
581 commit f86d9cb3250e36ac60bcd42d65fce9d63bfda90
582 Merge: 39fea6b 0d28a10
583 Author: Khaled Atef <kaa2168@columbia.edu>
584 Date:    Sun Dec 20 20:40:33 2015 -0500
585
586     Merge branch 'master' of https://github.com/DavidWatkins/Dice
587
588 commit 0d28a10d1ae9333877cdadd0f7eb7c99a587d561
589 Author: David Watkins <davidw@tkins.me>
590 Date:    Sun Dec 20 20:39:55 2015 -0500
591
592     Fixed file io
593
594 commit 39fea6ba072e0eb973deadf72c28dc70140432c3
595 Author: Khaled Atef <kaa2168@columbia.edu>
596 Date:    Sun Dec 20 20:23:11 2015 -0500
597
598     new tests
599
600 commit f989f8fcd03394dd759d65be5fa93406e7300fe8
601 Merge: ea1fc65 83d8ac3
602 Author: David Watkins <DavidWatkins@users.noreply.github.com>
603 Date:    Sun Dec 20 19:17:14 2015 -0500
604
605     Merge pull request #135 from DavidWatkins/fix-overrides-check
606
607     check for overridden methods takes ret type into account
608
609 commit ea1fc652a4bdde559280c96e38a01cd5ac165783
610 Merge: 0c7039c 3163d40
611 Author: David Watkins <DavidWatkins@users.noreply.github.com>
612 Date:    Sun Dec 20 19:16:39 2015 -0500
613
614     Merge pull request #134 from DavidWatkins/subclass_assignment
```



```
615
616     Subclass assignment
617
618     commit 0c7039c8d05f1a359ce8af67ed3fc0c581770539
619     Author: David Watkins <davidw@tkins.me>
620     Date:    Sun Dec 20 19:11:32 2015 -0500
621
622     Fixed assignment of obj_access problem
623
624     commit 6aeaa4c8a0d3fe6852c80263c918334a0d22dc06
625     Author: David Watkins <davidw@tkins.me>
626     Date:    Sun Dec 20 18:51:03 2015 -0500
627
628     Fixed stringClassReverse
629
630     commit 37ac35175eb27c39665b4bf77ee71d4a566bab4a
631     Author: David Watkins <davidw@tkins.me>
632     Date:    Sun Dec 20 18:26:16 2015 -0500
633
634     Added array access on obj_access
635
636     commit 83d8ac3fa9a130f8667cd6cf82691e8738bc94d4
637     Author: Emily Chen <emchennyc@gmail.com>
638     Date:    Sun Dec 20 18:12:23 2015 -0500
639
640     check for overridden methods takes ret type into account
641
642     commit 15d429843e5c9a584fa4914936df1ba3783b212f
643     Author: David Watkins <davidw@tkins.me>
644     Date:    Sun Dec 20 18:05:48 2015 -0500
645
646     Fixed array create initialize
647
648     commit f2390b94a80cfff1c217b533cacf61d954bdfac3
649     Author: Khaled Atef <kaa2168@columbia.edu>
650     Date:    Sun Dec 20 17:49:22 2015 -0500
651
652     tests...
653
654     commit 3163d400ace38ecdc60f41b643a27b9fa60dcd26
655     Author: Emily Chen <emchennyc@gmail.com>
656     Date:    Sun Dec 20 17:44:45 2015 -0500
657
658     fixed formatting
659
660     commit ab4a07e9e55a5ce2db8f30782faa018b0762a53a
661     Author: David Watkins <davidw@tkins.me>
662     Date:    Sun Dec 20 17:39:11 2015 -0500
663
```

```
664     Changed function naming collision schema
665
666     commit e91e642ad5fbfd8a64bea0b5e2295aaeb3ff4145
667     Author: Emily Chen <emchennyc@gmail.com>
668     Date:    Sun Dec 20 17:20:20 2015 -0500
669
670     fixed subclass assignment not to raise exception with reg object creation
671
672     commit dba6456b40bf8fc2c032b34984c210c27352a4e2
673     Author: Emily Chen <emchennyc@gmail.com>
674     Date:    Sun Dec 20 16:48:52 2015 -0500
675
676     checks subclass assignment
677
678     commit 0b512528037bec86727f7e721a08d636759ef845
679     Author: Khaled Atef <kaa2168@columbia.edu>
680     Date:    Sun Dec 20 16:45:20 2015 -0500
681
682     more tests and fixes
683
684     commit dc3d893e18172bfa7fdb9733fb9990b22f26a3dc
685     Author: Khaled Atef <kaa2168@columbia.edu>
686     Date:    Sun Dec 20 16:12:12 2015 -0500
687
688     cyclical inheritance test added
689
690     commit 00009886c90714b113bd2e9066df7c0314fe99be
691     Author: Khaled Atef <kaa2168@columbia.edu>
692     Date:    Sun Dec 20 15:52:35 2015 -0500
693
694     inheritance object passed in arg test
695
696     commit 79585bfacf986d5b013396ecdea2c4ce1f078edd
697     Merge: ae4bcc4 b5d6640
698     Author: David Watkins <davidw@tkins.me>
699     Date:    Sun Dec 20 15:41:22 2015 -0500
700
701     Merge branch 'master' of https://github.com/DavidWatkins/Dice
702
703     commit ae4bcc4ec6860484529e4431d96531ce245a3823
704     Author: David Watkins <davidw@tkins.me>
705     Date:    Sun Dec 20 15:40:50 2015 -0500
706
707     Fixed way accessing inherited methods checker thing grammar english pls
708
709     commit b5d6640ecfe55fa20bc69d109be8ef38cb2df82a
710     Merge: 777db46 da9452f
711     Author: Khaled Atef <kaa2168@columbia.edu>
712     Date:    Sun Dec 20 15:30:29 2015 -0500
```

```
713
714     Merge branch 'master' of https://github.com/DavidWatkins/Dice
715
716 commit 777db465f5de4f9ade562b56254806d86f884f88
717 Author: Khaled Atef <kaa2168@columbia.edu>
718 Date:   Sun Dec 20 15:30:18 2015 -0500
719
720     more tests
721
722 commit b15dd23dd09a127b4b45eeef83bc8f284c86f3de
723 Author: Khaled Atef <kaa2168@columbia.edu>
724 Date:   Sun Dec 20 15:02:14 2015 -0500
725
726     tests =0
727
728 commit da9452feecda712b24ae53419fc3858db4f7ffbb
729 Author: David Watkins <davidw@tkins.me>
730 Date:   Sun Dec 20 15:00:04 2015 -0500
731
732     Fixed empty main problem
733
734 commit 7d23e2a16c131048d43fafa146b577ca5f18a8fb
735 Author: Khaled Atef <kaa2168@columbia.edu>
736 Date:   Sun Dec 20 14:52:01 2015 -0500
737
738     fixed tests
739
740 commit dddd825bf32500fdd232c563c41b77a3e4426c44
741 Merge: 6b689f2 46d105a
742 Author: David Watkins <davidw@tkins.me>
743 Date:   Sun Dec 20 14:51:10 2015 -0500
744
745     Merge branch 'master' of https://github.com/DavidWatkins/Dice
746
747 commit 6b689f2c8446921678637a0d876c4411bbaa360b
748 Author: David Watkins <davidw@tkins.me>
749 Date:   Sun Dec 20 14:50:51 2015 -0500
750
751     Added casting to subtypes
752
753 commit 46d105aef7000673550854485f86d0359b0c8b00
754 Author: Khaled Atef <kaa2168@columbia.edu>
755 Date:   Sun Dec 20 14:39:13 2015 -0500
756
757     more tests including cyclical includes
758
759 commit 81392df3b88074c974fe897d35ee65b3cfe026d4
760 Merge: 9ace750 9301a8c
761 Author: nethacker11 <philip.schiffrin@gmail.com>
```

```
762 Date:    Sun Dec 20 14:06:38 2015 -0500
763
764     Merge branch 'master' of https://github.com/DavidWatkins/Dice
765
766 commit 9ace75050be810f9e0e460d47c409e972aaaa990
767 Author: nethacker11 <philip.schiffrin@gmail.com>
768 Date:    Sun Dec 20 14:06:23 2015 -0500
769
770     added 2 tests
771
772 commit 9301a8c8bebadeb4cf67f4199b1084c9d25107b3
773 Merge: f9503b9 20c6b6c
774 Author: David Watkins <davidw@tkins.me>
775 Date:    Sun Dec 20 14:05:44 2015 -0500
776
777     Merge branch 'master' of https://github.com/DavidWatkins/Dice
778
779 commit f9503b95b010f8c9516093fe1b9cac3f6e8a7f3c
780 Merge: df64b34 f17b85f
781 Author: David Watkins <davidw@tkins.me>
782 Date:    Sun Dec 20 14:05:29 2015 -0500
783
784     Merge branch 'master' of https://github.com/DavidWatkins/Dice
785
786 commit 20c6b6c16425120bbe1da4d355178c054b384698
787 Author: Khaled Atef <kaa2168@columbia.edu>
788 Date:    Sun Dec 20 14:01:26 2015 -0500
789
790     more tests passing
791
792 commit df64b347fd6e07abb2d4f0834da862231ff35cba
793 Author: David Watkins <davidw@tkins.me>
794 Date:    Sun Dec 20 13:49:00 2015 -0500
795
796     Added some broken stuff
797
798 commit f17b85fedaf22ff07158c044185229a9d96f4f13
799 Author: nethacker11 <philip.schiffrin@gmail.com>
800 Date:    Sun Dec 20 13:46:57 2015 -0500
801
802     added getchar()
803
804 commit 034b4a4e8a56c49e0de21385534708706f88f3af
805 Author: David Watkins <davidw@tkins.me>
806 Date:    Sun Dec 20 12:58:20 2015 -0500
807
808     Functions now have working private scope
809
810 commit fef6f2a5139dd5dda3d0d00cb349898d584ac0da
```

```
811 Author: David Watkins <davidw@tkins.me>
812 Date: Sun Dec 20 12:32:55 2015 -0500
813
814     main args is now working
815
816 commit 47a6d182878aa980a372554b5eb7bd331cf60e7f
817 Author: David Watkins <davidw@tkins.me>
818 Date: Sun Dec 20 11:26:54 2015 -0500
819
820     Break and continue now work
821
822 commit a9be4f6c34ee4230620875dc92bd7f7489d66c5f
823 Author: David Watkins <davidw@tkins.me>
824 Date: Sun Dec 20 10:01:28 2015 -0500
825
826     Added code for checking if break or continue is valid
827
828 commit 795773d726798b0b7d698e35293f4ee76c2acdf4
829 Author: David Watkins <davidw@tkins.me>
830 Date: Sun Dec 20 09:37:20 2015 -0500
831
832     Added basic private checking, not working for inheritance
833
834 commit 2e1c681369eb3397f0de724572cdf413988efbaa
835 Author: David Watkins <davidw@tkins.me>
836 Date: Sun Dec 20 08:54:08 2015 -0500
837
838     Added casting at the beginning of overridden function
839
840 commit ca425b48bfa72b4f26d4f2be8bc92f69a4cb4fdf
841 Author: David Watkins <davidw@tkins.me>
842 Date: Sun Dec 20 08:35:36 2015 -0500
843
844     Added default constructor
845
846 commit 98e3f63c3121a86e40c4445ff4bdd7f7dff36893
847 Author: David Watkins <davidw@tkins.me>
848 Date: Sun Dec 20 08:06:45 2015 -0500
849
850     Virtual function resolution works
851
852 commit 145101c510c43fb8809e5fe2ccdd7de2e8ece722
853 Author: David Watkins <davidw@tkins.me>
854 Date: Sun Dec 20 06:56:25 2015 -0500
855
856     Added working vtbl
857
858 commit 21f7e5cc757e7f94f3d41e71c95590188119a15b
859 Author: David Watkins <davidw@tkins.me>
```

```
860 Date:    Sun Dec 20 05:26:16 2015 -0500
861
862     Cleaned up use of types in SAST
863
864 commit 064f098e6ced5aa733a3beabf8edd3dda5173db3
865 Author: David Watkins <davidw@tkins.me>
866 Date:    Sun Dec 20 05:12:03 2015 -0500
867
868     Added unused integer to all scalls
869
870 commit 9ee2d0ef828eff03f3acd0ed117610481d012135
871 Merge: 2042484 76746fd
872 Author: David Watkins <davidw@tkins.me>
873 Date:    Sun Dec 20 05:01:45 2015 -0500
874
875     Merge branch 'master' of https://github.com/DavidWatkins/Dice
876
877 commit 2042484a2a9e8778eb1c4a86c00cb0ba8e5e0625
878 Author: David Watkins <davidw@tkins.me>
879 Date:    Sun Dec 20 05:01:23 2015 -0500
880
881     Incorporated Emily's changes to Analyzer
882
883 commit 76746fdb001845cb72dd757f870fc985b4f2261a
884 Merge: fa8e2ee c0eedeb
885 Author: Khaled Atef <kaa2168@columbia.edu>
886 Date:    Sun Dec 20 03:20:05 2015 -0500
887
888     Merge branch 'master' of https://github.com/DavidWatkins/Dice
889
890 commit fa8e2eea360f9b10e068fa1937317cafb003df12
891 Author: Khaled Atef <kaa2168@columbia.edu>
892 Date:    Sun Dec 20 03:19:53 2015 -0500
893
894     more tests
895
896 commit c0eedebd7866f602cd79bf581ba5030f5a9a53e4
897 Author: David Watkins <davidw@tkins.me>
898 Date:    Sun Dec 20 03:15:15 2015 -0500
899
900     Reformatted some code, fixed exit bug
901
902 commit 0a275a096762f01c506384a281c827a0689e8ab5
903 Author: Khaled Atef <kaa2168@columbia.edu>
904 Date:    Sun Dec 20 02:20:27 2015 -0500
905
906     modified dice.ml to pass exceptions
907
908 commit d61f20801707ee4ac695135909823b3ee4b09073
```

```
909 Author: nethacker11 <philip.schiffrin@gmail.com>
910 Date: Sun Dec 20 00:18:20 2015 -0500
911
912 took out print stmt in stdlib
913
914 commit e1bc841aa24a9ef597e94232e04735c26c4276cd
915 Author: Khaled Atef <kaa2168@columbia.edu>
916 Date: Sun Dec 20 00:06:16 2015 -0500
917
918 More tests =)
919
920 commit 60a80460f04a4ffe25d7bbe319734bab7c8ebc82
921 Author: nethacker11 <philip.schiffrin@gmail.com>
922 Date: Sat Dec 19 22:45:15 2015 -0500
923
924 fixed concat in stdlib
925
926 commit 7ad7480ee90a8759271b0961507d7f084990a162
927 Author: Khaled Atef <kaa2168@columbia.edu>
928 Date: Sat Dec 19 21:25:12 2015 -0500
929
930 Added more tests and modified dice.ml to account for an exception to make the test script work
931
932 commit 1eeea68662d793173c0dd4587cd244eb379e3176
933 Merge: 3529056 50a7529
934 Author: David Watkins <davidw@tkins.me>
935 Date: Sat Dec 19 17:20:15 2015 -0500
936
937 Merge branch 'master' of https://github.com/DavidWatkins/Dice
938
939 commit 3529056aae15850c8e3ce00eb314e0393d5a1ff3
940 Author: David Watkins <davidw@tkins.me>
941 Date: Sat Dec 19 17:19:43 2015 -0500
942
943 Added changes to allow for exit
944
945 commit 50a7529746b3b7488fb038d75817983dcef56713
946 Merge: d2b04d3 3fd9fbf
947 Author: nethacker11 <philip.schiffrin@gmail.com>
948 Date: Sat Dec 19 17:16:14 2015 -0500
949
950 Merge branch 'master' of https://github.com/DavidWatkins/Dice
951
952 commit d2b04d339c239b0000ccfdde4b93ee3bfe13a878
953 Author: nethacker11 <philip.schiffrin@gmail.com>
954 Date: Sat Dec 19 17:15:51 2015 -0500
955
956 updated stdlib to include Integer and String has reverse()
957
```

```
958 commit 3fd9fbf47a382b0c8bc02e6f13e32c810f7f9807
959 Merge: 8ac9eed 14e1b19
960 Author: Khaled Atef <kaa2168@columbia.edu>
961 Date: Sat Dec 19 16:46:38 2015 -0500
962
963 Merge branch 'master' of https://github.com/DavidWatkins/Dice
964
965 commit 8ac9eed3f00065424b59350a074749246d411869
966 Author: Khaled Atef <kaa2168@columbia.edu>
967 Date: Sat Dec 19 16:46:20 2015 -0500
968
969 more tweaks to tests and script
970
971 commit 14e1b190bfb5972a1a0394a23be43f178eef971b
972 Author: David Watkins <davidw@tkins.me>
973 Date: Sat Dec 19 16:31:22 2015 -0500
974
975 Fixed codegen for char_lits to i8_t
976
977 commit d984aff231ee6eb90e5921994f5d6fd14e044a79
978 Author: nethacker11 <philip.schiffrin@gmail.com>
979 Date: Sat Dec 19 16:19:40 2015 -0500
980
981 added test cases and updated stdlib
982
983 commit ff79fff82264ba8743377b3515514df0a988d7fc
984 Merge: b336d0a 602dc41
985 Author: David Watkins <davidw@tkins.me>
986 Date: Sat Dec 19 15:57:18 2015 -0500
987
988 Merge branch 'master' of https://github.com/DavidWatkins/Dice
989
990 commit b336d0a6333387906a3a63d44541953e1c6a4616
991 Author: David Watkins <davidw@tkins.me>
992 Date: Sat Dec 19 15:57:02 2015 -0500
993
994 Added modulo
995
996 commit 602dc4179efef1a87778934fb87428ddd5ee72d90
997 Author: Khaled Atef <kaa2168@columbia.edu>
998 Date: Sat Dec 19 15:55:55 2015 -0500
999
1000 corrected tester script to account for errors from exception tests
1001
1002 commit cedf61d44d4d5a1faf2424eb50cf983df9df22f3
1003 Author: David Watkins <davidw@tkins.me>
1004 Date: Sat Dec 19 15:24:25 2015 -0500
1005
1006 Fixed function element access
```



```
1007
1008 commit 664bef08cd785fcd8f862874acfce3ced40bc5d2
1009 Author: nethacker11 <philip.schiffrin@gmail.com>
1010 Date: Sat Dec 19 15:16:10 2015 -0500
1011
1012     added stdlib2 test and updated stdlib
1013
1014 commit f4a81c401d29969e5b341c99f2e68e003318bb2e
1015 Merge: 285aa85 3b7465c
1016 Author: nethacker11 <philip.schiffrin@gmail.com>
1017 Date: Sat Dec 19 15:14:14 2015 -0500
1018
1019     Merge branch 'master' of https://github.com/DavidWatkins/Dice
1020
1021 commit 3b7465cf892745766ce5a4bf08e4fdbdb28468eb
1022 Author: David Watkins <davidw@tkins.me>
1023 Date: Sat Dec 19 15:13:45 2015 -0500
1024
1025     This time for sure!
1026
1027 commit 285aa8594fe6d3b1fea5e2983e5599cc19bec253
1028 Merge: 3425edc d7ed17e
1029 Author: nethacker11 <philip.schiffrin@gmail.com>
1030 Date: Sat Dec 19 15:10:53 2015 -0500
1031
1032     Merge branch 'master' of https://github.com/DavidWatkins/Dice
1033
1034 commit d7ed17e991fa0322eac0a63f0b55c84d4e2c1115
1035 Author: David Watkins <davidw@tkins.me>
1036 Date: Sat Dec 19 15:10:09 2015 -0500
1037
1038     Fixed function param passing bug
1039
1040 commit 3425edceaa05209d8f57da67867dac753d0ea0bc
1041 Merge: 41afbc1 97de937
1042 Author: nethacker11 <philip.schiffrin@gmail.com>
1043 Date: Sat Dec 19 15:02:04 2015 -0500
1044
1045     Merge branch 'master' of https://github.com/DavidWatkins/Dice
1046
1047 commit 41afbc17bf2426ee44dc27bd254b550edbd78245
1048 Author: nethacker11 <philip.schiffrin@gmail.com>
1049 Date: Sat Dec 19 15:02:02 2015 -0500
1050
1051     updated codegen for lseek
1052
1053 commit 97de93788f1701bcc7e334d8061fe58fca6a5d35
1054 Author: David Watkins <davidw@tkins.me>
1055 Date: Sat Dec 19 15:01:22 2015 -0500
```

```
1056
1057     Fixed codegen_call for lseek
1058
1059     commit 3d58076d10a7060f85dfb363ec5cbce759038257
1060     Merge: 7413f9b 464fc4c
1061     Author: nethacker11 <philip.schiffrin@gmail.com>
1062     Date:   Sat Dec 19 14:57:34 2015 -0500
1063
1064     Merge branch 'master' of https://github.com/DavidWatkins/Dice
1065
1066     commit 464fc4c5d6119034866a6118cce83e59a56b3520
1067     Merge: 87f4d52 7a63abf
1068     Author: David Watkins <davidw@tkins.me>
1069     Date:   Sat Dec 19 14:55:19 2015 -0500
1070
1071     Merge branch 'master' of https://github.com/DavidWatkins/Dice
1072
1073     commit 87f4d52f2e6d185d46bc29b8635c6f98d7eb7853
1074     Author: David Watkins <davidw@tkins.me>
1075     Date:   Sat Dec 19 14:55:02 2015 -0500
1076
1077     Added lseek syntax to analyzer
1078
1079     commit 7413f9b0e14a20d00314556df6e6c4890fd243f3
1080     Merge: 3c1c15b 7a63abf
1081     Author: nethacker11 <philip.schiffrin@gmail.com>
1082     Date:   Sat Dec 19 14:22:33 2015 -0500
1083
1084     Merge branch 'master' of https://github.com/DavidWatkins/Dice
1085
1086     commit 7a63abffd7edafb87ecf82df2225e7ea2148eeb8
1087     Merge: c482260 afae098
1088     Author: Khaled Atef <kaa2168@columbia.edu>
1089     Date:   Sat Dec 19 14:22:02 2015 -0500
1090
1091     Merge branch 'master' of https://github.com/DavidWatkins/Dice
1092
1093     commit c48226078ee263889c6de75ff8d0f6f572a6a7ee
1094     Author: Khaled Atef <kaa2168@columbia.edu>
1095     Date:   Sat Dec 19 14:21:41 2015 -0500
1096
1097     added stdlib string
1098
1099     commit 3c1c15b99113b0e570fa6625ba4a2a0ee1c917e5
1100     Merge: 480dc4d afae098
1101     Author: nethacker11 <philip.schiffrin@gmail.com>
1102     Date:   Sat Dec 19 14:19:44 2015 -0500
1103
1104     Merge branch 'master' of https://github.com/DavidWatkins/Dice
```

```
1105
1106 commit 480dc4d0c15a9c5cd5bccfb7c8d05aebb423b9e7
1107 Author: nethacker11 <philip.schiffrin@gmail.com>
1108 Date: Sat Dec 19 14:18:18 2015 -0500
1109
1110 changed stdlib
1111
1112 commit afae098e32e66e69b0349e9809ce6d237f451179
1113 Merge: acbea61 404c6df
1114 Author: David Watkins <davidw@tkins.me>
1115 Date: Sat Dec 19 14:17:52 2015 -0500
1116
1117 Merge branch 'master' of https://github.com/DavidWatkins/Dice
1118
1119 commit acbea6113ddccfa59ce06c8288a4bfe81b134f6f
1120 Author: David Watkins <davidw@tkins.me>
1121 Date: Sat Dec 19 14:17:31 2015 -0500
1122
1123 Fixed right associativity of parser
1124
1125 commit 404c6df62cc80b61ceffed8cc666f9591757d5e0
1126 Merge: 782ca3f 3e4e5e6
1127 Author: Khaled Atef <kaa2168@columbia.edu>
1128 Date: Sat Dec 19 14:15:07 2015 -0500
1129
1130 Merge branch 'master' of https://github.com/DavidWatkins/Dice
1131
1132 commit 17c1362a3d24b7edf544491948a259fb816524a4
1133 Merge: c248f39 3e4e5e6
1134 Author: nethacker11 <philip.schiffrin@gmail.com>
1135 Date: Sat Dec 19 14:15:07 2015 -0500
1136
1137 Merge branch 'master' of https://github.com/DavidWatkins/Dice
1138
1139 commit c248f394794dc1a26b0052db05a4a2abffe5ba89
1140 Author: nethacker11 <philip.schiffrin@gmail.com>
1141 Date: Sat Dec 19 14:15:05 2015 -0500
1142
1143 updated stdlib
1144
1145 commit 782ca3fa5c903b0c87d7402724934c25a3cf3a30
1146 Author: Khaled Atef <kaa2168@columbia.edu>
1147 Date: Sat Dec 19 14:14:49 2015 -0500
1148
1149 modified tests
1150
1151 commit 3e4e5e6b27248dbe9de6af579040dbc991f2b5be
1152 Author: David Watkins <davidw@tkins.me>
1153 Date: Sat Dec 19 14:13:16 2015 -0500
```

```
1154
1155     Fixed array access for chars
1156
1157 commit cbcdf6c41b458da3355bf3aeb58a5d3549752e
1158 Merge: 3ca5e39 0c9870c
1159 Author: David Watkins <davidw@tkins.me>
1160 Date:   Sat Dec 19 13:54:44 2015 -0500
1161
1162     Merge branch 'master' of https://github.com/DavidWatkins/Dice
1163
1164 commit 3ca5e39a56c7a6c239d38e9c58eabd03304f1526
1165 Author: David Watkins <davidw@tkins.me>
1166 Date:   Sat Dec 19 13:54:14 2015 -0500
1167
1168     Fixed array acces for strings
1169
1170 commit 0c9870c3948b1e193926f363ec551830d8aae9ae
1171 Author: Khaled Atef <kaa2168@columbia.edu>
1172 Date:   Sat Dec 19 13:54:04 2015 -0500
1173
1174     added more tests
1175
1176 commit 91c9bc47dff55afd6269202ad1654145cf55b5da
1177 Author: David Watkins <davidw@tkins.me>
1178 Date:   Sat Dec 19 05:07:25 2015 -0500
1179
1180     Fixed stdlib
1181
1182 commit c603715b9036aa50daa30a423ee6e0b30fd9e8ce
1183 Author: David Watkins <davidw@tkins.me>
1184 Date:   Sat Dec 19 04:08:52 2015 -0500
1185
1186     While loops work
1187
1188 commit 27b53ff8e9131b2e686ed29755d54690936a2131
1189 Author: David Watkins <davidw@tkins.me>
1190 Date:   Sat Dec 19 04:02:09 2015 -0500
1191
1192     Fixed bug with array length
1193
1194 commit 64b72feeb55e71b92c1fd7810e5ccb82ae736f41
1195 Author: David Watkins <davidw@tkins.me>
1196 Date:   Sat Dec 19 03:39:57 2015 -0500
1197
1198     Fixed odd incorrect ordering bug
1199
1200 commit 170e4fd2e2285c0d7f106426651199a48c5b20e6
1201 Author: David Watkins <davidw@tkins.me>
1202 Date:   Sat Dec 19 03:34:00 2015 -0500
```

```
1203
1204     Fixed includes bug, fixed char array assignment of int length
1205
1206 commit 7c8d274ea55d5118e70db8f3d11dd5cff42d36e4
1207 Author: David Watkins <davidw@tkins.me>
1208 Date:   Sat Dec 19 01:36:29 2015 -0500
1209
1210     Migrated files and folders to appropriate place for new makefile schema
1211
1212 commit a1ae8ffbc1d1fe84c755abf98a44392680a63c20
1213 Author: nethacker11 <philip.schiffrin@gmail.com>
1214 Date:   Fri Dec 18 22:57:30 2015 -0500
1215
1216     updated stdlib and analyzer and codegen for built in functions
1217
1218 commit 1a5244813f0c299c673096a48a09dad022133599
1219 Author: David Watkins <davidw@tkins.me>
1220 Date:   Fri Dec 18 20:01:52 2015 -0500
1221
1222     Fixed \0, its now \000
1223
1224 commit a2d07124a44c96af5b158996c049fead07644dc5
1225 Author: nethacker11 <philip.schiffrin@gmail.com>
1226 Date:   Fri Dec 18 20:02:58 2015 -0500
1227
1228     updated stdlib.dice
1229
1230 commit e9c8d476beb76ebd9a4f4d1a23f5cf722d741744
1231 Author: David Watkins <davidw@tkins.me>
1232 Date:   Fri Dec 18 19:47:00 2015 -0500
1233
1234     backslash zero yo
1235
1236 commit d6be8f34690274401b8123cf491254274e8030b9
1237 Author: David Watkins <davidw@tkins.me>
1238 Date:   Fri Dec 18 19:33:09 2015 -0500
1239
1240     works now?
1241
1242 commit 8ad670e00d5b7cf8020581861306cf89ab17b8a6
1243 Merge: aec396d c6aflee
1244 Author: David Watkins <davidw@tkins.me>
1245 Date:   Fri Dec 18 19:17:00 2015 -0500
1246
1247     Merge branch 'master' of https://github.com/DavidWatkins/Dice
1248
1249 commit aec396db7c9a6714ce6e5de976596b42c1d03c8e
1250 Author: David Watkins <davidw@tkins.me>
1251 Date:   Fri Dec 18 19:16:41 2015 -0500
```

```
1252
1253     Works *crosses fingers*
1254
1255     commit c6af1eecd3362b57591589d61493c14707c11479
1256     Author: nethacker11 <philip.schiffrin@gmail.com>
1257     Date:   Fri Dec 18 19:13:08 2015 -0500
1258
1259     updated stdlib.dice
1260
1261     commit b0e033a148286f9de9c2cef0b37c799fb5ec36d0
1262     Author: David Watkins <davidw@tkins.me>
1263     Date:   Fri Dec 18 18:43:07 2015 -0500
1264
1265     So uh, nested comments are a thing
1266
1267     commit 0e91f6aca66d2804747918f460114f356842befd
1268     Author: Khaled Atef <kaa2168@columbia.edu>
1269     Date:   Fri Dec 18 17:37:31 2015 -0500
1270
1271     Exceptions folder created, need to add more tests here
1272
1273     commit 643197852baaf3fff864761ab7376bf32e6bacf0
1274     Author: nethacker11 <philip.schiffrin@gmail.com>
1275     Date:   Fri Dec 18 17:12:04 2015 -0500
1276
1277     added stdlib.dice, passes analyzer but not tested
1278
1279     commit b9c354db5a56e4d8e9543a1c00147260283e5d51
1280     Merge: 75cb0da 5ae669c
1281     Author: Khaled Atef <kaa2168@columbia.edu>
1282     Date:   Fri Dec 18 03:46:41 2015 -0500
1283
1284     Merge branch 'master' of https://github.com/DavidWatkins/Dice
1285
1286     commit 75cb0daf1e69f062f9cb1e6c66079639ededd3e0
1287     Author: Khaled Atef <kaa2168@columbia.edu>
1288     Date:   Fri Dec 18 03:41:15 2015 -0500
1289
1290     modified test script
1291
1292     commit 5ae669cf25734ab2bdfb6c989bfd933b98bdebb9
1293     Author: David Watkins <davidw@tkins.me>
1294     Date:   Thu Dec 17 19:26:41 2015 -0500
1295
1296     Works?
1297
1298     commit 1cfe2ae2cf20eb203f45617097d9daa93abf3793
1299     Author: nethacker11 <philip.schiffrin@gmail.com>
1300     Date:   Thu Dec 17 19:24:59 2015 -0500
```

```
1301
1302     added write function
1303
1304     commit 013f06fe8fcbf6d8db7dcf2cd32af311d47b7f2c
1305     Merge: 4554586 b0dcfe9
1306     Author: nethacker11 <philip.schiffrin@gmail.com>
1307     Date:   Thu Dec 17 18:59:31 2015 -0500
1308
1309     Merge branch 'master' of https://github.com/DavidWatkins/Dice
1310
1311     commit 4554586421327badd3daf2acb2c212fb98303a53
1312     Author: nethacker11 <philip.schiffrin@gmail.com>
1313     Date:   Thu Dec 17 18:59:29 2015 -0500
1314
1315     added more build in function declarations
1316
1317     commit b0dcfe9708793c4946a123dbf45afea8c305027e
1318     Author: David Watkins <davidw@tkins.me>
1319     Date:   Thu Dec 17 18:58:19 2015 -0500
1320
1321     Fixed shift/reduce, added linking of c functions
1322
1323     commit 9c7a140e1e036a70bb4af3159d21265a2799bcaf
1324     Author: nethacker11 <philip.schiffrin@gmail.com>
1325     Date:   Thu Dec 17 18:06:58 2015 -0500
1326
1327     added c function declarations in codegen.ml under built in functions
1328
1329     commit d04c2b99e7c467914839a1b6429d7284d8c78725
1330     Author: nethacker11 <philip.schiffrin@gmail.com>
1331     Date:   Thu Dec 17 17:37:41 2015 -0500
1332
1333     added folder for c library extensions for .bc files to be linked in dice.ml
1334
1335     commit d058e9c00fc86b609da0dce4a906b10718ca3430
1336     Author: David Watkins <davidw@tkins.me>
1337     Date:   Wed Dec 16 16:55:34 2015 -0500
1338
1339     Added delete command to free memory
1340
1341     commit 9414ee274b553debcc02a052fff0fd34e46e14e8
1342     Author: David Watkins <davidw@tkins.me>
1343     Date:   Wed Dec 16 16:29:17 2015 -0500
1344
1345     Added multi-dimensional c code
1346
1347     commit a08e96f67a96dd181abf6c67b769d371c326fa03
1348     Author: David Watkins <davidw@tkins.me>
1349     Date:   Wed Dec 16 16:28:52 2015 -0500
```

```
1350
1351     Array length working, also added multi-dimensional c code
1352
1353 commit 59e4b9b012b92b799cdac22849c667130731163a
1354 Author: David Watkins <davidw@tkins.me>
1355 Date:   Wed Dec 16 01:41:37 2015 -0500
1356
1357     Array primitives work
1358
1359 commit 3ab1e0ff494e1bdc57460aa3d930b5f15fe3c0a6
1360 Author: David Watkins <davidw@tkins.me>
1361 Date:   Tue Dec 15 23:45:42 2015 -0500
1362
1363     Fixed single dimension arrays
1364
1365 commit f4ccfe7371bdd8c051db4735db872885a0578f42
1366 Author: nethacker11 <philip.schiffrin@gmail.com>
1367 Date:   Tue Dec 15 22:20:45 2015 -0500
1368
1369     build_array_malloc in progress
1370
1371 commit 3e27ec7a42f5d91620f8c16390cf53a82f9e858f
1372 Author: nethacker11 <philip.schiffrin@gmail.com>
1373 Date:   Tue Dec 15 19:20:37 2015 -0500
1374
1375     changing to single dimensional arrays, compiles but looks like arraycreate is not accessed again
1376
1377 commit 10e87f3b9c82258c06247f972144d63f582dbc4c
1378 Author: David Watkins <davidw@tkins.me>
1379 Date:   Tue Dec 15 18:44:34 2015 -0500
1380
1381     Working status
1382
1383 commit c71bfa88710ef0a7c39f98fb4ece382a6dbb877c
1384 Author: David Watkins <davidw@tkins.me>
1385 Date:   Sat Dec 12 19:04:56 2015 -0500
1386
1387     ArrayCreate doesn't work, added code for array deref
1388
1389 commit b9ed042660504b766617f397d21c8858756f4f95
1390 Author: David Watkins <davidw@tkins.me>
1391 Date:   Sat Dec 12 18:57:28 2015 -0500
1392
1393     Added basic array methods
1394
1395 commit cdc675d5c824d42a7e82749a2472bb1da8726008
1396 Author: David Watkins <davidw@tkins.me>
1397 Date:   Fri Dec 11 15:28:10 2015 -0500
1398
```



```
1399     Fixed bug where constructors weren't being checked by name
1400
1401     commit 8346a0009480db6799587fa8a1b3ab0178c5ea43
1402     Author: David Watkins <DavidWatkins@users.noreply.github.com>
1403     Date:   Thu Dec 10 18:18:39 2015 -0500
1404
1405     Update README.md
1406
1407     commit b33b3a318fc92dbcdf434c72888f0c2399b3173f
1408     Author: David Watkins <davidw@tkins.me>
1409     Date:   Tue Dec 8 17:23:53 2015 -0500
1410
1411     Added help printing to compiler with no arguments
1412
1413     commit ec57d8062f137244729246260d34a7cd47641525
1414     Merge: ae65af0 bb7a89b
1415     Author: David Watkins <DavidWatkins@users.noreply.github.com>
1416     Date:   Sun Dec 6 17:21:44 2015 -0500
1417
1418     Merge pull request #79 from DavidWatkins/Kappa
1419
1420     Kappa
1421
1422     commit bb7a89b50cd1045cd8c0b711288d7bead3f8af20
1423     Merge: ae65af0 43e4e3b
1424     Author: David Watkins <davidw@tkins.me>
1425     Date:   Sun Dec 6 17:21:21 2015 -0500
1426
1427     Merge branch 'Kappa' of https://github.com/DavidWatkins/Dice into Kappa
1428
1429     commit ae65af04ea8c138768db9f1e25249d4c564d9882
1430     Merge: 914b15a df7d695
1431     Author: David Watkins <DavidWatkins@users.noreply.github.com>
1432     Date:   Sat Dec 5 21:31:11 2015 -0500
1433
1434     Merge pull request #77 from DavidWatkins/ObjAccess
1435
1436     Obj access
1437
1438     commit df7d695d9f61cb7709ac5bd24422a23691d969dc
1439     Author: David Watkins <davidw@tkins.me>
1440     Date:   Sat Dec 5 21:29:47 2015 -0500
1441
1442     Classes are now working, fixed tests to match up with new rules
1443
1444     commit 3547bd54ce8e66a8d984ecac37ef478f43d1d773
1445     Author: David Watkins <davidw@tkins.me>
1446     Date:   Fri Dec 4 15:39:07 2015 -0500
1447
```

```
1448     Sigh
1449
1450     commit 914b15a3301e9de97ff5b9fcbf57f7731fbd90a0
1451     Merge: bc5da4f b474701
1452     Author: Khaled Atef <kaa2168@columbia.edu>
1453     Date:   Fri Dec 4 01:27:57 2015 -0500
1454
1455     Merge branch 'master' of https://github.com/DavidWatkins/Dice
1456
1457     commit bc5da4f925a2a6995b3d79cff92fff0f87f0384d
1458     Author: Khaled Atef <kaa2168@columbia.edu>
1459     Date:   Fri Dec 4 01:27:04 2015 -0500
1460
1461     added else if tests
1462
1463     commit 43e4e3bf1d4a64e5fa71b3642a21250f37bb7334
1464     Author: Khaled Atef <kaa2168@columbia.edu>
1465     Date:   Fri Dec 4 01:14:26 2015 -0500
1466
1467     unop working
1468
1469     commit 2fedba447dd85d89582b3aad84c0a470db87de7c
1470     Author: David Watkins <davidw@tkins.me>
1471     Date:   Wed Dec 2 17:14:26 2015 -0500
1472
1473     Still WIP
1474
1475     commit a0c3cbf70c80847b0892ef61c9bf34c109ca1f49
1476     Author: David Watkins <davidw@tkins.me>
1477     Date:   Wed Dec 2 15:56:52 2015 -0500
1478
1479     Added sample test script
1480
1481     commit a639719f7a7d885a4008be87ac94cbe5ec170695
1482     Author: David Watkins <davidw@tkins.me>
1483     Date:   Wed Dec 2 15:56:03 2015 -0500
1484
1485     WIP
1486
1487     commit b47470171b10bbe3b8f7bcc9f7f0e52bf73a01e1
1488     Author: David Watkins <DavidWatkins@users.noreply.github.com>
1489     Date:   Wed Dec 2 07:33:33 2015 -0500
1490
1491     Update README.md
1492
1493     commit 15e55374aea051650f8f627205dbcc8160544a75
1494     Author: David Watkins <davidw@tkins.me>
1495     Date:   Wed Dec 2 06:48:25 2015 -0500
1496
```

```
1497     Function parameters are working
1498
1499     commit Off181573ba6a1fea1105ecc4b72f8fb269db965
1500     Author: David Watkins <davidw@tkins.me>
1501     Date:    Wed Dec 2 06:00:30 2015 -0500
1502
1503     Added basic function calls to compiler
1504
1505     commit d99e2cc2f5b17ce3826ffe4aa0c6bc39e8297465
1506     Author: Khaled Atef <kaa2168@columbia.edu>
1507     Date:    Wed Dec 2 04:26:03 2015 -0500
1508
1509     unop implemented, but not working. All tests are failing.
1510
1511     commit aa1368f6872e5c20d669f38614fd431e3b21c65
1512     Author: David Watkins <davidw@tkins.me>
1513     Date:    Wed Dec 2 03:48:01 2015 -0500
1514
1515     Added lazy evaluation and fixed error with function names
1516
1517     commit d0fa8223f546f315afc023d637f240af34329e36
1518     Author: David Watkins <davidw@tkins.me>
1519     Date:    Wed Dec 2 03:06:35 2015 -0500
1520
1521     Changed wording in helper
1522
1523     commit 74059d062fdbbdc1679dac574052c05459751c08
1524     Author: David Watkins <davidw@tkins.me>
1525     Date:    Wed Dec 2 03:04:36 2015 -0500
1526
1527     Added the ability to compile to a file
1528
1529     commit 2078c5fdb94b2cc6d265617c7d12d81e507c7e57
1530     Author: Khaled Atef <kaa2168@columbia.edu>
1531     Date:    Wed Dec 2 02:15:27 2015 -0500
1532
1533     corrected test-bool4.dice
1534
1535     commit c0d5caee65fb50c2aa083309957bd1b20dba1c1c
1536     Author: David Watkins <davidw@tkins.me>
1537     Date:    Wed Dec 2 01:58:56 2015 -0500
1538
1539     Float comparison expressions now evaluate properly
1540
1541     commit c6bb01085947ef3f51cbdc885238c3964039b708
1542     Author: Khaled Atef <kaa2168@columbia.edu>
1543     Date:    Wed Dec 2 01:29:37 2015 -0500
1544
1545     fixed tests and added more for bools
```

```
1546
1547 commit 0d9c3a0dfe3b893c500f75090b5f04c89bb4401c
1548 Merge: 63fdb09 a2300ae
1549 Author: David Watkins <davidw@tkins.me>
1550 Date: Wed Dec 2 01:25:58 2015 -0500
1551
1552 Merge branch 'master' of https://github.com/DavidWatkins/Dice
1553
1554 commit 63fdb093f7254bc4934fdde8b564b1d97463eada
1555 Author: David Watkins <davidw@tkins.me>
1556 Date: Wed Dec 2 01:25:27 2015 -0500
1557
1558 Added printing string representations of boolean values to codgen
1559
1560 commit a2300aedb2fe11c3fa612e1ae8f7d60537fb3019
1561 Author: Khaled Atef <kaa2168@columbia.edu>
1562 Date: Wed Dec 2 00:36:29 2015 -0500
1563
1564 Fixed syntax error
1565
1566 commit 861aee2ddb899d888a13e2a42e5a81c0a1528cd4
1567 Merge: e7494e3 b0ab4a8
1568 Author: Khaled Atef <kaa2168@columbia.edu>
1569 Date: Wed Dec 2 00:16:32 2015 -0500
1570
1571 Merge branch 'master' of https://github.com/DavidWatkins/Dice
1572
1573 commit b0ab4a8e92319f72c3d1bb2376475b424cbf1887
1574 Author: David Watkins <davidw@tkins.me>
1575 Date: Wed Dec 2 00:16:10 2015 -0500
1576
1577 Reverted change to printing floats
1578
1579 commit e7494e3b6488dc49d28bb3bcad6e77f7ea42d265
1580 Merge: d969ca2 21ac0fa
1581 Author: Khaled Atef <kaa2168@columbia.edu>
1582 Date: Wed Dec 2 00:11:39 2015 -0500
1583
1584 wMerge branch 'master' of https://github.com/DavidWatkins/Dice
1585
1586 commit d969ca2fc12093e18f946e893328b3cdb788ff43
1587 Author: Khaled Atef <kaa2168@columbia.edu>
1588 Date: Wed Dec 2 00:11:08 2015 -0500
1589
1590 nested if tests added with boolean tests of logical operators
1591
1592 commit 21ac0fa10db8c24347a0a56ed39cfc1b92e7ae19
1593 Author: David Watkins <davidw@tkins.me>
1594 Date: Wed Dec 2 00:07:50 2015 -0500
```

1595

1596 Fixed printing of floats

1597

1598 commit 4ca9ff15d8016c5fe78f81a23eb2b5bc19a443de

1599 Author: David Watkins <davidw@tkins.me>

1600 Date: Tue Dec 1 23:52:18 2015 -0500

1601

1602 Added exception for invalid integer operation in codegen

1603

1604 commit 9c25e446d76918a3b11be98a9d0aef72f2345e57

1605 Merge: c45b5f8 72718b2

1606 Author: David Watkins <DavidWatkins@users.noreply.github.com>

1607 Date: Tue Dec 1 23:50:17 2015 -0500

1608

1609 Merge pull request #68 from DavidWatkins/Kappa

1610

1611 Kappa

1612

1613 commit 72718b24c9c77818965340c2642b9746452517f9

1614 Merge: 2031096 c45b5f8

1615 Author: David Watkins <davidw@tkins.me>

1616 Date: Tue Dec 1 23:49:47 2015 -0500

1617

1618 Merge branch 'master' into Kappa

1619

1620 commit 203109635a92704afcaf6ba8f7686e4bc56ee463

1621 Author: Khaled Atef <kaa2168@columbia.edu>

1622 Date: Tue Dec 1 22:40:47 2015 -0500

1623

1624 fixed unused match warnings but matching AST type instead of llvalue. David determined that the Oc

1625

1626 commit c45b5f88281cfa8c5989fbc883bbe97230bac8c2

1627 Merge: 3707602 7630cb1

1628 Author: David Watkins <DavidWatkins@users.noreply.github.com>

1629 Date: Tue Dec 1 21:27:59 2015 -0500

1630

1631 Merge pull request #66 from DavidWatkins/emily

1632

1633 Emily

1634

1635 commit 7630cb139714b189697761faeb495d9a6d8055ad

1636 Author: Emily Chen <emchennyc@gmail.com>

1637 Date: Tue Dec 1 21:26:16 2015 -0500

1638

1639 raised wrong exception when trying to instantiate undefined class

1640

1641 commit 9d0040a4aa5506d46024e2c870dee099527cb6db

1642 Author: Emily Chen <emchennyc@gmail.com>

1643 Date: Tue Dec 1 21:13:34 2015 -0500

```
1644
1645     threw wrong exception for UndefinedClass case
1646
1647 commit 63765ae27d235ca0664cb329e72324475f80d6c0
1648 Author: Emily Chen <emchennyc@gmail.com>
1649 Date:   Tue Dec 1 20:26:28 2015 -0500
1650
1651     object creation flags when actuals don't match any existing constructor
1652
1653 commit 1d3c59c8bf8ce3d1c621697025a9c529f0285a2c
1654 Author: Emily Chen <emchennyc@gmail.com>
1655 Date:   Tue Dec 1 17:26:18 2015 -0500
1656
1657     types of actuals printed in same order as types of formals
1658
1659 commit 045fc2aa1cd78c1c93f58ce4c4412ccceada0b39
1660 Author: Emily Chen <emchennyc@gmail.com>
1661 Date:   Tue Dec 1 16:52:13 2015 -0500
1662
1663     can print types of formals and actuals
1664
1665 commit 5e2ea6f7870c893d0e8fa6df422f3be55a240555
1666 Author: Khaled Atef <kaa2168@columbia.edu>
1667 Date:   Tue Dec 1 16:06:31 2015 -0500
1668
1669     Test cases for arith negation added and build_global_stringptr modified for debugging
1670
1671 commit 4913954cb8166998c6aae53a2c1f733c06473890
1672 Author: Khaled Atef <kaa2168@columbia.edu>
1673 Date:   Tue Dec 1 08:43:55 2015 -0500
1674
1675     added cast test (float+int)
1676
1677 commit b4f2afc359fb3ad8dcf1e658d428480c84c183a7
1678 Author: Khaled Atef <kaa2168@columbia.edu>
1679 Date:   Tue Dec 1 07:25:26 2015 -0500
1680
1681     Compiles git add codegen.ml !
1682
1683 commit 3ea9139620f9b937d7c6892cf1be2209cad34635
1684 Author: Emily Chen <emchennyc@gmail.com>
1685 Date:   Tue Dec 1 03:13:43 2015 -0500
1686
1687     check_object_creation raises exception if instantiating unknown class
1688
1689 commit d98122c25680d734687c5e95d67832d620830d84
1690 Author: Emily Chen <emchennyc@gmail.com>
1691 Date:   Tue Dec 1 02:41:32 2015 -0500
1692
```

1693 checks object decl to see if the class is available

1694

1695 commit 3bd51afa9cf5b2041543025837ed50d93ffe7d52

1696 Author: Khaled Atef <kaa2168@columbia.edu>

1697 Date: Tue Dec 1 01:48:21 2015 -0500

1698

1699 fought through several rounds of compilation errors.

1700

1701 commit 4494d69fc57cdac1a944a92ea7660f899a906a9f

1702 Author: Khaled Atef <kaa2168@columbia.edu>

1703 Date: Tue Dec 1 01:22:02 2015 -0500

1704

1705 Rough draft of handle_binop implemented. Still need to compile it, but pushing to access on VM. I h

1706

1707 commit 37076028c622a12be9c222ca2331f265c99ac625

1708 Author: David Watkins <DavidWatkins@users.noreply.github.com>

1709 Date: Mon Nov 30 23:49:19 2015 -0500

1710

1711 Update README.md

1712

1713 commit 34586a39bbe449392a730dcbcf3e85dd2b70941c

1714 Author: David Watkins <davidw@tkins.me>

1715 Date: Mon Nov 30 23:46:51 2015 -0500

1716

1717 Merged Emily's changes to master

1718

1719 commit a2446010f6af0c06f465581a0b09bde85d4f1a3c

1720 Author: David Watkins <davidw@tkins.me>

1721 Date: Mon Nov 30 23:41:58 2015 -0500

1722

1723 Fixed pretty printer and loops

1724

1725 commit 9b8880077317b5dafd319becca52528d2fa8a393

1726 Merge: 889c3b7 6f8b207

1727 Author: David Watkins <DavidWatkins@users.noreply.github.com>

1728 Date: Mon Nov 30 23:35:44 2015 -0500

1729

1730 Merge pull request #56 from DavidWatkins/emily

1731

1732 Emily

1733

1734 commit 6f8b20749beb30748ed3912f631ad30cbfdf9ab0

1735 Author: David Watkins <davidw@tkins.me>

1736 Date: Mon Nov 30 23:34:52 2015 -0500

1737

1738 Added primitive variables

1739

1740 commit 1bfb88e3e588de2b2d097d9efa76cee25753129d

1741 Author: Emily Chen <emchennyc@gmail.com>

```
1742 Date:   Mon Nov 30 23:27:52 2015 -0500
1743
1744     remove debugging statements
1745
1746 commit 0fe96a727e2fbd895fdc20e4bc3b4b423fcec17f
1747 Merge: ef16286 889c3b7
1748 Author: Emily Chen <emchennyc@gmail.com>
1749 Date:   Mon Nov 30 22:41:57 2015 -0500
1750
1751     Merge branch 'master' of https://github.com/DavidWatkins/Dice into emily
1752
1753 commit ef1628630066d0d7d20112a5def6a221fc38827c
1754 Author: Emily Chen <emchennyc@gmail.com>
1755 Date:   Mon Nov 30 22:41:07 2015 -0500
1756
1757     converting local to slocal works for primitive types
1758
1759 commit 16491e2e0c6a513271acd0519f77b97818555344
1760 Author: Emily Chen <emchennyc@gmail.com>
1761 Date:   Mon Nov 30 22:01:09 2015 -0500
1762
1763     local var decls are tracked even without assignment expr
1764
1765 commit 2adbb32da2aa5cccc60561594cb402d00b2e9c7bf
1766 Author: Emily Chen <emchennyc@gmail.com>
1767 Date:   Mon Nov 30 21:48:09 2015 -0500
1768
1769     local var decl is added to env when statement includes nonempty expr
1770
1771 commit 889c3b715b50b63391a454a75a5d9d7dfbdd2657
1772 Merge: 3064464 3d3154c
1773 Author: David Watkins <davidw@tkins.me>
1774 Date:   Mon Nov 30 19:49:10 2015 -0500
1775
1776     Merge branch 'master' of https://github.com/DavidWatkins/Dice
1777
1778 commit 306446425bd29b931a6325b47b3da0cc3b84e04f
1779 Author: David Watkins <davidw@tkins.me>
1780 Date:   Mon Nov 30 19:48:49 2015 -0500
1781
1782     Added pretty printing of sast and ast in JSON
1783
1784 commit 3d3154cea0c622561c7a63946e5e85ec8eb07e8d
1785 Author: David Watkins <DavidWatkins@users.noreply.github.com>
1786 Date:   Mon Nov 30 16:18:51 2015 -0500
1787
1788     Update README.md
1789
1790 commit 64d255b692d1d3f156a210253bb4db2b1bd123ba
```



```
1791 Author: Khaled Atef <kaa2168@columbia.edu>
1792 Date:   Mon Nov 30 13:26:06 2015 -0500
1793
1794     modified test script to perform automatic compilation of Dice Executable at the beginning of each s
1795
1796 commit f4312c13faa500a2601e08b1fbd53568750df70b
1797 Author: Khaled Atef <kaa2168@columbia.edu>
1798 Date:   Mon Nov 30 12:14:07 2015 -0500
1799
1800     corrected syntax error
1801
1802 commit b562f21c0f9e17dc946ddf2d1faa206346607e5d
1803 Merge: 338553e db99c23
1804 Author: David Watkins <davidw@tkins.me>
1805 Date:   Mon Nov 30 08:10:05 2015 -0500
1806
1807     Merge branch 'emily'
1808
1809 commit db99c2314ba0bdf2bba05501e971dd379e1a0bbc
1810 Merge: 9f1d6c7 338553e
1811 Author: David Watkins <davidw@tkins.me>
1812 Date:   Mon Nov 30 08:09:54 2015 -0500
1813
1814     Merge branch 'master' into emily
1815
1816 commit 338553e016ab991c9a5b278eb4e6fccb3e632121
1817 Author: David Watkins <davidw@tkins.me>
1818 Date:   Mon Nov 30 03:18:29 2015 -0500
1819
1820     Added code for building for loops
1821
1822 commit ed35422de1e3530e82a4ecdb99a07c01774707cb
1823 Merge: ac53ca3 7fe5c5c
1824 Author: David Watkins <davidw@tkins.me>
1825 Date:   Mon Nov 30 02:35:32 2015 -0500
1826
1827     Merge branch 'master' of https://github.com/DavidWatkins/Dice
1828
1829 commit ac53ca3084d06312dabbe3058ab51694774c6bc3
1830 Author: David Watkins <davidw@tkins.me>
1831 Date:   Mon Nov 30 02:35:07 2015 -0500
1832
1833     Fixed elseless if problem
1834
1835 commit 7fe5c5ca653e8a5973228953b681f431833a16bd
1836 Merge: 9b5f7d1 50d5298
1837 Author: Khaled Atef <kaa2168@columbia.edu>
1838 Date:   Mon Nov 30 02:15:10 2015 -0500
1839
```

```
1840     Merge branch 'master' of https://github.com/DavidWatkins/Dice
1841
1842     commit 9b5f7d18f8f895d4979b9a6bc32164262cb9bc31
1843     Author: Khaled Atef <kaa2168@columbia.edu>
1844     Date:   Mon Nov 30 02:14:35 2015 -0500
1845
1846     basic inhertiance test added
1847
1848     commit 50d52984ad083eac3722a630cd027a0714537459
1849     Merge: 75a41c3 1932754
1850     Author: David Watkins <davidw@tkins.me>
1851     Date:   Mon Nov 30 01:39:36 2015 -0500
1852
1853     Merge branch 'master' of https://github.com/DavidWatkins/Dice
1854
1855     commit 75a41c3cf0da048bb6b76875f97d428cf84d8e41
1856     Author: David Watkins <davidw@tkins.me>
1857     Date:   Mon Nov 30 01:39:06 2015 -0500
1858
1859     Ifs semi-implemented, multi-line programs work now
1860
1861     commit 9f1d6c75a4454794ac10e636bb6dd07291cbd642
1862     Merge: a874f50 1932754
1863     Author: Emily Chen <emchennyc@gmail.com>
1864     Date:   Mon Nov 30 01:38:40 2015 -0500
1865
1866     Merge branch 'master' of https://github.com/DavidWatkins/Dice into emily
1867
1868     commit a874f50a43d711086c8038dc92c06014bf11a39c
1869     Author: Emily Chen <emchennyc@gmail.com>
1870     Date:   Mon Nov 30 01:37:41 2015 -0500
1871
1872     check_binop succeeds when only literal operands; doesn't handle IDs yet
1873
1874     commit 19327541aec867c8b3617ce7a55c2b8c30afc56a
1875     Author: Khaled Atef <kaa2168@columbia.edu>
1876     Date:   Mon Nov 30 01:35:32 2015 -0500
1877
1878     array tests added for single and multidimensional arrays.
1879
1880     commit d1a88f6cc7a112e34869a03c36f0fbb8c95dea73
1881     Author: Khaled Atef <kaa2168@columbia.edu>
1882     Date:   Sun Nov 29 23:56:36 2015 -0500
1883
1884     mroe tests
1885
1886     commit 01c53a3905d6dda85bd7d407ce1024c049401f3f
1887     Merge: 0052631 96c9f21
1888     Author: Emily Chen <emchennyc@gmail.com>
```

```
1889 Date: Sat Nov 28 16:48:45 2015 -0500
1890
1891 Merge pull request #50 from DavidWatkins/epsilon
1892
1893 Epsilon
1894
1895 commit 96c9f21a876921f9fe7b54c7c5520d2684926080
1896 Merge: 0828f97 0052631
1897 Author: Emily Chen <ec2805@columbia.edu>
1898 Date: Sat Nov 28 16:43:56 2015 -0500
1899
1900 Merge branch 'master' of https://github.com/DavidWatkins/Dice into epsilon
1901
1902 commit 0828f97195361fcde1f315f76c0b9b40602fdfa6
1903 Author: Emily Chen <ec2805@columbia.edu>
1904 Date: Sat Nov 28 16:43:40 2015 -0500
1905
1906 current state of LRM, WIP
1907
1908 commit 00526316b382f1fcdbf7e20dd8116d76f3c0af49
1909 Author: David Watkins <davidw@tkins.me>
1910 Date: Thu Nov 26 03:53:13 2015 -0500
1911
1912 Added environments as return types for expressions and statements
1913
1914 commit 7bd0f08fd5735207d23ef0282f75571779c17032
1915 Author: David Watkins <davidw@tkins.me>
1916 Date: Thu Nov 26 03:28:16 2015 -0500
1917
1918 Added assignment type checking
1919
1920 commit 91f50320126a774977e963b002952cffcdaf8c0b
1921 Author: David Watkins <davidw@tkins.me>
1922 Date: Thu Nov 26 03:18:53 2015 -0500
1923
1924 Reorganized analyser unop
1925
1926 commit eb1e72d42ffccfa994b21db18c5ee7594b3086cb
1927 Author: David Watkins <davidw@tkins.me>
1928 Date: Thu Nov 26 03:09:06 2015 -0500
1929
1930 Print will now accept variable number of arguments and print integers
1931
1932 commit 2697f7d36eee3267d4d08acd72ee36218cfe885f
1933 Author: David Watkins <davidw@tkins.me>
1934 Date: Thu Nov 26 02:43:40 2015 -0500
1935
1936 Added reserved functions to analyzer
1937
```

```
1938 commit f016c356c05016b220b3503f7ef331c0cc6fe9e9
1939 Author: David Watkins <davidw@tkins.me>
1940 Date: Wed Nov 25 23:14:40 2015 -0500
1941
1942 Analyzer now uses SExpr instead of expr
1943
1944 commit 405feab53aeb98996d924e1d0c054b2c057893b8
1945 Author: David Watkins <davidw@tkins.me>
1946 Date: Wed Nov 25 20:46:37 2015 -0500
1947
1948 Added test ocaml code to produce llvm
1949
1950 commit fb92dc93387bc04a842ce20414642a8e0d6be079
1951 Merge: a70917b d3bfd36
1952 Author: Emily Chen <ec2805@columbia.edu>
1953 Date: Wed Nov 25 14:19:20 2015 -0500
1954
1955 Merge branch 'master' of https://github.com/DavidWatkins/Dice into epsilon
1956
1957 commit d3bfd36c6a493a1c4b768bcc86049b5245975fdc
1958 Author: David Watkins <davidw@tkins.me>
1959 Date: Mon Nov 23 03:55:35 2015 -0500
1960
1961 Added a lot
1962
1963 commit 18c53d74b916b57cf79523da4bb5532408f0d623
1964 Merge: e714714 c3635ab
1965 Author: David Watkins <DavidWatkins@users.noreply.github.com>
1966 Date: Sat Nov 21 22:30:11 2015 -0500
1967
1968 Merge pull request #46 from DavidWatkins/Kreygasm
1969
1970 Kreygasm
1971
1972 commit c3635ab4859a46182ea3be101ffe08c80567da83
1973 Author: nethacker11 <philip.schiffrin@gmail.com>
1974 Date: Sat Nov 21 22:28:57 2015 -0500
1975
1976 duplicates checked in stringmaps
1977
1978 commit 347bb718b69bba5cedcf8d920e81fb46a9857602
1979 Author: David Watkins <davidw@tkins.me>
1980 Date: Sat Nov 21 21:58:42 2015 -0500
1981
1982 fubic
1983
1984 commit 796dad808edb739bb79e140ae8284affc416ba8e
1985 Author: nethacker11 <philip.schiffrin@gmail.com>
1986 Date: Sat Nov 21 20:30:52 2015 -0500
```

```
1987
1988     analyzer broken
1989
1990 commit 83453a96dc22e2b0cb3c0b5fadda6c38cd34f84d
1991 Author: nethacker11 <philip.schiffrin@gmail.com>
1992 Date:   Fri Nov 20 15:34:00 2015 -0500
1993
1994     updated analyzer for global table
1995
1996 commit a70917bebd8b122fc1456a0de6d647f27e124378
1997 Author: Emily Chen <ec2805@columbia.edu>
1998 Date:   Tue Nov 17 05:39:53 2015 -0500
1999
2000     specify wraparound behavior for char overflow during addition operation
2001
2002 commit 5ef7d4040ae2a3f384dc8e49108df9f911da54e8
2003 Author: Emily Chen <ec2805@columbia.edu>
2004 Date:   Tue Nov 17 05:32:45 2015 -0500
2005
2006     fixed typos in Type section
2007
2008 commit 519ecd38eb0ff36345e404500a58799ab6e6f22e
2009 Author: Emily Chen <ec2805@columbia.edu>
2010 Date:   Tue Nov 17 05:32:15 2015 -0500
2011
2012     fixed typos in Type section
2013
2014 commit 1dbea9cdd7ce99e8afb045d825c10cf7b61da1e6
2015 Author: Emily Chen <ec2805@columbia.edu>
2016 Date:   Tue Nov 17 05:28:27 2015 -0500
2017
2018     lrm pdf
2019
2020 commit 218cdd226af54fc7a12aa54174686808b9c0c080
2021 Author: Emily Chen <ec2805@columbia.edu>
2022 Date:   Tue Nov 17 05:27:01 2015 -0500
2023
2024     expressions emulate K&R reference
2025
2026 commit a23065b93cfa8ea563b2e5cafe47e4001364329f
2027 Author: Emily Chen <ec2805@columbia.edu>
2028 Date:   Tue Nov 17 04:14:47 2015 -0500
2029
2030     remove examples from Types section
2031
2032 commit e71471403e598ff74fff7e1c18b6c26f84db7c4e
2033 Author: Emily Chen <ec2805@columbia.edu>
2034 Date:   Tue Nov 17 01:26:58 2015 -0500
2035
```

```
2036     update regex for int, float
2037
2038     commit 01369938d06a83b7a411e97ea7f3105355ecb1c7
2039     Author: David Watkins <davidw@tkins.me>
2040     Date:   Mon Nov 16 21:09:30 2015 -0500
2041
2042         Added new keyword, fixed pretty printing, allowed varied variable declaration
2043
2044     commit 90ac3e878efdb7d8471a49ac07b2717d568394ec
2045     Author: David Watkins <davidw@tkins.me>
2046     Date:   Mon Nov 16 05:00:47 2015 -0500
2047
2048         Hello world demo code
2049
2050     commit 4a37b8b8e8fe6d695354db10b78f4273584ece35
2051     Author: David Watkins <davidw@tkins.me>
2052     Date:   Mon Nov 16 04:55:33 2015 -0500
2053
2054         Added escape characters to string literals
2055
2056     commit cc898068bdd320886cbf6d6d950edc00a5cb8afe
2057     Merge: be88ee9 d8ed6e5
2058     Author: David Watkins <DavidWatkins@users.noreply.github.com>
2059     Date:   Sun Nov 15 15:24:08 2015 -0500
2060
2061         Merge pull request #7 from DavidWatkins/Delta
2062
2063         Delta
2064
2065     commit be88ee9635bd7a60d134eac92cd8516dc08ccd06
2066     Author: David Watkins <davidw@tkins.me>
2067     Date:   Sun Nov 15 03:00:33 2015 -0500
2068
2069         Removed bindings.c
2070
2071     commit b35203029ea05992df4d7356c556d8250379ec3e
2072     Merge: 1a79286 bdfc46f
2073     Author: David Watkins <davidw@tkins.me>
2074     Date:   Sun Nov 15 02:55:59 2015 -0500
2075
2076         Merge branch 'Delta' of https://github.com/DavidWatkins/Dice into HEAD
2077
2078     commit d8ed6e5ac77c66a1e28e43a91d1c7c90d10d096c
2079     Merge: 819c652 bdfc46f
2080     Author: David Watkins <davidw@tkins.me>
2081     Date:   Sun Nov 15 02:49:47 2015 -0500
2082
2083         Merge branch 'Delta' of https://github.com/DavidWatkins/Dice into Delta
2084
```

```
2085 commit bdfc46f92ab376ea29d6c672efa1c95cdc547f78
2086 Author: Khaled Atef <kaa2168@columbia.edu>
2087 Date: Sun Nov 15 02:49:23 2015 -0500
2088
2089     fixed cleaning up of temp files
2090
2091 commit 819c652f8af6058175b61339d75612f798b7f446
2092 Author: David Watkins <davidw@tkins.me>
2093 Date: Sun Nov 15 02:43:59 2015 -0500
2094
2095     Added unary minus
2096
2097 commit b79952cc5f486e080c31a7ae00b8977fa6812aa2
2098 Author: David Watkins <davidw@tkins.me>
2099 Date: Sun Nov 15 02:37:18 2015 -0500
2100
2101     Removed - from int and float literals
2102
2103 commit b7e306b4eb8dbb8859a79a7242064713f923605d
2104 Author: David Watkins <davidw@tkins.me>
2105 Date: Sun Nov 15 02:29:54 2015 -0500
2106
2107     Fixed rule with return
2108
2109 commit 30877b0821d29c59f8e86dbe8a0d4437d63dc6bc
2110 Author: Khaled Atef <kaa2168@columbia.edu>
2111 Date: Sun Nov 15 02:21:01 2015 -0500
2112
2113     tester corrected to work with lli
2114
2115 commit f037b3b5dc89409d59d55b5be4ed2a816b317be6
2116 Merge: e843f0e ae1d756
2117 Author: Khaled Atef <kaa2168@columbia.edu>
2118 Date: Sun Nov 15 02:19:51 2015 -0500
2119
2120     Merge branch 'Delta' of https://github.com/DavidWatkins/Dice into Delta
2121
2122 commit e843f0ef31580d6846586a8ca49c2840222276c9
2123 Author: Khaled Atef <kaa2168@columbia.edu>
2124 Date: Sun Nov 15 02:19:34 2015 -0500
2125
2126     Corrected syntax errors in test case code
2127
2128 commit ae1d7560e16dd7eecdbc34a291d8f4e41a97eeeb
2129 Author: David Watkins <DavidWatkins@users.noreply.github.com>
2130 Date: Sun Nov 15 02:10:16 2015 -0500
2131
2132     Update README.md
2133
```

```
2134 commit 026fd5026bf957515f9b0902aa2d278b48197fe0
2135 Author: David Watkins <DavidWatkins@users.noreply.github.com>
2136 Date: Sun Nov 15 01:55:44 2015 -0500
2137
2138     Update README.md
2139
2140 commit 5ca0b69612e326f2523c5c7542f1ae5ea02d6f24
2141 Author: David Watkins <davidw@tkins.me>
2142 Date: Sat Nov 14 20:14:41 2015 -0500
2143
2144     Small edit to readme
2145
2146 commit 1c2547eef86a517507a6cbec1655f38df7875290
2147 Author: David Watkins <davidw@tkins.me>
2148 Date: Sat Nov 14 20:12:53 2015 -0500
2149
2150     Small changes
2151
2152 commit 54d7539d119aa459278dca7b3bcb68c248054948
2153 Author: David Watkins <davidw@tkins.me>
2154 Date: Sat Nov 14 20:09:16 2015 -0500
2155
2156     Added to README
2157
2158 commit d88306a8e86159b467a5da701bd315ce8e713d5a
2159 Author: David Watkins <davidw@tkins.me>
2160 Date: Sat Nov 14 19:57:11 2015 -0500
2161
2162     Compiler works, run build.sh
2163
2164 commit 32d87fa0fd8a81778ddd2d61936d64f5ef6aebc6
2165 Author: Khaled Atef <kaa2168@columbia.edu>
2166 Date: Sat Nov 14 17:40:47 2015 -0500
2167
2168     modified test script to use lli
2169
2170 commit 0db46dad8c7f8f97385be5602b4340efb7485c44
2171 Merge: 8aa9ed2 2287265
2172 Author: David Watkins <davidw@tkins.me>
2173 Date: Sat Nov 14 16:38:18 2015 -0500
2174
2175     Merge branch 'Delta' of https://github.com/DavidWatkins/Dice into Delta
2176
2177 commit 8aa9ed21512f946604e9824e72d0f32f48460cb9
2178 Author: David Watkins <davidw@tkins.me>
2179 Date: Sat Nov 14 16:37:59 2015 -0500
2180
2181     Works!!!!!!
2182
```



```
2183 commit 228726581d0b46dd87aeeacfe2fc66b276ecd434
2184 Merge: 01e738d 65f6ba6
2185 Author: Khaled Atef <kaa2168@columbia.edu>
2186 Date: Sat Nov 14 16:19:25 2015 -0500
2187
2188 Merge branch 'Delta' of https://github.com/DavidWatkins/Dice into Delta
2189
2190 commit 01e738dae2e637fd7a2265aa16bc3135172b24e2
2191 Author: Khaled Atef <kaa2168@columbia.edu>
2192 Date: Sat Nov 14 16:11:58 2015 -0500
2193
2194 testing script and basic test cases
2195
2196 commit 65f6ba6989f389b3db5b95cb4f0eaf0438f10157
2197 Author: David <davidw@tkins.me>
2198 Date: Sat Nov 14 13:41:36 2015 -0500
2199
2200 Made changes yo
2201
2202 commit 7dbe37512674db6a5f911bcc336878c45c5c1aca
2203 Author: David <davidw@tkins.me>
2204 Date: Sat Nov 14 03:38:45 2015 -0500
2205
2206 I give up for now
2207
2208 commit 04c4053bf43276ac073c4141ce1d2f79ddbc3452
2209 Author: David <davidw@tkins.me>
2210 Date: Sat Nov 14 03:31:54 2015 -0500
2211
2212 iWhatever
2213
2214 commit 1bfde45790ff5c37a06fed8a732d95343d6b09fe
2215 Author: David Watkins <djwt146@columbia.edu>
2216 Date: Fri Nov 13 23:51:52 2015 -0500
2217
2218 Again WIP
2219
2220 commit 6bc13cfadb5458a77b2d521390683981502d3cd1
2221 Author: David Watkins <djwt146@columbia.edu>
2222 Date: Fri Nov 13 16:25:05 2015 -0500
2223
2224 Added new way to make, figuring out layout for code based on tutorial
2225
2226 commit 1a79286feb4a6b21d8ded437dda312143a485f9b
2227 Author: David Watkins <davidw@tkins.me>
2228 Date: Thu Nov 12 20:37:37 2015 -0400
2229
2230 Wrong rule for utils
2231
```

```
2232 commit 550cd68be9b44ca6b4bb57dd3ca6539ae4ee04ca
2233 Author: David Watkins <djrival7@gmail.com>
2234 Date: Wed Nov 11 15:44:05 2015 -0500
2235
2236 Created base code for compiler and improved processinclude
2237
2238 commit e37f596018482de09bad4e87b85a9b346c33b374
2239 Author: David Watkins <dju2146@columbia.edu>
2240 Date: Wed Nov 11 02:27:24 2015 -0500
2241
2242 Added more descriptive error messages to dice files with incorrect syntax
2243
2244 commit 2c93957d1cf2401c8eae282123bcaa397290c194
2245 Author: David Watkins <dju2146@columbia.edu>
2246 Date: Wed Nov 11 01:14:35 2015 -0500
2247
2248 Changed primitive arrays to support inclusion of expressions and fixed
2249 escaped char literals
2250
2251 commit bfe58d3de921dd7428b6fb18d2ba2240336a0164
2252 Merge: 13e70b1 77193cc
2253 Author: Khaled Atef <kaa2168@columbia.edu>
2254 Date: Mon Nov 9 02:28:11 2015 -0500
2255
2256 Merge branch 'master' of https://github.com/DavidWatkins/Dice
2257
2258 commit 13e70b1a603b87eb5c59dd96a7908a68e2b4286e
2259 Author: Khaled Atef <kaa2168@columbia.edu>
2260 Date: Mon Nov 9 02:27:48 2015 -0500
2261
2262 testing script implemented for Scanner tokenizer with some basic test cases. More to follow soon
2263
2264 commit 77193cc351212bbe8fd9ff02c73f71c2a958cf91
2265 Author: Emily Chen <ec2805@columbia.edu>
2266 Date: Mon Nov 9 05:16:18 2015 +0000
2267
2268 updated roles, re-rendered LRM pdf
2269
2270 commit 833154914d227be8b3d75ff2266cb5b1d016f185
2271 Author: Emily Chen <ec2805@columbia.edu>
2272 Date: Mon Nov 9 05:13:39 2015 +0000
2273
2274 updated roles
2275
2276 commit 48b5419853a7dad44ae3ef590ddb48f9f0fc40fb
2277 Author: Emily Chen <ec2805@columbia.edu>
2278 Date: Mon Nov 9 05:08:07 2015 +0000
2279
2280 updated LRM pdf
```

```
2281
2282 commit 62673cf7dcffe60a07ba5711e4df02c9d4542589
2283 Author: Emily Chen <ec2805@columbia.edu>
2284 Date: Sun Nov 8 02:53:38 2015 +0000
2285
2286     add description for logical operators and member access operator
2287
2288 commit 650993c85b05f1415aa1fedf4f995358d5e94f6b
2289 Author: Emily Chen <ec2805@columbia.edu>
2290 Date: Sun Nov 8 02:24:12 2015 +0000
2291
2292     add example of inheritance using "extends" kw
2293
2294 commit eb8488828a35dfe92828ff403eb3fa60b734eaf2
2295 Author: Emily Chen <ec2805@columbia.edu>
2296 Date: Sun Nov 8 02:05:59 2015 +0000
2297
2298     updated examples so they don't declare and initialize in same statement
2299
2300 commit ac91d9956899f8a7246bef381942afcf505f95b6
2301 Author: Emily Chen <ec2805@columbia.edu>
2302 Date: Sun Nov 8 01:06:22 2015 +0000
2303
2304     no class name collisions within module or between modules
2305
2306 commit 662d69a4aa9be43b54f85db25ec23ced249629b6
2307 Author: Emily Chen <ec2805@columbia.edu>
2308 Date: Sun Nov 8 00:49:20 2015 +0000
2309
2310     change .di to .dice; specify that recursive includes are not supported
2311
2312 commit 61afc126141b1ceaa5618ca4af55e1b6229a9f2f
2313 Author: Emily Chen <ec2805@columbia.edu>
2314 Date: Sun Nov 8 00:39:14 2015 +0000
2315
2316     describe Include statement
2317
2318 commit 9d04c148421a6793a71f17543fba74b0179b5a9e
2319 Author: Emily Chen <ec2805@columbia.edu>
2320 Date: Sat Nov 7 23:59:35 2015 +0000
2321
2322     updated array declaration, initialization, access in LRM
2323
2324 commit c73e469da24918a33fbc32fadf46eef0bb34a83d
2325 Merge: 51f4c2e 18a9ca2
2326 Author: David Watkins <djrival7@gmail.com>
2327 Date: Sat Nov 7 13:55:09 2015 -0500
2328
2329     Merge branch 'master' of https://github.com/DavidWatkins/DiceLanguage
```

```
2330
2331 commit 51f4c2e5e49ea49f1b9d7fdc84b116f79da7ef74
2332 Author: David Watkins <djrival7@gmail.com>
2333 Date: Sat Nov 7 12:16:59 2015 -0500
2334
2335     Added nested primitive arrays to parser
2336
2337 commit 18a9ca288d017dffc5eb5240f6ed4e6551f4484
2338 Author: Emily Chen <ec2805@columbia.edu>
2339 Date: Fri Nov 6 23:24:59 2015 +0000
2340
2341     updated operator precedence in parser
2342
2343 commit 5a8cef865cec749060472a2da08f68e5f66ab603
2344 Merge: 8a3a33b 233b9ae
2345 Author: David Watkins <DavidWatkins@users.noreply.github.com>
2346 Date: Thu Nov 5 01:55:34 2015 -0500
2347
2348     Merge pull request #3 from DavidWatkins/DavidFix
2349
2350     David fix
2351
2352 commit 233b9ae4d2ad329217da5f89a02208e65f0f1056
2353 Merge: 2339f25 8a3a33b
2354 Author: David Watkins <DavidWatkins@users.noreply.github.com>
2355 Date: Thu Nov 5 01:51:37 2015 -0500
2356
2357     Merge pull request #2 from DavidWatkins/master
2358
2359     Merge pull request #1 from DavidWatkins/DavidFix
2360
2361 commit 2339f25231f1ee437ed6703545d6c7b010ec99a7
2362 Author: David Watkins <djwt146@columbia.edu>
2363 Date: Thu Nov 5 01:47:31 2015 -0500
2364
2365     Added AST printing method by using menhir inside ocaml
2366
2367 commit 8a3a33b299306b2322b4c87df2c42f559bb0f612
2368 Merge: 9507d54 d095d57
2369 Author: David Watkins <DavidWatkins@users.noreply.github.com>
2370 Date: Thu Nov 5 00:57:10 2015 -0500
2371
2372     Merge pull request #1 from DavidWatkins/DavidFix
2373
2374     David fix
2375
2376 commit d095d57af5fb83805fcbd5d70ff133facad7f64a
2377 Author: David Watkins <djwt146@columbia.edu>
2378 Date: Thu Nov 5 00:37:34 2015 -0500
```

```
2379
2380     Pretty printer bug fixed, tokenizer now prints line numbers
2381
2382 commit 7c800f61455e0a09d421afb4d384e1d0a07f5283
2383 Author: David Watkins <djwt146@columbia.edu>
2384 Date:   Wed Nov 4 22:12:26 2015 -0500
2385
2386     Made changes to front end, pretty print and tokenizer work
2387
2388 commit b2a0bbd31c3484a2a914a1fb01cd8e05ecbc074f
2389 Author: David Watkins <djwt146@columbia.edu>
2390 Date:   Wed Nov 4 20:41:00 2015 -0500
2391
2392     Fixed makefile?
2393
2394 commit 14f5ace95044a80e3d8d2a49e5e6a645f704a6ae
2395 Author: David Watkins <djwt146@columbia.edu>
2396 Date:   Wed Nov 4 20:35:55 2015 -0500
2397
2398     This is just a test run of additional useful files, needs to be compiled
2399     on a unix system
2400
2401 commit a87e9b9a3706cb33fb79f260391b786bf53c6a40
2402 Author: David Watkins <djrival7@gmail.com>
2403 Date:   Wed Nov 4 15:57:22 2015 -0500
2404
2405     Fixed parser with class keyword, removed array keyword
2406
2407 commit 41825d291a910847c3aa0d5d67f5c60f3698fcef
2408 Author: David Watkins <djrival7@gmail.com>
2409 Date:   Wed Nov 4 15:50:46 2015 -0500
2410
2411     Revert "Fixed operator precedence"
2412
2413     This reverts commit d132fdb8d21ba69a8d9d1c71c0ab71af5231eac0.
2414
2415 commit d132fdb8d21ba69a8d9d1c71c0ab71af5231eac0
2416 Author: David Watkins <djrival7@gmail.com>
2417 Date:   Wed Nov 4 15:24:56 2015 -0500
2418
2419     Fixed operator precedence
2420
2421 commit 0726d908374df6c447d82290e06a83b84d0fdd0a
2422 Author: David Watkins <djrival7@gmail.com>
2423 Date:   Wed Nov 4 15:23:35 2015 -0500
2424
2425     Fixed bucket_args to refer to general expr list
2426
2427 commit ffec7d32cba2126885009518d79f072feeaa88628
```

```
2428 Author: David Watkins <djrival7@gmail.com>
2429 Date: Wed Nov 4 15:21:37 2015 -0500
2430
2431 Added datatypes to primitive arrays
2432
2433 commit 096f5a8bd45b44f23a42abaf837ad1f55597bce3
2434 Author: David Watkins <djrival7@gmail.com>
2435 Date: Wed Nov 4 15:18:01 2015 -0500
2436
2437 Fixed bug, apparently no issues wot
2438
2439 commit f30730a1cd041a3df6010c1ee62a59b17842e68f
2440 Author: David Watkins <dju2146@columbia.edu>
2441 Date: Wed Nov 4 14:43:20 2015 -0500
2442
2443 Fixed Menhir errors
2444
2445 commit 89a3de8e2e78735910eb385a3b51f4795e115c62
2446 Author: David Watkins <djrival7@gmail.com>
2447 Date: Wed Nov 4 13:38:03 2015 -0500
2448
2449 Removed extraneous files
2450
2451 commit c6d4db34bcca5aa6cc86a4e6f670a858c2f0b6bc
2452 Author: David Watkins <djrival7@gmail.com>
2453 Date: Wed Nov 4 13:36:38 2015 -0500
2454
2455 Cleaned up git directory
2456
2457 commit 4d157027e77211c94f295f34cef0d03a19c7f102
2458 Author: David Watkins <djrival7@gmail.com>
2459 Date: Wed Nov 4 13:20:24 2015 -0500
2460
2461 Found a more elegant solution to array problem
2462
2463 commit d7f28aa1dad4d1e788ab7b2aaab962372dfe1e71
2464 Author: David Watkins <djrival7@gmail.com>
2465 Date: Wed Nov 4 00:06:05 2015 -0500
2466
2467 No shift reduce but not ideal
2468
2469 commit d29a030e18489e489f0f8941cfbc70cc85c81a03
2470 Author: David Watkins <djrival7@gmail.com>
2471 Date: Tue Nov 3 23:45:34 2015 -0500
2472
2473 Super close, just ambiguity surroundign array access
2474
2475 commit 5473e62f147ec34eaccd4289930c4b2144d7c968
2476 Author: David Watkins <djrival7@gmail.com>
```

```
2477 Date: Tue Nov 3 18:04:37 2015 -0500
2478
2479 More stuff
2480
2481 commit dc6ad10dc89c389534ec225082c64da35a512268
2482 Author: David Watkins <djrival7@gmail.com>
2483 Date: Tue Nov 3 14:10:00 2015 -0500
2484
2485 Shift/Reduce down to 2, fixed layout of cdecl and cbody
2486
2487 commit 9507d5426cd130cc927941a4620383040c895717
2488 Author: Khaled Atef <kaa2168@columbia.edu>
2489 Date: Mon Nov 2 13:07:36 2015 -0500
2490
2491 BIBLETHUMP delimiter still in this version. AST modified to remove actions not used
2492
2493 commit 5419ec530d153c46c7ee4a3c12765a73c0fcb0c0
2494 Author: Khaled Atef <kaa2168@columbia.edu>
2495 Date: Mon Nov 2 01:32:27 2015 -0500
2496
2497 Corrected the array access production to account for multidimensional arrays
2498
2499 commit ce9ed98c205d39871c544a3640d92a6e1c77c31f
2500 Author: Khaled Atef <kaa2168@columbia.edu>
2501 Date: Mon Nov 2 01:29:07 2015 -0500
2502
2503 Parser compiles w/o any errors, but not tested yet. Multidimensional arrays implemented
2504
2505 commit 8ee5b859b0fea8a91154fe05694cd875c05fbd9e
2506 Author: Philip Schiffrin <philip.schiffrin@gmail.com>
2507 Date: Thu Oct 29 22:58:33 2015 -0400
2508
2509 hacked a solution for the last shift/reduce by adding token FUN at beginning of fdecls
2510
2511 commit bd6ecf78c93229b46957e67a50056e975ddcc072
2512 Author: Philip Schiffrin <philip.schiffrin@gmail.com>
2513 Date: Thu Oct 29 21:29:48 2015 -0400
2514
2515 fixed all reduce/reduce and most shift/reduce errors in the parser, lost most of our logic
2516
2517 commit d8cdf93ba6240b597307acb59407165c5b8eeff2
2518 Merge: df1979e 5a7f132
2519 Author: David Watkins <djw2146@columbia.edu>
2520 Date: Mon Oct 26 20:12:34 2015 -0400
2521
2522 Merge branch 'master' of https://github.com/DavidWatkins/DiceLanguage
2523
2524 commit df1979e32572f5bb0e3e5e4c95d22ff99ae77c53
2525 Author: David Watkins <djw2146@columbia.edu>
```

```
2526 Date:   Mon Oct 26 20:12:23 2015 -0400
2527
2528     Fixed language to dice again
2529
2530 commit 5a7f13237ad7c32bc9609653e30b3f71b9c3aaf1
2531 Author: David Watkins <DavidWatkins@users.noreply.github.com>
2532 Date:   Mon Oct 26 16:30:45 2015 -0400
2533
2534     Update README.md
2535
2536 commit f88282699ce5b7e3ea3c672c51e1ac1a69527687
2537 Author: David Watkins <djw2146@columbia.edu>
2538 Date:   Mon Oct 26 02:57:24 2015 -0400
2539
2540     Fixed with edits
2541
2542 commit 6778395705ceea2284b4f1e13c022ddfd2c45f64
2543 Author: David Watkins <djw2146@columbia.edu>
2544 Date:   Mon Oct 26 02:38:35 2015 -0400
2545
2546     Fixed intro
2547
2548 commit 3905c2317321473a3e481e2b52902461ba03d5bb
2549 Merge: 764eeb4 f207e7c
2550 Author: David Watkins <djw2146@columbia.edu>
2551 Date:   Mon Oct 26 02:28:08 2015 -0400
2552
2553     Merge branch 'master' of https://github.com/DavidWatkins/DiceLanguage
2554
2555 commit 764eeb44f8ffa599a2451b037b67078683d6eef9
2556 Author: David Watkins <djw2146@columbia.edu>
2557 Date:   Mon Oct 26 02:27:49 2015 -0400
2558
2559     Finished final draft of LRM
2560
2561 commit f207e7c0a231b852c21103503bbee8fa6edb483a
2562 Merge: d37bf92 0c85801
2563 Author: Khaled Atef <kaa2168@columbia.edu>
2564 Date:   Mon Oct 26 00:34:51 2015 -0400
2565
2566     Merge branch 'master' of https://github.com/DavidWatkins/JFlat
2567
2568 commit d37bf92df1077fd3f612827f14c34cb7e6095af4
2569 Author: Khaled Atef <kaa2168@columbia.edu>
2570 Date:   Mon Oct 26 00:34:34 2015 -0400
2571
2572     Parser compiles, but produces 457 reduce/reduce errors.
2573
2574 commit 0c858019e2b58ae63d8ec581a185953a558a464f
```



```
2575 Author: David Watkins <djw2146@columbia.edu>
2576 Date: Sun Oct 25 22:18:37 2015 -0400
2577
2578 Removed extraneous file
2579
2580 commit fe8bd9d0a8abf0f521a6ff44da6d7d615fa259b1
2581 Author: Emily Chen <ec2805@columbia.edu>
2582 Date: Sun Oct 25 04:29:11 2015 +0000
2583
2584 adding content for statements section; TODO include statements
2585
2586 commit 35b69e9bdf8fb8623f19d7d75790c7197c188c74
2587 Author: Emily Chen <ec2805@columbia.edu>
2588 Date: Sun Oct 25 03:30:32 2015 +0000
2589
2590 remove elseif keyword
2591
2592 commit 71bfd4fc327bcce124455bea1a731351a884ea25
2593 Author: Emily Chen <ec2805@columbia.edu>
2594 Date: Sun Oct 25 02:33:32 2015 +0000
2595
2596 update Statements sections
2597
2598 commit c991aa20fcad8e2a7ddab6f0552c105cd943e752
2599 Author: David Watkins <djrival7@gmail.com>
2600 Date: Sat Oct 24 22:23:32 2015 -0400
2601
2602 Whatevs
2603
2604 commit b4a0296539e07421a7436d7530154bbd8208158d
2605 Author: David Watkins <djrival7@gmail.com>
2606 Date: Sat Oct 24 21:34:36 2015 -0400
2607
2608 More
2609
2610 commit b6d7bdda712a8bbc77c923861fa2dd1161f383e9
2611 Merge: 00c7c8b 8b447ba
2612 Author: David Watkins <djrival7@gmail.com>
2613 Date: Sat Oct 24 21:28:34 2015 -0400
2614
2615 Merge branch 'master' of https://github.com/DavidWatkins/DiceLanguage
2616
2617 commit 00c7c8bffa49163684020efa60cccd3ae70e481c4
2618 Author: David Watkins <djrival7@gmail.com>
2619 Date: Sat Oct 24 21:27:44 2015 -0400
2620
2621 Stuff
2622
2623 commit 8b447ba1a5c2cbfdb18ec5edf26527a926f28e0c
```

```
2624 Author: Emily Chen <ec2805@columbia.edu>
2625 Date: Sun Oct 25 01:01:54 2015 +0000
2626
2627     updated constructor definition
2628
2629 commit 00225f13ceeb3ac4ba51811a308cf069046b58af
2630 Author: Emily Chen <ec2805@columbia.edu>
2631 Date: Sun Oct 25 00:21:34 2015 +0000
2632
2633     method names cannot be same as class name
2634
2635 commit 02cc3c7206ed0626bf93ba93bb7924f9896431ea
2636 Merge: fed5273 1335b3e
2637 Author: Emily Chen <ec2805@columbia.edu>
2638 Date: Sat Oct 24 23:16:24 2015 +0000
2639
2640     Merge branch 'master' of https://github.com/DavidWatkins/JFlat
2641
2642 commit 1335b3e88c32dbbcdfe8e0954bb9fd0b89ce4903
2643 Merge: 45adcd4 d56fbb6
2644 Author: Khaled Atef <kaa2168@columbia.edu>
2645 Date: Sat Oct 24 19:24:12 2015 -0400
2646
2647     Merge branch 'master' of https://github.com/DavidWatkins/JFlat
2648
2649 commit d56fbb6510545503ea3a1ed8eeb0dd38a20cd42c
2650 Author: Philip Schiffrin <philip.schiffrin@gmail.com>
2651 Date: Sat Oct 24 19:23:49 2015 -0400
2652
2653     changed doubles to floats
2654
2655 commit 45adcd4da21201da272a7807a47ebde99fbb1e77
2656 Author: Khaled Atef <kaa2168@columbia.edu>
2657 Date: Sat Oct 24 19:23:04 2015 -0400
2658
2659     changed double to float
2660
2661 commit be271ea2be45684e4cc07fb2723a75343509bf6a
2662 Author: David Watkins <djrival7@gmail.com>
2663 Date: Sat Oct 24 19:18:21 2015 -0400
2664
2665     LOL more stuff
2666
2667 commit fed5273dc9e7adae462996c0a7fadc4bdfb91bdd
2668 Merge: 119f1a5 d56fbb6
2669 Author: Emily Chen <ec2805@columbia.edu>
2670 Date: Sat Oct 24 23:15:30 2015 +0000
2671
2672     Merge branch 'master' of https://github.com/DavidWatkins/JFlat
```

```
2673
2674 commit 119f1a57fef847539f3ab76ba3af1bbbe0ef91b2
2675 Author: Emily Chen <ec2805@columbia.edu>
2676 Date: Sat Oct 24 23:13:31 2015 +0000
2677
2678     update lexical elements to replace double w float
2679
2680 commit a793510d204f60c87c8d44ec5f03c6ea7713c1f0
2681 Author: David Watkins <djrival7@gmail.com>
2682 Date: Sat Oct 24 18:49:37 2015 -0400
2683
2684     Whatever
2685
2686 commit 4ef1e1b02c3fbaace4d0c4471e4bd048b5c7e8e1
2687 Author: David Watkins <djwt146@columbia.edu>
2688 Date: Sat Oct 24 18:20:07 2015 -0400
2689
2690     Added array and object creation to parser
2691
2692 commit 923b6e6ee2e37705f88441be373b71e478475326
2693 Author: David Watkins <djwt146@columbia.edu>
2694 Date: Sat Oct 24 17:35:10 2015 -0400
2695
2696     Added Program def
2697
2698 commit 0873e88184cb95d37ce0e6ceb7a320fda04f3dbe
2699 Author: David Watkins <djwt146@columbia.edu>
2700 Date: Sat Oct 24 16:52:25 2015 -0400
2701
2702     Added more info to the parser
2703
2704 commit 049a058293023065afb6f90def5b215c298e5511
2705 Author: Khaled Atef <kaa2168@columbia.edu>
2706 Date: Sat Oct 24 14:36:56 2015 -0400
2707
2708     Added negative doubles in scanner
2709
2710 commit b10a719cae082c8add0a509266264312d528a7df
2711 Author: Khaled Atef <kaa2168@columbia.edu>
2712 Date: Sat Oct 24 13:42:59 2015 -0400
2713
2714     Corrected precedence chart by removing modulo reference
2715
2716 commit 32497722d8a7efbba2baaaa8ab80ddf899b8f429
2717 Merge: 1ca3359 388cada
2718 Author: Emily Chen <ec2805@columbia.edu>
2719 Date: Sat Oct 24 05:46:04 2015 +0000
2720
2721     Merge branch 'master' of https://github.com/DavidWatkins/JFlat
```

```
2722
2723 commit 1ca3359bdd54bca5247219fddcd6d7de94dcfaa8
2724 Author: Emily Chen <ec2805@columbia.edu>
2725 Date: Sat Oct 24 05:45:12 2015 +0000
2726
2727     renamed lexical elements LRM
2728
2729 commit 5c18e1c07073971ecd666ee761db3e757693d631
2730 Author: Emily Chen <ec2805@columbia.edu>
2731 Date: Sat Oct 24 05:44:06 2015 +0000
2732
2733     classes LRM
2734
2735 commit 388cada09bd0c3b9d43c052780c708b636c958ba
2736 Merge: 7489892 865accd
2737 Author: Khaled Atef <kaa2168@columbia.edu>
2738 Date: Sat Oct 24 00:00:54 2015 -0400
2739
2740     Merge branch 'master' of https://github.com/DavidWatkins/JFlat
2741
2742 commit 865accd230bbb198523fac3a74aadf14a0e157ff
2743 Author: Philip Schiffrin <philip.schiffrin@gmail.com>
2744 Date: Fri Oct 23 23:54:06 2015 -0400
2745
2746     updated LRM_Phil.txt with structure, scope, and arrays
2747
2748 commit 74898925d7d70a982dd70e3ff2f14b73c7d26f0e
2749 Author: Khaled Atef <kaa2168@columbia.edu>
2750 Date: Fri Oct 23 23:52:04 2015 -0400
2751
2752     Expressions/Operators portion of LRM
2753
2754 commit 3b92c848a16d0e7a8f9315d680c13bfbc5f17888
2755 Author: Emily Chen <ec2805@columbia.edu>
2756 Date: Sat Oct 24 01:36:09 2015 +0000
2757
2758     add 'this' keyword
2759
2760 commit 1aecbbdfefb15eee2c260c3703274b9dba43777b
2761 Merge: 3650409 42a62d0
2762 Author: Emily Chen <ec2805@columbia.edu>
2763 Date: Sat Oct 24 01:12:16 2015 +0000
2764
2765     Merge branch 'master' of https://github.com/DavidWatkins/JFlat
2766
2767 commit 3650409be97c9ed8cd94802464950aaced1993c5
2768 Author: Emily Chen <ec2805@columbia.edu>
2769 Date: Sat Oct 24 01:11:34 2015 +0000
2770
```

```
2771     lexical elements section for LRM
2772
2773     commit 42a62d0bf9d983cd3a0f15d4b8f1cb09bf41598e
2774     Author: David Watkins <djrival7@gmail.com>
2775     Date:   Fri Oct 23 19:14:43 2015 -0400
2776
2777     More stuff
2778
2779     commit 8e3398b458b45005129ab57c82ff6a13f9abafcb
2780     Author: Philip Schiffrin <philip.schiffrin@gmail.com>
2781     Date:   Fri Oct 23 16:54:05 2015 -0400
2782
2783     added lrm text for data types
2784
2785     commit fe46f41b37d368df2a0c5b552e7166f47c6b227d
2786     Author: David Watkins <DavidWatkins@users.noreply.github.com>
2787     Date:   Fri Oct 23 16:45:12 2015 -0400
2788
2789     Update README.md
2790
2791     commit ac7ee22f9b3ea22922761752306918e49ce148c8
2792     Merge: 8456847 8aa535a
2793     Author: David Watkins <djrival7@gmail.com>
2794     Date:   Fri Oct 23 16:40:21 2015 -0400
2795
2796     Merge branch 'master' of https://github.com/DavidWatkins/DiceLanguage
2797
2798     commit 84568474c4b9868e460decac0c99e730fecfc92d
2799     Author: David Watkins <djrival7@gmail.com>
2800     Date:   Fri Oct 23 16:39:44 2015 -0400
2801
2802     Added stuff
2803
2804     commit 8aa535ad4d802f7205c73c896b93b13dd8180239
2805     Author: David Watkins <davidw@tkins.me>
2806     Date:   Mon Oct 12 11:32:44 2015 -0400
2807
2808     Added isprime ll
2809
2810     commit 4812638376210accb89e6774c0442a918daa4b2f
2811     Author: David Watkins <davidw@tkins.me>
2812     Date:   Mon Oct 12 11:09:09 2015 -0400
2813
2814     Code working with llvm helloworld example, but does not return result
2815
2816     commit 4cc504c6d2d3732f2a96bc999ef2a28098a320a1
2817     Author: Emily Chen <ec2805@columbia.edu>
2818     Date:   Mon Oct 12 03:03:44 2015 -0400
2819
```

```
2820     remove obsolete TODO
2821
2822 commit 491935e26c6e27de4c14fb38f5d911eab53e0127
2823 Author: Emily Chen <ec2805@columbia.edu>
2824 Date:   Mon Oct 12 02:52:38 2015 -0400
2825
2826     host first sends the number of expected bytes it's sending
2827
2828 commit 546776b3750cfa4dfbe9bdb2388ce16ced7b83b4
2829 Author: Emily Chen <ec2805@columbia.edu>
2830 Date:   Mon Oct 12 02:19:24 2015 -0400
2831
2832     child proc successfully executes command and writes results to file
2833
2834 commit a129ab12519bfdc1d5805b31ce6a966eeaaef52b
2835 Author: Emily Chen <ec2805@columbia.edu>
2836 Date:   Mon Oct 12 01:45:24 2015 -0400
2837
2838     worker no longer blocks after host is done sending all file data
2839
2840 commit 3c7dd62f15c82bf9a8e75c4b9e3a8b5b006a8d60
2841 Author: David Watkins <davidw@tkins.me>
2842 Date:   Sun Oct 11 20:02:24 2015 -0400
2843
2844     Code now compiles and added test.py
2845
2846 commit 89b213ca24cacf788cf48f740a2beaecfa58bb50
2847 Merge: 682c38f bb4d7c3
2848 Author: David Watkins <djrival7@gmail.com>
2849 Date:   Sun Oct 11 19:25:53 2015 -0400
2850
2851     Merge branch 'master' of https://github.com/DavidWatkins/DiceLanguage
2852
2853 commit 682c38fb01a5ccd93c73c1e1e050d3763fcf5e03
2854 Author: David Watkins <djrival7@gmail.com>
2855 Date:   Sun Oct 11 19:25:30 2015 -0400
2856
2857     Added new code for worker and Makefile
2858
2859 commit bb4d7c386dd1f202183ca3a39fcc2ff184b28fb9
2860 Author: Philip Schiffrin <philip.schiffrin@gmail.com>
2861 Date:   Sun Oct 11 16:37:52 2015 -0400
2862
2863     added isprime for llvm test - llvm code generated by llvm from isprime.c
2864
2865 commit 6ad9e088cdb74598d64d6ce3d1c55d8064295342
2866 Author: David Watkins <djrival7@gmail.com>
2867 Date:   Fri Oct 9 23:18:24 2015 -0400
2868
```

```
2869     Added initial C code
2870
2871 commit 87f6d1a0d5c756bce028ee1e0622b51957665906
2872 Author: David Watkins <DavidWatkins@users.noreply.github.com>
2873 Date:   Fri Oct 9 23:17:27 2015 -0400
2874
2875     Initial commit
```

Software Development Environment

From the beginning of the project we agreed to the following development environment with the following software versions:

- **Ubuntu 15.10** - Very simple to use linux distribution that had the LLVM software and OCaml software easily accessible. Ubuntu was used within Virtualbox to ensure consistency across hardware as well.
- **LLVM-3.7** - The latest version of LLVM and allowed for easy code generation in OCaml using the LLVM module
- **OCaml Packages** - There were some features, such as JSON manipulation, that required additional OCaml packages. Therefore we included the following four OCaml packages in our development process: core, batteries, llvm, and yojson.
- **Slack** - We agreed that the Slack chat messaging platform was the most convenient and efficient way to share code snippets and communicate. It also brought up morale in the group in the form of emojis.
- **Github** - In order to version control our software and maintain a working version at any time, we used Github as our go to source code repository. It made integration with the team simpler and everyone was able to view the repository conveniently in their browser.
- **Latex** - In order to compile the documentation we made sure to all use Latex to ensure high quality material being produced for the project.
- **Vim/Sublime** - We could not create a consensus on which text editor to use, but in the end it did not matter to much which members used which.

Programming Style Guide

We adhered to the following style guide as much as possible:

- No lines greater than 80 characters
- Ensure that pattern matches are on the same indent with respect to each other
- Use tabbed indentation as opposed to spaces. Ensure that the tab width is 4 spaces.

5. ARCHITECTURE

The Compiler

To give a quick overview of our compiler, we have a total of 8 modules:

- **analyzer.ml** - Semantically checks incoming AST representation to make sure that it includes existing files, adheres to the rules of inheritance, and expressions are properly type-checked
- **codegen.ml** - Converts a semantically checked AST into a working LLVM code by producing LLVM IR
- **dice.ml** - Main module that calls on all the other modules depending on compiler flags passed to it
- **filepath.ml** - Uses system calls to determine the absolute path to any file in the system. Useful for uniquely checking if an include statement refers to the same files
- **parser.mly** - Reads in tokens from the scanner to produce an AST representation of the program
- **processor.ml** - Handles communication between scanner and parser so that error messages regarding invalid input can be handled better
- **scanner.mll** - Reads a source file and tokenizes it to the corresponding token output
- **utils.ml** - Contains several functions for printing out the string representation of various intermediate representations in our language. Most critically used for debugging

and we have 4 interfaces

- **ast.ml** - Representation of program after parser
- **conf.ml** - Contains paths for accessing standard library and bindings
- **exceptions.ml** - All exceptions in the compiler
- **sast.ml** - The semantically checked representation of the language

and we have 2 library files

- **bindings.c** - A c file containing critical functions written in c that are usable in the language. This is compiled to LLVM bitcode and then linked with all source files compiled in our language
- **stdlib.dice** - A file containing user defined classes written in dice that are usable by the user

The Scanner

The Scanner scans through the input file and tokenizes the input, discarding characters which are no longer need such as whitespace.

The Parser

The parser scans the tokens passed to it by the scanner and constructs an abstract syntax tree based on the definitions provided and the input tokens. The top level of the abstract syntax tree is a structure containing all classes and a structure containing all include statements. The Parser produces the following layout:

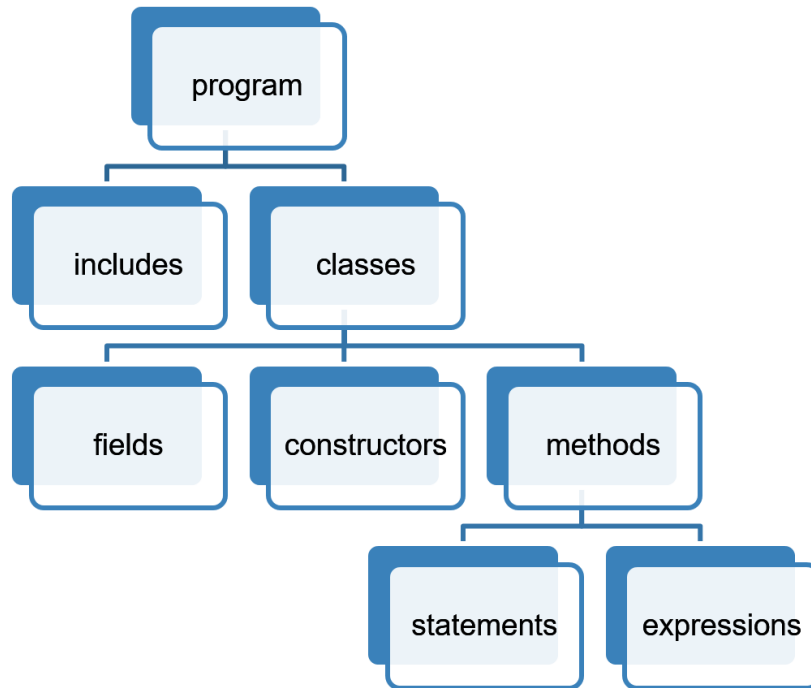


Figure 5.1: AST program representation.

The Semantic Analyzer

The first job of the Analyzer is to run the Scanner and Parser on any files contained in the includes statements of the given abstract syntax tree. The process of building an abstract syntax tree is the same for these files as for the originally compiled file. If any of these new abstract syntax trees contain include statements, the same process is run until there are no more includes. Similarly, each time a new included file's abstract syntax tree is passed to the Analyzer, all classes contained in the class structure of the new abstract syntax tree are appended to the original class list contained in the original class structure which was in the original abstract syntax tree. Once this process is complete, the analyzer is left with a class structure which contains every class defined in every file which was included with the originally compiled file.

Next, the Analyzer performs an inheritance analysis by looking through the class list contained in the class structure and performs an analysis to determine whether any classes are children or parents of other classes. If there are any such relationships, the fields of each parent class are added to the front of its child's fields list, and the methods of each parent class are added to the child's method's list. However, if the child has declared a method or field which shares the same name as the parent's field or method, the child's field or method is not overwritten by the parent. As the inheritance analysis is performed, the list of fields for each class is also assigned a integer key beginning with 0 which will serve as the key to a lookup table which, at runtime, contains pointers to every function for each class.

Once the inheritance analysis is performed, semantic analysis is performed on each statement and expression in each block of code in every method for every class. This semantic analysis consists of making sure that

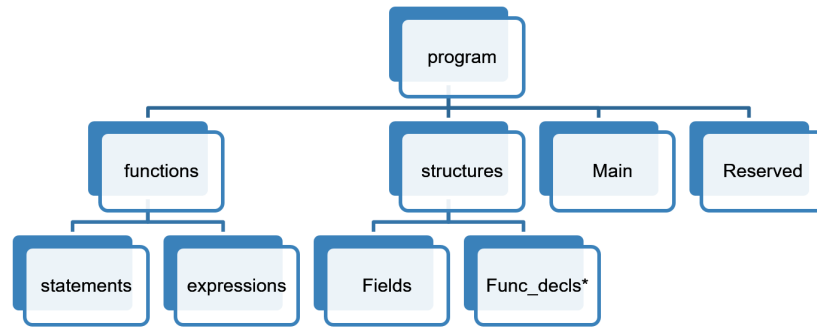


Figure 5.2: SAST representation.

types are consistent in every expression, making sure variables are declared and in the proper scope, and making sure that variables are only declared once. For instance, if an integer `x` is declared and `x` is assigned to the return of a method, the analyzer checks that the called method returns the type of `x`, namely an integer.

As this analysis is performed, the analyzer is simultaneously constructing a semantic abstract syntax tree. The purpose of this new data structure is to provide the code generator with data that is organized more similarly to the LLVM code that it will eventually produce. Thus, instead of classes containing methods and fields, the top level program structure now contains separate sections for methods and fields. This is useful for the code generator because the LLVM code that is produced uses structs to store the fields of a class and functions to store the code within a class's methods. Thus, there is no inherent connection between the functions and the structs in LLVM. However, the analyzer modifies each method so that an instance of the structure containing the fields of the given class is passed in as the first argument to every function for that class. In this way, functions can access each field of a given class by accessing the data inside of the structure.

The Code Generator

The code generator uses the semantic abstract syntax tree passed to it by the analyzer to construct the LLVM IR file which contains the final instructions for the program.

Structs and Inheritance

All structs are given an integer key at the beginning of their definition which will allow them to directly get their own virtual function table. Even if a subclass inherits from a parent class, it will be initialized with a specific key that is unique to the class at the beginning of each struct. For inherited fields they are organized in the order they were inherited, allowing multiple levels of inheritance. However it was too complex of a problem to solve multiple inheritance so we chose not to implement it.

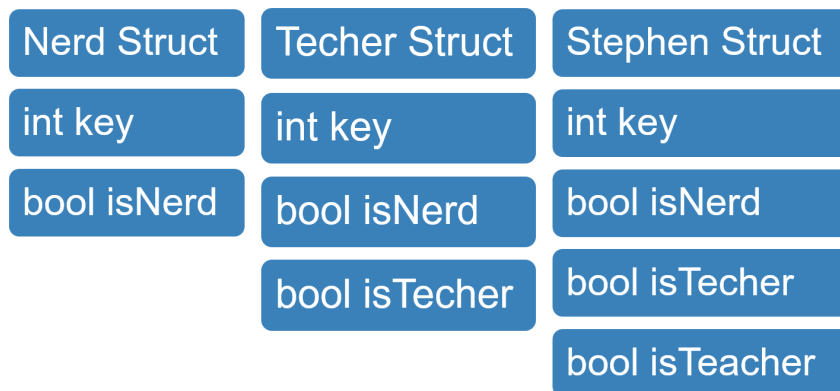


Figure 5.3: Structs example with inheritance.

The Virtual Function Table

At compile time, an intermediate representation of the virtual function table is produced in LLVM IR. It is a function defined as "lookup" that is able to lookup a classes virtual function array by its class index and a function index unique to that function. The function index is generated from the Func_decl list of a struct in the SAST. This way all subclasses have the same index for referring to the same function. Take for example

Class Indexes→	Nerd	Techer	Stephen
Function Indexes→	isNerd:Nerd	isNerd:Nerd	isNerd:Stephen
			isTeacher:Stephen

Figure 5.4: Virtual Function Table Example.

a class Nerd which has a subclass Techer, which itself has a subclass Stephen. Nerd has an isNerd method defined, Techer then inherits that method. Stephen would inherit that method but instead overrides them with its own implementation. But if a Nerd type variable is assigned to a Stephen type variable, the casted struct would still have the corresponding key to the Stephen class, and the function call would receive the correct index of 1 if isNerd were called.

Expressions and Bindings

Once the inheritance code is generated, the code generator iterates through the entire semantic abstract syntax tree and produces the necessary LLVM code for each function, statement, and expression. This code generation is done using the OCaml LLVM library, which uses OCaml functions to produce the desired LLVM code. We then link the resulting LLVM module with a precompiled bindings.bc which allows for the custom C functions we wrote to be incorporated into a user program in LLVM.

The Utilities

Using the `utils.ml` module we were able to pretty print, print to JSON for AST and SAST, and print out the tokens for any given program. This made debugging the semantic analyzer much easier as we were able to see what went into it and what it produces at any time. The following is an example of what the SAST looks like in JSON.

```
1 {
2   "sprogram": {
3     "classes": [
4       { "scdecl": { "scname": "test", "sfields": [], "sfuncs": [] } }
5     ],
6     "functions": [
7       {
8         "sfdecl": {
9           "sfname": "test.constructor",
10          "sreturnType": "class test",
11          "sformals": [],
12          "sbody": [
13            {
14              "slocal": {
15                "datatype": "class test",
16                "name": "this",
17                "val": {
18                  "call": {
19                    "name": "cast",
20                    "params": [
21                      {
22                        "call": {
23                          "name": "malloc",
24                          "params": [
25                            {
26                              "call": {
27                                "name": "sizeof",
28                                "params": [
29                                  {
30                                    "id": {
31                                      "name": "ignore",
32                                      "datatype": "class test"
33                                    }
34                                  ]
35                                },
36                                "index": 0,
37                                "datatype": "int"
38                              }
39                            ],
40                            "index": 0,
41                            "datatype": "char[]"
42                          }
43                        ]
44                      }
45                    ]
46                  }
47                }
48              }
49            ]
50          }
51        }
52      ]
53    }
54  }
55 }
```

```

44         }
45     ],
46     "index": 0,
47     "datatype": "class test"
48 }
49 }
50 }
51 },
52 {
53     "sexpr": {
54         "expr": {
55             "assign": {
56                 "lhs": {
57                     "objaccess": {
58                         "lhs": {
59                             "id": { "name":
↪ "this", "datatype": "class test" }
60
61                             },
62                             "op": ".",
63                             "rhs": {
64                                 "id": { "name":
↪ ".key", "datatype": "int" }
65
66                                 },
67                                 "datatype": "int"
68                             }
69                         },
70                         "op": "=",
71                         "rhs": { "int_lit": { "val": 0,
↪ "datatype": "int" } },
72                     }
73                 },
74                 "datatype": "int"
75             }
76         },
77         "sreturn": {
78             "return": {
79                 "id": { "name": "this", "datatype": "class test"
↪ }
80
81                 },
82                 "datatype": "class test"
83             }
84         },
85         "func_type": "user"
86     }
87 }
88 ],

```

```

89     "main": {
90         "sfdecl": {
91             "sfname": "main",
92             "sreturnType": "void",
93             "sformals": [
94                 { "name": "this", "datatype": "class test" },
95                 { "name": "args", "datatype": "char[] []" }
96             ],
97             "sbody": [
98                 {
99                     "slocal": {
100                         "datatype": "class test",
101                         "name": "this",
102                         "val": {
103                             "call": {
104                                 "name": "cast",
105                                 "params": [
106                                     {
107                                         "call": {
108                                             "name": "malloc",
109                                             "params": [
110                                                 {
111                                                     "call": {
112                                                         "name": "sizeof",
113                                                         "params": [
114                                                             {
115                                                                 "id": {
116                                                                     "name":
117                                                                     "datatype":
118                                                                 }
119                                                             ],
120                                                         "index": 0,
121                                                         "datatype": "int"
122                                                     }
123                                                 }
124                                             ],
125                                             "index": 0,
126                                             "datatype": "char[] "
127                                         }
128                                     }
129                                 ],
130                                 "index": 0,
131                                 "datatype": "class test"
132                             }
133                         }
134                     }
135                 }

```

```

136         },
137         {
138             "sexpr": {
139                 "expr": {
140                     "assign": {
141                         "lhs": {
142                             "objaccess": {
143                                 "lhs": {
144                                     "id": { "name": "this",
↪ "datatype": "class test" }
145                                 },
146                                 "op": ".",
147                                 "rhs": { "id": { "name": ".key",
↪ "datatype": "int" } },
148                                 "datatype": "int"
149                             }
150                         },
151                         "op": "=",
152                         "rhs": { "int_lit": { "val": 0, "datatype": "int" } },
153                         "datatype": "int"
154                     }
155                 },
156                 "datatype": "int"
157             }
158         },
159         {
160             "sexpr": {
161                 "expr": {
162                     "call": {
163                         "name": "print",
164                         "params": [
165                             {
166                                 "string_lit": {
167                                     "val": "Hello, World!",
168                                     "datatype": "char[]"
169                                 }
170                             }
171                         ],
172                         "index": 0,
173                         "datatype": "void"
174                     }
175                 },
176                 "datatype": "void"
177             }
178         },
179         ],
180         "func_type": "user"
181     }
182 },

```

```
183     "reserved": [  
184     {  
185         "sfdecl": {  
186             "sfname": "print",  
187             "sreturnType": "void",  
188             "sformals": [ { "Many": "Any" } ],  
189             "sbody": [],  
190             "func_type": "reserved"  
191         }  
192     },  
193     {  
194         "sfdecl": {  
195             "sfname": "malloc",  
196             "sreturnType": "char[]",  
197             "sformals": [ { "name": "size", "datatype": "int" } ],  
198             "sbody": [],  
199             "func_type": "reserved"  
200         }  
201     },  
202     {  
203         "sfdecl": {  
204             "sfname": "cast",  
205             "sreturnType": "Any",  
206             "sformals": [ { "name": "in", "datatype": "Any" } ],  
207             "sbody": [],  
208             "func_type": "reserved"  
209         }  
210     },  
211     {  
212         "sfdecl": {  
213             "sfname": "sizeof",  
214             "sreturnType": "int",  
215             "sformals": [ { "name": "in", "datatype": "Any" } ],  
216             "sbody": [],  
217             "func_type": "reserved"  
218         }  
219     },  
220     {  
221         "sfdecl": {  
222             "sfname": "open",  
223             "sreturnType": "int",  
224             "sformals": [  
225                 { "name": "path", "datatype": "char[]" },  
226                 { "name": "flags", "datatype": "int" }  
227             ],  
228             "sbody": [],  
229             "func_type": "reserved"  
230         }  
231     },
```



```
232     {
233         "sfdecl": {
234             "sfname": "close",
235             "sreturnType": "int",
236             "sformals": [ { "name": "fd", "datatype": "int" } ],
237             "sbody": [],
238             "func_type": "reserved"
239         }
240     },
241     {
242         "sfdecl": {
243             "sfname": "read",
244             "sreturnType": "int",
245             "sformals": [
246                 { "name": "fd", "datatype": "int" },
247                 { "name": "buf", "datatype": "char[]" },
248                 { "name": "nbyte", "datatype": "int" }
249             ],
250             "sbody": [],
251             "func_type": "reserved"
252         }
253     },
254     {
255         "sfdecl": {
256             "sfname": "write",
257             "sreturnType": "int",
258             "sformals": [
259                 { "name": "fd", "datatype": "int" },
260                 { "name": "buf", "datatype": "char[]" },
261                 { "name": "nbyte", "datatype": "int" }
262             ],
263             "sbody": [],
264             "func_type": "reserved"
265         }
266     },
267     {
268         "sfdecl": {
269             "sfname": "lseek",
270             "sreturnType": "int",
271             "sformals": [
272                 { "name": "fd", "datatype": "int" },
273                 { "name": "offset", "datatype": "int" },
274                 { "name": "whence", "datatype": "int" }
275             ],
276             "sbody": [],
277             "func_type": "reserved"
278         }
279     },
280     {
```

```
281         "sfdecl": {
282             "sfname": "exit",
283             "sreturnType": "void",
284             "sformals": [ { "name": "status", "datatype": "int" } ],
285             "sbody": [],
286             "func_type": "reserved"
287         }
288     },
289     {
290         "sfdecl": {
291             "sfname": "getchar",
292             "sreturnType": "int",
293             "sformals": [],
294             "sbody": [],
295             "func_type": "reserved"
296         }
297     },
298     {
299         "sfdecl": {
300             "sfname": "input",
301             "sreturnType": "char[]",
302             "sformals": [],
303             "sbody": [],
304             "func_type": "reserved"
305         }
306     }
307 ]
308 }
309 }
```

Supplementary Code

The Standard Library

The standard library was written in order to provide the user with a solid foundation on which to start writing interesting programs. To that end we provide for basic file i/o and string and integer manipulation.

String

Provide useful functionality for string manipulation.

Fields

String has no public fields. Private fields include a char array `my_string` which stores the given string and an int to store the length of the string.

Constructors

String(char[] a) Accepts a char array, such as a string literal or a char array. This string is copied into the `my_string` field of the object and the private `length()` method is run to get the length of the input string.

Methods

private int length_internal(char[] input) Returns the length of the given char array.

private char[] copy_internal(char[] input) Creates a new char array into which it copies the given char array.

public char[] string() Returns the char array contained in the my_string field.

public char getChar(int index) Returns the char contained at the given index in the my_string field.

public int length() Returns the length of the my_string field

public int toInteger() Converts the char array in the my_string field to an integer and returns that int. If the char array contained in the my_string field is not a string representation of an int, the behavior is undefined.

public int toDigit(char digit) Returns the integer corresponding to the character passed in.

public class String copy(class String input) Returns a copy of the current object.

public int indexOf(char input) Returns the index of the input character in the my_string field. Returns -1 if the character is not found in the field.

public class String reverse() Returns a string object with the my_string field containing the reverse of the current my_string char array.

public class String concat(class String temp) Returns a string object with the my_string field containing the concatenation of the current my_string field with the temp's my_string field.

public bool compare(class String input) Returns true if the my_string field of the input String is equal to the my_string field of the current String object.

public bool contains(class String check) Returns true if the my_string field of the input String is contained in the my_string field of the current String object.

public void free() Frees the memory for the my_string field of the current String object.

File

The File class constructor takes two arguments: a char[] that points to an already opened file on which the user wishes to operate and a boolean indicating whether the user wishes to open the file for writing. If the boolean is true the file is opened for reading and writing, and if false the file is opened as read only. The constructor stores the given path in a field and then calls open() on the given path and, if successful, sets the object's file descriptor field to the return of open(). If open() fails, the program exits with error.

Fields

File has no public fields. Private fields are the class String filePath, private bool isWriteEnabled, and the private int fd.

Constructors

File(char[] path, bool isWriteEnabled) Accepts a char array to open a file on, then creates a file object with the file descriptor. `isWriteEnabled` is a parameter that is used to determine whether the file can be written to or just read from.

Methods

private int openfile(class String path, bool isWriteEnabled) Returns the file descriptor of the opened file if successful, and -1 otherwise.

public char[] readfile(int num) Reads num bytes from the open file and returns the bytes in a char array.

public int writefile(char[] arr, int offset) Writes the contents of the char[] array to the file. If offset is -1 the write starts at the beginning of the file, if 0 it starts at the end of the file, and with any other positive integer it starts writing offset bytes from the beginning of the file.

public void closefile() Closes the open file. On error, the program exits with error.

Integer

The Integer class provides for integers to be converted to char arrays.

Fields

Integer has no public fields. There is one private field `my_int` which stores the given integer.

Constructors

Integer(int input) Accepts an integer which is stored in the field `my_int`.

Methods

public int num() Returns the integer stored in the `my_int` field.

public char toChar(int digit) Returns in teh input digit as a character.

public class String toString() Converts the integer stored in the `my_int` field into a string using the `toChar()` method. Returns a string object.

Built-in Functions

These are functions which are mapped from Dice to the C standard library, which is accessed through LLVM IR. The following function names may not be declared by the user since they are reserved. These are the only functions in dice which are not called as the method of an object; instead the user calls them directly with no dot operator.

int print(...)

The print function can take a char array, int, float and boolean. For char arrays, the contents of the array are printed to stdout. For every other type, the type is converted to the proper variable identifier as used in the C standard library printf function, and then the identifier is replaced with the value of the passed in type when the string is printed to standard out. Arguments can be in any order and must be comma separated.

char[] malloc(int size)

Returns a char pointer to an area of allocated memory on the heap of size bytes.

int open(char[] path)

Attempts to open the file located at the path specified and, if successful, returns a file descriptor to the open file. Returns -1 on failure.

int close(int fd)

Closes the open file identified by the integer fd. Returns 0 if successful and -1 on error.

int read(int fd, char[] buf, int num)

Reads num bytes from the open file identified by fd and stores the resulting string in the char array buf. If successful the number of bytes read is returned. Otherwise returns -1.

int write(int fd, char[] buf, int size)

Writes the contents of the char array buf, which contains size bytes, to the open file identified by fd. If successful the number of bytes written is returned. Otherwise returns -1.

int lseek(int fd, int offset, int whence)

The lseek() function repositions the offset of the open file associated with the file descriptor fd to the argument offset according to the directive whence as follows: 0 - the offset is set to offset bytes, 1 - The offset is set to its current location plus offset bytes, 2 - The offset is set to the size of the file plus offset bytes.

void exit(int flag)

Exits the program. Program exits without error if flag is 0 and exits with error if flag is set to any other integer.

int getchar()

Gets a character from stdin. Returns the character cast to an int.

Functions Implemented in C

With LLVM IR dice is able to compile functions written in C to LLVM. The following functions for dice were written in C.

Declarations

char[] input()

The input function reads from stdin with the C standard library `getchar()` function, storing each character in a malloc'd char array, until a newline character is read. The resulting array is returned.

long[] init_arr(int[] dims, int dimc)

Takes a list of dimensions in the form of ints and initialize a dimc-dimensional array in a one-dimension malloc call. To access element `arr[1][2]`, first dereference `a[1]`, and cast the value to a `long*`, which is an address to the array at position 1. Then dereference `arr[2]` and then cast that to a `long*` and the value is located at that position. This function is implemented in `bindings.c`, but was never incorporated directly into the language.

6. TEST PLAN

We embodied a "Test Driven Development" approach while creating our programming language. This process entailed writing tests for specific features of our language before starting to implement them. Every test should start by failing in an automated script and then the script should be executed after every modification to any portion of the compiler (from scanner to code generation). This way the team members would know if any modifications made resulted in other tests failing that had previously passed.

The majority of the test cases in our suite check the code generation through a comparison of print statement outputs from the code and our expected output. We created a test for every component of our language from basic variable declaration and assignment to class inheritance and method overriding. If it's in our language, there's a test case for it.

Testing Phases

Unit Testing

In the beginning of the testing process, we set out to thoroughly check the scanner and parser; however, the course instructor suggested we focus on the overall output of the project because testing end-to-end flow was his recommendation. To simplify checking of the Abstract Syntax Tree (AST) and the semantically checked AST (SAST), our manager created a pretty printer that would output the trees in a Javascript Object Notation (JSON) format for quick visual confirmation of their structure. In addition to quick visual feedback JSON objects provide, we also considered using an OCaml JSON visualization package known as `yojson` to render a visual tree of the data. We then compared the results of this output to the expected results based on the input.

Integration Testing

In addition to running the test suite routinely, we streamlined creation of new test cases by allowing any member of the team to create a git issue (labeled with "Testing") whenever a test case idea came to mind. Khaled (Test Suite Creator) would then screen all the open testing issues and add/modify the test according to schedule set by the manager.

During the development process, we also realized that in addition to checking proper output from our programs, we should also check if our analyzer was correctly identifying semantically invalid code. For example, if trying to assign a float type number to an integer variable (a feature we do not support), the analyzer should throw the proper exception. We accounted for these cases and placed all the tests in a separate folder with an identifying prefix to easily determine the category of test case.

Automation

Testing was very simple using `./tester.sh`. We can verify that a test works individually by running `lli` on the outputted `ll` file

Test Suites

We created a total of 121 tests divided into two categories. One checks that the compiler is properly recognizing invalid code. The other checks that the compiler accepts valid code and tests the output program.

7. LESSONS LEARNED

David

Most critically I learned that if you want to make something good, put as much effort as physically possible into it. I was told frequently "get started early" with respect to this project. After starting early I also learned that working often and with purpose helped not only myself get through the project but also the rest of my team.

As project manager the most critical decision I made was to gain consensus on the development environment that each team member was using. My main takeaway was to make sure that everyone agrees to use the same tools and systems. Having incompatible hardware/software can create unnecessary tension in what is already a stressful situation.

One final note is that I really did not know what to expect from OCaml coming into this class. It seemed very mysterious at first, but after looking through previous examples of compilers from other groups and writing out the Analyzer for my language, I quickly grew to enjoy the language. It certainly was not as daunting as it seemed at first.

Emily

If you're collaborating with someone to implement a feature where there are design decisions affecting different components of the compiler, then both of you should iterate on your respective parts simultaneously and communicate with each other. In other words, before your teammate has a chance to prototype their part, implement the bare minimum to test whether the overall design works. Also, OCaml turned out to be a good tool for writing a compiler (because of all the tree traversals we did for type-checking and implementing inheritance) so I think learning it was a good investment.

Khaled

Read the lessons learned from previous projects and prioritize (with your group) which of them you will implement. You will not be able to do them all, but if you can agree as a group on which mistakes you can avoid, you're already ahead. For our group, we determined that we will ACTUALLY start early, which we did.

Fortunately, we had a very organized and decisive manager that made sure we were all on track throughout the semester. Make sure you nominate a person with same qualities if you don't want to spend the last week of the semester pulling all-nighters for this project (save that for your other exams).

Track tasks with Github's issue tracking. Keep this issue tracker open during meetings with the Professor/TAs in order to avoid forgetting discussed to-do items. Ensure the manager of the group delegates through this system.

To spare your team members pain, don't use the diff command's output in your test script. Just label the program's output and your expected output and place them on top of each other for easy reading.

Philip

This project was a good overall lesson in how important it is to plan ahead when constructing a piece of software with a large, complex codebase. Our manager did a great job of making sure that we always had a plan of action when attacking each new problem, which was key in making sure the project came to fruition. Also, watch out for any rogue characters, especially 'h'.

8. CODE LISTING

Code

analyzer.ml

```
1  open Sast
2  open Ast
3  open Processor
4  open Utils
5  open Filepath
6  open Conf
7
8  module StringMap = Map.Make (String)
9
10 module StringSet = Set.Make (String)
11
12 let struct_indexes:(string, int) Hashtbl.t = Hashtbl.create 10
13 let predecessors:(string, string list) Hashtbl.t = Hashtbl.create 10
14
15 module SS = Set.Make(
16     struct
17         let compare = Pervasives.compare
18         type t = datatype
19     end )
20
21 type class_map = {
22     field_map      : Ast.field StringMap.t;
23     func_map       : Ast.func_decl StringMap.t;
24     constructor_map : Ast.func_decl StringMap.t;
25     reserved_map   : sfunc_decl StringMap.t;
26     cdecl          : Ast.class_decl;
27 }
28
29 type env = {
30     env_class_maps: class_map StringMap.t;
31     env_name      : string;
32     env_cmap      : class_map;
33     env_locals    : datatype StringMap.t;
34     env_parameters: Ast.formal StringMap.t;
35     env_returnType: datatype;
36     env_in_for    : bool;
37     env_in_while  : bool;
```

```

38     env_reserved : sfunc_decl list;
39 }
40
41 let update_env_name env env_name =
42 {
43     env_class_maps = env.env_class_maps;
44     env_name       = env_name;
45     env_cmap       = env.env_cmap;
46     env_locals     = env.env_locals;
47     env_parameters = env.env_parameters;
48     env_returnType = env.env_returnType;
49     env_in_for     = env.env_in_for;
50     env_in_while   = env.env_in_while;
51     env_reserved   = env.env_reserved;
52 }
53
54 let update_call_stack env in_for in_while =
55 {
56     env_class_maps = env.env_class_maps;
57     env_name       = env.env_name;
58     env_cmap       = env.env_cmap;
59     env_locals     = env.env_locals;
60     env_parameters = env.env_parameters;
61     env_returnType = env.env_returnType;
62     env_in_for     = in_for;
63     env_in_while   = in_while;
64     env_reserved   = env.env_reserved;
65 }
66
67 let append_code_to_constructor fbody cname ret_type =
68     let key = Hashtbl.find struct_indexes cname in
69     let init_this = [SLocal(
70         ret_type,
71         "this",
72         SCall(
73             "cast",
74             [SCall("malloc",
75                 [
76                     SCall("sizeof", [SId("ignore",
77 ↪ ret_type)], Datatype(Int_t), 0)
78                 ],
79                 Arraytype(Char_t, 1), 0)
80             ],
81             ret_type,
82             0
83         )
84     );
85     SExpr(
86         SAssign(
87             SObjAccess(

```

```

86         SId("this", ret_type),
87         SId(".key", Datatype(Int_t)),
88         Datatype(Int_t)
89     ),
90     SInt_Lit(key),
91     Datatype(Int_t)
92 ),
93     Datatype(Int_t)
94 )
95 ]
96 in
97 let ret_this =
98     [
99         SReturn(
100             SId("this", ret_type),
101             ret_type
102         )
103     ]
104 in
105     (* Need to check for duplicate default constructs *)
106     (* Also need to add malloc around other constructors *)
107     init_this @ fbody @ ret_this
108
109 let default_constructor_body cname =
110     let ret_type = Datatype(Objecttype(cname)) in
111     let fbody = [] in
112     append_code_to_constructor fbody cname ret_type
113
114 let default_sc cname =
115 {
116     sfname                = Ast.FName (cname ^ "." ^ "constructor");
117     sreturnType           = Datatype(Objecttype(cname));
118     sformals              = [];
119     sbody                 = default_constructor_body cname;
120     func_type             = Sast.User;
121     overrides             = false;
122     source                = "NA";
123 }
124
125 let default_c cname =
126 {
127     scope                = Ast.Public;
128     fname                = Ast.Constructor;
129     returnType           = Datatype(ConstructorType);
130     formals              = [];
131     body                 = [];
132     overrides            = false;
133     root_cname           = None;
134 }

```

```

135
136 let process_includes filename includes classes =
137     (* Bring in each include *)
138     let processInclude include_statement =
139         let file_in = open_in include_statement in
140         let lexbuf = Lexing.from_channel file_in in
141         let token_list = Processor.build_token_list lexbuf in
142         let program = Processor.parser include_statement token_list in
143         ignore(close_in file_in);
144         program
145     in
146     let rec iterate_includes classes m = function
147         [] -> classes
148         | (Include h) :: t ->
149             let h = if h = "stdlib" then Conf.stdlib_path else h in
150             (* Check each include against the map *)
151             let realpath = Filepath.realpath h in
152             if StringMap.mem realpath m then
153                 iterate_includes (classes) (m) (t)
154             else
155                 let result = processInclude realpath in
156                 match result with Program(i,c) ->
157                     iterate_includes (classes @ c) (StringMap.add realpath 1
158     ↪ m) (i @ t)
159     in
160     iterate_includes classes (StringMap.add (Filepath.realpath filename) 1
161     ↪ StringMap.empty) includes
162
163 let get_name cname fdecl =
164     (* We use '.' to separate types so llvm will recognize the function name and it
165     ↪ won't conflict *)
166     (* let params = List.fold_left (fun s -> (function Formal(t, _) -> s ^ "." ^
167     ↪ Utils.string_of_datatype t | _ -> "")) "" fdecl.formals in *)
168     let name = Utils.string_of_fname fdecl.fname in
169     if name = "main"
170     then "main"
171     else cname ^ "." ^ name(* ^ params *)
172
173 let get_constructor_name cname fdecl =
174     let params = List.fold_left (fun s -> (function Formal(t, _) -> s ^ "." ^
175     ↪ Utils.string_of_datatype t | _ -> "")) "" fdecl.formals in
176     let name = Utils.string_of_fname fdecl.fname in
177     cname ^ "." ^ name ^ params
178
179 let get_name_without_class fdecl =
180     (* We use '.' to separate types so llvm will recognize the function name and it
181     ↪ won't conflict *)
182     let params = List.fold_left (fun s -> (function Formal(t, _) -> s ^ "." ^
183     ↪ Utils.string_of_datatype t | _ -> "")) "" fdecl.formals in

```

```

177     let name = Utils.string_of_fname fdecl.fname in
178     let ret_type = Utils.string_of_datatype fdecl.returnType in
179     ret_type ^ "." ^ name ^ "." ^ params
180
181     (* Generate list of all classes to be used for semantic checking *)
182     let build_class_maps reserved cdecls =
183         let reserved_map = List.fold_left (fun m f -> StringMap.add
184 ↪ (Utils.string_of_fname f.sfname) f m) StringMap.empty reserved in
185         let helper m (cdecl:Ast.class_decl) =
186             let fieldfun = (fun m -> (function Field(s, d, n) -> if (StringMap.mem
187 ↪ (n) m) then raise(Exceptions.DuplicateField) else (StringMap.add n (Field(s, d, n))
188 ↪ m))) in
189             let funcname = get_name cdecl.cname in
190             let funcfun m fdecl =
191                 if (StringMap.mem (funcname fdecl) m)
192                 then raise(Exceptions.DuplicateFunction(funcname fdecl))
193                 else if (StringMap.mem (Utils.string_of_fname fdecl.fname)
194 ↪ reserved_map)
195                 then
196 ↪ raise(Exceptions.CannotUseReservedFuncName(Utils.string_of_fname fdecl.fname))
197                 else (StringMap.add (funcname fdecl) fdecl m)
198             in
199             let constructor_name = get_constructor_name cdecl.cname in
200             let constructorfun m fdecl =
201                 if fdecl.formals = [] then m
202                 else if StringMap.mem (constructor_name fdecl) m
203                 then raise(Exceptions.DuplicateConstructor)
204                 else (StringMap.add (constructor_name fdecl) fdecl m)
205             in
206             let default_c = default_c cdecl.cname in
207             let constructor_map = StringMap.add (get_constructor_name cdecl.cname
208 ↪ default_c) default_c StringMap.empty in
209             (if (StringMap.mem cdecl.cname m) then raise
210 ↪ (Exceptions.DuplicateClassName(cdecl.cname)) else
211                 StringMap.add cdecl.cname
212                 {
213                     field_map = List.fold_left fieldfun StringMap.empty
214 ↪ cdecl.cbody.fields;
215                     func_map = List.fold_left funcfun StringMap.empty
216 ↪ cdecl.cbody.methods;
217                     constructor_map = List.fold_left constructorfun
218 ↪ constructor_map cdecl.cbody.constructors;
219                     reserved_map = reserved_map;
220                     cdecl = cdecl }
221                 m) in
222         List.fold_left helper StringMap.empty cdecls
223
224     let rec get_all_descendants cname accum =
225         if Hashtbl.mem predecessors cname then
226             let direct_descendants = Hashtbl.find predecessors cname in

```

```

216         let add_childs_descendants desc_set direct_descendant =
217             get_all_descendants direct_descendant (StringSet.add direct_descendant
↳ desc_set)
218         in
219         List.fold_left add_childs_descendants accum direct_descendants
220         else accum
221
222 let inherited potential_predec potential_child =
223     match potential_predec, potential_child with
224     | Datatype(Objecttype(predec_cname)), Datatype(Objecttype(child_cname)) ->
225         let descendants = get_all_descendants predec_cname StringSet.empty in
226         if (predec_cname = child_cname) || (StringSet.mem child_cname descendants) then
↳ true
227         else raise (Exceptions.LocalAssignTypeMismatch(predec_cname, child_cname))
228     | _ , _ -> false
229
230 let get_equality_binop_type type1 type2 se1 se2 op =
231     (* Equality op not supported for float operands. The correct way to test floats
232         for equality is to check the difference between the operands in question *)
233     if (type1 = Datatype(Float_t) || type2 = Datatype(Float_t)) then raise
↳ (Exceptions.InvalidBinopExpression "Equality operation is not supported for Float
↳ types")
234     else
235         match type1, type2 with
236         | Datatype(Char_t), Datatype(Int_t)
237         | Datatype(Int_t), Datatype(Char_t)
238         | Datatype(Objecttype(_), Datatype(Null_t))
239         | Datatype(Null_t), Datatype(Objecttype(_))
240         | Datatype(Null_t), Arraytype(_, _)
241         | Arraytype(_, _), Datatype(Null_t) -> SBinop(se1, op, se2,
↳ Datatype(Bool_t))
242     | _ ->
243         if type1 = type2 then SBinop(se1, op, se2, Datatype(Bool_t))
244         else raise (Exceptions.InvalidBinopExpression "Equality operator can't
↳ operate on different types, with the exception of Int_t and Char_t")
245
246 let get_logical_binop_type se1 se2 op = function
247     (Datatype(Bool_t), Datatype(Bool_t)) -> SBinop(se1, op, se2, Datatype(Bool_t))
248     | _ -> raise (Exceptions.InvalidBinopExpression "Logical operators only operate
↳ on Bool_t types")
249
250 let get_comparison_binop_type type1 type2 se1 se2 op =
251     let numerics = SS.of_list [Datatype(Int_t); Datatype(Char_t); Datatype(Float_t)]
252     in
253         if SS.mem type1 numerics && SS.mem type2 numerics
254         then SBinop(se1, op, se2, Datatype(Bool_t))
255         else raise (Exceptions.InvalidBinopExpression "Comparison operators
↳ operate on numeric types only")
256

```



```

257
258 let get_arithmetic_binop_type se1 se2 op = function
259     (Datatype(Int_t), Datatype(Float_t))
260     | (Datatype(Float_t), Datatype(Int_t))
261     | (Datatype(Float_t), Datatype(Float_t))      -> SBinop(se1,
↪ op, se2, Datatype(Float_t))
262
263     | (Datatype(Int_t), Datatype(Char_t))
264     | (Datatype(Char_t), Datatype(Int_t))
265     | (Datatype(Char_t), Datatype(Char_t))      -> SBinop(se1, op,
↪ se2, Datatype(Char_t))
266
267     | (Datatype(Int_t), Datatype(Int_t))      ->
↪ SBinop(se1, op, se2, Datatype(Int_t))
268
269     | _ -> raise (Exceptions.InvalidBinopExpression "Arithmetic operators
↪ don't support these types")
270
271 let rec get_ID_type env s =
272     try StringMap.find s env.env_locals
273     with | Not_found ->
274         try let formal = StringMap.find s env.env_parameters in
275             (function Formal(t, _) -> t | Many t -> t ) formal
276         with | Not_found -> raise (Exceptions.UndefinedID s)
277
278 and check_array_primitive env el =
279     let rec iter t sel = function
280         [] -> sel, t
281         | e :: el ->
282             let se, _ = expr_to_sexpr env e in
283             let se_t = get_type_from_sexpr se in
284             if t = se_t
285             then iter t (se :: sel) el
286             else
287                 let t1 = Utils.string_of_datatype t in
288                 let t2 = Utils.string_of_datatype se_t in
289                 raise (Exceptions.InvalidArrayPrimitiveConsecutiveTypes(t1,
↪ t2))
290
291     in
292     let se, _ = expr_to_sexpr env (List.hd el) in
293     let el = List.tl el in
294     let se_t = get_type_from_sexpr se in
295     let sel, t = iter se_t ([se]) el in
296     let se_t = match t with
297         Datatype(x) -> Arraytype(x, 1)
298         | Arraytype(x, n) -> Arraytype(x, n+1)
299         | _ as t ->
↪ raise (Exceptions.InvalidArrayPrimitiveType(Utils.string_of_datatype t))
    in

```

```

300     SArrayPrimitive(sel, se_t)
301
302 and check_array_init env d el =
303     (* Get dimension size for the array being created *)
304     let array_complexity = List.length el in
305     let check_elem_type e =
306         let sexpr, _ = expr_to_sexpr env e in
307         let sexpr_type = get_type_from_sexpr sexpr in
308         if sexpr_type = Datatype(Int_t)
309             then sexpr
310             else raise(Exceptions.MustPassIntegerTypeToArrayCreate)
311     in
312     let convert_d_to_arraytype = function
313         Datatype(x) -> Arraytype(x, array_complexity)
314     |
315         _ as t ->
316         let error_msg = Utils.string_of_datatype t in
317         raise (Exceptions.ArrayInitTypeInvalid(error_msg))
318     in
319     let sexpr_type = convert_d_to_arraytype d in
320     let sel = List.map check_elem_type el in
321     SArrayCreate(d, sel, sexpr_type)
322
323 and check_array_access env e el =
324     (* Get dimensions of array, ex: foo[10][4][2] is dimen=3 *)
325     let array_dimensions = List.length el in
326     (* Check every e in el is of type Datatype(Int_t). Ensure all indices are ints *)
327     let check_elem_type arg =
328         let sexpr, _ = expr_to_sexpr env arg in
329         let sexpr_type = get_type_from_sexpr sexpr in
330         if sexpr_type = Datatype(Int_t)
331             then sexpr
332             else raise(Exceptions.MustPassIntegerTypeToArrayAccess)
333     in
334     (* converting e to se also checks if the array id has been declared *)
335     let se, _ = expr_to_sexpr env e in
336     let se_type = get_type_from_sexpr se in
337
338     (* Check that e has enough dimens as e's in el. Return overall datatype of
339     ↪ access*)
340     let check_array_dim_vs_params num_params = function
341         Arraytype(t, n) ->
342             if num_params < n then
343                 Arraytype(t, (n-num_params))
344             else if num_params = n then
345                 Datatype(t)
346             else
347                 raise
348     ↪ (Exceptions.ArrayAccessInvalidParamLength(string_of_int num_params, string_of_int n))
349     |
350         _ as t ->

```

```

347         let error_msg = Utils.string_of_datatype t in
348         raise (Exceptions.ArrayAccessExpressionNotArray(error_msg))
349     in
350     let sexpr_type = check_array_dim_vs_params array_dimensions se_type in
351     let sel = List.map check_elem_type el in
352
353     SArrayAccess(se, sel, sexpr_type)
354
355 and check_obj_access env lhs rhs =
356     let check_lhs = function
357         This                                     -> SId("this",
↪ Datatype(Objecttype(env.env_name)))
358         | Id s                                     -> SId(s, get_ID_type env s)
359         | ArrayAccess(e, el)                     -> check_array_access env e el
360         | _ as e                                   -> raise (Exceptions.LHSofRootAccessMustBeIDorFunc
↪ (Utils.string_of_expr e))
361     in
362     let ptype_name parent_type = match parent_type with
363         Datatype(Objecttype(name))               -> name
364         | _ as d                                   -> raise
↪ (Exceptions.ObjAccessMustHaveObjectType (Utils.string_of_datatype d))
365     in
366     let rec check_rhs (env) parent_type (top_level_env) =
367         let pt_name = ptype_name parent_type in
368         let get_id_type_from_object env (id) cname tlenv =
369             let cmap = StringMap.find cname env.env_class_maps in
370             let match_field f = match f with
371                 Field(scope, d, n) ->
372                     (* Have to update this with all parent classes
↪ checks *)
373                     if scope = Ast.Private && tlenv.env_name <>
↪ env.env_name then
374                         raise(Exceptions.CannotAccessPrivateFieldInNonProperScope
↪ env.env_name, tlenv.env_name))
375                     else d
376             in
377             try match_field (StringMap.find id cmap.field_map)
378             with | Not_found -> raise
↪ (Exceptions.UnknownIdentifierForClass(id, cname))
379         in
380         function
381             (* Check fields in parent *)
382             Id s                                     -> SId(s,
↪ (get_id_type_from_object env s pt_name top_level_env)), env
383             (* Check functions in parent *)
384             | Call(fname, el)                       ->
385                 let env = update_env_name env pt_name in
386                 check_call_type top_level_env true env fname el, env
387             (* Set parent, check if base is field *)

```

```

388         | ObjAccess(e1, e2) ->
389             let old_env = env in
390             let lhs, env = check_rhs env parent_type top_level_env e1
↪ in
391             let lhs_type = get_type_from_sexpr lhs in
392
393             let pt_name = ptype_name lhs_type in
394             let lhs_env = update_env_name env pt_name in
395
396             let rhs, env = check_rhs lhs_env lhs_type top_level_env
↪ e2 in
397             let rhs_type = get_type_from_sexpr rhs in
398             SObjAccess(lhs, rhs, rhs_type), old_env
399         | _ as e -> raise
↪ (Exceptions.InvalidAccessLHS (Utils.string_of_expr e))
400     in
401     let arr_lhs, _ = expr_to_sexpr env lhs in
402     let arr_lhs_type = get_type_from_sexpr arr_lhs in
403     match arr_lhs_type with
404     | Arraytype(Char_t, 1) -> raise (Exceptions.CannotAccessLengthOfCharArray)
405     | Arraytype(_, _) ->
406         let rhs = match rhs with
407             | Id("length") -> SId("length", Datatype(Int_t))
408             | _ -> raise (Exceptions.CanOnlyAccessLengthOfArray)
409         in
410         SObjAccess(arr_lhs, rhs, Datatype(Int_t))
411     | _ ->
412         let lhs = check_lhs lhs in
413         let lhs_type = get_type_from_sexpr lhs in
414
415         let ptype_name = ptype_name lhs_type in
416         let lhs_env = update_env_name env ptype_name in
417
418         let rhs, _ = check_rhs lhs_env lhs_type env rhs in
419         let rhs_type = get_type_from_sexpr rhs in
420         SObjAccess(lhs, rhs, rhs_type)
421
422 and check_call_type top_level_env isObjAccess env fname e1 =
423     let sel, env = expr1_to_sexpr1 env e1 in
424     (* check that 'env.env_name' is in the list of defined classes *)
425     let cmap =
426         try StringMap.find env.env_name env.env_class_maps
427         with | Not_found -> raise (Exceptions.UndefinedClass env.env_name)
428     in
429
430     let handle_param formal param =
431         let fty = match formal with Formal(d, _) -> d | _ -> Datatype(Void_t) in
432         let pty = get_type_from_sexpr param in
433         match fty, pty with

```

```

434         Datatype(Objecttype(f)), Datatype(Objecttype(p)) ->
435             if f <> p then
436                 try let descendants = Hashtbl.find predecessors f in
437                     let _ = try List.find (fun d -> p = d) descendants
438                         with | Not_found ->
439 ↪ raise(Exceptions.CannotPassNonInheritedClassesInPlaceOfOthers(f, p))
440                     in
441                         let rt = Datatype(Objecttype(f)) in
442                             SCall("cast", [param; SId("ignore", rt)], rt, 0)
443                             with | Not_found ->
444 ↪ raise(Exceptions.ClassIsNotExtendedBy(f, p))
445                     else param
446             | _ -> if fty = pty then param else
447 ↪ raise(Exceptions.IncorrectTypePassedToFunction(fname, Utils.string_of_datatype pty))
448         in
449             let index fdecl fname =
450                 let cdecl = cmap.cdecl in
451                 (* Have to update this with all parent classes checks *)
452                 let _ =
453                     if fdecl.scope = Ast.Private && top_level_env.env_name <>
454 ↪ env.env_name then
455                     raise(Exceptions.CannotAccessPrivateFunctionInNonProperScope(get_name
456 ↪ env.env_name fdecl, env.env_name, top_level_env.env_name))
457                 in
458                 (* Not exactly sure why there needs to be a list.rev *)
459                 let fns = List.rev cdecl.cbody.methods in
460                 let rec find x lst =
461                     match lst with
462                     | [] -> raise (Failure ("Could not find " ^ fname))
463                     | fdecl :: t ->
464                         let search_name = (get_name env.env_name fdecl) in
465                         if x = search_name then 0
466                         else if search_name = "main" then find x t
467                         else 1 + find x t
468                 in
469                 find fname fns
470         in
471             let handle_params (formals) params =
472                 match formals, params with
473                 | [Many(Any)], _ -> params
474                 | [], [] -> []
475                 | [], _
476                 | _, [] -> raise(Exceptions.IncorrectTypePassedToFunction(fname,
477 ↪ Utils.string_of_datatype (Datatype(Void_t))))
478                 | _ ->
479                     let len1 = List.length formals in
480                     let len2 = List.length params in

```

```

477         if len1 <> len2 then raise(Exceptions.IncorrectNumberOfArguments(fname,
↪ len1, len2))
478         else
479             List.map2 handle_param formals sel
480         in
481
482         let sfname = env.env_name ^ "." ^ fname in
483         try let func = StringMap.find fname cmap.reserved_map in
484             let actuals = handle_params func.sformals sel in
485             SCall(fname, actuals, func.sreturnType, 0)
486         with | Not_found ->
487         try let f = StringMap.find sfname cmap.func_map in
488             let actuals = handle_params f.formals sel in
489             let index = index f sfname in
490             SCall(sfname, actuals, f.returnType, index)
491         with | Not_found -> raise(Exceptions.FunctionNotFound(env.env_name, sfname)) | _
↪ as ex -> raise ex
492
493 and check_object_constructor env s e1 =
494     let sel, env = expr1_to_sexpr1 env e1 in
495     (* check that 'env.env_name' is in the list of defined classes *)
496     let cmap =
497         try StringMap.find s env.env_class_maps
498         with | Not_found -> raise (Exceptions.UndefinedClass s)
499     in
500     (* get a list of the types of the actuals to match against defined function
↪ formals *)
501     let params = List.fold_left (fun s e -> s ^ "." ^ (Utils.string_of_datatype
↪ (get_type_from_sexpr e))) "" sel in
502     let constructor_name = s ^ "." ^ "constructor" ^ params in
503     let _ =
504         try StringMap.find constructor_name cmap.constructor_map
505         with | Not_found -> raise (Exceptions.ConstructorNotFound
↪ constructor_name)
506     in
507     let ftype = Datatype(Objecttype(s)) in
508     (* Add a reference to the class in front of the function call *)
509     (* Must properly handle the case where this is a reserved function *)
510     SObjectCreate(constructor_name, sel, ftype)
511
512 and check_assign env e1 e2 =
513     let se1, env = expr_to_sexpr env e1 in
514     let se2, env = expr_to_sexpr env e2 in
515     let type1 = get_type_from_sexpr se1 in
516     let type2 = get_type_from_sexpr se2 in
517     match (type1, se2) with
518         Datatype(Objecttype(_)), SNull
519     |      Arraytype(_, _), SNull -> SAssign(se1, se2, type1)
520     |      _ ->

```

```

521     match type1, type2 with
522     | Datatype(Char_t), Datatype(Int_t)
523     | Datatype(Int_t), Datatype(Char_t) -> SAssign(se1, se2, type1)
524     | Datatype(Objecttype(d)), Datatype(Objecttype(t)) ->
525       if d = t then SAssign(se1, se2, type1)
526       else if inherited type1 type2 then
527         SAssign(se1, SCall("cast", [se2; SId("ignore", type1)], type1,
↪ 0), type1)
528       else raise (Exceptions.AssignmentTypeMismatch(Utills.string_of_datatype
↪ type1, Utills.string_of_datatype type2))
529     | _ ->
530       if type1 = type2
531       then SAssign(se1, se2, type1)
532       else raise (Exceptions.AssignmentTypeMismatch(Utills.string_of_datatype
↪ type1, Utills.string_of_datatype type2))
533
534 and check_unop env op e =
535   let check_num_unop t = function
536     Sub -> t
537     | _ -> raise (Exceptions.InvalidUnaryOperation)
538   in
539   let check_bool_unop = function
540     Not -> Datatype(Bool_t)
541     | _ -> raise (Exceptions.InvalidUnaryOperation)
542   in
543   let se, env = expr_to_sexpr env e in
544   let t = get_type_from_sexpr se in
545   match t with
546   | Datatype(Int_t)
547   | Datatype(Float_t) -> SUnop(op, se, check_num_unop t op)
548   | Datatype(Bool_t) -> SUnop(op, se, check_bool_unop op)
549   | _ -> raise (Exceptions.InvalidUnaryOperation)
550
551 and check_binop env e1 op e2 =
552   let se1, env = expr_to_sexpr env e1 in
553   let se2, env = expr_to_sexpr env e2 in
554   let type1 = get_type_from_sexpr se1 in
555   let type2 = get_type_from_sexpr se2 in
556   match op with
557   | Equal | Neq -> get_equality_binop_type type1 type2 se1 se2 op
558   | And | Or -> get_logical_binop_type se1 se2 op (type1, type2)
559   | Less | Leq | Greater | Geq -> get_comparison_binop_type type1 type2 se1 se2 op
560   | Add | Mult | Sub | Div | Mod -> get_arithmetic_binop_type se1 se2 op (type1,
↪ type2)
561   | _ -> raise (Exceptions.InvalidBinopExpression ((Utills.string_of_op op) ^ " is
↪ not a supported binary op"))
562
563 and check_delete env e =
564   let se, _ = expr_to_sexpr env e in

```

```

565     let t = get_type_from_sexpr se in
566     match t with
567         Arraytype(_, _) | Datatype(Objecttype(_)) -> SDelete(se)
568     | _ -> raise(Exceptions.CanOnlyDeleteObjectsOrArrays)
569
570 and expr_to_sexpr env = function
571     Int_Lit i          -> SInt_Lit(i), env
572   | Boolean_Lit b      -> SBoolean_Lit(b), env
573   | Float_Lit f        -> SFloat_Lit(f), env
574   | String_Lit s       -> SString_Lit(s), env
575   | Char_Lit c         -> SChar_Lit(c), env
576   | This              -> SId("this", Datatype(Objecttype(env.env_name))), env
577   | Id s              -> SId(s, get_ID_type env s), env
578   | Null              -> SNull, env
579   | Noexpr            -> SNoexpr, env
580
581   | ObjAccess(e1, e2)  -> check_obj_access env e1 e2, env
582   | ObjectCreate(s, e1) -> check_object_constructor env s e1, env
583   | Call(s, e1)        -> check_call_type env false env s e1, env
584
585   | ArrayCreate(d, e1) -> check_array_init env d e1, env
586   | ArrayAccess(e, e1) -> check_array_access env e e1, env
587   | ArrayPrimitive e1  -> check_array_primitive env e1, env
588
589   | Assign(e1, e2)     -> check_assign env e1 e2, env
590   | Unop(op, e)        -> check_unop env op e, env
591   | Binop(e1, op, e2)  -> check_binop env e1 op e2, env
592   | Delete(e)          -> check_delete env e, env
593
594
595 and get_type_from_sexpr = function
596     SInt_Lit(_)          -> Datatype(Int_t)
597   | SBoolean_Lit(_)      -> Datatype(Bool_t)
598   | SFloat_Lit(_)        -> Datatype(Float_t)
599   | SString_Lit(_)       -> Arraytype(Char_t, 1)
600   | SChar_Lit(_)         -> Datatype(Char_t)
601   | SId(_, d)            -> d
602   | SBinop(_, _, _, d)   -> d
603   | SAssign(_, _, d)     -> d
604   | SNoexpr              -> Datatype(Void_t)
605   | SArrayCreate(_, _, d) -> d
606   | SArrayAccess(_, _, d) -> d
607   | SObjAccess(_, _, d)  -> d
608   | SCall(_, _, d, _)    -> d
609   | SObjectCreate(_, _, d) -> d
610   | SArrayPrimitive(_, d) -> d
611   | SUnop(_, _, d)       -> d
612   | SNull                -> Datatype(Null_t)
613   | SDelete _            -> Datatype(Void_t)

```



```

614
615 and expr1_to_sexpr1 env e1 =
616   let env_ref = ref(env) in
617   let rec helper = function
618     head::tail ->
619       let a_head, env = expr_to_sexpr !env_ref head in
620       env_ref := env;
621       a_head::(helper tail)
622   | [] -> []
623   in (helper e1), !env_ref
624
625 let rec local_handler d s e env =
626   if StringMap.mem s env.env_locals
627   then raise (Exceptions.DuplicateLocal s)
628   else
629     let se, env = expr_to_sexpr env e in
630     let t = get_type_from_sexpr se in
631     if t = Datatype(Void_t) || t = Datatype(Null_t) || t = d ||
↳ (inherited d t)
632     then
633       let new_env = {
634         env_class_maps = env.env_class_maps;
635         env_name = env.env_name;
636         env_cmap = env.env_cmap;
637         env_locals = StringMap.add s d env.env_locals;
638         env_parameters = env.env_parameters;
639         env_returnType = env.env_returnType;
640         env_in_for = env.env_in_for;
641         env_in_while = env.env_in_while;
642         env_reserved = env.env_reserved;
643       } in
644       (* if the user-defined type being declared is not in global classes map, it is an
↳ undefined class *)
645       (match d with
646         Datatype(Objecttype(x)) ->
647           (if not (StringMap.mem
↳ (Utils.string_of_object d) env.env_class_maps)
648           then raise
↳ (Exceptions.UndefinedClass (Utils.string_of_object d))
649           else
650             let local = if inherited d t then SLocal(t, s, se) else
↳ SLocal(d, s, se)
651             in local, new_env)
652         | _ -> SLocal(d, s, se), new_env)
653       else
654         (let type1 = (Utils.string_of_datatype t) in
655          let type2 = (Utils.string_of_datatype d) in
656          let ex = Exceptions.LocalAssignTypeMismatch(type1, type2)
↳ in

```

```

657         raise ex)
658
659 let rec check_sblock s1 env = match s1 with
660     [] -> SBlock([SExpr(SNoexpr, Datatype(Void_t))]), env
661   | _ ->
662     let s1, _ = convert_stmt_list_to_sstmt_list env s1 in
663     SBlock(s1), env
664
665 and check_expr_stmt e env =
666     let se, env = expr_to_sexpr env e in
667     let t = get_type_from_sexpr se in
668     SExpr(se, t), env
669
670 and check_return e env =
671     let se, _ = expr_to_sexpr env e in
672     let t = get_type_from_sexpr se in
673     match t, env.env_returnType with
674     | Datatype(Null_t), Datatype(Objecttype(_))
675     | Datatype(Null_t), Arraytype(_, _) -> SReturn(se, t), env
676   | _ ->
677     if t = env.env_returnType
678     then SReturn(se, t), env
679     else raise (Exceptions.ReturnTypeMismatch(Utils.string_of_datatype t,
680 ↪ Utils.string_of_datatype env.env_returnType))
681
682 and check_if e s1 s2 env =
683     let se, _ = expr_to_sexpr env e in
684     let t = get_type_from_sexpr se in
685     let ifbody, _ = parse_stmt env s1 in
686     let elsebody, _ = parse_stmt env s2 in
687     if t = Datatype(Bool_t)
688     then SIf(se, ifbody, elsebody), env
689     else raise Exceptions.InvalidIfStatementType
690
691 and check_for e1 e2 e3 s env =
692     let old_val = env.env_in_for in
693     let env = update_call_stack env true env.env_in_while in
694
695     let se1, _ = expr_to_sexpr env e1 in
696     let se2, _ = expr_to_sexpr env e2 in
697     let se3, _ = expr_to_sexpr env e3 in
698     let forbody, _ = parse_stmt env s in
699     let conditional = get_type_from_sexpr se2 in
700     let sfor =
701         if (conditional = Datatype(Bool_t) || conditional = Datatype(Void_t))
702         then SFor(se1, se2, se3, forbody)
703         else raise Exceptions.InvalidForStatementType
704     in

```

```

705     let env = update_call_stack env old_val env.env_in_while in
706     sfor, env
707
708 and check_while e s env =
709     let old_val = env.env_in_while in
710     let env = update_call_stack env env.env_in_for true in
711
712     let se, _ = expr_to_sexpr env e in
713     let t = get_type_from_sexpr se in
714     let sstmt, _ = parse_stmt env s in
715     let swhile =
716         if (t = Datatype(Bool_t) || t = Datatype(Void_t))
717             then SWhile(se, sstmt)
718             else raise Exceptions.InvalidWhileStatementType
719     in
720
721     let env = update_call_stack env env.env_in_for old_val in
722     swhile, env
723
724 and check_break env =
725     if env.env_in_for || env.env_in_while then
726         SBreak, env
727     else
728         raise Exceptions.CannotCallBreakOutsideOfLoop
729
730 and check_continue env =
731     if env.env_in_for || env.env_in_while then
732         SContinue, env
733     else
734         raise Exceptions.CannotCallContinueOutsideOfLoop
735
736 and parse_stmt env = function
737     Block s1                -> check_sblock s1 env
738     | Expr e                 -> check_expr_stmt e env
739     | Return e               -> check_return e env
740     | If(e, s1, s2)          -> check_if e s1 s2      env
741     | For(e1, e2, e3, e4)    -> check_for e1 e2 e3 e4 env
742     | While(e, s)            -> check_while e s env
743     | Break                  -> check_break env (*
↳ Need to check if in right context *)
744     | Continue               -> check_continue env (* Need to
↳ check if in right context *)
745     | Local(d, s, e)         -> local_handler d s e env
746
747 (* Update this function to return an env object *)
748 and convert_stmt_list_to_sstmt_list env stmt_list =
749     let env_ref = ref(env) in
750     let rec iter = function
751         head::tail ->

```

```

752         let a_head, env = parse_stmt !env_ref head in
753         env_ref := env;
754         a_head::(iter tail)
755     | [] -> []
756 in
757 let sstmt_list = (iter stmt_list), !env_ref in
758 sstmt_list
759
760 let append_code_to_main fbody cname ret_type =
761     let key = Hashtbl.find struct_indexes cname in
762     let init_this = [SLocal(
763         ret_type,
764         "this",
765         SCall(
766             "cast",
767             [SCall("malloc",
768                 [
769                     SCall("sizeof", [SId("ignore",
770 ↪ ret_type)]), Datatype(Int_t), 0)
771                 ],
772             Arraytype(Char_t, 1), 0)
773             ],
774             ret_type, 0
775         )
776     );
777     SExpr(
778         SAssign(
779             SObjAccess(
780                 SId("this", ret_type),
781                 SId(".key", Datatype(Int_t)),
782                 Datatype(Int_t)
783             ),
784             SInt_Lit(key),
785             Datatype(Int_t)
786         ),
787         Datatype(Int_t)
788     )
789 in
790 init_this @ fbody
791
792 let convert_constructor_to_sfdecl class_maps reserved class_map cname constructor =
793     let env = {
794         env_class_maps      = class_maps;
795         env_name            = cname;
796         env_cmap            = class_map;
797         env_locals          = StringMap.empty;
798         env_parameters      = List.fold_left (fun m f -> match f with Formal(d,
799 ↪ s) -> (StringMap.add s f m) | _ -> m) StringMap.empty constructor.formals;
800         env_returnType      = Datatype(Objecttype(cname));

```

```

799         env_in_for           = false;
800         env_in_while         = false;
801         env_reserved          = reserved;
802     } in
803     let fbody = fst (convert_stmt_list_to_sstmt_list env constructor.body) in
804     {
805         sfname                  = Ast.FName (get_constructor_name cname
↪ constructor);
806         sreturnType = Datatype(Objecttype(cname));
807         sformals                = constructor.formals;
808         sbody                   = append_code_to_constructor fbody cname
↪ (Datatype(Objecttype(cname)));
809         func_type               = Sast.User;
810         overrides               = false;
811         source                  = "NA";
812     }
813
814 let check_fbody fname fbody returnType =
815     let len = List.length fbody in
816     if len = 0 then () else
817     let final_stmt = List.hd (List.rev fbody) in
818     match returnType, final_stmt with
819     | Datatype(Void_t), _ -> ()
820     | _, SReturn(_, _) -> ()
821     | _ -> raise(Exceptions.AllNonVoidFunctionsMustEndWithReturn(fname))
822
823 let convert_fdecl_to_sfdecl class_maps reserved class_map cname fdecl =
824     let root_cname = match fdecl.root_cname with
825     | Some(x) -> x
826     | None -> cname
827     in
828     let class_formal =
829         if fdecl.overrides then
830             Ast.Formal(Datatype(Objecttype(root_cname)), "this")
831         else
832             Ast.Formal(Datatype(Objecttype(cname)), "this")
833     in
834     let env_param_helper m fname = match fname with
835     | Formal(d, s) -> (StringMap.add s fname m)
836     | _ -> m
837     in
838     let env_params = List.fold_left env_param_helper StringMap.empty (class_formal ::
↪ fdecl.formals) in
839     let env = {
840         env_class_maps          = class_maps;
841         env_name                 = cname;
842         env_cmap                 = class_map;
843         env_locals               = StringMap.empty;
844         env_parameters           = env_params;

```

```

845         env_returnType      = fdecl.returnType;
846         env_in_for           = false;
847         env_in_while         = false;
848         env_reserved         = reserved;
849     }
850     in
851     let fbody = fst (convert_stmt_list_to_sstmt_list env fdecl.body) in
852     let fname = (get_name cname fdecl) in
853     ignore(check_fbody fname fbody fdecl.returnType);
854     let fbody = if fname = "main"
855         then (append_code_to_main fbody cname (Datatype(Objecttype(cname))))
856         else fbody
857     in
858     (* We add the class as the first parameter to the function for codegen *)
859     {
860         sfname                = Ast.FName (get_name cname fdecl);
861         sreturnType           = fdecl.returnType;
862         sformals               = class_formal :: fdecl.formals;
863         sbody                  = fbody;
864         func_type              = Sast.User;
865         overrides              = fdecl.overrides;
866         source                  = cname;
867     }
868
869 let convert_cdecl_to_sast sfdecls (cdecl:Ast.class_decl) =
870     {
871         scname = cdecl.cname;
872         sfields = cdecl.cbody.fields;
873         sfuncs = sfdecls;
874     }
875
876 (*
877  * Given a list of func_decls for the base class and a single func_decl
878  * for the child class, replaces func_decls for the base class if any of them
879  * have the same method signature
880  *)
881 let replace_fdecl_in_base_methods base_cname base_methods child_fdecl =
882     let replace base_fdecl accum =
883         let get_root_cname = function
884             None -> Some(base_cname)
885             | Some(x) -> Some(x)
886         in
887         let modify_child_fdecl =
888             {
889                 scope = child_fdecl.scope;
890                 fname = child_fdecl.fname;
891                 returnType = child_fdecl.returnType;
892                 formals = child_fdecl.formals;
893                 body = child_fdecl.body;

```

```

894         overrides = true;
895         root_cname = get_root_cname base_fdecl.root_cname;
896     }
897     in
898     if (get_name_without_class base_fdecl) = (get_name_without_class
↪ child_fdecl)
899         then modify_child_fdecl::accum
900         else base_fdecl::accum
901     in
902     List.fold_right replace base_methods []
903
904 let merge_methods base_cname base_methods child_methods =
905     let check_overrides child_fdecl accum =
906         let base_checked_for_overrides =
907             replace_fdecl_in_base_methods base_cname (fst accum) child_fdecl
908         in
909         if (fst accum) = base_checked_for_overrides
910             then ((fst accum), child_fdecl::(snd accum))
911             else (base_checked_for_overrides, (snd accum))
912     in
913     let updated_base_and_child_fdecls =
914         List.fold_right check_overrides child_methods (base_methods, [])
915     in
916     (fst updated_base_and_child_fdecls) @ (snd updated_base_and_child_fdecls)
917
918 let merge_cdecls base_cdecl child_cdecl =
919     (* return a cdecl in which cdecl.cbody.fields contains the fields of
920 the extended class, concatenated by the fields of the child class *)
921     let child_cbody =
922         {
923             fields = base_cdecl.cbody.fields @ child_cdecl.cbody.fields;
924             constructors = child_cdecl.cbody.constructors;
925             methods = merge_methods base_cdecl.cname
↪ base_cdecl.cbody.methods child_cdecl.cbody.methods
926         }
927     in
928     {
929         cname = child_cdecl.cname;
930         extends = child_cdecl.extends;
931         cbody = child_cbody
932     }
933
934 (* returns a list of cdecls that contains inherited fields *)
935 let inherit_fields_cdecls cdecls inheritance_forest =
936     (* iterate through cdecls to make a map for lookup *)
937     let cdecl_lookup = List.fold_left (fun a litem -> StringMap.add litem.cname litem
↪ a) StringMap.empty cdecls in
938     let add_key key pred maps =
939         let elem1 = StringSet.add key (fst maps) in

```

```

940         let accum acc child = StringSet.add child acc in
941         let elem2 = List.fold_left (accum) (snd maps) pred in
942         (elem1, elem2)
943     in
944     let empty_s = StringSet.empty in
945     let res = StringMap.fold add_key inheritance_forest (empty_s, empty_s) in
946     let roots = StringSet.diff (fst res) (snd res) in
947     let rec add_inherited_fields predec desc map_to_update =
948         let merge_fields accum descendant =
949             let updated_predec_cdecl = StringMap.find predec accum in
950             let descendant_cdecl_to_update = StringMap.find descendant
↳ cdecl_lookup in
951                 let merged = merge_cdecls updated_predec_cdecl
↳ descendant_cdecl_to_update in
952                 let updated = (StringMap.add descendant merged accum) in
953                 if (StringMap.mem descendant inheritance_forest) then
954                     let descendants_of_descendant = StringMap.find descendant
↳ inheritance_forest in
955                         add_inherited_fields descendant descendants_of_descendant
↳ updated
956                     else updated
957                 in
958                 List.fold_left merge_fields map_to_update desc
959     in
960     (* map class name of every class_decl in 'cdecls' to its inherited cdecl *)
961     let inherited_cdecls =
962         let traverse_tree tree_root accum =
963             let tree_root_descendant = StringMap.find tree_root
↳ inheritance_forest in
964                 let accum_with_tree_root_mapping = StringMap.add tree_root
↳ (StringMap.find tree_root cdecl_lookup) accum in
965                 add_inherited_fields tree_root tree_root_descendant
↳ accum_with_tree_root_mapping
966                 in
967                 StringSet.fold traverse_tree roots StringMap.empty
968     in
969     (* build a list of updated cdecls corresponding to the sequence of cdecls in
↳ 'cdecls' *)
970     let add_inherited_cdecl cdecl accum =
971         let inherited_cdecl =
972             try StringMap.find cdecl.cname inherited_cdecls
973             with | Not_found -> cdecl
974         in
975         inherited_cdecl::accum
976     in
977     let result = List.fold_right add_inherited_cdecl cdecls [] in
978     result
979
980 let convert_cdecls_to_sast class_maps reserved (cdecls:Ast.class_decl list) =

```



```

981     let find_main = (fun f -> match f.sfname with FName n -> n = "main" | _ -> false)
↪ in
982     let get_main func_list =
983         let mains = (List.find_all find_main func_list) in
984         if List.length mains < 1 then
985             raise Exceptions.MainNotDefined
986         else if List.length mains > 1 then
987             raise Exceptions.MultipleMainsDefined
988         else List.hd mains
989     in
990     let remove_main func_list =
991         List.filter (fun f -> not (find_main f)) func_list
992     in
993     let find_default_constructor cdecl clist =
994         let default_cname = cdecl.cname ^ "." ^ "constructor" in
995         let find_default_c f =
996             match f.sfname with FName n -> n = default_cname | _ -> false
997         in
998         try let _ = List.find find_default_c clist in
999             clist
1000         with | Not_found ->
1001             let default_c = default_sc cdecl.cname in
1002             default_c :: clist
1003     in
1004     let handle_cdecl cdecl =
1005         let class_map = StringMap.find cdecl.cname class_maps in
1006         let sconstructor_list = List.fold_left (fun l c ->
↪ (convert_constructor_to_sfdecl class_maps reserved class_map cdecl.cname c) :: l) []
↪ cdecl.cbody.constructors in
1007         let sconstructor_list = find_default_constructor cdecl sconstructor_list
↪ in
1008         let func_list = List.fold_left (fun l f -> (convert_fdecl_to_sfdecl
↪ class_maps reserved class_map cdecl.cname f) :: l) [] cdecl.cbody.methods in
1009         let sfunc_list = remove_main func_list in
1010         let scdecl = convert_cdecl_to_sast sfunc_list cdecl in
1011         (scdecl, func_list @ sconstructor_list)
1012     in
1013     let iter_cdecls t c =
1014         let scdecl = handle_cdecl c in
1015         (fst scdecl :: fst t, snd scdecl @ snd t)
1016     in
1017     let scdecl_list, func_list = List.fold_left iter_cdecls ([], []) cdecls in
1018     let main = get_main func_list in
1019     let funcs = remove_main func_list in
1020     (* let funcs = (add_default_constructors cdecls class_maps) @ funcs in *)
1021     {
1022         classes                = scdecl_list;
1023         functions              = funcs;
1024         main                   = main;

```

```

1025         reserved                = reserved;
1026     }
1027
1028 let add_reserved_functions =
1029     let reserved_stub name return_type formals =
1030         {
1031             sfname                = FName(name);
1032             sreturnType           = return_type;
1033             sformals              = formals;
1034             sbody                 = [];
1035             func_type             = Sast.Reserved;
1036             overrides             = false;
1037             source                = "NA";
1038         }
1039     in
1040     let i32_t = Datatype(Int_t) in
1041     let void_t = Datatype(Void_t) in
1042     let str_t = Arraytype(Char_t, 1) in
1043     let mf t n = Formal(t, n) in (* Make formal *)
1044     let reserved = [
1045         reserved_stub "print"      (void_t)      ([Many(Any)]);
1046         reserved_stub "malloc"    (str_t)        ([mf i32_t "size"]);
1047         reserved_stub "cast"      (Any)          ([mf Any "in"]);
1048         reserved_stub "sizeof"    (i32_t)        ([mf Any "in"]);
1049         reserved_stub "open"      (i32_t)        ([mf str_t "path"; mf i32_t
↪ "flags"]);
1050         reserved_stub "close"     (i32_t)        ([mf i32_t "fd"]);
1051         reserved_stub "read"      (i32_t)        ([mf i32_t "fd"; mf str_t
↪ "buf"; mf i32_t "nbyte"]);
1052         reserved_stub "write"     (i32_t)        ([mf i32_t "fd"; mf str_t
↪ "buf"; mf i32_t "nbyte"]);
1053         reserved_stub "lseek"     (i32_t)        ([mf i32_t "fd"; mf i32_t
↪ "offset"; mf i32_t "whence"]);
1054         reserved_stub "exit"      (void_t)        ([mf i32_t "status"]);
1055         reserved_stub "getchar"   (i32_t)         ([]);
1056         reserved_stub "input"     (str_t)         ([]);
1057     ] in
1058     reserved
1059
1060 let build_inheritance_forest cdecls cmap =
1061     let handler a cdecl =
1062         match cdecl.extends with
1063         Parent(s)          ->
1064             let new_list = if (StringMap.mem s a) then
1065                 cdecl.cname::(StringMap.find s a)
1066             else
1067                 [cdecl.cname]
1068         in
1069         Hashtbl.add predecessors s new_list;

```

```

1070             (StringMap.add s new_list a)
1071         |      NoParent      -> a
1072     in
1073     let forest = List.fold_left handler StringMap.empty cdecls in
1074
1075     let handler key value =
1076         if not (StringMap.mem key cmap) then
1077             raise (Exceptions.UndefinedClass key)
1078     in
1079     ignore(StringMap.iter handler forest);
1080     forest
1081
1082 let merge_maps m1 m2 =
1083     StringMap.fold (fun k v a -> StringMap.add k v a) m1 m2
1084
1085 let update_class_maps map_type cmap_val cname cmap_to_update =
1086     let update m map_type =
1087         if map_type = "field_map" then
1088             {
1089                 field_map = cmap_val;
1090                 func_map = m.func_map;
1091                 constructor_map = m.constructor_map;
1092                 reserved_map = m.reserved_map;
1093                 cdecl = m.cdecl;
1094             }
1095         else m
1096     in
1097     let updated = StringMap.find cname cmap_to_update in
1098     let updated = update updated map_type in
1099     let updated = StringMap.add cname updated cmap_to_update in
1100     updated
1101
1102 let inherit_fields class_maps predecessors =
1103     (* Get basic inheritance map *)
1104     let add_key key pred map = StringMap.add key pred map in
1105     let cmaps_inherit = StringMap.fold add_key class_maps StringMap.empty in
1106     (* Perform accumulation of child classes *)
1107     let add_key key pred maps =
1108         let elem1 = StringSet.add key (fst maps) in
1109         let accum acc child = StringSet.add child acc in
1110         let elem2 = List.fold_left (accum) (snd maps) pred in
1111         (elem1, elem2)
1112     in
1113     let empty_s = StringSet.empty in
1114     let res = StringMap.fold add_key predecessors (empty_s, empty_s) in
1115     let roots = StringSet.diff (fst res) (snd res) in
1116     (*in let _ = print_set_members roots*)
1117     let rec add_inherited_fields predec desc cmap_to_update =
1118         let cmap_inherit accum descendant =

```

```

1119         let predec_field_map = (StringMap.find predec accum).field_map in
1120         let desc_field_map = (StringMap.find descendant accum).field_map
1121     ↪ in
1122         let merged = merge_maps predec_field_map desc_field_map in
1123         let updated = update_class_maps "field_map" merged descendant
1124     ↪ accum in
1125         if (StringMap.mem descendant predecessors) then
1126             let descendants_of_descendant = StringMap.find descendant
1127     ↪ predecessors in
1128             add_inherited_fields descendant descendants_of_descendant
1129     ↪ updated
1130         else updated
1131     in
1132     List.fold_left cmap_inherit cmap_to_update desc
1133     (* end of add_inherited_fields *)
1134 in
1135     let result = StringSet.fold (fun x a -> add_inherited_fields x (StringMap.find x
1136     ↪ predecessors) a) roots cmap_inherit
1137     (*in let _ = print_map result*)
1138     in result
1139
1140 (* TODO Check that this actually works *)
1141 let check_cyclical_inheritance cdecls predecessors =
1142     let handle_predecessor cdecl parent predecessor =
1143         if cdecl.cname = predecessor then
1144             raise(Exceptions.CyclicalDependencyBetween(cdecl.cname, parent))
1145     in
1146     let handle_cdecl cdecl =
1147         if StringMap.mem cdecl.cname predecessors
1148         then
1149             let pred_list = StringMap.find cdecl.cname predecessors
1150     ↪ in
1151             List.iter (handle_predecessor cdecl (List.hd pred_list))
1152     ↪ pred_list
1153         else ()
1154     in
1155     List.iter handle_cdecl cdecls
1156
1157 let build_func_map_inherited_lookup cdecls_inherited =
1158     let build_func_map cdecl =
1159         let add_func m fdecl = StringMap.add (get_name cdecl.cname fdecl) fdecl m
1160     ↪ in
1161         List.fold_left add_func StringMap.empty cdecl.cbody.methods
1162     in
1163     let add_class_func_map m cdecl = StringMap.add cdecl.cname (build_func_map cdecl)
1164     ↪ m in
1165     List.fold_left add_class_func_map StringMap.empty cdecls_inherited
1166
1167 let add_inherited_methods cmap cdecls func_maps_inherited =

```

```

1159     let find_cdecl cname =
1160         try List.find (fun cdecl -> cdecl.cname = cname) cdecls
1161         with | Not_found -> raise Not_found
1162     in
1163     let update_with_inherited_methods cname cmap =
1164         let fmap = StringMap.find cname func_maps_inherited in
1165         let cdecl = find_cdecl cname in
1166         {
1167             field_map = cmap.field_map;
1168             func_map = fmap;
1169             constructor_map = cmap.constructor_map;
1170             reserved_map = cmap.reserved_map;
1171             cdecl = cdecl;
1172         }
1173     in
1174     let add_updated_cmap cname cmap accum = StringMap.add cname
1175 ↪ (update_with_inherited_methods cname cmap) accum in
1176     StringMap.fold add_updated_cmap cmaps StringMap.empty
1177 let handle_inheritance cdecls class_maps =
1178     let predecessors = build_inheritance_forest cdecls class_maps in
1179     let cdecls_inherited = inherit_fields_cdecls cdecls predecessors in
1180     let func_maps_inherited = build_func_map_inherited_lookup cdecls_inherited in
1181     ignore(check_cyclical_inheritance cdecls predecessors);
1182     let cmaps_with_inherited_fields = inherit_fields class_maps predecessors in
1183     let cmaps_inherited = add_inherited_methods cmaps_with_inherited_fields
1184 ↪ cdecls_inherited func_maps_inherited in
1185     cmaps_inherited, cdecls_inherited
1186 let generate_struct_indexes cdecls =
1187     let cdecl_handler index cdecl =
1188         Hashtbl.add struct_indexes cdecl.cname index
1189     in
1190     List.iteri cdecl_handler cdecls
1191
1192 (* Main method for analyzer *)
1193 let analyze filename program = match program with
1194     Program(includes, classes) ->
1195         (* Include code from external files *)
1196         let cdecls = process_includes filename includes classes in
1197         ignore(generate_struct_indexes cdecls);
1198
1199         (* Add built-in functions *)
1200         let reserved = add_reserved_functions in
1201         (* Generate the class_maps for look up in checking functions *)
1202         let class_maps = build_class_maps reserved cdecls in
1203         let class_maps, cdecls = handle_inheritance cdecls class_maps in
1204         let sast = convert_cdecls_to_sast class_maps reserved cdecls in
1205         sast

```

ast.ml

```

1  type op = Add | Sub | Mult | Div | Equal | Neq | Less | Leq | Greater | Geq | And | Not |
    ↪ Or | Mod
2  type scope = Private | Public
3  type primitive = Int_t | Float_t | Void_t | Bool_t | Char_t | Objecttype of string |
    ↪ ConstructorType | Null_t
4  type datatype = Arraytype of primitive * int | Datatype of primitive | Any
5
6  type extends = NoParent | Parent of string
7  type fname = Constructor | FName of string
8  type formal = Formal of datatype * string | Many of datatype
9
10 type expr =
11     Int_Lit of int
12   | Boolean_Lit of bool
13   | Float_Lit of float
14   | String_Lit of string
15   | Char_Lit of char
16   | This
17   | Id of string
18   | Binop of expr * op * expr
19   | Assign of expr * expr
20   | Noexpr
21   | ArrayCreate of datatype * expr list
22   | ArrayAccess of expr * expr list
23   | ObjAccess of expr * expr
24   | Call of string * expr list
25   | ObjectCreate of string * expr list
26   | ArrayPrimitive of expr list
27   | Unop of op * expr
28   | Null
29   | Delete of expr
30
31 type stmt =
32     Block of stmt list
33   | Expr of expr
34   | Return of expr
35   | If of expr * stmt * stmt
36   | For of expr * expr * expr * stmt
37   | While of expr * stmt
38   | Break
39   | Continue
40   | Local of datatype * string * expr
41
42 type field = Field of scope * datatype * string
43 type include_stmt = Include of string
44
45 type func_decl = {

```

```
46     scope : scope;
47     fname : fname;
48     returnType : datatype;
49     formals : formal list;
50     body : stmt list;
51     overrides : bool;
52     root_cname : string option;
53 }
54
55 type cbody = {
56     fields : field list;
57     constructors : func_decl list;
58     methods : func_decl list;
59 }
60
61 type class_decl = {
62     cname : string;
63     extends : extends;
64     cbody: cbody;
65 }
66
67 type program = Program of include_stmt list * class_decl list
```

bindings.c

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  #define INIT_SIZE 100
5
6  struct s {
7      int x;
8      int y;
9  };
10
11 char* input() {
12     int initial_size = INIT_SIZE;
13     char* str = malloc(initial_size);
14     int index = 0;
15     char tmp = '0';
16     while((tmp = getchar()) != '\n') {
17         if(index >= initial_size - 1) {
18             str = realloc(str, initial_size *= 2);
19         }
20         str[index++] = tmp;
21     }
22     str[index] = '\0';
23     return str;
24 }
25
26 void rec_init(long* arr, int curr_offset, int* static_offsets, int* indexes, int* dims,
27 ↪ int dimc, int dim_curr) {
28
29     //Assign length
30     arr[curr_offset] = dims[dim_curr];
31
32     if(dim_curr + 1 >= dimc)
33         return;
34
35     //Determine the static offset and the dynamic offset
36     int static_offset = static_offsets[dim_curr];
37     int dynamic_offset = 0;
38     for(int i = 0; i < dim_curr; i++) {
39         int tmp = indexes[i];
40         for(int j = i + 1; j <= dim_curr; j++) {
41             tmp *= dims[j];
42         }
43         dynamic_offset += tmp;
44     }
45
46     //Iterate through position and initialize subarrays
47     //Set local indexes to pointers to the subarrays
```



```
47     for(int i = 0; i < dims[dim_curr]; i++) {
48         int offset = (static_offset + (dynamic_offset + i) * (dims[dim_curr + 1]
↳ + 1));
49
50         long* sub = arr + offset;
51         arr[curr_offset + 1 + i] = (long) sub;
52
53         indexes[dim_curr] = i;
54         rec_init(arr, offset, static_offsets, indexes, dims, dimc, dim_curr + 1);
55     }
56 }
57
58 long* init_arr(int* dims, int dimc) {
59
60     int static_offsets[dimc];
61     int total = 0;
62     for(int i = 0; i < dimc; i++) {
63         static_offsets[i] = 1;
64         for(int j = 0; j < i; j++) {
65             static_offsets[i] *= dims[j];
66         }
67         static_offsets[i] *= dims[i] + 1;
68         static_offsets[i] += total;
69         total = static_offsets[i];
70     }
71
72     int indexes[dimc];
73     for(int i = 0; i < dimc; i++) {
74         indexes[i] = 0;
75     }
76
77     //Get total length of array
78     int length = 0;
79     for(int i = 0; i < dimc; i++) {
80         int tmp = 1;
81         for(int j = i - 1; j >= 0; j--) {
82             tmp *= dims[j];
83         }
84         tmp *= dims[i] + 1;
85         length += tmp;
86     }
87
88     //Malloc array
89     long* arr = malloc(length);
90
91     //Set all values to 0 initially
92     for(int i = 0 ; i < length; i++) {
93         arr[i] = 0;
94     }
```

```
95
96     //Initialize the entire array
97     rec_init(arr, 0, static_offsets, indexes, dims, dimc, 0);
98
99     return arr;
100 }
101
102 // int main() {
103
104     //     //Array creation
105     //     int dims[5] = {2, 3, 4, 5, 6};
106     //     int dimc = 5;
107
108     //     long* arr = init_arr(dims, dimc);
109
110     //     //Get total length of array
111     //     int length = 0;
112     //     for(int i = 0; i < dimc; i++) {
113     //         int tmp = 1;
114     //         for(int j = i - 1; j >= 0; j--) {
115     //             tmp *= dims[j];
116     //         }
117     //         tmp *= dims[i] + 1;
118     //         length += tmp;
119     //     }
120
121     //     for(int i = 0; i < length; i++) {
122     //         printf("val: %ld | addr: %ld\n", arr[i], (long) arr + i);
123     //     }
124     //     printf("\n");
125     // }
```

codegen.ml

```

1 (* =====
2    * Code Generation
3    *=====*)
4
5 open LlvM
6 open Ast
7 open Sast
8 open Analyzer
9 open Exceptions
10 open Batteries
11 open Hashtbl
12 open Conf
13
14 open Llvm.MemoryBuffer
15 open Llvm_bitreader
16
17 let context = global_context ()
18 let the_module = create_module context "Dice Codegen"
19 let builder = builder context
20 let named_values:(string, llvalue) Hashtbl.t = Hashtbl.create 50
21 let named_params:(string, llvalue) Hashtbl.t = Hashtbl.create 50
22 let struct_types:(string, lltype) Hashtbl.t = Hashtbl.create 10
23 let struct_field_indexes:(string, int) Hashtbl.t = Hashtbl.create 50
24
25 let i32_t = i32_type context;;
26 let i8_t = i8_type context;;
27 let f_t = double_type context;;
28 let i1_t = i1_type context;;
29 let str_t = pointer_type i8_t;;
30 let i64_t = i64_type context;;
31 let void_t = void_type context;;
32
33 let str_type = Arraytype(Char_t, 1)
34
35 let (br_block) = ref (block_of_value (const_int i32_t 0))
36 let (cont_block) = ref (block_of_value (const_int i32_t 0))
37 let is_loop = ref false
38
39 let debug = fun s ->
40     print_endline ("^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^s");
41     dump_module the_module;
42     print_endline ("^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^s");
43     ()
44
45 let rec get_ptr_type datatype = match datatype with
46     | Arraytype(t, 0) -> get_type (Datatype(t))
47     | Arraytype(t, 1) -> pointer_type (get_type (Datatype(t)))

```

```

48         |           Arraytype(t, i) -> pointer_type (get_ptr_type (Arraytype(t, (i-1))))
49         |           _ -> raise(Exceptions.InvalidStructType "Array Pointer Type")
50
51 and find_struct name =
52     try Hashtbl.find struct_types name
53     with | Not_found -> raise(Exceptions.InvalidStructType name)
54
55 and get_type (datatype:Ast.datatype) = match datatype with
56     Datatype(Int_t) -> i32_t
57     |           Datatype(Float_t) -> f_t
58     |           Datatype(Bool_t) -> i1_t
59     |           Datatype(Char_t) -> i8_t
60     |           Datatype(Void_t) -> void_t
61     |           Datatype(Null_t) -> i32_t
62     |           Datatype(Objecttype(name)) -> pointer_type(find_struct name)
63     |           Arraytype(t, i) -> get_ptr_type (Arraytype(t, (i)))
64     |           d -> raise(Exceptions.InvalidStructType (Utils.string_of_datatype d))
65
66 (* cast will return an llvalue of the desired type *)
67 (* The commented out casts are unsupported actions in Dice *)
68 let cast lhs rhs lhsType rhsType llbuilder =
69     match (lhsType, rhsType) with
70     (* int to, __ ) ( using const_sitofp for signed ints *)
71     (Datatype(Int_t), Datatype(Int_t))                                     ->
    ↪ (lhs, rhs), Datatype(Int_t)
72     |           (Datatype(Int_t), Datatype(Char_t))                               ->
    ↪ (build_uitofp lhs i8_t "tmp" llbuilder, rhs), Datatype(Char_t)
73     (* /           (Datatype(Int_t),
    ↪ Datatype(Bool_t))                                     -> (lhs, const_zext rhs i32_t) *)
74     |           (Datatype(Int_t), Datatype(Float_t))                               -> (build_sitofp
    ↪ lhs f_t "tmp" llbuilder, rhs), Datatype(Float_t)
75
76     (* char to, __ ) ( using uitofp since char isn't signed *)
77     |           (Datatype(Char_t), Datatype(Int_t))                               -> (lhs,
    ↪ build_uitofp rhs i8_t "tmp" llbuilder), Datatype(Char_t)
78     |           (Datatype(Char_t), Datatype(Char_t))                               -> (lhs, rhs),
    ↪ Datatype(Char_t)
79     (* /           (Datatype(Char_t), Datatype(Bool_t))                               -> (lhs,
    ↪ const_zext rhs i8_t) *)
80     (* /           (Datatype(Char_t), Datatype(Float_t))                               ->
    ↪ (const_uitofp lhs f_t, rhs) *)
81
82     (* bool to, __ ) ( zext fills the empty bits with zeros, zero extension *)
83     (* /           (Datatype(Bool_t), Datatype(Int_t))                               ->
    ↪ (const_zext lhs i32_t, rhs) *)
84     (* /           (Datatype(Bool_t), Datatype(Char_t))                               ->
    ↪ (const_zext lhs i8_t, rhs) *)
85     |           (Datatype(Bool_t), Datatype(Bool_t))                               -> (lhs,
    ↪ rhs), Datatype(Bool_t)

```

```

86      (* /      (Datatype(Bool_t), Datatype(Float_t))      ->
↪    (const_uitofp lhs f_t, rhs) *)
87
88      (* float to, __) ( using fptosi for signed ints *)
89      |    (Datatype(Float_t), Datatype(Int_t))      -> (lhs,
↪    build_sitofp rhs f_t "tmp" llbuilder), Datatype(Float_t)
90      (* /      (Datatype(Float_t), Datatype(Char_t))      ->
↪    (lhs, const_uitofp rhs f_t) *)
91      (* /      (Datatype(Float_t), Datatype(Bool_t))      ->
↪    (lhs, const_uitofp rhs f_t) *)
92      |    (Datatype(Float_t), Datatype(Float_t))      -> (lhs, rhs),
↪    Datatype(Float_t)
93
94      | Datatype(Objecttype(d)), Datatype(Null_t)      -> (lhs, rhs),
↪    lhsType
95      | Datatype(Null_t), Datatype(Objecttype(d))      -> (rhs, lhs),
↪    rhsType
96      | Datatype(Objecttype(d)), t      ->
↪    raise(Exceptions.CanOnlyCompareObjectsWithNull(d, (Utils.string_of_datatype t)))
97
98      | Arraytype(d, s), Datatype(Null_t)      ->
↪    (lhs, rhs), lhsType
99      | Datatype(Null_t), Arraytype(d, s)      -> (rhs,
↪    lhs), rhsType
100     | Arraytype(d, _), t
↪
↪
↪    raise(Exceptions.CanOnlyCompareArraysWithNull(Utils.string_of_primitive d,
↪    (Utils.string_of_datatype t)))
101
102     |      _
↪
↪
↪    raise (Exceptions.CannotCastTypeException(Utils.string_of_datatype lhsType,
↪    Utils.string_of_datatype rhsType))
103
104 let rec handle_binop e1 op e2 d llbuilder =
105     (* Get the types of e1 and e2 *)
106     let type1 = Analyzer.get_type_from_sexpr e1 in
107     let type2 = Analyzer.get_type_from_sexpr e2 in
108
109     (* Generate llvalues from e1 and e2 *)
110
111     let e1 = codegen_sexpr llbuilder e1 in
112     let e2 = codegen_sexpr llbuilder e2 in
113
114     let float_ops op e1 e2 =
115     match op with
116         Add      -> build_fadd e1 e2 "flt_addtmp" llbuilder
117         |      Sub      -> build_fsub e1 e2 "flt_subtmp" llbuilder
118         |      Mult     -> build_fmuls e1 e2 "flt_multtmp" llbuilder

```

```

119         |      Div      -> build_fdiv e1 e2 "flt_divtmp" llbuilder
120         |      Mod      -> build_frem e1 e2 "flt_sremtmp" llbuilder
121         |      Equal     -> build_fcmp Fcmp.Oeq e1 e2 "flt_eqtmp"
↳ llbuilder
122         |      Neq      -> build_fcmp Fcmp.One e1 e2 "flt_neqtmp" llbuilder
123         |      Less     -> build_fcmp Fcmp.Ult e1 e2 "flt_lesstmp"
↳ llbuilder
124         |      Leq      -> build_fcmp Fcmp.Ole e1 e2 "flt_leqtmp" llbuilder
125         |      Greater  -> build_fcmp Fcmp.Ogt e1 e2 "flt_sgttmp"
↳ llbuilder
126         |      Geq      -> build_fcmp Fcmp.Oge e1 e2 "flt_sgetmp" llbuilder
127         |      -        -> raise Exceptions.FloatOpNotSupported
128
129     in
130
131     (* chars are considered ints, so they will use int_ops as well*)
132     let int_ops op e1 e2 =
133     match op with
134         Add      -> build_add e1 e2 "addtmp" llbuilder
135         |      Sub      -> build_sub e1 e2 "subtmp" llbuilder
136         |      Mult     -> build_mul e1 e2 "multtmp" llbuilder
137         |      Div      -> build_sdiv e1 e2 "divtmp" llbuilder
138         |      Mod      -> build_srem e1 e2 "sremtmp" llbuilder
139         |      Equal     -> build_icmp Icmp.Eq e1 e2 "eqtmp" llbuilder
140         |      Neq      -> build_icmp Icmp.Ne e1 e2 "neqtmp" llbuilder
141         |      Less     -> build_icmp Icmp.Slt e1 e2 "lesstmp" llbuilder
142         |      Leq      -> build_icmp Icmp.Sle e1 e2 "leqtmp" llbuilder
143         |      Greater  -> build_icmp Icmp.Sgt e1 e2 "sgtmp" llbuilder
144         |      Geq      -> build_icmp Icmp.Sge e1 e2 "sgetmp" llbuilder
145         |      And      -> build_and e1 e2 "andtmp" llbuilder
146         |      Or       -> build_or e1 e2 "ortmp" llbuilder
147         |      -        -> raise Exceptions.IntOpNotSupported
148     in
149
150     let obj_ops op e1 e2 =
151     match op with
152         Equal -> build_is_null e1 "tmp" llbuilder
153         |      Neq -> build_is_not_null e1 "tmp" llbuilder
154         |      -   -> raise
↳ (Exceptions.ObjOpNotSupported(Utills.string_of_op op))
155     in
156
157     let (e1, e2), d = cast e1 e2 type1 type2 llbuilder in
158
159     let type_handler d = match d with
160         Datatype(Float_t) -> float_ops op e1 e2
161         |      Datatype(Int_t)
162         |      Datatype(Bool_t)
163         |      Datatype(Char_t) -> int_ops op e1 e2

```

```

164         |           Datatype(Objecttype(_))
165         |           Arraytype(_, _) -> obj_ops op e1 e2
166         |           _ -> raise Exceptions.InvalidBinopEvaluationType
167     in
168
169     type_handler d
170
171 and handle_unop op e d llbuilder =
172     (* Get the type of e *)
173     let eType = Analyzer.get_type_from_sexpr e in
174     (* Get llvalue *)
175     let e = codegen_sexpr llbuilder e in
176
177     let unops op eType e = match (op, eType) with
178     (Sub, Datatype(Int_t))          -> build_neg e "int_unoptmp"
179   ↪ llbuilder
180     | (Sub, Datatype(Float_t))      -> build_fneg e "flt_unoptmp"
181   ↪ llbuilder
182     | (Not, Datatype(Bool_t))      -> build_not e "bool_unoptmp" llbuilder
183     | _ -> raise Exceptions.UnopNotSupported in
184
185     let unop_type_handler d = match d with
186     Datatype(Float_t)
187     | Datatype(Int_t)
188     | Datatype(Bool_t) -> unops op eType e
189     | _ -> raise Exceptions.InvalidUnopEvaluationType
190   in
191
192     unop_type_handler d
193
194 and func_lookup fname =
195     match (lookup_function fname the_module) with
196     None -> raise (Exceptions.LLVMFunctionNotFound fname)
197     | Some f -> f
198
199 and codegen_print e1 llbuilder =
200     let printf = func_lookup "printf" in
201     let tmp_count = ref 0 in
202     let incr_tmp = fun x -> incr tmp_count in
203
204     let map_expr_to_printfexpr expr =
205         let exprType = Analyzer.get_type_from_sexpr expr in
206         match exprType with
207         Datatype(Bool_t) ->
208             incr_tmp ();
209             let tmp_var = "tmp" ^ (string_of_int !tmp_count) in
210             let trueStr = SString_Lit("true") in
211             let falseStr = SString_Lit("false") in
212             let id = SId(tmp_var, str_type) in

```

```

211         ignore(codegen_stmt llbuilder (SLocal(str_type, tmp_var,
↳ SNoexpr)));
212         ignore(codegen_stmt llbuilder (SIf(expr,
213                                     SExpr(SAssign(i
↳ trueStr, str_type), str_type),
214                                     SExpr(SAssign(i
↳ falseStr, str_type), str_type)
215                                     ));
216         codegen_sexpr llbuilder id
217         | _ -> codegen_sexpr llbuilder expr
218         in
219
220         let params = List.map map_expr_to_printfexpr el in
221         let param_types = List.map (Analyzer.get_type_from_sexpr) el in
222
223         let map_param_to_string = function
224             Arraytype(Char_t, 1)      -> "%s"
225             | Datatype(Int_t)          -> "%d"
226             | Datatype(Float_t)        -> "%f"
227             | Datatype(Bool_t)         -> "%s"
228             | Datatype(Char_t)         -> "%c"
229             | _                        -> raise
↳ (Exceptions.InvalidTypePassedToPrintf)
230         in
231         let const_str = List.fold_left (fun s t -> s ^ map_param_to_string t) ""
↳ param_types in
232         let s = codegen_sexpr llbuilder (SString_Lit(const_str)) in
233         let zero = const_int i32_t 0 in
234         let s = build_in_bounds_gep s [| zero |] "tmp" llbuilder in
235         build_call printf (Array.of_list (s :: params)) "tmp" llbuilder
236
237 and codegen_func_call fname el d llbuilder =
238     let f = func_lookup fname in
239     let params = List.map (codegen_sexpr llbuilder) el in
240     match d with
241         Datatype(Void_t) -> build_call f (Array.of_list params) "" llbuilder
242         | _ -> build_call f (Array.of_list
↳ params) "tmp" llbuilder
243
244 and codegen_sizeof el llbuilder =
245     let type_of = Analyzer.get_type_from_sexpr (List.hd el) in
246     let type_of = get_type type_of in
247     let size_of = size_of type_of in
248     build_bitcast size_of i32_t "tmp" llbuilder
249
250 and codegen_cast el d llbuilder =
251     let cast_malloc_to_objtype lhs currType newType llbuilder = match newType with
252         Datatype(Objecttype(x)) ->
253         let obj_type = get_type (Datatype(Objecttype(x))) in

```



```

254         build_pointercast lhs obj_type "tmp" llbuilder
255         | _ as t -> raise
↳ (Exceptions.CannotCastTypeException(Utils.string_of_datatype currType,
↳ Utils.string_of_datatype t))
256     in
257     let expr = List.hd el in
258     let t = Analyzer.get_type_from_sexpr expr in
259     let lhs = match expr with
260     | Sast.SId(id, d) -> codegen_id false false id d llbuilder
261     | SObjAccess(e1, e2, d) -> codegen_obj_access false e1 e2 d llbuilder
262     | SArrayAccess(se, sel, d) -> codegen_array_access true se sel d
↳ llbuilder
263     | _ -> codegen_sexpr llbuilder expr
264     in
265     cast_malloc_to_objtype lhs t d llbuilder
266
267 and codegen_call llbuilder d el = function
268     "print" -> codegen_print el llbuilder
269     | "sizeof" -> codegen_sizeof el llbuilder
270     | "cast" -> codegen_cast el d llbuilder
271     | "malloc" -> codegen_func_call "malloc" el d llbuilder
272     | "open" -> codegen_func_call "open" el d llbuilder
273     | "write" -> codegen_func_call "write" el d llbuilder
274     | "close" -> codegen_func_call "close" el d llbuilder
275     | "read" -> codegen_func_call "read" el d llbuilder
276     | "lseek" -> codegen_func_call "lseek" el d llbuilder
277     | "exit" -> codegen_func_call "exit" el d llbuilder
278     | "input" -> codegen_func_call "input" el d llbuilder
279     | "getchar" -> codegen_func_call "getchar" el d llbuilder
280     | _ as fname -> raise
↳ (Exceptions.UnableToCallFunctionWithoutParent fname) (* codegen_func_call fname el
↳ llbuilder *)
281
282 and codegen_id isDeref checkParam id d llbuilder =
283     if isDeref then
284         try Hashtbl.find named_params id
285         with | Not_found ->
286         try let _val = Hashtbl.find named_values id in
287             build_load _val id llbuilder
288         with | Not_found -> raise (Exceptions.UnknownVariable id)
289     else
290         try Hashtbl.find named_values id
291         with | Not_found ->
292             try
293                 let _val = Hashtbl.find named_params id in
294                 if checkParam then raise (Exceptions.CannotAssignParam
↳ id)
295             else _val
296         with | Not_found -> raise (Exceptions.UnknownVariable id)

```

```

297
298 and codegen_assign lhs rhs d llbuilder =
299     let rhsType = Analyzer.get_type_from_sexpr rhs in
300     (* Special case '=' because we don't want to emit the LHS as an
301      * expression. *)
302     let lhs, isObjAccess = match lhs with
303     | Sast.SId(id, d) -> codegen_id false false id d llbuilder, false
304     | SObjAccess(e1, e2, d) -> codegen_obj_access false e1 e2 d llbuilder,
↪ true
305     | SArrayAccess(se, sel, d) -> codegen_array_access true se sel d
↪ llbuilder, true
306     | _ -> raise Exceptions.AssignLHSMustBeAssignable
307     in
308     (* Codegen the rhs. *)
309     let rhs = match rhs with
310     | Sast.SId(id, d) -> codegen_id false false id d llbuilder
311     | SObjAccess(e1, e2, d) -> codegen_obj_access true e1 e2 d llbuilder
312     | _ -> codegen_sexpr llbuilder rhs
313     in
314     let rhs = match d with
315     | Datatype(Objecttype(_)) ->
316         if isObjAccess then rhs
317         else build_load rhs "tmp" llbuilder
318     | Datatype(Null_t) -> const_null (get_type d)
319     | _ -> rhs
320     in
321     let rhs = match d, rhsType with
322     | Datatype(Char_t), Datatype(Int_t) -> build_uitofp rhs i8_t "tmp"
↪ llbuilder
323     | Datatype(Int_t), Datatype(Char_t) -> build_uitofp rhs i32_t "tmp"
↪ llbuilder
324     | _ -> rhs
325     in
326     (* Lookup the name. *)
327     ignore(build_store rhs lhs llbuilder);
328     rhs
329
330 and deref ptr t llbuilder =
331     build_gep ptr (Array.of_list [ptr]) "tmp" llbuilder
332
333 and codegen_obj_access isAssign lhs rhs d llbuilder =
334     let codegen_func_call param_ty fptr parent_expr e1 d llbuilder =
335         let match_sexpr se = match se with
336         | SId(id, d) -> let isDeref = match d with
337             Datatype(Objecttype(_)) -> false
338             _ -> true
339         in codegen_id isDeref false id d llbuilder
340         | se -> codegen_sexpr llbuilder se
341     in

```

```

342         let parent_expr = build_pointercast parent_expr param_ty "tmp" llbuilder
↪   in
343         let params = List.map match_sexpr el in
344         match d with
345         | Datatype(Void_t) -> build_call fptr (Array.of_list (parent_expr
↪   :: params)) "" llbuilder
346         | _ -> build_call fptr (Array.of_list (parent_expr :: params))
↪   "tmp" llbuilder
347         in
348         let check_lhs = function
349         | SId(s, d) -> codegen_id false false s d
↪   llbuilder
350         | SArrayAccess(e, el, d) -> codegen_array_access false e el d
↪   llbuilder
351         | se -> raise (Exceptions.LHSofRootAccessMustBeIDorFunc
↪   (Utils.string_of_sexpr se))
352         in
353         (* Needs to be changed *)
354         let rec check_rhs isLHS parent_expr parent_type =
355         let parent_str = Utils.string_of_object parent_type in
356         function
357         (* Check fields in parent *)
358         | SId(field, d) ->
359             let search_term = (parent_str ^ "." ^ field) in
360             let field_index = Hashtbl.find struct_field_indexes
↪   search_term in
361             let _val = build_struct_gep parent_expr field_index field
↪   llbuilder in
362             let _val = match d with
363             | Datatype(Objecttype(_)) ->
364                 if not isAssign then _val
365                 else build_load _val field llbuilder
366             | _ ->
367                 if not isAssign then
368                     _val
369                 else
370                     build_load _val field llbuilder
371             in
372             _val
373
374         | SArrayAccess(e, el, d) ->
375
376         let ce = check_rhs false parent_expr parent_type e in
377         let index = codegen_sexpr llbuilder (List.hd el) in
378         let index = match d with
379         | Datatype(Char_t) -> index
380         | _ -> build_add index (const_int i32_t 1) "tmp"
↪   llbuilder
381         in

```

```

382         let _val = build_gep ce [| index |] "tmp" llbuilder in
383         if isLHS && isAssign
384             then _val
385             else build_load _val "tmp" llbuilder
386
387         (* Check functions in parent *)
388         | SCall(fname, e1, d, index) ->
389             let index = const_int i32_t index in
390             let c_index = build_struct_gep parent_expr 0 "cindex"
391
392             ↪ llbuilder in
393
394             let c_index = build_load c_index "cindex" llbuilder in
395             let lookup = func_lookup "lookup" in
396             let fptr = build_call lookup [| c_index; index |] "fptr"
397
398             ↪ llbuilder in
399
400             let fptr2 = func_lookup fname in
401             let f_ty = type_of fptr2 in
402             let param1 = param fptr2 0 in
403             let param_ty = type_of param1 in
404             let fptr = build_pointercast fptr f_ty fname llbuilder in
405             let ret = codegen_func_call param_ty fptr parent_expr e1
406
407             ↪ d llbuilder in
408
409             let ret = ret
410                 (* if not isLHS && not isAssign then
411                    build_load ret "tmp" llbuilder
412                 else
413                    ret *)
414
415             in
416             ret
417         (* Set parent, check if base is field *)
418         | SObjAccess(e1, e2, d) ->
419             let e1_type = Analyzer.get_type_from_sexpr e1 in
420             let e1 = check_rhs true parent_expr parent_type e1 in
421             let e2 = check_rhs true e1 e1_type e2 in
422             e2
423         | _ as e -> raise (Exceptions.InvalidAccessLHS
424             ↪ (Utils.string_of_sexpr e))
425
426         in
427         let lhs_type = Analyzer.get_type_from_sexpr lhs in
428         match lhs_type with
429         | Arraytype(_, _) ->
430             let lhs = codegen_sexpr llbuilder lhs in
431             let _ = match rhs with
432                 | SId("length", _) -> "length"
433                 | _ -> raise (Exceptions.CanOnlyAccessLengthOfArray)
434             in
435             let _val = build_gep lhs [| (const_int i32_t 0) |] "tmp"
436
437             ↪ llbuilder in
438
439             build_load _val "tmp" llbuilder
440         | _ ->

```

```

426         let lhs = check_lhs lhs in
427         let rhs = check_rhs true lhs lhs_type rhs in
428         rhs
429
430 and codegen_obj_create fname el d llbuilder =
431     let f = func_lookup fname in
432     let params = List.map (codegen_sexpr llbuilder) el in
433     let obj = build_call f (Array.of_list params) "tmp" llbuilder in
434     obj
435
436 and codegen_string_lit s llbuilder =
437     if s = "true" then build_global_stringptr "true" "tmp" llbuilder
438     else if s = "false" then build_global_stringptr "false" "tmp" llbuilder
439     else build_global_stringptr s "tmp" llbuilder
440
441 and codegen_array_access isAssign e el d llbuilder =
442     let index = codegen_sexpr llbuilder (List.hd el) in
443     let index = match d with
444         Datatype(Char_t) -> index
445         | _ -> build_add index (const_int i32_t 1) "tmp" llbuilder
446     in
447     let arr = codegen_sexpr llbuilder e in
448     let _val = build_gep arr [| index |] "tmp" llbuilder in
449     if isAssign
450         then _val
451         else build_load _val "tmp" llbuilder
452
453 and initialise_array arr arr_len init_val start_pos llbuilder =
454     let new_block label =
455         let f = block_parent (insertion_block llbuilder) in
456         append_block (global_context ()) label f
457     in
458     let bbcurr = insertion_block llbuilder in
459     let bbcond = new_block "array.cond" in
460     let bbbody = new_block "array.init" in
461     let bbdone = new_block "array.done" in
462     ignore (build_br bbcond llbuilder);
463     position_at_end bbcond llbuilder;
464
465     (* Counter into the length of the array *)
466     let counter = build_phi [const_int i32_t start_pos, bbcurr] "counter" llbuilder in
467     add_incoming ((build_add counter (const_int i32_t 1) "tmp" llbuilder), bbbody) counter;
468     let cmp = build_icmp Icmp.Slt counter arr_len "tmp" llbuilder in
469     ignore (build_cond_br cmp bbbody bbdone llbuilder);
470     position_at_end bbbody llbuilder;
471
472     (* Assign array position to init_val *)
473     let arr_ptr = build_gep arr [| counter |] "tmp" llbuilder in
474     ignore (build_store init_val arr_ptr llbuilder);

```

```

475   ignore (build_br bbcond llbuilder);
476   position_at_end bbdone llbuilder
477
478 and codegen_array_create llbuilder t expr_type el =
479     if(List.length el > 1) then raise(Exceptions.ArrayLargerThan1Unsupported)
480     else
481     match expr_type with
482     Arraytype(Char_t, 1) ->
483         let e = List.hd el in
484         let size = (codegen_sexpr llbuilder e) in
485         let t = get_type t in
486         let arr = build_array_malloc t size "tmp" llbuilder in
487         let arr = build_pointercast arr (pointer_type t) "tmp" llbuilder in
488         (* initialise_array arr size (const_int i32_t 0) 0 llbuilder; *)
489         arr
490     | _ ->
491         let e = List.hd el in
492         let t = get_type t in
493
494         (* This will not work for arrays of objects *)
495         let size = (codegen_sexpr llbuilder e) in
496         let size_t = build_intcast (size_of t) i32_t "tmp" llbuilder in
497         let size = build_mul size_t size "tmp" llbuilder in
498         let size_real = build_add size (const_int i32_t 1) "arr_size" llbuilder
499
500         let arr = build_array_malloc t size_real "tmp" llbuilder in
501         let arr = build_pointercast arr (pointer_type t) "tmp" llbuilder in
502
503         let arr_len_ptr = build_pointercast arr (pointer_type i32_t) "tmp"
504         llbuilder in
505
506         (* Store length at this position *)
507         ignore(build_store size_real arr_len_ptr llbuilder);
508         initialise_array arr_len_ptr size_real (const_int i32_t 0) 0 llbuilder;
509         arr
510 and codegen_array_prim d el llbuilder =
511     let t = d in
512     let size = (const_int i32_t ((List.length el))) in
513     let size_real = (const_int i32_t ((List.length el) + 1)) in
514     let t = get_type t in
515     let arr = build_array_malloc t size_real "tmp" llbuilder in
516     let arr = build_pointercast arr t "tmp" llbuilder in
517     let size_casted = build_bitcast size t "tmp" llbuilder in
518     ignore(if d = Arraytype(Char_t, 1) then ignore(build_store size_casted arr
519     llbuilder)); (* Store length at this position *)
520     (* initialise_array arr size_real (const_int i32_t 0) 1 llbuilder; *)

```

```

521     let llvalues = List.map (codegen_sexpr llbuilder) el in
522     List.iteri (fun i llval ->
523         let arr_ptr = build_gep arr [| (const_int i32_t (i+1)) |]
524     ↪ "tmp" llbuilder in
525         ignore(build_store llval arr_ptr llbuilder); ) llvalues;
526     arr
527 and codegen_delete e llbuilder =
528     let ce = match e with
529         SId(id, d) -> codegen_id false false id d llbuilder
530     | _ -> codegen_sexpr llbuilder e
531     in
532     build_free ce llbuilder
533
534 and codegen_sexpr llbuilder = function
535     SInt_Lit(i)                -> const_int i32_t i
536 | SBoolean_Lit(b)            -> if b then const_int i1_t 1 else
537 ↪ const_int i1_t 0
538 | SFloat_Lit(f)              -> const_float f_t f
539 | SString_Lit(s)             -> codegen_string_lit s llbuilder
540 | SChar_Lit(c)               -> const_int i8_t (Char.code c)
541 | SId(id, d)                 -> codegen_id true false id d llbuilder
542 | SBinop(e1, op, e2, d)      -> handle_binop e1 op e2 d llbuilder
543 | SAssign(e1, e2, d)         -> codegen_assign e1 e2 d llbuilder
544 | SNoexpr                    -> build_add (const_int i32_t 0) (const_int
545 ↪ i32_t 0) "nop" llbuilder
546 | SArrayCreate(t, e1, d)     -> codegen_array_create llbuilder t d e1
547 | SArrayAccess(e, e1, d)     -> codegen_array_access false e e1 d
548 ↪ llbuilder
549 | SObjAccess(e1, e2, d)      -> codegen_obj_access true e1 e2 d
550 ↪ llbuilder
551 | SCall(fname, e1, d, _)     -> codegen_call llbuilder d e1 fname
552 | SObjectCreate(id, e1, d)   -> codegen_obj_create id e1 d llbuilder
553 | SArrayPrimitive(e1, d)     -> codegen_array_prim d e1 llbuilder
554 | SUnop(op, e, d)           -> handle_unop op e d llbuilder
555 | SNull                     -> const_null i32_t
556 | SDelete e                  -> codegen_delete e
557 ↪ llbuilder
558
559 and codegen_if_stmt exp then_ (else_:Sast.sstmt) llbuilder =
560     let cond_val = codegen_sexpr llbuilder exp in
561
562     (* Grab the first block so that we might later add the conditional branch
563      * to it at the end of the function. *)
564     let start_bb = insertion_block llbuilder in
565     let the_function = block_parent start_bb in
566
567     let then_bb = append_block context "then" the_function in
568
569

```

```

564      (* Emit 'then' value. *)
565      position_at_end then_bb llbuilder;
566      let _ (* then_val *) = codegen_stmt llbuilder then_ in
567
568      (* Codegen of 'then' can change the current block, update then_bb for the
569       * phi. We create a new name because one is used for the phi node, and the
570       * other is used for the conditional branch. *)
571      let new_then_bb = insertion_block llbuilder in
572
573      (* Emit 'else' value. *)
574      let else_bb = append_block context "else" the_function in
575      position_at_end else_bb llbuilder;
576      let _ (* else_val *) = codegen_stmt llbuilder else_ in
577
578      (* Codegen of 'else' can change the current block, update else_bb for the
579       * phi. *)
580      let new_else_bb = insertion_block llbuilder in
581
582
583      let merge_bb = append_block context "ifcont" the_function in
584      position_at_end merge_bb llbuilder;
585      (* let then_bb_val = value_of_block new_then_bb in *)
586      let else_bb_val = value_of_block new_else_bb in
587      (* let incoming = [(then_bb_val, new_then_bb); (else_bb_val, new_else_bb)] in *)
588      (* let phi = build_phi incoming "iftmp" llbuilder in *)
589
590      (* Return to the start block to add the conditional branch. *)
591      position_at_end start_bb llbuilder;
592      ignore (build_cond_br cond_val then_bb else_bb llbuilder);
593
594      (* Set a unconditional branch at the end of the 'then' block and the
595       * 'else' block to the 'merge' block. *)
596      position_at_end new_then_bb llbuilder; ignore (build_br merge_bb llbuilder);
597      position_at_end new_else_bb llbuilder; ignore (build_br merge_bb llbuilder);
598
599      (* Finally, set the builder to the end of the merge block. *)
600      position_at_end merge_bb llbuilder;
601
602      else_bb_val (* phi *)
603
604  and codegen_for init_ cond_ inc_ body_ llbuilder =
605      let old_val = !is_loop in
606      is_loop := true;
607
608      let the_function = block_parent (insertion_block llbuilder) in
609
610      (* Emit the start code first, without 'variable' in scope. *)
611      let _ = codegen_sexpr llbuilder init_ in
612

```



```

613      (* Make the new basic block for the loop header, inserting after current
614      * block. *)
615      let loop_bb = append_block context "loop" the_function in
616      (* Insert maintenance block *)
617      let inc_bb = append_block context "inc" the_function in
618      (* Insert condition block *)
619      let cond_bb = append_block context "cond" the_function in
620      (* Create the "after loop" block and insert it. *)
621      let after_bb = append_block context "afterloop" the_function in
622
623      let _ = if not old_val then
624          cont_block := inc_bb;
625          br_block := after_bb;
626      in
627
628      (* Insert an explicit fall through from the current block to the
629      * loop_bb. *)
630      ignore (build_br cond_bb llbuilder);
631
632      (* Start insertion in loop_bb. *)
633      position_at_end loop_bb llbuilder;
634
635      (* Emit the body of the loop. This, like any other expr, can change the
636      * current BB. Note that we ignore the value computed by the body, but
637      * don't allow an error *)
638      ignore (codegen_stmt llbuilder body_);
639
640      let bb = insertion_block llbuilder in
641      move_block_after bb inc_bb;
642      move_block_after inc_bb cond_bb;
643      move_block_after cond_bb after_bb;
644      ignore (build_br inc_bb llbuilder);
645
646      (* Start insertion in loop_bb. *)
647      position_at_end inc_bb llbuilder;
648      (* Emit the step value. *)
649      let _ = codegen_sexpr llbuilder inc_ in
650      ignore (build_br cond_bb llbuilder);
651
652      position_at_end cond_bb llbuilder;
653
654      let cond_val = codegen_sexpr llbuilder cond_ in
655      ignore (build_cond_br cond_val loop_bb after_bb llbuilder);
656
657      (* Any new code will be inserted in after_bb. *)
658      position_at_end after_bb llbuilder;
659
660      is_loop := old_val;
661

```

```

662      (* for expr always returns 0.0. *)
663      const_null f_t
664
665  and codegen_while cond_ body_ llbuilder =
666      let null_sexpr = SInt_Lit(0) in
667      codegen_for null_sexpr cond_ null_sexpr body_ llbuilder
668
669  and codegen_alloc datatype var_name expr llbuilder =
670      let t = match datatype with
671          Datatype(Objecttype(name)) -> find_struct name
672          | _ -> get_type datatype
673      in
674      let alloc = build_alloc t var_name llbuilder in
675      Hashtbl.add named_values var_name alloc;
676      let lhs = SId(var_name, datatype) in
677      match expr with
678          SNoexpr -> alloc
679          | _ -> codegen_assign lhs expr datatype llbuilder
680
681  and codegen_ret d expr llbuilder =
682      match expr with
683          SId(name, d) ->
684              (match d with
685                  | Datatype(Objecttype(_)) -> build_ret (codegen_id false false
686 ↪ name d llbuilder) llbuilder
687                  | _ -> build_ret (codegen_id true true name d llbuilder)
688 ↪ llbuilder)
689              | SObjAccess(e1, e2, d) -> build_ret (codegen_obj_access true e1 e2 d
690 ↪ llbuilder) llbuilder
691              | SNoexpr -> build_ret_void llbuilder
692              | _ -> build_ret (codegen_sexpr llbuilder expr) llbuilder
693
694  and codegen_break llbuilder =
695      let block = fun () -> !br_block in
696      build_br (block ()) llbuilder
697
698  and codegen_continue llbuilder =
699      let block = fun () -> !cont_block in
700      build_br (block ()) llbuilder
701
702  and codegen_stmt llbuilder = function
703      SBlock s1 -> List.hd(List.map
704 ↪ (codegen_stmt llbuilder) s1)
705      | SExpr(e, d) -> codegen_sexpr llbuilder e
706      | SReturn(e, d) -> codegen_ret d e llbuilder
707      | SIf (e, s1, s2) -> codegen_if_stmt e s1 s2 llbuilder
708      | SFor (e1, e2, e3, s) -> codegen_for e1 e2 e3 s llbuilder
709      | SWhile (e, s) -> codegen_while e s llbuilder
710      | SBreak -> codegen_break llbuilder

```

```

707         | SContinue                                -> codegen_continue llbuilder
708         | SLocal(d, s, e)                          -> codegen_alloca d s e llbuilder
709
710 let codegen_funcstub sfdecl =
711     let fname = (Utils.string_of_fname sfdecl.sfname) in
712     let is_var_arg = ref false in
713     let params = List.rev (List.fold_left (fun l -> (function Formal(t, _) ->
714 ↪ get_type t :: l | _ -> is_var_arg := true; l)) [] sfdecl.sformals) in
715     let fty = if !is_var_arg
716 ↪ then var_arg_function_type (get_type sfdecl.sreturnType)
717 ↪ (Array.of_list params)
718 ↪ else function_type (get_type sfdecl.sreturnType) (Array.of_list
719 ↪ params)
720 ↪ in
721     define_function fname fty the_module
722
723 let init_params f formals =
724     let formals = Array.of_list (formals) in
725     Array.iteri (fun i a ->
726 ↪ let n = formals.(i) in
727 ↪ let n = Utils.string_of_formal_name n in
728 ↪ set_value_name n a;
729 ↪ Hashtbl.add named_params n a;
730 ↪ ) (params f)
731
732 let codegen_func sfdecl =
733     Hashtbl.clear named_values;
734     Hashtbl.clear named_params;
735     let fname = (Utils.string_of_fname sfdecl.sfname) in
736     let f = func_lookup fname in
737     let llbuilder = builder_at_end context (entry_block f) in
738     let _ = init_params f sfdecl.sformals in
739     let _ = if sfdecl.overrides then
740 ↪ let this_param = Hashtbl.find named_params "this" in
741 ↪ let source = Datatype(Objecttype(sfdecl.source)) in
742 ↪ let casted_param = build_pointercast this_param (get_type source)
743 ↪ "casted" llbuilder in
744 ↪ Hashtbl.replace named_params "this" casted_param;
745 ↪ in
746     let _ = codegen_stmt llbuilder (SBlock (sfdecl.sbody)) in
747     if sfdecl.sreturnType = Datatype(Void_t)
748     then ignore(build_ret_void llbuilder);
749     ()
750
751 let codegen_vtbl sdecls =
752     let rt = pointer_type i64_t in
753     let void_pt = pointer_type i64_t in
754     let void_ppt = pointer_type void_pt in

```

```

752     let f = func_lookup "lookup" in
753     let llbuilder = builder_at_end context (entry_block f) in
754
755     let len = List.length sdecls in
756     let total_len = ref 0 in
757     let sdecl_llvm_arr = build_array_alloca void_pt (const_int i32_t len) "tmp"
↪ llbuilder in
758
759     let handle_sdecl sdecl =
760         let index = Hashtbl.find Analyzer.struct_indexes sdecl.sname in
761         let len = List.length sdecl.sfuncs in
762         let sfdecl_llvm_arr = build_array_alloca void_pt (const_int i32_t len)
↪ "tmp" llbuilder in
763
764         let handle_fdecl i sfdecl =
765             let fptr = func_lookup (Utils.string_of_fname sfdecl.sfname) in
766             let fptr = build_pointercast fptr void_pt "tmp" llbuilder in
767
768             let ep = build_gep sfdecl_llvm_arr [| (const_int i32_t i) |]
↪ "tmp" llbuilder in
769             ignore(build_store fptr ep llbuilder);
770             in
771             List.iteri handle_fdecl sdecl.sfuncs;
772             total_len := !total_len + len;
773
774             let ep = build_gep sdecl_llvm_arr [| (const_int i32_t index) |] "tmp"
↪ llbuilder in
775             ignore(build_store sfdecl_llvm_arr ep llbuilder);
776             in
777             List.iter handle_sdecl sdecls;
778
779     let c_index = param f 0 in
780     let f_index = param f 1 in
781     set_value_name "c_index" c_index;
782     set_value_name "f_index" f_index;
783
784     if !total_len == 0 then
785         build_ret (const_null rt) llbuilder
786     else
787         let vtbl = build_gep sdecl_llvm_arr [| c_index |] "tmp" llbuilder in
788         let vtbl = build_load vtbl "tmp" llbuilder in
789         let fptr = build_gep vtbl [| f_index |] "tmp" llbuilder in
790         let fptr = build_load fptr "tmp" llbuilder in
791
792         build_ret fptr llbuilder
793
794 let codegen_library_functions () =
795     (* C Std lib functions *)
796     let printf_ty = var_arg_function_type i32_t [| pointer_type i8_t |] in

```

```

797     let _ = declare_function "printf" printf_ty the_module in
798     let malloc_ty = function_type (str_t) [| i32_t |] in
799     let _ = declare_function "malloc" malloc_ty the_module in
800 let open_ty = function_type i32_t [| (pointer_type i8_t); i32_t |] in
801     let _ = declare_function "open" open_ty the_module in
802     let close_ty = function_type i32_t [| i32_t |] in
803     let _ = declare_function "close" close_ty the_module in
804     let read_ty = function_type i32_t [| i32_t; pointer_type i8_t; i32_t |] in
805     let _ = declare_function "read" read_ty the_module in
806     let write_ty = function_type i32_t [| i32_t; pointer_type i8_t; i32_t |] in
807     let _ = declare_function "write" write_ty the_module in
808     let lseek_ty = function_type i32_t [| i32_t; i32_t; i32_t |] in
809     let _ = declare_function "lseek" lseek_ty the_module in
810     let exit_ty = function_type void_t [| i32_t |] in
811     let _ = declare_function "exit" exit_ty the_module in
812         let realloc_ty = function_type str_t [| str_t; i32_t |] in
813         let _ = declare_function "realloc" realloc_ty the_module in
814     let getchar_ty = function_type (i32_t) [| |] in
815     let _ = declare_function "getchar" getchar_ty the_module in
816
817     (* Dice defined functions *)
818     let fty = function_type (pointer_type i64_t) [| i32_t; i32_t |] in
819     let _ = define_function "lookup" fty the_module in
820 let rec_init_ty = function_type void_t [| (pointer_type i64_t); i32_t; (pointer_type
↪ i32_t); (pointer_type i32_t); (pointer_type i32_t); i32_t; i32_t |] in
821     let _ = declare_function "rec_init" rec_init_ty the_module in
822 let init_arr_ty = function_type (pointer_type i64_t) [| (pointer_type i32_t); i32_t
↪ |] in
823     let _ = declare_function "init_arr" init_arr_ty the_module in
824     let input_ty = function_type str_t [| |] in
825     let _ = declare_function "input" input_ty the_module in
826     ()
827
828 let codegen_struct_stub s =
829     let struct_t = named_struct_type context s.scname in
830     Hashtbl.add struct_types s.scname struct_t
831
832 let codegen_struct s =
833     let struct_t = Hashtbl.find struct_types s.scname in
834     let type_list = List.map (function Field(_, d, _) -> get_type d) s.sfields in
835     let name_list = List.map (function Field(_, _, s) -> s) s.sfields in
836
837     (* Add key field to all structs *)
838     let type_list = i32_t :: type_list in
839     let name_list = ".key" :: name_list in
840
841     let type_array = (Array.of_list type_list) in
842     List.iteri (fun i f ->
843         let n = s.scname ^ "." ^ f in

```

```

844     Hashtbl.add struct_field_indexes n i;
845   ) name_list;
846     struct_set_body struct_t type_array true
847
848 let init_args argv args argc llbuilder =
849     let new_block label =
850         let f = block_parent (insertion_block llbuilder) in
851         append_block (global_context ()) label f
852     in
853     let bbcurr = insertion_block llbuilder in
854     let bbcond = new_block "args.cond" in
855     let bbbody = new_block "args.init" in
856     let bbdone = new_block "args.done" in
857     ignore (build_br bbcond llbuilder);
858     position_at_end bbcond llbuilder;
859
860     (Counter into the length of the array)
861     let counter = build_phi [const_int i32_t 0, bbcurr] "counter" llbuilder in
862     add_incoming ((build_add counter (const_int i32_t 1) "tmp" llbuilder), bbbody)
863   ↪ counter;
864     let cmp = build_icmp Icmp.Slt counter argc "tmp" llbuilder in
865     ignore (build_cond_br cmp bbbody bbdone llbuilder);
866     position_at_end bbbody llbuilder;
867
868     (Assign array position to init_val)
869     let arr_ptr = build_gep args [| counter |] "tmp" llbuilder in
870     let argv_val = build_gep argv [| counter |] "tmp" llbuilder in
871     let argv_val = build_load argv_val "tmp" llbuilder in
872     ignore (build_store argv_val arr_ptr llbuilder);
873     ignore (build_br bbcond llbuilder);
874     position_at_end bbdone llbuilder
875
876 let construct_args argc argv llbuilder =
877     let str_pt = pointer_type str_t in
878     let size_real = build_add argc (const_int i32_t 1) "arr_size" llbuilder in
879
880     let arr = build_array_malloc str_pt size_real "args" llbuilder in
881     let arr = build_pointercast arr str_pt "args" llbuilder in
882     let arr_len_ptr = build_pointercast arr (pointer_type i32_t) "argc_len" llbuilder
883   ↪ in
884     let arr_1 = build_gep arr [| const_int i32_t 1 |] "arr_1" llbuilder in
885
886     (Store length at this position)
887     ignore (build_store argc arr_len_ptr llbuilder);
888     ignore (init_args argv arr_1 argc llbuilder);
889     arr
890
891 let codegen_main main =
892     Hashtbl.clear named_values;

```

```
891     Hashtbl.clear named_params;
892     let fty = function_type i32_t [| i32_t; pointer_type str_t |] in
893     let f = define_function "main" fty the_module in
894     let llbuilder = builder_at_end context (entry_block f) in
895
896     let argc = param f 0 in
897     let argv = param f 1 in
898     set_value_name "argc" argc;
899     set_value_name "argv" argv;
900     let args = construct_args argc argv llbuilder in
901     Hashtbl.add named_params "args" args;
902
903     let _ = codegen_stmt llbuilder (SBlock (main.sbody)) in
904     build_ret (const_int i32_t 0) llbuilder
905
906 let linker filename =
907     let llctx = Llvm.global_context () in
908     let llmem = Llvm.MemoryBuffer.of_file filename in
909     let llm = Llvm_bitreader.parse_bitcode llctx llmem in
910     ignore(Llvm_linker.link_modules the_module llm)
911
912 let codegen_sprogram sprogram =
913     let _ = codegen_library_functions () in
914     let _ = List.map (fun s -> codegen_struct_stub s) sprogram.classes in
915     let _ = List.map (fun s -> codegen_struct s) sprogram.classes in
916     let _ = List.map (fun f -> codegen_funcstub f) sprogram.functions in
917     let _ = List.map (fun f -> codegen_func f) sprogram.functions in
918     let _ = codegen_main sprogram.main in
919     let _ = codegen_vtbl sprogram.classes in
920     let _ = linker Conf.bindings_path in
921     the_module
922
923 (* Need to handle assignment of two different types *)
924 (* Need to handle private/public access *)
```

conf.ml

```
1 let bindings_path = "_includes/bindings.bc"
2 let stdlib_path = "_includes/stdlib.dice"
```


dice.ml

```

1  open Llvm
2  open Llvm_analysis
3  open Analyzer
4  open Utils
5  open Ast
6  open Yojson
7  open Exceptions
8  open Filepath
9
10 type action = Tokens | TokenEndl | PrettyPrint | Ast | Sast | Compile | CompileToFile |
    ↪ Help
11
12 let get_action = function
13     "-tendl"      -> TokenEndl
14     | "-t"        -> Tokens
15     | "-p"        -> PrettyPrint
16     | "-ast"      -> Ast
17     | "-sast"     -> Sast
18     | "-h"        -> Help
19     | "-c"        -> Compile
20     | "-f"        -> CompileToFile
21     | _ as s      -> raise (Exceptions.InvalidCompilerArgument s)
22
23 let check_single_argument = function
24     "-h"          -> Help, ""
25     | "-tendl"
26     | "-t"
27     | "-p"
28     | "-ast"
29     | "-sast"
30     | "-c"
31     | "-f"        -> raise (Exceptions.NoFileArgument)
32     | _ as s      -> CompileToFile, s
33
34 let dice_name filename =
35     let basename = Filename.basename filename in
36     let filename = Filename.chop_extension basename in
37     filename ^ ".ll"
38
39 let help_string = (
40     "Usage: dice [optional-option] <source file>\n" ^
41     "optional-option:\n" ^
42     "\t-h: Print help text\n" ^
43     "\t-tendl: Prints tokens with newlines intact\n" ^
44     "\t-t: Prints token stream\n" ^
45     "\t-p: Pretty prints Ast as a program\n" ^
46     "\t-ast: Prints abstract syntax tree as json\n" ^

```

```

47         "\t-sast: Prints semantically checked syntax tree as json\n" ^
48         "\t-c: Compiles source\n" ^
49         "\t-f: Compiles source to file (<filename>.<ext> -> <filename>.ll)\n" ^
50         "Option defaults to \"-f\"\n"
51     )
52
53 let _ =
54     ignore(Printexc.record_backtrace true);
55     try
56         let action, filename =
57             if Array.length Sys.argv = 1 then
58                 Help, ""
59             else if Array.length Sys.argv = 2 then
60                 check_single_argument (Sys.argv.(1))
61             else if Array.length Sys.argv = 3 then
62                 get_action Sys.argv.(1), Sys.argv.(2)
63             else raise (Exceptions.InvalidNumberCompilerArguments
↪ (Array.length Sys.argv))
64         in
65         (* Added fun () -> <x> so that each is evaluated only when requested *)
66         let filename = Filepath.realpath filename in
67         let file_in = fun () -> open_in filename in
68         let lexbuf = fun () -> Lexing.from_channel
↪ (file_in ()) in
69         let token_list = fun () -> Processor.build_token_list (lexbuf
↪ ()) in
70         let program = fun () -> Processor.parser filename (token_list
↪ ()) in
71         let sprogram = fun () -> Analyzer.analyze filename (program ())
↪ in
72         let llm = fun () -> Codegen.codegen_sprogram (sprogram
↪ ()) in
73         (* let _ = Llum_analysis.assert_valid_module llm in *)
74         match action with
75             | Help -> print_string help_string
76             | Tokens -> print_string
↪ (Utils.token_list_to_string (token_list ()))
77             | TokenEndl -> print_string
↪ (Utils.token_list_to_string_endl (token_list ()))
78             | Ast -> print_string (pretty_to_string
↪ (Utils.print_tree (program ())))
79             | Sast -> print_string (pretty_to_string
↪ (Utils.map_sprogram_to_json (sprogram ())))
80             | PrettyPrint -> print_string (Utils.string_of_program
↪ (program ()))
81             | Compile -> dump_module (llm ())
82             | CompileToFile -> print_module (dice_name filename) (llm
↪ ())
83     with

```

```

84         Exceptions.IllegalCharacter(filename, c, ln) ->
85             print_string
86             (
87                 "In \"^ filename ^ \"\", Illegal Character, '\" ^
88                 Char.escaped c ^ '\", line \" ^ string_of_int ln ^ \"\n"
89             )
90         | Exceptions.UnmatchedQuotation(ln)          -> print_endline("Unmatched
↪ Quotation, line \" ^ string_of_int ln)
91         | Exceptions.IllegalToken(tok)              -> print_endline("Illegal
↪ token \" ^ tok)
92         | Exceptions.MissingEOF                      ->
↪ print_endline("Missing EOF")
93         | Parsing.Parse_error ->
94             print_string
95             (
96                 "File \"^ !Processor.filename ^ \"\", \" ^
97                 "line \" ^ string_of_int !Processor.line_number ^ \"\", \" ^
98                 "character \" ^ string_of_int !Processor.char_num ^ \"\", \" ^
99                 "Syntax Error, token \" ^ Utils.string_of_token
↪ !Processor.last_token ^ \"\n"
100             )
101
102         | Exceptions.InvalidNumberCompilerArguments i -> print_endline ("Invalid
↪ argument passed \" ^ (string_of_int i)); print_string help_string
103         | Exceptions.InvalidCompilerArgument s        -> print_endline
↪ ("Invalid argument passed \" ^ s); print_string help_string
104         | Exceptions.NoFileArgument                  ->
↪ print_string ("Must include file argument\n\" ^ help_string)
105
106         | Exceptions.IncorrectNumberOfArgumentsException
↪ -> print_endline("Incorrect number of arguments passed to
↪ function")
107         | Exceptions.ConstructorNotFound(cname)
↪ -> print_endline("Constructor\" ^ cname ^ \"
↪ not found")
108         | Exceptions.DuplicateClassName(cname)
↪ -> print_endline("Class \" ^ cname ^ \" not
↪ found")
109         | Exceptions.DuplicateField
↪ ->
↪ print_endline("Duplicate field defined")
110         | Exceptions.DuplicateFunction(fname)
↪ -> print_endline("Duplicate function defined
↪ \" ^ fname)
111         | Exceptions.DuplicateConstructor
↪ -> print_endline("Duplicate
↪ constructor found")

```

```

112         |           Exceptions.DuplicateLocal(lname)
↪                                     -> print_endline("Duplicate local
↪ variable defined " ^ lname)
113         |           Exceptions.UndefinedClass(cname)
↪                                     -> print_endline("Undefined class " ^
↪ cname)
114         |           Exceptions.UnknownIdentifier(id)
↪                                     -> print_endline("Unkown identifier "
↪ ^ id)
115         |           Exceptions.InvalidBinopExpression(binop)
↪                                     -> print_endline("Invalid binary expression " ^
↪ binop)
116         |           Exceptions.InvalidIfStatementType
↪                                     -> print_endline("Invalid type passed
↪ to if statement, must be bool")
117         |           Exceptions.InvalidForStatementType
↪                                     -> print_endline("Invalid type passed
↪ to for loop, must be bool")
118         |           Exceptions.ReturnTypeMismatch(t1,
↪ t2)
↪                                     -> print_endline("Incorrect return type "
↪ ^ t1 ^ " expected " ^ t2)
119         |           Exceptions.MainNotDefined
↪                                     ->
↪ print_endline("Main not found in program")
120         |           Exceptions.MultipleMainsDefined
↪                                     ->
↪ print_endline("Multiple mains defined, can only define 1")
121         |           Exceptions.InvalidWhileStatementType
↪                                     -> print_endline("Invalid type passed to
↪ while loop, must be bool")
122         |           Exceptions.LocalAssignTypeMismatch(t1, t2)
↪                                     -> print_endline("Invalid assignment of " ^ t1 ^ " to
↪ " ^ t2)
123         |           Exceptions.InvalidUnaryOperation
↪                                     -> print_endline("Invalid unary
↪ operator")
124         |           Exceptions.AssignmentTypeMismatch(t1, t2)
↪                                     -> print_endline("Invalid assignment of " ^ t1 ^ " to
↪ " ^ t2)
125         |           Exceptions.FunctionNotFound(fname, scope)
↪                                     -> print_endline("function " ^ fname ^ " not found in
↪ scope " ^ scope)
126         |           Exceptions.UndefinedID(id)
↪                                     ->
↪ print_endline("Undefined id " ^ id)
127         |           Exceptions.InvalidAccessLHS(t)
↪                                     -> print_endline("Invalid LHS
↪ expression of dot operator with " ^ t)

```

```

128         |           Exceptions.LHSofRootAccessMustBeIDorFunc(lhs)
↪           -> print_endline("Dot operator expects ID, not " ^ lhs)
129         |           Exceptions.ObjAccessMustHaveObjectType(t)
↪           -> print_endline("Can only dereference objects, not "
↪ ^ t)
130         |           Exceptions.UnknownIdentifierForClass(c, id)                               ->
↪ print_endline("Unknown id " ^ id ^ " for class " ^ c)
131         |           Exceptions.CannotUseReservedFuncName(f)
↪           -> print_endline("Cannot use name " ^ f ^ " because
↪ it is reserved")
132         |           Exceptions.InvalidArrayPrimitiveConsecutiveTypes(t1,t2)               ->
↪ print_endline("Array primitive types must be equal, not " ^ t1 ^ " " ^ t2)
133         |           Exceptions.InvalidArrayPrimitiveType(t)
↪           -> print_endline("Array primitive type invalid, " ^
↪ t)
134         |           Exceptions.MustPassIntegerTypeToArrayCreate                               ->
↪ print_endline("Only integer types can be passed to an array initializer")
135         |           Exceptions.ArrayInitTypeInvalid(t)
↪           -> print_endline("Only integer types
↪ can be passed to an array initializer, not " ^ t)
136         |           Exceptions.MustPassIntegerTypeToArrayAccess                               ->
↪ print_endline("Only integer types can be passed to an array access")
137         |           Exceptions.ArrayAccessInvalidParamLength(o,a)
↪           -> print_endline("Only arrays can have access to length, not
↪ " ^ o ^ " " ^ a)
138         |           Exceptions.ArrayAccessExpressionNotArray(a)                             ->
↪ print_endline("This expression is not an array " ^ a)
139         |           Exceptions.CanOnlyAccessLengthOfArray
↪           -> print_endline("Can only access the length
↪ of an array")
140         |           Exceptions.CanOnlyDeleteObjectsOrArrays
↪           -> print_endline("Can only delete objects or arrays")
141         |           Exceptions.CannotAccessLengthOfCharArray
↪           -> print_endline("Cannot access the length of a char
↪ array")
142         |           Exceptions.AllNonVoidFunctionsMustEndWithReturn(f)                   ->
↪ print_endline("Non-void function " ^ f ^ " does not end in return")
143         |           Exceptions.CyclicalDependencyBetween(c1, c2)                           ->
↪ print_endline("Class " ^ c1 ^ " and " ^ c2 ^ " have a cylical dependence")
144         |           Exceptions.CannotAccessPrivateFieldInNonProperScope(f, cp, cc) ->
↪ print_endline("Cannot access private field " ^ f ^ " in scope " ^ cp ^ " from object
↪ " ^ cc)
145         |           Exceptions.CannotCallBreakOutsideOfLoop
↪           -> print_endline("Cannot call break outside of loop")
146         |           Exceptions.CannotCallContinueOutsideOfLoop                               ->
↪ print_endline("Cannot call continue outside of loop")

```

```

147         |           Exceptions.CannotAccessPrivateFunctionInNonProperScope(f, cp, cc) ->
↪ print_endline("Cannot access private function " ^ f ^ " in scope " ^ cp ^ " from
↪ object " ^ cc)
148         |           Exceptions.CannotPassNonInheritedClassesInPlaceOfOthers(c1, c2)
↪         -> print_endline("Cannot pass non-inherited classe" ^ c1 ^ " to parameter " ^
↪ c2)
149         |           Exceptions.IncorrectTypePassedToFunction(id, t)
↪         -> print_endline("Canot pass type " ^ t ^ "
↪ to " ^ id)
150         |           Exceptions.IncorrectNumberOfArguments(f, a1, a2) ->
↪ print_endline("Cannot pass " ^ string_of_int a1 ^ " args when expecting " ^
↪ string_of_int a2 ^ " in " ^ f)
151         |           Exceptions.ClassIsNotExtendedBy(c1, c2) ->
↪ print_endline("Class " ^ c1 ^ " not extended by " ^ c2)
152
153         |           Exceptions.InvalidTypePassedToPrintf ->
↪ print_endline("Invalid type passed to print")
154         |
↪         Exceptions.InvalidBinaryOperator ->
↪ print_endline("Invalid binary operator")
155         |           Exceptions.UnknownVariable(id)
↪         -> print_endline("Unknown variable "
↪ ^ id)
156         |           Exceptions.AssignLHSMustBeAssignable ->
↪ print_endline("Assignment lhs must be assignable")
157         |           Exceptions.CannotCastTypeException(t1, t2) ->
↪ print_endline("Cannot cast " ^ t1 ^ " to " ^ t2)
158         |           Exceptions.InvalidBinopEvaluationType
↪         -> print_endline("Invalid binary expression
↪ evaluation type")
159         |           Exceptions.FloatOpNotSupported
↪         -> print_endline("Float operation not
↪ supported")
160         |           Exceptions.IntOpNotSupported
↪         -> print_endline("Integer operation
↪ not supported")
161         |           Exceptions.LLVMFunctionNotFound(f) ->
↪ print_endline("LLVM function " ^ f ^ " not found")
162         |           Exceptions.InvalidStructType(t)
↪         -> print_endline("Invalid structure type " ^
↪ t)
163         |           Exceptions.UnableToCallFunctionWithoutParent(f) ->
↪ print_endline("Unable to call function " ^ f ^ " without parent")
164         |           Exceptions.CannotAssignParam(p)
↪         -> print_endline("Cannot assign to param " ^
↪ p)
165         |           Exceptions.InvalidUnopEvaluationType ->
↪ print_endline("Invalid unary expression evaluation type")

```

```
166         |           Exceptions.UnopNotSupported
↪                                     -> print_endline("Unary operator not
↪ supported")
167         |           Exceptions.ArrayLargerThan1Unsupported
↪                                     -> print_endline("Array dimensions greater than 1 not
↪ supported")
168         |           Exceptions.CanOnlyCompareObjectsWithNull(e1, e2)           ->
↪ print_endline("Can only compare objects with null " ^ e1 ^ " " ^ e2)
169         |           Exceptions.ObjOpNotSupported(op)
↪                                     -> print_endline("Object operator not
↪ supported " ^ op)
170         |           Exceptions.CanOnlyCompareArraysWithNull(e1, e2)           ->
↪ print_endline("Can only compare arrays with null " ^ e1 ^ " " ^ e2)
```

exceptions.ml

```
1  (* Dice Exceptions *)
2  exception InvalidNumberCompilerArguments of int
3  exception InvalidCompilerArgument of string
4  exception NoFileArgument
5
6  (* Processor Exceptions *)
7  exception MissingEOF
8
9  (* Scanner Exceptions *)
10 exception IllegalCharacter of string * char * int
11 exception UnmatchedQuotation of int
12 exception IllegalToken of string
13
14 (* Analyzer Exceptions *)
15 exception IncorrectNumberOfArgumentsException
16 exception ConstructorNotFound of string
17 exception DuplicateClassName of string
18 exception DuplicateField
19 exception DuplicateFunction of string
20 exception DuplicateConstructor
21 exception DuplicateLocal of string
22 exception UndefinedClass of string
23 exception UnknownIdentifier of string
24 exception InvalidBinopExpression of string
25 exception InvalidIfStatementType
26 exception InvalidForStatementType
27 exception ReturnTypeMismatch of string * string
28 exception MainNotDefined
29 exception MultipleMainsDefined
30 exception InvalidWhileStatementType
31 exception LocalAssignTypeMismatch of string * string
32 exception InvalidUnaryOperation
33 exception AssignmentTypeMismatch of string * string
34 exception FunctionNotFound of string * string
35 exception UndefinedID of string
36 exception InvalidAccessLHS of string
37 exception LHSofRootAccessMustBeIDorFunc of string
38 exception ObjAccessMustHaveObjectType of string
39 exception UnknownIdentifierForClass of string * string
40 exception CannotUseReservedFuncName of string
41 exception InvalidArrayPrimitiveConsecutiveTypes of string * string
42 exception InvalidArrayPrimitiveType of string
43 exception MustPassIntegerTypeToArrayCreate
44 exception ArrayInitTypeInvalid of string
45 exception MustPassIntegerTypeToArrayAccess
46 exception ArrayAccessInvalidParamLength of string * string
47 exception ArrayAccessExpressionNotArray of string
```



```
48 exception CanOnlyAccessLengthOfArray
49 exception CanOnlyDeleteObjectsOrArrays
50 exception CannotAccessLengthOfCharArray
51 exception AllNonVoidFunctionsMustEndWithReturn of string
52 exception CyclicalDependencyBetween of string * string
53 exception CannotAccessPrivateFieldInNonProperScope of string * string * string
54 exception CannotCallBreakOutsideOfLoop
55 exception CannotCallContinueOutsideOfLoop
56 exception CannotAccessPrivateFunctionInNonProperScope of string * string * string
57 exception CannotPassNonInheritedClassesInPlaceOfOthers of string * string
58 exception IncorrectTypePassedToFunction of string * string
59 exception IncorrectNumberOfArguments of string * int * int
60 exception ClassIsNotExtendedBy of string * string
61
62 (* Codegen Exceptions *)
63 exception InvalidTypePassedToPrintf
64 exception InvalidBinaryOperator
65 exception UnknownVariable of string
66 exception AssignLHSMustBeAssignable
67 exception CannotCastTypeException of string * string
68 exception InvalidBinopEvaluationType
69 exception FloatOpNotSupported
70 exception IntOpNotSupported
71 exception LLVMFunctionNotFound of string
72 exception InvalidStructType of string
73 exception UnableToCallFunctionWithoutParent of string
74 exception CannotAssignParam of string
75 exception InvalidUnopEvaluationType
76 exception UnopNotSupported
77 exception ArrayLargerThan1Unsupported
78 exception CanOnlyCompareObjectsWithNull of string * string
79 exception ObjOpNotSupported of string
80 exception CanOnlyCompareArraysWithNull of string * string
```

filepath.ml

```

1  open Filename
2  open Unix
3
4  exception Safe_exception of (string * string list ref)
5
6  let raise_safe fmt =
7    let do_raise msg = raise @@ Safe_exception (msg, ref []) in
8    Printf.ksprintf do_raise fmt
9
10 let reraise_with_context ex fmt =
11   let do_raise context =
12     let () = match ex with
13       | Safe_exception (_, old_contexts) -> old_contexts := context :: !old_contexts
14       | _ -> Printf.eprintf "warning: Attempt to add note '%s' to non-Safe_exception!"
15         ↪ context
16     in
17     raise ex
18   in Printf.ksprintf do_raise fmt
19
20 module StringMap = struct
21   include Map.Make(String)
22   let find_nf = find
23   let find_safe key map = try find key map with Not_found -> raise_safe "BUG: Key '%s'
24     ↪ not found in StringMap!" key
25   let find key map = try Some (find key map) with Not_found -> None
26   let map_bindings fn map = fold (fun key value acc -> fn key value :: acc) map []
27 end
28
29 type path_component =
30   | Filename of string (* foo/ *)
31   | ParentDir   (* ../ *)
32   | CurrentDir  (* ./ *)
33   | EmptyComponent (* / *)
34
35 type filepath = string
36
37 let on_windows = Filename.dir_sep <> "/"
38
39 let path_is_absolute path = not (Filename.is_relative path)
40
41 let string_tail s i =
42   let len = String.length s in
43   if i > len then failwith ("String '" ^ s ^ "' too short to split at " ^ (string_of_int
44     ↪ i))
45   else String.sub s i (len - i)

```

```
45 let split_path_str path =
46   let l = String.length path in
47   let is_sep c = (c = '/' || (on_windows && c = '\\')) in
48
49   (* Skip any leading slashes and return the rest *)
50   let rec find_rest i =
51     if i < l then (
52       if is_sep path.[i] then find_rest (i + 1)
53       else string_tail path i
54     ) else (
55       ""
56     ) in
57
58   let rec find_slash i =
59     if i < l then (
60       if is_sep path.[i] then (String.sub path 0 i, find_rest (i + 1))
61       else find_slash (i + 1)
62     ) else (
63       (path, "")
64     )
65   in
66   find_slash 0
67
68 let split_first path =
69   if path = "" then
70     (CurrentDir, "")
71   else (
72     let (first, rest) = split_path_str path in
73     let parsed =
74       if first = Filename.parent_dir_name then ParentDir
75       else if first = Filename.current_dir_name then CurrentDir
76       else if first = "" then EmptyComponent
77       else Filename first in
78     (parsed, rest)
79   )
80
81 let normpath path : filepath =
82   let rec explode path =
83     match split_first path with
84     | CurrentDir, "" -> []
85     | CurrentDir, rest -> explode rest
86     | first, "" -> [first]
87     | first, rest -> first :: explode rest in
88
89   let rec remove_parents = function
90     | checked, [] -> checked
91     | (Filename _name :: checked), (ParentDir :: rest) -> remove_parents (checked, rest)
92     | checked, (first :: rest) -> remove_parents ((first :: checked), rest) in
93
```

```

94   let to_string = function
95     | Filename name -> name
96     | ParentDir -> Filename.parent_dir_name
97     | EmptyComponent -> ""
98     | CurrentDir -> assert false in
99   String.concat Filename.dir_sep @@ List.rev_map to_string @@ remove_parents ([], explode
↳ path)
100
101
102 let abspath path =
103   let (+/) = Filename.concat in
104   normpath (
105     if path_is_absolute path then path
106     else (Sys.getcwd ()) +/ path
107   )
108
109 let realpath path =
110   let (+/) = Filename.concat in   (* Faster version, since we know the path is relative
↳ *)
111
112   (* Based on Python's version *)
113   let rec join_realpath path rest seen =
114     (* Printf.printf "join_realpath <%s> + <%s>\n" path rest; *)
115     (* [path] is already a realpath (no symlinks). [rest] is the bit to join to it. *)
116     match split_first rest with
117     | Filename name, rest -> (
118       (* path + name/rest *)
119       let newpath = path +/ name in
120       let link = try Some (Unix.readlink newpath) with Unix.Unix_error _ -> None in
121       match link with
122       | Some target ->
123         (* path + symlink/rest *)
124         begin match StringMap.find newpath seen with
125         | Some (Some cached_path) -> join_realpath cached_path rest seen
126         | Some None -> (normpath (newpath +/ rest), false)   (* Loop; give up *)
127         | None ->
128           (* path + symlink/rest -> realpath(path + target) + rest *)
129           match join_realpath path target (StringMap.add newpath None seen) with
130           | path, false ->
131             (normpath (path +/ rest), false)   (* Loop; give up *)
132           | path, true -> join_realpath path rest (StringMap.add newpath (Some path)
↳ seen)
133         end
134       | None ->
135         (* path + name/rest -> path/name + rest (name is not a symlink) *)
136         join_realpath newpath rest seen
137     )
138     | CurrentDir, "" ->
139       (path, true)

```

```

140 | CurrentDir, rest ->
141   (* path + ./rest *)
142   join_realpath path rest seen
143 | ParentDir, rest ->
144   (* path + ../rest *)
145   if String.length path > 0 then (
146     let name = Filename.basename path in
147     let path = Filename.dirname path in
148     if name = Filename.parent_dir_name then
149       join_realpath (path + "/" + name) rest seen      (* path/.. + ../rest ->
↳ path/../../.. + rest *)
150     else
151       join_realpath path rest seen                      (* path/name + ../rest ->
↳ path + rest *)
152   ) else (
153     join_realpath Filename.parent_dir_name rest seen    (* "" + ../rest -> .. + rest
↳ *)
154   )
155 | EmptyComponent, rest ->
156   (* [rest] is absolute; discard [path] and start again *)
157   join_realpath Filename.dir_sep rest seen
158 in
159
160 try
161   if on_windows then
162     abspath path
163   else (
164     fst @@ join_realpath (Sys.getcwd ()) path StringMap.empty
165   )
166 with Safe_exception _ as ex -> reraise_with_context ex "... in realpath(%s)" path

```

parser.mly

```

1  %{  open Ast  %}
2
3  %token CLASS EXTENDS CONSTRUCTOR INCLUDE DOT THIS PRIVATE PUBLIC
4  %token INT FLOAT BOOL CHAR VOID NULL TRUE FALSE
5  %token SEMI LPAREN RPAREN LBRACE RBRACE LBRACKET RBRACKET COMMA
6  %token AND NOT OR PLUS MINUS TIMES DIVIDE ASSIGN MODULO
7  %token EQ NEQ LT LEQ GT GEQ BAR
8  %token RETURN IF ELSE FOR WHILE BREAK CONTINUE NEW DELETE
9  %token <int> INT_LITERAL
10 %token <float> FLOAT_LITERAL
11 %token <string> STRING_LITERAL
12 %token <string> ID
13 %token <char> CHAR_LITERAL
14 %token EOF
15
16 %nonassoc NOELSE
17 %nonassoc ELSE
18 %right ASSIGN
19 %left AND OR
20 %left EQ NEQ
21 %left LT GT LEQ GEQ
22 %left PLUS MINUS
23 %left TIMES DIVIDE MODULO
24 %right NOT
25 %right DELETE
26 %right RBRACKET
27 %left LBRACKET
28 %right DOT
29
30 %start program
31 %type <Ast.program> program
32
33 %%
34
35 program:
36     includes cdecls EOF { Program($1, $2) }
37
38     /*****
39         INCLUDE
40     *****/
41
42 includes:
43     /* nothing */ { [] }
44     | include_list { List.rev $1 }
45
46 include_list:
47     include_decl { [$1] }

```

```

48         |           include_list include_decl { $2::$1 }
49
50 include_decl:
51     INCLUDE LPAREN STRING_LITERAL RPAREN SEMI { Include($3) }
52
53
54 /*****
55 CLASSES
56 *****/
57 cdecls:
58     cdecl_list    { List.rev $1 }
59
60 cdecl_list:
61     cdecl          { [$1] }
62     | cdecl_list cdecl { $2::$1 }
63
64 cdecl:
65     CLASS ID LBRACE cbody RBRACE { {
66         cname = $2;
67         extends = NoParent;
68         cbody = $4
69     } }
70     | CLASS ID EXTENDS ID LBRACE cbody RBRACE { {
71         cname = $2;
72         extends = Parent($4);
73         cbody = $6
74     } }
75
76 cbody:
77     /* nothing */ { {
78         fields = [];
79         constructors = [];
80         methods = [];
81     } }
82     | cbody field { {
83         fields = $2 :: $1.fields;
84         constructors = $1.constructors;
85         methods = $1.methods;
86     } }
87     | cbody constructor { {
88         fields = $1.fields;
89         constructors = $2 :: $1.constructors;
90         methods = $1.methods;
91     } }
92     | cbody fdecl { {
93         fields = $1.fields;
94         constructors = $1.constructors;
95         methods = $2 :: $1.methods;
96     } }

```

```

97
98
99  /*****
100  CONSTRUCTORS
101  *****/
102
103  constructor:
104      CONSTRUCTOR LPAREN formals_opt RPAREN LBRACE stmt_list RBRACE {
105          {
106              scope = Public;
107              fname = Constructor;
108              returnType = Datatype(ConstructorType);
109              formals = $3;
110              body = List.rev $6;
111              overrides = false;
112              root_cname = None;
113          }
114      }
115
116  /*****
117  FIELDS
118  *****/
119
120  scope:
121      PRIVATE { Private }
122      |      PUBLIC { Public }
123
124  /* public UserObj name; */
125  field:
126      scope datatype ID SEMI { Field($1, $2, $3) }
127
128  /*****
129  METHODS
130  *****/
131
132  fname:
133      ID { $1 }
134
135  fdecl:
136      scope datatype fname LPAREN formals_opt RPAREN LBRACE stmt_list RBRACE
137      {
138          {
139              scope = $1;
140              fname = FName($3);
141              returnType = $2;
142              formals = $5;
143              body = List.rev $8;
144              overrides = false;
145              root_cname = None;

```



```

146         }
147     }
148
149     /*****
150     FORMALS/PARAMETERS & VARIABLES & ACTUALS
151     *****/
152
153     formals_opt:
154         /* nothing */ { [] }
155         |
156         formal_list { List.rev $1 }
157
158     formal_list:
159         formal { [$1] }
160         |
161         formal_list COMMA formal { $3 :: $1 }
162
163     formal:
164         datatype ID { Formal($1, $2) }
165
166     actuals_opt:
167         /* nothing */ { [] }
168         |
169         actuals_list { List.rev $1 }
170
171     actuals_list:
172         expr { [$1] }
173         |
174         actuals_list COMMA expr { $3 :: $1 }
175
176     /*****
177     DATATYPES
178     *****/
179
180     primitive:
181         INT { Int_t }
182         |
183         FLOAT { Float_t }
184         |
185         CHAR { Char_t }
186         |
187         BOOL { Bool_t }
188         |
189         VOID { Void_t }
190
191     name:
192         CLASS ID { Objecttype($2) }
193
194     type_tag:
195         primitive { $1 }
196         |
197         name { $1 }
198
199     array_type:
200         type_tag LBRACKET brackets RBRACKET { Arraytype($1, $3) }
201
202     datatype:
203         type_tag { Datatype($1) }

```

```

195         |          array_type { $1 }
196
197 brackets:
198         /* nothing */ { 1 }
199         |          brackets RBRACKET LBRACKET { $1 + 1 }
200
201 /*****
202  EXPRESSIONS
203 *****/
204
205 stmt_list:
206         /* nothing */ { [] }
207         | stmt_list stmt { $2 :: $1 }
208
209 stmt:
210         expr SEMI { Expr($1) }
211         | RETURN expr SEMI { Return($2) }
212         | RETURN SEMI { Return(Noexpr) }
213         | LBRACE stmt_list RBRACE { Block(List.rev $2) }
214         | IF LPAREN expr RPAREN stmt %prec NOELSE { If($3, $5,
↪ Block([Expr(Noexpr)])) }
215         | IF LPAREN expr RPAREN stmt ELSE stmt { If($3, $5, $7) }
216         | FOR LPAREN expr_opt SEMI expr_opt SEMI expr_opt RPAREN stmt
217         { For($3, $5, $7, $9) }
218         | WHILE LPAREN expr RPAREN stmt { While($3, $5) }
219         | BREAK SEMI { Break }
220         | CONTINUE SEMI { Continue }
221         | datatype ID SEMI { Local($1, $2, Noexpr) }
222         | datatype ID ASSIGN expr SEMI { Local($1, $2, $4) }
223
224 expr_opt:
225         /* nothing */ { Noexpr }
226         | expr { $1 }
227
228 expr:
229         literals { $1 }
230         | expr PLUS expr { Binop($1,
↪ Add, $3) }
231         | expr MINUS expr { Binop($1,
↪ Sub, $3) }
232         | expr TIMES expr { Binop($1,
↪ Mult, $3) }
233         | expr DIVIDE expr { Binop($1,
↪ Div, $3) }
234         | expr EQ expr { Binop($1,
↪ Equal, $3) }
235         | expr NEQ expr { Binop($1,
↪ Neq, $3) }

```

```

236      |      expr LT      expr      { Binop($1,
↪ Less,  $3) }
237      |      expr LEQ    expr      { Binop($1,
↪ Leq,   $3) }
238      |      expr GT      expr      { Binop($1,
↪ Greater, $3) }
239      |      expr GEQ     expr      { Binop($1,
↪ Geq,    $3) }
240      |      expr AND     expr      { Binop($1,
↪ And,    $3) }
241      |      expr MODULO  expr      { Binop($1,
↪ Mod,    $3) }
242      |      NOT expr      {
↪ Unop (Not, $2) }
243      |      expr OR      expr      { Binop($1,
↪ Or,     $3) }
244      |      expr DOT     expr      {
↪ ObjAccess($1, $3) }
245      |      expr ASSIGN  expr      { Assign($1,
↪ $3) }
246      |      DELETE expr      {
↪ Delete($2) }
247      |      MINUS expr      { Unop
↪ (Sub, $2) }
248      |      ID LPAREN actuals_opt RPAREN      { Call($1, $3) }
249      |      NEW ID LPAREN actuals_opt RPAREN    { ObjectCreate($2, $4) }
250      |      NEW type_tag bracket_args RBRACKET { ArrayCreate(Datatype($2),
↪ List.rev $3) }
251      |      expr bracket_args RBRACKET          { ArrayAccess($1,
↪ List.rev $2) }
252      |      LPAREN expr RPAREN                  { $2 }
253
254 bracket_args:
255      LBRACKET expr                                { [$2] }
256      |      bracket_args RBRACKET LBRACKET expr { $4 :: $1 }
257
258 literals:
259      INT_LITERAL                                { Int_Lit($1) }
260      |      FLOAT_LITERAL                       { Float_Lit($1) }
261      |      TRUE                                { Boolean_Lit(true) }
262      |      FALSE                               { Boolean_Lit(false) }
263      |      STRING_LITERAL                      { String_Lit($1) }
264      |      CHAR_LITERAL                       { Char_Lit($1) }
265      |      THIS                                { This }
266      |      ID                                  { Id($1) }
267      |      NULL                                { Null }
268      |      BAR array_prim BAR                  { ArrayPrimitive($2) }
269
270 /* ARRAY LITERALS */

```

```
271
272 array_prim:
273     expr                                { [$1] }
274     | array_prim COMMA expr            { $3 :: $1 }
```

processor.ml

```

1  open Parser
2
3  type token_attr = {
4    lineno: int;
5    cnum: int;
6  }
7
8  let line_number = ref 1
9  let last_token = ref EOF
10 let char_num = ref 1
11 let filename = ref ""
12
13 let build_token_list lexbuf =
14     Scanner.filename := !filename;
15     let rec helper prev_cnum prev_lineno lexbuf token_list =
16         let token = Scanner.token lexbuf in
17         let lineno = !Scanner.lineno in
18         let cnum = (Lexing.lexeme_start_p lexbuf).Lexing.pos_cnum in
19         let prev_cnum = if lineno > prev_lineno then cnum else prev_cnum in
20         let cnum = cnum - prev_cnum in
21         match token with
22         | EOF as eof -> (eof, { lineno = lineno; cnum = cnum } )::token_list
23         | t           -> (t, { lineno = lineno; cnum = cnum } )::(helper prev_cnum lineno
24 ↪ lexbuf token_list)
25     in helper 0 0 lexbuf []
26
27 let parser file token_list =
28     let token_list = ref(token_list) in
29     let tokenizer _ =
30         match !token_list with
31         | (head, curr) :: tail ->
32             filename := file;
33             line_number := curr.lineno;
34             char_num := curr.cnum;
35             last_token := head;
36             token_list := tail;
37             head
38         | [] -> raise (Exceptions.MissingEOF)
39     in
40     let program = Parser.program tokenizer (Lexing.from_string "") in
41     program

```

sast.ml

```

1  open Ast
2
3  type sexpr =
4      SInt_Lit of int
5      | SBoolean_Lit of bool
6      | SFloat_Lit of float
7      | SString_Lit of string
8      | SChar_Lit of char
9      | SId of string * datatype
10     | SBinop of sexpr * op * sexpr * datatype
11     | SAssign of sexpr * sexpr * datatype
12     | SNoexpr
13     | SArrayCreate of datatype * sexpr list * datatype
14     | SArrayAccess of sexpr * sexpr list * datatype
15     | SObjAccess of sexpr * sexpr * datatype
16     | SCall of string * sexpr list * datatype * int
17     | SObjectCreate of string * sexpr list * datatype
18     | SArrayPrimitive of sexpr list * datatype
19     | SUnop of op * sexpr * datatype
20     | SNull
21     | SDelete of sexpr
22
23  type sstmt =
24      SBlock of sstmt list
25      | SExpr of sexpr * datatype
26      | SReturn of sexpr * datatype
27      | SIf of sexpr * sstmt * sstmt
28      | SFor of sexpr * sexpr * sexpr * sstmt
29      | SWhile of sexpr * sstmt
30      | SBreak
31      | SContinue
32      | SLocal of datatype * string * sexpr
33
34  type func_type = User | Reserved
35
36  type sfunc_decl = {
37      sfname : fname;
38      sreturnType : datatype;
39      sformals : formal list;
40      sbody : sstmt list;
41      func_type : func_type;
42      source : string;
43      overrides : bool;
44  }
45
46  type sclass_decl = {
47      scname : string;

```

```
48         sfields : field list;
49         sfuncs: sfunc_decl list;
50     }
51
52     (* Class Declarations / All method declarations / Main entry method *)
53     type sprogram = {
54         classes : sclass_decl list;
55         functions : sfunc_decl list;
56         main : sfunc_decl;
57         reserved : sfunc_decl list;
58     }
```

scanner.mll

```

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

```

```

48 | ">="      { GEQ }
49 | "and"     { AND }
50 | "or"      { OR }
51 | "not"     { NOT }
52 | ' .'      { DOT }
53 | '['       { LBRACKET }
54 | ']'       { RBRACKET }
55 | '|'       { BAR }
56
57 (* Branch Control *)
58 | "if"      { IF }
59 | "else"    { ELSE }
60 | "for"     { FOR }
61 | "while"   { WHILE }
62 | "return"  { RETURN }
63
64 (* Data Types *)
65 | "int"     { INT }
66 | "float"   { FLOAT }
67 | "bool"    { BOOL }
68 | "char"    { CHAR }
69 | "void"    { VOID }
70 | "null"    { NULL }
71 | "true"    { TRUE }
72 | "false"   { FALSE }
73
74 (* Classes *)
75 | "class"    { CLASS }
76 | "constructor" { CONSTRUCTOR }
77 | "public"   { PUBLIC }
78 | "private"  { PRIVATE }
79 | "extends"  { EXTENDS }
80 | "include"  { INCLUDE }
81 | "this"     { THIS }
82 | "break"    { BREAK }
83 | "continue" { CONTINUE }
84 | "new"      { NEW }
85 | "delete"   { DELETE }
86
87 | int as lxm          { INT_LITERAL(int_of_string lxm) }
88 | float as lxm        { FLOAT_LITERAL(float_of_string lxm) }
89 | char as lxm         { CHAR_LITERAL( String.get lxm 1 ) }
90 | escape_char as lxm { CHAR_LITERAL( String.get (unescape lxm) 1) }
91 | string              { STRING_LITERAL(unescape s) }
92 | id as lxm           { ID(lxm) }
93 | eof                 { EOF }
94
95 | '""'               { raise (Exceptions.UnmatchedQuotation(!lineno)) }
96 | _ as illegal      { raise (Exceptions.IllegalCharacter(!filename, illegal, !lineno)) }

```

```
97
98 and comment = parse
99     return      { incr lineno; comment lexbuf }
100    |      "*" "    { decr depth; if !depth > 0 then comment lexbuf else
↪ token lexbuf }
101    |      "(" "    { incr depth; comment lexbuf }
102    |      _      { comment lexbuf }
```

stdlib.dice

```
1  class Integer {
2
3      private int my_int;
4
5      constructor(int input) {
6          this.my_int = input;
7      }
8
9      public int num() {
10         return this.my_int;
11     }
12
13
14     public char toChar(int digit) {
15
16         if (digit == 0) {
17             return '0';
18         } else if (digit == 1) {
19             return '1';
20         } else if (digit == 2) {
21             return '2';
22         } else if (digit == 3) {
23             return '3';
24         } else if (digit == 4) {
25             return '4';
26         } else if (digit == 5) {
27             return '5';
28         } else if (digit == 6) {
29             return '6';
30         } else if (digit == 7) {
31             return '7';
32         } else if (digit == 8) {
33             return '8';
34         } else if (digit == 9) {
35             return '9';
36         }
37
38         return 'z';
39     }
40
41
42
43
44
45     public class String toString() {
46
47         (* integer cannot be greater than 10 digits in 32 bit *)
```

```
48     int temp = this.my_int;
49     int i = 0;
50     char[] str = new char[9];
51
52     int digit = temp % 10;
53     str[i] = this.toChar(digit);
54     i = i + 1;
55     temp = temp / 10;
56     while (temp > 0) {
57
58         digit = temp % 10;
59         str[i] = this.toChar(digit);
60         temp = temp / 10;
61         i = i + 1;
62     }
63
64     str[i] = 0;
65     class String newString = new String(str);
66     class String a = newString.reverse();
67     return newString.reverse();
68 }
69 }
70
71
72
73 class String {
74
75     private char[] my_string;
76     private int length;
77
78     constructor(char[] input) {
79
80         this.my_string = this.copy_internal(input);
81
82         this.length = this.length();
83     }
84
85     (* PRIVATE CLASSES ----- *)
86
87     private int length_internal(char[] input) {
88         int length = 0;
89
90         while(input[length] != 0) {
91             length = length + 1;
92         }
93
94         return length;
95     }
96 }
```

```
97     private char[] copy_internal(char[] input) {
98
99         char[] newString = new char[this.length_internal(input) + 1];
100
101         int i = 0;
102         for (; input[i] != 0; i = i + 1) {
103             newString[i] = input[i];
104         }
105
106         newString[i] = 0;
107         return newString;
108     }
109
110     (* PUBLIC CLASSES ----- *)
111
112     public char[] string() {
113         return this.my_string;
114     }
115
116     public char getChar(int index) {
117
118         return this.my_string[index];
119     }
120
121     public int length() {
122
123         int length = 0;
124
125         while(this.my_string[length] != 0){
126             length = length + 1;
127         }
128
129         return length;
130     }
131
132     public int toInteger() {
133
134         char[] temp = this.string();
135         int ndigit = 0;
136         int i;
137         int j;
138         for (i = 0; i < this.length; i = i + 1) {
139
140             int exp = 1;
141             int xdigit = this.toDigit(temp[i]);
142             for (j = 0; j < (this.length-i-1); j = j + 1) {
143                 exp = exp * 10;
144             }
145             xdigit = xdigit * exp;
```

```
146         ndigit = ndigit + xdigit;
147     }
148
149     return ndigit;
150 }
151
152 public int toDigit(char digit) {
153
154     if (digit == '0') {
155         return 0;
156     } else if (digit == '1') {
157         return 1;
158     } else if (digit == '2') {
159         return 2;
160     } else if (digit == '3') {
161         return 3;
162     } else if (digit == '4') {
163         return 4;
164     } else if (digit == '5') {
165         return 5;
166     } else if (digit == '6') {
167         return 6;
168     } else if (digit == '7') {
169         return 7;
170     } else if (digit == '8') {
171         return 8;
172     } else if (digit == '9') {
173         return 9;
174     }
175
176     return -1;
177 }
178
179
180 public class String copy(class String input) {
181
182     char[] newArray = this.copy_internal(input.string());
183     class String newString = new String(newArray);
184     return newString;
185 }
186
187 public int indexOf(char x) {
188
189     int i = 0;
190     for (; this.getChar(i) != x and this.getChar(i) != 0; i = i + 1) {
191     }
192
193     (* If the char was not found, return -1 *)
194     if (i == this.length()) {
```

```
195         return -1;
196     }
197
198     return i;
199 }
200
201 public class String reverse() {
202
203     class String newString;
204
205     char[] temp = new char[this.length + 1];
206     int i = this.length;
207     for (; i > 0; i = i - 1) {
208
209         temp[this.length - i] = this.getChar(i-1);
210     }
211     temp[this.length] = 0;
212     newString = new String(temp);
213     return newString;
214 }
215
216 public class String concat(class String temp) {
217
218     char[] temparray = new char[this.length() + temp.length() + 1];
219
220     (* Copy over the current string into a new char array *)
221     int i = 0;
222     for (; this.getChar(i) != 0; i = i + 1) {
223         temparray[i] = this.getChar(i);
224     }
225
226     (* Append the new string *)
227     int j = 0;
228     for (; temp.getChar(j) != 0; j = j + 1) {
229         temparray[i+j] = temp.getChar(j);
230     }
231
232     temparray[this.length() + temp.length()] = 0;
233     class String newString = new String(temparray);
234     return newString;
235 }
236
237 public bool compare(class String check) {
238
239     if (check.length != this.length) {
240         return false;
241     }
242
243     int i = 0;
```

```
244
245     for (; i < check.length(); i = i + 1) {
246
247         if (check.getChar(i) != this.getChar(i)) {
248             return false;
249         }
250     }
251
252     return true;
253 }
254
255 public bool contains(class String check) {
256
257
258     if (this.length < check.length) {
259         return false;
260     } else if (this.compare(check)) {
261         return true;
262     } else {
263
264         int diff = this.length - check.length + 1;
265         int i;
266         int j;
267         for ( i = 0; i < diff; i = i + 1)
268
269             for ( j = 0; j < check.length; j = j + 1) {
270
271                 if (this.getChar(i+j) != check.getChar(j)) {
272                     break;
273                 }
274
275                 if (j == check.length - 1) {
276                     return true;
277                 }
278             }
279         }
280     return false;
281 }
282
283 public void free() {
284
285     delete(this.my_string);
286 }
287
288 }
289
290
291
292 class File {
```



```
293
294 private class String filePath;
295 private bool isWriteEnabled;
296 private int fd;
297
298 constructor(char[] path, bool isWriteEnabled) {
299
300     this.filePath = new String(path);
301     this.isWriteEnabled = isWriteEnabled;
302     class String a = this.filePath;
303     this.fd = this.openfile(a, this.isWriteEnabled);
304     if (this.fd < 0) {
305         print("open failed");
306         exit(1);
307     }
308 }
309
310 (* PRIVATE CLASSES ----- *)
311
312 private int openfile(class String path, bool isWriteEnabled) {
313
314     if (isWriteEnabled) {
315         (* 2 is the value for O_RDWR *)
316         return open(path.string(), 2);
317     }
318
319     (* 0 is the value for O_RDONLY *)
320     return open(path.string(), 0);
321 }
322
323 (* PUBLIC CLASSES ----- *)
324
325 public void closefile() {
326
327     if (close(this.fd) < 0) {
328         print("close failed");
329     }
330 }
331
332 public char[] readfile(int bytes) {
333
334     char[] buf = new char[bytes];
335
336     int ret = read(this.fd, buf, bytes);
337
338     if (ret < 0) {
339         print("read failed");
340     }
341 }
```

```
342     return buf;
343 }
344
345 public int writefile(char[] buf, int offset) {
346
347     class String temp = new String(buf);
348     int err;
349     (* seek to desired offset from beginning of file *)
350     if (offset > 0) {
351         err = lseek(this.fd, offset, 0);
352     } else if (offset == -1) {
353         err = lseek(this.fd, 0, 0);
354     } else {
355         (* Seek to the end of the file by default *)
356         err = lseek(this.fd, 0, 2);
357     }
358
359     if (err < 0) {
360         print("seek failed");
361     }
362
363     err = write(this.fd, temp.string(), temp.length());
364     if (err < 0) {
365         print("write failed");
366     }
367     return err;
368 }
369
370 }
```

utils.ml

```

1  (* Pretty Printer *)
2  open Ast
3  open Sast
4  open Parser
5  open Processor
6  open Yojson
7
8  let save file string =
9      let channel = open_out file in
10     output_string channel string;
11     close_out channel
12
13  let replace input output =
14      Str.global_replace (Str.regexp_string input) output
15
16  (* Print data types *)
17
18  let string_of_scope = function
19      Public      -> "public"
20      | Private -> "private"
21
22  let string_of_primitive = function
23      Int_t          -> "int"
24      | Float_t      -> "float"
25      | Void_t       -> "void"
26      | Bool_t       -> "bool"
27      | Char_t       -> "char"
28      | Objecttype(s) -> "class " ^ s
29      | ConstructorType -> "constructor"
30      | Null_t       -> "null"
31
32  let string_of_object = function
33      Datatype(Objecttype(s)) -> s
34      | _ -> ""
35
36  let rec print_brackets = function
37      1 -> "["
38      | a -> "[" ^ print_brackets (a - 1)
39
40  let string_of_datatype = function
41      Arraytype(p, i) -> (string_of_primitive p) ^ (print_brackets i)
42      | Datatype(p) -> (string_of_primitive p)
43      | Any -> "Any"
44
45  (* Print expressions *)
46
47  let string_of_op = function

```

```

48         Add                                -> "+"
49     |     Sub                                -> "-"
50     |     Mult                               -> "*"
51     |     Div                                -> "/"
52     |     Equal                             -> "=="
53     |     Neq                                -> "!="
54     |     Less                               -> "<"
55     |     Leq                                -> "<="
56     |     Greater                            -> ">"
57     |     Geq                                -> ">="
58     |     And                                -> "and"
59     |     Not                                -> "not"
60     |     Or                                 -> "or"
61     |     Mod                                -> "%"
62
63 let rec string_of_bracket_expr = function
64     []                                     -> ""
65     | head :: tail                         -> "[" ^ (string_of_expr head) ^ "]" ^
    ↪ (string_of_bracket_expr tail)
66 and string_of_array_primitive = function
67     []                                     -> ""
68     | [last]                               -> (string_of_expr last)
69     | head :: tail                         -> (string_of_expr head) ^ ", " ^
    ↪ (string_of_array_primitive tail)
70 and string_of_expr = function
71     Int_Lit(i)                             -> string_of_int i
72     | Boolean_Lit(b)                       -> if b then "true" else "false"
73     | Float_Lit(f)                         -> string_of_float f
74     | String_Lit(s)                        -> "\"" ^ (String.escaped s) ^ "\""
75     | Char_Lit(c)                          -> Char.escaped c
76     | This                                 -> "this"
77     | Id(s)                                -> s
78     | Binop(e1, o, e2)                     -> (string_of_expr e1) ^ " " ^
    ↪ (string_of_op o) ^ " " ^ (string_of_expr e2)
79     | Assign(e1, e2)                       -> (string_of_expr e1) ^ " = " ^
    ↪ (string_of_expr e2)
80     | Noexpr                               -> ""
81     | ObjAccess(e1, e2)                   -> (string_of_expr e1) ^ "." ^
    ↪ (string_of_expr e2)
82     | Call(f, el)                         -> f ^ "(" ^ String.concat ",
    ↪ " (List.map string_of_expr el) ^ ")"
83     | ArrayPrimitive(el)                  -> "[" ^ (string_of_array_primitive
    ↪ el) ^ "]"
84     | Unop(op, e)                         -> (string_of_op op) ^ "("
    ↪ ^ string_of_expr e ^ ")"
85     | Null                                 -> "null"
86     | ArrayCreate(d, el)                  -> "new " ^ string_of_datatype d ^
    ↪ string_of_bracket_expr el

```

```

87      |   ArrayAccess(e, el)          -> (string_of_expr e) ^ (string_of_bracket_expr
↪   el)
88      |   ObjectCreate(s, el)         -> "new " ^ s ^ "(" ^ String.concat ", "
↪   (List.map string_of_expr el) ^ ")"
89      |   Delete(e)                   -> "delete (" ^
↪   (string_of_expr e) ^ ")"
90  ;;
91
92  let rec string_of_bracket_sexpr = function
93      []                               -> ""
94      |   head :: tail                 -> "[" ^ (string_of_sexpr head) ^ "]" ^
↪   (string_of_bracket_sexpr tail)
95  and string_of_sarray_primitive = function
96      []                               -> ""
97      |   [last]                      -> (string_of_sexpr last)
98      |   head :: tail                 -> (string_of_sexpr head) ^ ", " ^
↪   (string_of_sarray_primitive tail)
99  and string_of_sexpr = function
100      |   SInt_Lit(i)                  -> string_of_int i
101      |   SBoolean_Lit(b)              -> if b then "true" else
↪   "false"
102      |   SFloat_Lit(f)                -> string_of_float f
103      |   SString_Lit(s)               -> "\"" ^ (String.escaped
↪   s) ^ "\""
104      |   SChar_Lit(c)                 -> Char.escaped c
105      |   SId(s, _)                    -> s
106      |   SBinop(e1, o, e2, _)         -> (string_of_sexpr e1) ^ " " ^
↪   (string_of_op o) ^ " " ^ (string_of_sexpr e2)
107      |   SAssign(e1, e2, _)           -> (string_of_sexpr e1) ^ " =
↪   " ^ (string_of_sexpr e2)
108      |   SNoexpr                      -> ""
109      |   SObjAccess(e1, e2, _)        -> (string_of_sexpr e1) ^ "." ^
↪   (string_of_sexpr e2)
110      |   SCall(f, el, _, _)           -> f ^ "(" ^ String.concat ", "
↪   " (List.map string_of_sexpr el) ^ ")"
111      |   SArrayPrimitive(el, _)       -> "|" ^
↪   (string_of_sarray_primitive el) ^ "|"
112      |   SUNop(op, e, _)              -> (string_of_op op) ^
↪   "(" ^ string_of_sexpr e ^ ")"
113      |   SNull                        -> "null"
114      |   SArrayCreate(d, el, _)       -> "new " ^ string_of_datatype d ^
↪   string_of_bracket_sexpr el
115      |   SArrayAccess(e, el, _)       -> (string_of_sexpr e) ^
↪   (string_of_bracket_sexpr el)
116      |   SObjectCreate(s, el, _)      -> "new " ^ s ^ "(" ^ String.concat ", "
↪   (List.map string_of_sexpr el) ^ ")"
117      |   SDelete(e)                   -> "delete (" ^
↪   (string_of_sexpr e) ^ ")"
118  ;;

```

```

119
120 let string_of_local_expr = function
121     Noexpr -> ""
122     |      e      -> " = " ^ string_of_expr e
123
124 (* Print statements *)
125
126 let rec string_of_stmt indent =
127     let indent_string = String.make indent '\t' in
128     let get_stmt_string = function
129
130         Block(stmts) ->
131             indent_string ^ "{\n" ^
132             String.concat "" (List.map (string_of_stmt
133     ↪ (indent+1)) stmts) ^
134             indent_string ^ "}\n"
135
136     | Expr(expr) ->
137         indent_string ^ string_of_expr expr ^ ";\n";
138
139     | Return(expr) ->
140         indent_string ^ "return " ^ string_of_expr expr ^ ";\n";
141
142     | If(e, s, Block([Expr(Noexpr)])) ->
143         indent_string ^ "if (" ^ string_of_expr e ^ ")\n" ^
144         (string_of_stmt (indent+1) s)
145
146     | If(e, s1, s2) ->
147         indent_string ^ "if (" ^ string_of_expr e ^ ")\n" ^
148         string_of_stmt (indent+1) s1 ^
149         indent_string ^ "else\n" ^
150         string_of_stmt (indent+1) s2
151
152     | For(e1, e2, e3, s) ->
153         indent_string ^ "for (" ^ string_of_expr e1 ^ " ; " ^
154     ↪ string_of_expr e2 ^ " ; " ^ string_of_expr e3 ^ ")\n" ^
155         string_of_stmt (indent) s
156
157     | While(e, s) ->
158         indent_string ^ "while (" ^ string_of_expr e ^ ")\n" ^
159         string_of_stmt (indent) s
160
161     | Break -> indent_string
162     ↪ ^ "break;\n"
163
164     | Continue -> indent_string ^
165     ↪ "continue;\n"
166
167     | Local(d, s, e) -> indent_string ^
168     ↪ string_of_datatype d ^ " " ^ s ^ string_of_local_expr e ^ ";\n"
169     in get_stmt_string

```

```

163
164 let string_of_local_sexpr = function
165     SNoexpr          -> ""
166     | e              -> " = " ^ string_of_sexpr e
167
168 let rec string_of_sstmt indent =
169     let indent_string = String.make indent '\t' in
170     let get_stmt_string = function
171
172         SBlock(stmts)          ->
173             indent_string ^ "{\n" ^
174             String.concat "" (List.map (string_of_sstmt
175 ↪ (indent+1)) stmts) ^
176             indent_string ^ "}\n"
177
178     | SExpr(expr, _)          ->
179         indent_string ^ string_of_sexpr expr ^ ";\n";
180
181     | SReturn(expr, _)        ->
182         indent_string ^ "return " ^ string_of_sexpr expr ^ ";\n";
183
184     | SIf(e, s, SBlock([SExpr(SNoexpr, _)])) ->
185         indent_string ^ "if (" ^ string_of_sexpr e ^ ")\n" ^
186             (string_of_sstmt (indent+1) s)
187
188     | SIf(e, s1, s2)          ->
189         indent_string ^ "if (" ^ string_of_sexpr e ^ ")\n" ^
190             string_of_sstmt (indent+1) s1 ^
191             indent_string ^ "else\n" ^
192             string_of_sstmt (indent+1) s2
193
194     | SFor(e1, e2, e3, s)      ->
195         indent_string ^ "for (" ^ string_of_sexpr e1 ^ " ; " ^
196 ↪ string_of_sexpr e2 ^ " ; " ^ string_of_sexpr e3 ^ ")\n" ^
197             string_of_sstmt (indent) s
198
199     | SWhile(e, s)            ->
200         indent_string ^ "while (" ^ string_of_sexpr e ^ ")\n" ^
201             string_of_sstmt (indent) s
202
203     | SBreak                  -> indent_string
204 ↪ ^ "break;\n"
205
206     | SContinue                -> indent_string ^
207 ↪ "continue;\n"
208
209     | SLocal(d, s, e)         -> indent_string ^
210 ↪ string_of_datatype d ^ " " ^ s ^ string_of_local_sexpr e ^ ";\n"
211         in get_stmt_string
212
213 (* Print Function *)

```

```

207
208 let string_of_fname = function
209     Constructor -> "constructor"
210     | FName(s)   -> s
211
212 let string_of_formal = function
213     Formal(d, s) -> (string_of_datatype d) ^ " " ^ s
214     | _          -> ""
215
216 let string_of_formal_name = function
217     Formal(_, s) -> s
218     | _          -> ""
219
220 let string_of_func_decl fdecl =
221     "" ^ (string_of_scope fdecl.scope) ^ " " ^ (string_of_datatype fdecl.returnType)
222     ↪ ^ " " ^ (string_of_fname fdecl.fname) ^ " " ^
223     (* Formals *)
224     "(" ^ String.concat "," (List.map string_of_formal fdecl.formals) ^ ")" {\n" ^
225     (* body *)
226     String.concat "" (List.map (string_of_stmt 2) fdecl.body) ^
227     "\t}\n\n"
228
229 (* Class Printing *)
230
231 let string_of_extends = function
232     NoParent      -> ""
233     | Parent(s)   -> "extends " ^ s ^ " "
234
235 let string_of_field = function
236     Field(s, d, id) -> (string_of_scope s) ^ " " ^ (string_of_datatype d) ^ " " ^ id
237     ↪ ^ ";\n"
238
239 let string_of_cbody cbody =
240     String.concat "" (List.map (fun s -> "\t" ^ s) (List.map string_of_field
241     ↪ cbody.fields)) ^
242     String.concat "" (List.map (fun s -> "\t" ^ s) (List.map string_of_func_decl
243     ↪ cbody.constructors)) ^
244     String.concat "" (List.map (fun s -> "\t" ^ s) (List.map string_of_func_decl
245     ↪ cbody.methods))
246
247 let string_of_class_decl cdecl =
248     "class " ^ cdecl.cname ^ " " ^ (string_of_extends cdecl.extends) ^ "{\n" ^
249     (string_of_cbody cdecl.cbody) ^
250     "}\n"
251
252 (* Include Printing *)
253
254 let rec string_of_include = function
255     Include(s) -> "include(" ^ s ^ ");\n"
256
257

```



```

251  (* Print whole program *)
252
253  let string_of_program = function
254      Program(includes, cdecls) ->
255      String.concat "" (List.map string_of_include includes) ^ "\n" ^
256      String.concat "\n" (List.map string_of_class_decl cdecls)
257
258  (* Print AST tree representation *)
259
260  let includes_tree includes =
261      'List (List.map (function Include s -> 'String s) includes)
262
263  let map_fields_to_json fields =
264      'List (List.map (function Field(scope, datatype, s) ->
265          'Assoc [
266              ("name", 'String s);
267              ("scope", 'String (string_of_scope scope));
268              ("datatype", 'String (string_of_datatype datatype));
269          ]) fields)
270
271  let map_formals_to_json formals =
272      'List (List.map (function Formal(d, s) -> 'Assoc [
273
274      ↪ 'String s);
275
276      ↪ 'String (string_of_datatype d));
277
278      ↪
279      | Many d -> 'Assoc [("Many",
280
281      ↪ 'String (string_of_datatype d));]
282      ) formals)
283
284  let rec map_expr_to_json = function
285      Int_Lit(i) -> 'Assoc [("int_lit", 'Int i)]
286      | Boolean_Lit(b) -> 'Assoc [("bool_lit", 'Bool b)]
287      | Float_Lit(f) -> 'Assoc [("float_lit", 'Float f)]
288      | String_Lit(s) -> 'Assoc [("string_lit", 'String
289      ↪ s)]
290      | Char_Lit(c) -> 'Assoc [("char_lit",
291      ↪ 'String (Char.escaped c))]
292      | This -> 'String "this"
293      | Id(s) -> 'Assoc [("id", 'String
294      ↪ s)]
295      | Binop(e1, o, e2) -> 'Assoc [("binop", 'Assoc [("lhs",
296      ↪ map_expr_to_json e1); ("op", 'String (string_of_op o)); ("rhs", map_expr_to_json
297      ↪ e2)]]]
298      | Assign(e1, e2) -> 'Assoc [("assign", 'Assoc
299      ↪ [("lhs", map_expr_to_json e1); ("op", 'String "="); ("rhs", map_expr_to_json e2)]]]
300      | Noexpr -> 'String "noexpr"

```

```

290         |      ObjAccess(e1, e2)                -> 'Assoc [("objaccess", 'Assoc
↪  [("lhs", map_expr_to_json e1); ("op", 'String "."); ("rhs", map_expr_to_json e2))]]
291         |      Call(f, el)                      -> 'Assoc [("call", 'Assoc
↪  ([("name", 'String f); ("params", 'List (List.map map_expr_to_json el)); ]) )]
292         |      ArrayPrimitive(el)               -> 'Assoc [("arrayprimitive",
↪  'List(List.map map_expr_to_json el))]
293         |      Unop(op, e)                      -> 'Assoc [("Unop", 'Assoc
↪  [("op", 'String (string_of_op op)); ("operand", map_expr_to_json e)]]]
294         |      Null                             -> 'String "null"
295         |      ArrayCreate(d, el)               -> 'Assoc [("arraycreate", 'Assoc [("datatype",
↪  'String (string_of_datatype d)); ("args", 'List (List.map map_expr_to_json el))]])
296         |      ArrayAccess(e, el)               -> 'Assoc [("arrayaccess", 'Assoc [("array",
↪  map_expr_to_json e); ("args", 'List (List.map map_expr_to_json el))]])
297         |      ObjectCreate(s, el)              -> 'Assoc [("objectcreate", 'Assoc [("type",
↪  'String s); ("args", 'List (List.map map_expr_to_json el))]])
298         |      Delete(e)                       -> 'Assoc [("delete", 'Assoc
↪  [("expr", map_expr_to_json e)]]]
299
300 let rec map_stmt_to_json = function
301         |      Block(stmts)                     -> 'Assoc [("block", 'List (List.map
↪  (map_stmt_to_json) stmts))]
302         |      Expr(expr)                       -> 'Assoc [("expr",
↪  map_expr_to_json expr)]
303         |      Return(expr)                     -> 'Assoc [("return",
↪  map_expr_to_json expr)]
304         |      If(e, s1, s2)                    -> 'Assoc [("if", 'Assoc
↪  [("cond", map_expr_to_json e); ("ifbody", map_stmt_to_json s1)]; ("else",
↪  map_stmt_to_json s2)]
305         |      For(e1, e2, e3, s)               -> 'Assoc [("for", 'Assoc [("init",
↪  map_expr_to_json e1); ("cond", map_expr_to_json e2); ("inc", map_expr_to_json e3);
↪  ("body", map_stmt_to_json s))]])
306         |      While(e, s)                     -> 'Assoc [("while", 'Assoc
↪  [("cond", map_expr_to_json e); ("body", map_stmt_to_json s))]])
307         |      Break                           -> 'String "break"
308         |      Continue                        -> 'String "continue"
309         |      Local(d, s, e)                   -> 'Assoc [("local", 'Assoc
↪  [("datatype", 'String (string_of_datatype d)); ("name", 'String s); ("val",
↪  map_expr_to_json e)]]]
310
311 let map_methods_to_json methods =
312     'List (List.map (fun (fdecl:Ast.func_decl) ->
313         'Assoc [
314             ("name", 'String (string_of_fname fdecl.fname));
315             ("scope", 'String (string_of_scope fdecl.scope));
316             ("returnType", 'String (string_of_datatype fdecl.returnType));
317             ("formals", map_formals_to_json fdecl.formals);
318             ("body", 'List (List.map (map_stmt_to_json) fdecl.body));
319         ]) methods)
320

```

```

321
322 let cdecls_tree cdecls =
323     let map_cdecl_to_json cdecl =
324         'Assoc [
325             ("cname", 'String cdecl.cname);
326             ("extends", 'String (string_of_extends cdecl.extends));
327             ("fields", map_fields_to_json cdecl.cbody.fields);
328             ("methods", map_methods_to_json cdecl.cbody.methods);
329             ("constructors", map_methods_to_json cdecl.cbody.constructors)
330         ]
331     in
332     'List (List.map (map_cdecl_to_json) cdecls)
333
334 let print_tree = function
335     Program(includes, cdecls) ->
336         'Assoc [("program",
337             'Assoc([
338                 ("includes", includes_tree includes);
339                 ("classes", cdecls_tree cdecls)
340             ]))
341         ])
342
343 (* Print SAST tree representation *)
344
345 let rec map_sexpr_to_json =
346     let datatype d = [("datatype", 'String (string_of_datatype d))] in
347     function
348         SInt_Lit(i)                -> 'Assoc [("int_lit", 'Assoc ([("val", 'Int
↪ i)] @ (datatype (Datatype(Int_t)))))]
349         | SBoolean_Lit(b)          -> 'Assoc [("bool_lit", 'Assoc ([("val", 'Bool
↪ b)] @ (datatype (Datatype(Bool_t)))))]
350         | SFloat_Lit(f)            -> 'Assoc [("float_lit", 'Assoc ([("val", 'Float
↪ f)] @ (datatype (Datatype(Float_t)))))]
351         | SString_Lit(s)           -> 'Assoc [("string_lit", 'Assoc ([("val",
↪ 'String s)] @ (datatype (ArrayType(Char_t, 1)))))]
352         | SChar_Lit(c)             -> 'Assoc [("char_lit", 'Assoc ([("val", 'String
↪ (Char.escaped c)] @ (datatype (Datatype(Char_t)))))]
353         | SId(s, d)                -> 'Assoc [("id", 'Assoc ([("name", 'String s)] @
↪ (datatype d)))]
354         | SBinop(e1, o, e2, d)     -> 'Assoc [("binop", 'Assoc ([("lhs",
↪ map_sexpr_to_json e1); ("op", 'String (string_of_op o)); ("rhs", map_sexpr_to_json
↪ e2)] @ (datatype d)))]
355         | SAssign(e1, e2, d)       -> 'Assoc [("assign", 'Assoc ([("lhs",
↪ map_sexpr_to_json e1); ("op", 'String "="); ("rhs", map_sexpr_to_json e2)] @
↪ (datatype d)))]
356         | SNoexpr                  -> 'Assoc [("noexpr", 'Assoc (datatype
↪ (Datatype(Void_t)))]

```

```

357     | SArrayCreate(t, e1, d) -> 'Assoc [("arraycreate", 'Assoc ([("datatype",
  ↪ 'String (string_of_datatype d)); ("args", 'List (List.map map_sexpr_to_json e1))] @
  ↪ (datatype d)))]
358     | SArrayAccess(e, e1, d) -> 'Assoc [("arrayaccess", 'Assoc ([("array",
  ↪ map_sexpr_to_json e); ("args", 'List (List.map map_sexpr_to_json e1))] @ (datatype
  ↪ d)))]
359     | SObjAccess(e1, e2, d) -> 'Assoc [("objaccess", 'Assoc ([("lhs",
  ↪ map_sexpr_to_json e1); ("op", 'String "."); ("rhs", map_sexpr_to_json e2)] @
  ↪ (datatype d)))]
360     | SCall(fname, e1, d, i) -> 'Assoc [("call", 'Assoc ([("name", 'String fname);
  ↪ ("params", 'List (List.map map_sexpr_to_json e1)); ("index", 'Int i) ] @ (datatype
  ↪ d)) )]
361     | SObjectCreate(s, e1, d) -> 'Assoc [("objectcreate", 'Assoc ([("type", 'String
  ↪ s); ("args", 'List (List.map map_sexpr_to_json e1))] @ (datatype d)))]
362     | SArrayPrimitive(e1, d) -> 'Assoc [("arrayprimitive", 'Assoc
  ↪ ([("expressions", 'List (List.map map_sexpr_to_json e1))] @ (datatype d)))]
363     | SUnop(op, e, d) -> 'Assoc [("Unop", 'Assoc ([("op", 'String
  ↪ (string_of_op op)); ("operand", map_sexpr_to_json e)] @ (datatype d)))]
364     | SNull -> 'Assoc [("null", 'Assoc (datatype
  ↪ (Datatype(Void_t)))]
365     | SDelete(e) -> 'Assoc [("delete", 'Assoc
  ↪ ([("expr", map_sexpr_to_json e)] @ (datatype (Datatype(Void_t)))]))
366
367 let rec map_sstmt_to_json =
368     let datatype d = [("datatype", 'String (string_of_datatype d))] in
369     function
370         SBlock s1 -> 'Assoc [("sblock", 'List
  ↪ (List.map (map_sstmt_to_json) s1))]
371         | SExpr(e, d) -> 'Assoc [("sexpr", 'Assoc ([("expr",
  ↪ map_sexpr_to_json e)] @ (datatype d)))]
372         | SReturn(e, d) -> 'Assoc [("sreturn", 'Assoc
  ↪ ([("return", map_sexpr_to_json e)] @ (datatype d)))]
373         | SIf (e, s1, s2) -> 'Assoc [("sif", 'Assoc ([("cond",
  ↪ map_sexpr_to_json e); ("ifbody", map_sstmt_to_json s1)]; ("selse", map_sstmt_to_json
  ↪ s2))]
374         | SFor (e1, e2, e3, s) -> 'Assoc [("sfor", 'Assoc ([("init",
  ↪ map_sexpr_to_json e1); ("cond", map_sexpr_to_json e2); ("inc", map_sexpr_to_json e3);
  ↪ ("body", map_sstmt_to_json s)))]
375         | SWhile (e, s) -> 'Assoc [("swhile", 'Assoc
  ↪ ([("cond", map_sexpr_to_json e); ("body", map_sstmt_to_json s)))]
376         | SBreak -> 'String "sbreak"
377         | SContinue -> 'String "scontinue"
378         | SLocal(d, s, e) -> 'Assoc [("slocal", 'Assoc
  ↪ ([("datatype", 'String (string_of_datatype d)); ("name", 'String s); ("val",
  ↪ map_sexpr_to_json e)))]
379
380 let string_of_func_type = function
381     User -> "user" | Reserved -> "reserved"
382

```

```

383 let map_sfdecl_to_json sfdecl =
384   'Assoc [("sfdecl", 'Assoc [
385     ("sfname", 'String (string_of_fname sfdecl.sfname));
386     ("sreturnType", 'String (string_of_datatype sfdecl.sreturnType));
387     ("sformals", map_formals_to_json sfdecl.sformals);
388     ("sbody", 'List (List.map (map_sstmt_to_json) sfdecl.sbody));
389     ("func_type", 'String (string_of_func_type sfdecl.func_type));
390   ])]
391
392 let map_sfdecls_to_json sfdecls =
393   'List (List.map map_sfdecl_to_json sfdecls)
394
395 let map_scdecls_to_json scdecls =
396   'List (List.map (fun scdecl ->
397     'Assoc [("scdecl",
398       'Assoc [
399         ("scname", 'String
400           ↪ scdecl.scname);
401         ("sfields",
402           ↪ map_fields_to_json scdecl.sfields);
403         ("sfuncs",
404           ↪ map_sfdecls_to_json scdecl.sfuncs);
405       ])
406     ])
407   scdecls)
408
409 let map_sprogram_to_json sprogram =
410   'Assoc [("sprogram", 'Assoc [
411     ("classes", map_scdecls_to_json sprogram.classes);
412     ("functions", map_sfdecls_to_json sprogram.functions);
413     ("main", map_sfdecl_to_json sprogram.main);
414     ("reserved", map_sfdecls_to_json sprogram.reserved);
415   ])]
416
417 (* Print tokens *)
418
419 let string_of_token = function
420   LPAREN      -> "LPAREN"
421   | RPAREN    -> "RPAREN"
422   | LBRACE    -> "LBRACE"
423   | RBRACE    -> "RBRACE"
424   | SEMI      -> "SEMI"
425   | COMMA     -> "COMMA"
426   | PLUS      -> "PLUS"
427   | MINUS     -> "MINUS"
428   | TIMES     -> "TIMES"
429   | DIVIDE    -> "DIVIDE"
430   | ASSIGN    -> "ASSIGN"
431   | EQ        -> "EQ"

```

```

429         |           NEQ                               -> "NEQ"
430         |           LT                                -> "LT"
431         |           LEQ                               -> "LEQ"
432         |           GT                                -> "GT"
433         |           GEQ                               -> "GEQ"
434         |           AND                               -> "AND"
435         |           OR                                -> "OR"
436         |           NOT                               -> "NOT"
437         |           DOT                               -> "DOT"
438         |           LBRACKET                          -> "LBRACKET"
439         |           RBRACKET                          -> "RBRACKET"
440         |           BAR                               -> "BAR"
441         |           IF                                -> "IF"
442         |           ELSE                              -> "ELSE"
443         |           FOR                               -> "FOR"
444         |           WHILE                             -> "WHILE"
445         |           RETURN                            -> "RETURN"
446         |           INT                               -> "INT"
447         |           FLOAT                            -> "FLOAT"
448         |           BOOL                             -> "BOOL"
449         |           CHAR                             -> "CHAR"
450         |           VOID                             -> "VOID"
451         |           NULL                             -> "NULL"
452         |           TRUE                             -> "TRUE"
453         |           FALSE                            -> "FALSE"
454         |           CLASS                            -> "CLASS"
455         |           CONSTRUCTOR                       -> "CONSTRUCTOR"
456         |           PUBLIC                           -> "PUBLIC"
457         |           PRIVATE                           -> "PRIVATE"
458         |           EXTENDS                           -> "EXTENDS"
459         |           INCLUDE                           -> "INCLUDE"
460         |           THIS                             -> "THIS"
461         |           BREAK                             -> "BREAK"
462         |           CONTINUE                          -> "CONTINUE"
463         |     NEW                                     -> "NEW"
464         |           INT_LITERAL(i)                   -> "INT_LITERAL(" ^ string_of_int i ^ ")"
465         |           FLOAT_LITERAL(f)                  -> "FLOAT_LITERAL(" ^ string_of_float f ^ ")"
466         |           CHAR_LITERAL(c)                   -> "CHAR_LITERAL(" ^ Char.escaped c ^
↳      " ^ ")"
467         |           STRING_LITERAL(s)                 -> "STRING_LITERAL(" ^ s ^ ")"
468         |           ID(s)                             -> "ID(" ^ s ^ ")"
469         |           DELETE                             -> "DELETE"
470         |           MODULO                             -> "MODULO"
471         |           EOF                               -> "EOF"
472
473 let string_of_token_no_id = function
474     LPAREN                               -> "LPAREN"
475     RPAREN                               -> "RPAREN"
476     LBRACE                               -> "LBRACE"

```

```
477      |      RBRACE                -> "RBRACE"
478      |      SEMI                  -> "SEMI"
479      |      COMMA                 -> "COMMA"
480      |      PLUS                  -> "PLUS"
481      |      MINUS                 -> "MINUS"
482      |      TIMES                 -> "TIMES"
483      |      DIVIDE                -> "DIVIDE"
484      |      ASSIGN                -> "ASSIGN"
485      |      EQ                    -> "EQ"
486      |      NEQ                   -> "NEQ"
487      |      LT                    -> "LT"
488      |      LEQ                   -> "LEQ"
489      |      GT                    -> "GT"
490      |      GEQ                   -> "GEQ"
491      |      AND                   -> "AND"
492      |      OR                    -> "OR"
493      |      NOT                   -> "NOT"
494      |      DOT                   -> "DOT"
495      |      LBRACKET              -> "LBRACKET"
496      |      RBRACKET             -> "RBRACKET"
497      |      BAR                   -> "BAR"
498      |      IF                    -> "IF"
499      |      ELSE                  -> "ELSE"
500      |      FOR                   -> "FOR"
501      |      WHILE                 -> "WHILE"
502      |      RETURN                -> "RETURN"
503      |      INT                   -> "INT"
504      |      FLOAT                 -> "FLOAT"
505      |      BOOL                  -> "BOOL"
506      |      CHAR                  -> "CHAR"
507      |      VOID                  -> "VOID"
508      |      NULL                  -> "NULL"
509      |      TRUE                  -> "TRUE"
510      |      FALSE                 -> "FALSE"
511      |      CLASS                 -> "CLASS"
512      |      CONSTRUCTOR           -> "CONSTRUCTOR"
513      |      PUBLIC                -> "PUBLIC"
514      |      PRIVATE               -> "PRIVATE"
515      |      EXTENDS               -> "EXTENDS"
516      |      INCLUDE               -> "INCLUDE"
517      |      THIS                  -> "THIS"
518      |      BREAK                 -> "BREAK"
519      |      CONTINUE              -> "CONTINUE"
520      |      NEW                   -> "NEW"
521      |      INT_LITERAL(i)        -> "INT_LITERAL"
522      |      FLOAT_LITERAL(f)      -> "FLOAT_LITERAL"
523      |      CHAR_LITERAL(c)       -> "CHAR_LITERAL"
524      |      STRING_LITERAL(s)     -> "STRING_LITERAL"
525      |      ID(s)                 -> "ID"
```

```
526         |          DELETE          -> "DELETE"
527         |          MODULO          -> "MODULO"
528         |          EOF              -> "EOF"
529
530 let token_list_to_string_endl token_list =
531   let rec helper last_line_number = function
532     (token, curr)::tail ->
533       let line = curr.lineno in
534       (if line != last_line_number then "\n" ^ string_of_int line ^ ". " else "
↪ ") ^
535         string_of_token token ^ helper line tail
536     | [] -> "\n"
537   in helper 0 token_list
538
539 let token_list_to_string token_list =
540   let rec helper = function
541     (token, line)::tail ->
542       string_of_token_no_id token ^ " " ^ helper tail
543     | [] -> "\n"
544   in helper token_list
```


_tags

```
1 <filepath.*> or <**/*.native> or <**/*.byte>: package(unix)
```

Demo

Demo_Animals.dice

```
1  include("stdlib");
2
3  class Animal{
4      public int weight;
5      constructor(){
6          this.weight = 0;
7      }
8
9      constructor(int w){
10         this.weight = w;
11     }
12
13     public void move(){
14         print("Animals move in many ways");
15     }
16 }
17
18 class Bird extends Animal {
19     public int maxFlyingHeight;
20
21     constructor(){
22         this.weight = 0;
23         this.maxFlyingHeight = 0;
24     }
25
26     constructor(int w, int h){
27         this.weight = w;
28         this.maxFlyingHeight = h;
29     }
30
31     public void move(){
32         print("Birds fly!");
33     }
34 }
35
36
37 class Dog extends Animal {
38     public int speed;
39
40     constructor(){
41         this.weight = 0;
42         this.speed = 0;
43     }
44
45     constructor(int w, int s){
```

```
46         this.weight = w;
47         this.speed = s;
48     }
49
50     public void move(){
51         print("Dogs run!");
52     }
53 }
54
55 class Stephen extends Animal {
56     private bool isDone;
57
58     constructor() {
59         this.isDone = true;
60     }
61
62     constructor(bool isDone) {
63         this.isDone = isDone;
64     }
65
66     public void move() {
67         if(not this.isDone) {
68             print("I am a techer!");
69         } else {
70             print("Also my favorite number is 42");
71         }
72         this.isDone = true;
73     }
74 }
75
76
77 class Snake extends Animal {
78     public int slitherSpeed;
79
80     constructor(){
81         this.weight = 0;
82         this.slitherSpeed = 0;
83     }
84
85     constructor(int w, int s){
86         this.weight = w;
87         this.slitherSpeed = s;
88     }
89
90     public void move(){
91         print("Snakes slither!");
92     }
93 }
94
```

```
95 class Marnie extends Dog {
96     public int cuteness;
97
98     constructor(){
99         this.weight = 0;
100        this.speed = 0;
101    }
102
103    constructor(int w, int s){
104        this.weight = w;
105        this.speed = s;
106    }
107
108    constructor(int w, int s, int c){
109        this.weight = w;
110        this.speed = s;
111        this.cuteness = c;
112    }
113
114    public void move(){
115        class File a = new File("Demo/marnie1.txt", true);
116        char[] buf = a.readfile(4500);
117        a.closefile();
118        print(buf);
119        print("\n");
120    }
121 }
122
123 class test {
124     private bool isDone;
125     public void main(char[][] args) {
126         this.logo();
127         this.isDone = false;
128
129         bool keepGoing = true;
130         while(keepGoing){
131             this.animalsToChoose();
132             char[] buf = input();
133             print("\n");
134
135             int choice = this.getInt(buf[0]);
136
137             if(choice==5)
138                 break;
139             else
140                 this.printMovement(choice);
141
142             print("\n");
143         }
```

```
144
145     class Marnie a = new Marnie();
146     a.move();
147 }
148
149 public int getInt(char num){
150     if(num=='1')
151         return 1;
152     else if(num=='2')
153         return 2;
154     else if(num=='3')
155         return 3;
156     else if(num=='4')
157         return 4;
158     else if(num=='5')
159         return 5;
160
161     return 0;
162
163 }
164
165 public void printMovement(int choice){
166
167     class Animal b = new Bird();
168     class Animal d = new Dog();
169     class Animal s = new Snake();
170     class Animal stephen = new Stephen(this.isDone);
171
172     if(choice == 1)
173         b.move();
174     else if(choice == 2)
175         d.move();
176     else if(choice == 3)
177         s.move();
178     else if(choice == 4) {
179         stephen.move();
180         this.isDone = true;
181     }
182     else
183         print("Animal not selected!\n");
184
185     print("\n");
186 }
187
188 public void animalsToChoose(){
189     print("1-Bird\n2-Dog\n3-Snake\n4-Stephen\n5-Exit\nPlease choose an animal
↪ or exit(by selecting a number):");
190
191 }
```

```
192
193     public void logo(){
194         class File a = new File("Demo/logo.txt", true);
195         char[] buf = a.readfile(4500);
196         a.closefile();
197         print(buf);
198
199         int i;
200         for(i=0;i<3;i=i+1){
201             print("\n");
202         }
203
204         print("Welcome to the animal farm!\n\n");
205     }
206 }
```

logo.txt

```
1  .----- .----- .----- .-----
2  | .----- .| .----- .| .----- .| .----- |
3  | | _ _ _ _ _ | | | _ _ _ _ _ | | | _ _ _ _ _ | |
4  | | | _ _ _ _ _ ' | | | | _ _ _ _ _ | | | | _ _ _ _ _ | |
5  | | | | _ _ _ _ _ ' \ | | | | | | | / ' \ | | | | | | | \ | |
6  | | | | | | | | | | | | | | | | | | | | | | | | | | | |
7  | | | _ _ _ _ _ ' / | | | | _ _ _ _ _ | | | | \ ' _ _ _ _ _ \ | | | | _ _ _ _ _ / | |
8  | | | _ _ _ _ _ ' | | | | _ _ _ _ _ | | | | ' _ _ _ _ _ ' | | | | _ _ _ _ _ | |
9  | | | | | | | | | | | | | | | | | | | | | | | | | | | |
10 | ' _ _ _ _ _ _ _ ' | | ' _ _ _ _ _ _ _ ' | | ' _ _ _ _ _ _ _ ' | | ' _ _ _ _ _ _ _ ' |
11 ' _ _ _ _ _ _ _ ' ' _ _ _ _ _ _ _ ' ' _ _ _ _ _ _ _ ' ' _ _ _ _ _ _ _ '
12 ..... ~~~~~ .....
13 ..... ~~~~~ .....
14 ..... ~~~~~ .....
15 ..... ~~~~~ .....
16 ..... ~~~~~ .....
17 ..... ~~~~~ .....
18 ..... ~~~~~ .....
19 ..... ~~~~~ .....
20 ..... ~~~~~ .....
21 ..... ~~~~~ .....
22 ..... ~~~~~ .....
23 ..... ~~~~~ .....
24 ..... ~~~~~ .....
25 ..... ~~~~~ .....
26 ..... ~~~~~ .....
27 ..... ~~~~~ .....
28 ..... ~~~~~ .....
29 ..... ~~~~~ .....
30 ..... ~~~~~ .....
31 ..... ~~~~~ .....
32 ..... ~~~~~ .....
33 ..... ~~~~~ .....
34 ..... ~~~~~ .....
35 ..... ~~~~~ .....
36 ..... ~~~~~ .....
37 ..... ~~~~~ .....
38 ..... ~~~~~ .....
39 ..... ~~~~~ .....
40 ..... ~~~~~ .....
41 ..... ~~~~~ .....
42 ..... ~~~~~ .....
43 ..... ~~~~~ .....
44 ..... ~~~~~ .....
45 ..... ~~~~~ .....
46 ..... ~~~~~ .....
47 ..... ~~~~~ .....
```

```
48 ..... ~ ~ ..... ~ ~ ~ ..... , ~ ~ ~ ..... ~ ~ .....
49 ..... ~ ~ ~ ..... ~ ~ ..... ~ ~ ~ : ~ ..... ~ ~ .....
50 ..... ~ ~ ~ ..... ~ ~ ..... ~ ~ ..... ~ ~ .....
51 ..... ~ ~ ~ ~ ..... ~ ~ ..... ~ ~ ..... : ~ , ..... : ~ ~ ~ .....
52 ..... ~ ~ ~ ~ : ..... ~ ~ ..... ~ ~ ..... ~ ~ : ..... , ~ ~ ~ ~ .....
53 ..... ~ ~ ~ ~ ~ ..... ~ ~ ..... : ~ ~ ..... ~ ~ ..... ~ ~ ~ ~ .....
54 ..... ~ ~ ~ ~ ~ ..... ~ ~ ..... , ~ ~ ..... ~ ~ ..... ~ ~ ~ ~ .....
55 ..... ~ ~ ~ ~ ~ ..... ~ ~ ..... ~ ~ ~ ~ ~ ..... ~ ~ ~ ~ .....
56 ..... ~ ~ ~ ~ ~ , ~ ~ ~ ~ ..... ~ ~ ..... ~ ~ ~ ~ ..... : ~ ~ ~ ~ .....
57 ..... : ~ ~ ~ ~ : ..... ~ ~ ..... ~ ~ ..... , ~ ~ ~ ~ .....
58 ..... ~ ~ ~ ~ ~ : ..... ~ ~ ..... ~ ~ ..... , ~ ~ ~ ~ .....
59 ..... ~ ~ ~ ~ ~ ..... ~ ~ ..... ~ ~ ~ ~ .....
60 ..... ~ ~ ~ ~ ~ ..... ~ ~ ..... ~ ~ ~ ~ .....
61 ..... ~ ~ ~ ~ ~ ~ ~ ~ .....
62 ..... : ~ : .....
```


- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 10
- 11
- 12
- 13
- 14
- 15
- 16
- 17
- 18
- 19
- 20
- 21
- 22
- 23
- 24
- 25
- 26
- 27
- 28
- 29
- 30
- 31
- 32
- 33
- 34
- 35
- 36
- 37
- 38
- 39
- 40
- 41
- 42
- 43
- 44

224

marnie2.txt

```

1  ++++++
2  ++++++
3  ++++++
4  ++++++I+++++
5  ++++++I+++++?+I+++++II??I??+
6  ++++++?+I+++++IIII7IIIIII7I7I7++?+
7  ++++++I+++++I+++++?+?+IIIIIIIIII7I7I7??I?+
8  ++++++?+?+I?+I??+IIIIII?IIIIII7I777IIIIII?I+
9  ++++++?+?+I?+I??+IIIIII777I7IIII?+
10 ++++++I?+I?+I?+I??+IIIIII77IIIIIIII?+
11 ++++++?+?+I?+I??+I?+I??+IIIIII7IIIIIIIIIIII?+
12 ++++++?+?+?+?+?+?+?+?+I?+IIII?IIIIII77IIIIII+
13 ++++++I?+?+?+?+?+?+?+?+IIII?II?+IIIIIIIIIIII?+
14 ++++++IIII?+?+?+?+?+?+?+?+IIII?II?+IIIIIIIIIIII?+
15 ++++++IIII?+?+?+?+?+?+?+?+IIII?II?+IIIIII??I77+
16 ++++++I?+?+?+?+?+?+?+?+IIII777IIIIIIIIII7?+
17 ++++++I?+?+?+?+?+?+?+?+IIIIIIII?I??+IIII?+
18 ++++++I?+?+?+?+?+?+?+?+IIIIIIII?I??+IIII?+
19 ++++++I?+?+?+?+?+?+?+?+IIIIIIII?I??+IIII?+
20 ++++++I?+?+?+?+?+?+?+?+IIIIIIII?I??+IIII?+
21 ++++++I?+?+?+?+?+?+?+?+IIIIIIII?I??+IIII?+
22 ++++++I?+?+?+?+?+?+?+?+IIIIIIII?I??+IIII?+
23 ++++++I?+?+?+?+?+?+?+?+IIIIIIII?I??+IIII?+
24 ++++++I?+?+?+?+?+?+?+?+IIIIIIII?I??+IIII?+
25 ++++++I?+?+?+?+?+?+?+?+IIIIIIII?I??+IIII?+
26 ++++++I?+?+?+?+?+?+?+?+IIIIIIII?I??+IIII?+
27 ++++++I?+?+?+?+?+?+?+?+IIIIIIII?I??+IIII?+
28 ++++++I?+?+?+?+?+?+?+?+IIIIIIII?I??+IIII?+
29 ++++++I?+?+?+?+?+?+?+?+IIIIIIII?I??+IIII?+
30 ++++++I?+?+?+?+?+?+?+?+IIIIIIII?I??+IIII?+
31 ++++++I?+?+?+?+?+?+?+?+IIIIIIII?I??+IIII?+
32 ++++++I?+?+?+?+?+?+?+?+IIIIIIII?I??+IIII?+
33 ++++++I?+?+?+?+?+?+?+?+IIIIIIII?I??+IIII?+
34 ++++++I?+?+?+?+?+?+?+?+IIIIIIII?I??+IIII?+
35 ++++++I?+?+?+?+?+?+?+?+IIIIIIII?I??+IIII?+
36 ++++++I?+?+?+?+?+?+?+?+IIIIIIII?I??+IIII?+
37 ++++++I?+?+?+?+?+?+?+?+IIIIIIII?I??+IIII?+
38 ++++++I?+?+?+?+?+?+?+?+IIIIIIII?I??+IIII?+
39 ++++++I?+?+?+?+?+?+?+?+IIIIIIII?I??+IIII?+
40 ++++++I?+?+?+?+?+?+?+?+IIIIIIII?I??+IIII?+
41 ++++++I?+?+?+?+?+?+?+?+IIIIIIII?I??+IIII?+
42 ++++++I?+?+?+?+?+?+?+?+IIIIIIII?I??+IIII?+
43 ++++++I?+?+?+?+?+?+?+?+IIIIIIII?I??+IIII?+
44 ++++++I?+?+?+?+?+?+?+?+IIIIIIII?I??+IIII?+
45 ++++++I?+?+?+?+?+?+?+?+IIIIIIII?I??+IIII?+
46 ++++++I?+?+?+?+?+?+?+?+IIIIIIII?I??+IIII?+
47 ++++++I?+?+?+?+?+?+?+?+IIIIIIII?I??+IIII?+

```

```
48  ???II7IIIIIIII?I???~?I???+??+=+???+==++??=++?II????????????????????????IIIIII
49  ?+?I?IIIIIIII???I???=?III??II?+???I+++++?+?=++?III????????????????????????IIII
50  ?+++?I???IIII????I?III?III+??+=+++++++?+?=++?IIII????????????????????????I
51  ??++?I??+?????+??I?II????+??+=+++++++?=+++????IIII????????????????????????
```

Othello.dice

```
1  include("stdlib");
2
3
4  class Player {
5
6
7      public class LocationObj placeTile(bool retry)  {
8          return new LocationObj();
9      }
10
11     public void setResult(class LocationObj move) {
12     }
13 }
14
15 (*****
16
17 class HumanPlayer extends Player {
18     private class Board board;
19     public int myPieceType;
20
21     constructor()  {
22         this.board = new Board();
23         this.myPieceType = 2;
24         class Board b = this.board;
25         b.initializeBoard();
26     }
27
28     public class LocationObj placeTile(bool retry)  {
29         if (this.myPieceType == 2)
30             this.myPieceType = 1;
31         if (retry){
32             print("Last move was invalid. Retry.\n"); }
33         print("It's your turn\n");
34         class Board b = this.board;
35         b.printBoard();
36
37         print("Please enter your move\n");
38         class LocationObj move = this.getLocationObjChoice();
39         int temp = this.myPieceType;
40         b.setPlayerMove(move, temp);
41         return move;
42     }
43
44     public void setResult(class LocationObj move) {
45         int temp = this.myPieceType;
46         if (temp == 1) {
47             bool one = (move.getHorizontal() == 3);
```

```

48         bool two = (move.getHorizontal() == 4);
49         bool three = (move.getVertical() == 3);
50         bool four = (move.getVertical() == 4);
51         bool five = ((one or two ) and (three or four));
52         if(not five){
53             this.myPieceType = 0;
54         }
55     }
56
57     int opponentPieceType;
58     int temp2 = this.myPieceType;
59     if (temp2 == 0){
60         opponentPieceType = 1; }
61     else {
62         opponentPieceType = 0;}
63
64     class Board b = this.board;
65     b.setPlayerMove(move, opponentPieceType);
66 }
67
68 private class LocationObj getLocationObjChoice(){
69     char[] userInput;
70     class String uInput;
71     class Board b = new Board();
72     class LocationObj move = null;
73     int temp = this.myPieceType;
74     while (not (b.isValid(move, temp))) {
75         print("You are " , this.myPieceType , ". What is the x location of your next
↪ move?");
76         userInput = input();
77         uInput = new String(userInput);
78         int x = uInput.toInteger();
79         print("You are " , this.myPieceType , ". What is the y location of your next
↪ move?");
80         userInput = input();
81         uInput = new String(userInput);
82         int y = uInput.toInteger();
83         move = new LocationObj(x - 1, y - 1);
84         bool one = b.isValid(move,temp);
85         if (not one){
86             print("invalid move, try again.\n"); }
87     }
88     return move;
89 }
90 }
91
92
93
94 (*****)
```

```
95
96
97 class ComputerPlayer extends Player {
98
99     private class Board board;
100     public int myType;
101
102     constructor(){
103         this.board = new Board();
104         class Board board = this.board;
105         board.initializeBoard();
106         this.myType = 2;
107     }
108
109     public class LocationObj placeTile(bool retry) {
110         class Board board = this.board;
111         if(this.myType == 2){
112             this.myType = 1;
113         }
114
115
116         class LocationObj move = this.getBestMove(this.myType);
117         int temp = this.myType;
118         board.setPlayerMove(move, temp);
119         return move;
120     }
121
122     public void setResult(class LocationObj move) {
123         class Board board = this.board;
124         if(this.myType == 2 and
125             ((move.getHorizontal() == 3 or move.getHorizontal() == 4) == false) and
126             (move.getVertical() == 3 or move.getVertical() == 4)){
127             this.myType = 0;
128         }
129
130         int opponent;
131         if(this.myType == 1)
132             opponent = 0;
133         else
134             opponent = 1;
135
136         board.setPlayerMove(move, opponent);
137     }
138
139     private int[] createPointArray(){
140         int[] points = new int[64];
141         int i;
142         int j;
143         for(i = 0; i < 8; i = i + 1)
```

```

144     {
145         for(j = 0; j < 8; j = j + 1)
146         {
147             points[j*8+i] = 1;
148         }
149     }
150 (*
151     //[4*8+2*8+3*8+2*8+2*8+3*8+2*8+4]
152     //[2*8+3*8+1*8+1*8+1*8+1*8+3*8+2]
153     //[3*8+1*8+2*8+1*8+1*8+2*8+1*8+3]
154     //[2*8+1*8+1*8+0*8+0*8+1*8+1*8+2]
155     //[2*8+1*8+1*8+0*8+0*8+1*8+1*8+2]
156     //[3*8+1*8+2*8+1*8+1*8+2*8+1*8+3]
157     //[2*8+3*8+1*8+1*8+1*8+1*8+3*8+2]
158     //[4*8+2*8+3*8+2*8+2*8+3*8+2*8+4]
159 *)
160     points[(0*8)+0] = points[(7*8)+0] = points[(0*8)+7] = points[(7*8)+7] = 4;
161
162     points[2] = points[2*8] = points[2*8+7] = points[7*8+2] = 3;
163     points[1*8+1] = points[6*8+6] = points[1*8+6] = points[6*8+1] = 3;
164     points[5] = points[5*8+0] = points[5*8+7] = points[7*8+5] = 3;
165
166     points[0*8+1] = points[0*8+3] = points[0*8+4] = points[0*8+6] = 2;
167     points[1*8+0] = points[0*8+7] = 2;
168     points[2*8+2] = points[2*8+5] = 2;
169     points[3*8+0] = points[3*8+7] = 2;
170     points[4*8+0] = points[4*8+7] = 2;
171     points[5*8+2] = points[5*8+5] = 2;
172     points[6*8+0] = points[6*8+7] = 2;
173     points[7*8+1] = points[7*8+3] = points[7*8+4] = points[7*8+6] = 2;
174
175     points[4*8+4] = points[4*8+5] = points[5*8+4] = points[5*8+5] = 0;
176
177     return points;
178 }
179
180 private class LocationObj getBestMove(int turn){
181     class LocationObj best = null;
182     int currentValue = -2147483647;
183     int[] pointArray = createPointArray();
184     class Board board = this.board;
185     int i;
186     int j;
187     for(i = 0; i < 8; i = i + 1) {
188         for(j = 0; j < 8; j = j + 1){
189             class LocationObj l = new LocationObj(i,j);
190             if(board.isValid(l, turn) and pointArray[i*8+j] > currentValue){
191                 currentValue = pointArray[i*8+j];
192                 best = l;

```

```
193         }
194     }
195 }
196     return best;
197 }
198
199 }
200
201
202
203 (**
204  * Implementation of Location that has horizontal and verticale coordinates
205  *
206  * @author David Watkins
207  * @UNI djw2146
208  *)
209 (**
210  * Implementation of Location that has horizontal and verticale coordinates
211  *
212  * @author David Watkins
213  * @UNI djw2146
214  *)
215
216
217
218 class LocationObj {
219
220 public void main(char[] [] args) {}
221     (* ===== *)
222
223     private int horizontal;
224     private int vertical;
225
226     (**
227     * Creates a LocationObj with horizontal and vertical coordinates
228     *
229     * @param horizontal x coordinate
230     * @param vertical y coordinate
231     *)
232     constructor(int horizontal, int vertical){
233         this.horizontal = horizontal;
234         this.vertical = vertical;
235     }
236
237     (*
238     * @Return Horizontal coordinate
239     *)
240     (* @Override *) (* ===== *)
241     public int getHorizontal() {
```



```
242         return this.horizontal;
243     }
244
245     (*
246     * @Return Vertical coordinate
247     *)
248     (* @Override *) (* ===== *)
249     public int getVertical() {
250         return this.vertical;
251     }
252 }
253
254
255 (**
256 * Maintains and operates on the board.
257 * Has methods for checking if particular moves are valid and initializing the board.
258 *
259 * @author David Watkins
260 * UNI djw2146
261 *)
262
263
264
265 class Board {
266     private int[] board;
267
268     (**
269     * Initializes a new board with size Game.SIZE x Game.SIZE
270     *)
271
272     constructor(){
273         this.board = new int[64];
274         this.initializeBoard();
275     }
276
277     (**
278     * Initializes the board to have the center most four pieces in the correct formation
279     *
280     * @param type The type of player, as the user of the Board could differ
281     *)
282     public void initializeBoard(){
283         int i;
284         int j;
285         int[] board = this.board;
286         for (i = 0 ; i < 8 ; i = i + 1) {
287             for (j = 0 ; j < 8 ; j = j + 1) {
288                 board[i+j]=2;
289             }
290         }
```

```

291
292     board[(3*8)+4] = 1;
293     board[(4*8)+3] = 1;
294     board[(3*8)+3] = 0;
295     board[(4*8)+4] = 0;
296 }
297
298 (**
299  * Prints out a formatted version of the board to the console.
300  *)
301 public void printBoard() {
302     print("-----");
303     int i;
304     int j;
305     int[] board = this.board;
306
307     for(i = 0; i < 7; i=i+1){
308         (* Prints out each line individually *)
309         for( j = 0; j < 7; j = j + 1){
310
311             (//Prints out each section of a line of the board*)
312             if(board[(j*8+i)] == 0) { (//SELF player*)
313                 print("|(WHITE)");
314             } else if(board[(8*j)+i] == 1) { (//OPPONENT player*)
315                 print("|(BLACK)");
316             } else { (//No piece in location*)
317                 print("|(", (j+1), ", ", (i+1), ")");
318             }
319         }
320         print("|"); (//Finishes the line*)
321         print("\n");
322     }
323     print("-----");
324     print("\n");
325 }
326
327 (**
328  * Checks all possible indices of the board to determine whether or not a winner has
↪ been determined
329  * Counts the number of each piece and returns the type of winner
330  *
331  * @return The winner if there is a winner, null if no winner is determined yet
332  *)
333 public int thereIsWinner(){
334     int[] temp = this.totalCount();
335
336     if(temp[0] + temp[1] == 64){ (//If the total number of pieces equals the entire
↪ board *)
337         return this.whoHasMore();

```

```
338         } else { (*//If no winner yet *)
339             return -1;
340         }
341         return 9;
342     }
343
344     (**
345      * A method for determining which player has more pieces on the board.
346      *
347      * @return Which Piece has more positins in the board
348      *)
349     public int whoHasMore(){
350         int[] temp = this.totalCount();
351
352         if(temp[0] > temp[1]) {
353             return 0;
354         } else if(temp[1] > temp[0]) {
355             return 1;
356         } else { (* //If there is a tie *)
357             return 2;
358         }
359         (* default return *)
360         return 9;
361     }
362
363     (**
364      * Private helper metod for determining the current count of a particular type of
↪ player
365      *
366      * @return A size two vector containing whiteCount is [0] and blackCount in [1]
367      *)
368     private int[] totalCount(){
369         int whiteCount = 0;
370         int blackCount = 0;
371         int i;
372         int j;
373         int[] board = this.board;
374         for(i = 0; i < 8; i = i + 1) {
375             for(j = 0; j < 8; j = j + 1) {
376                 if(board[(i*8)+j] == 0) {
377                     whiteCount = whiteCount + 1;
378                 } else if(board[(i*8)+j] == 1) {
379                     blackCount = blackCount + 1;
380                 }
381             }
382         }
383         int[] temp = new int[2];
384         temp[0] = blackCount;
385         temp[1] = whiteCount;
```

```

386         return temp;
387     }
388
389     /**
390      * Will check to see if the particular direction is a valid move
391      * If bool updateBoard is true, will also update any pieces appropriately modified by
↪ a particular player move.
392      * Is only privately used by isValid and setPlayerMove to make sure a move is valid
393      * Returns 0 if the direction is invalid
394      *
395      * @param move The player's new move
396      * @param incx The coefficient of x
397      * @param incy The coefficient of y
398      * @param player The current player playing the new move
399      * @param updateBoard Whether or not to update pieces
400      * @return Whether the move is valid or not
401      *)
402
403
404     private bool checkDir(class LocationObj move, int incx, int incy, int player , bool
↪ updateBoard) {
405
406         int[] board = this.board;
407         int opponent; (//The opposite color of player*)
408         int x = move.getHorizontal();
409         int y = move.getVertical();
410
411         (//Current player's move *)
412         if (player == 1) {
413             opponent=0;
414         } else {
415             opponent=1;
416         }
417         ( //Modify the position by one *)
418         int dist = 0;
419         x = x + incx;
420         y = y + incy;
421
422         (//While x and y are in bounds and the current position is an opponent piece *)
423         while ((x < 8) and (x >= 0) and (y < 8) and (y >= 0) and (board[(x*8)+y] ==
↪ opponent)) {
424             x = x + incx;
425             y = y + incy;
426             dist = dist + 1;
427         }
428
429         ( //If x and y are still in bounds and the final position is a player piece,
↪ will *)

```

```

430         if ((dist != 0) and (x < 8) and (x >= 0) and (y < 8) and (y >= 0) and
↪ (board[(x*8)+y]==player)) {
431             if (updateBoard) { (* //Will update the board if true *)
432                 int j;
433                 for (j = 1 ; j <= dist ; j = j+1) {
434                     x = x - incx; (*//Decrease x by one*)
435                     y = y - incy; (*//Decrease y by one*)
436                     class LocationObj l = new LocationObj(x,y);
437                     this.setLoc(l,player); (*//Update location to player piece *)
438                 }
439                 return true; (* //The current distance from the initial position *)
440             } else {
441                 return false; (*//Not a valid direction *)
442             }
443         }
444         return false; (* default return *)
445     }
446
447     (**
448     * Will set the location defined by move to the type player
449     *
450     * @param move The position to be modified
451     * @param player The player's type
452     *)
453
454     private void setLoc(class LocationObj move, int player){
455
456         this.board[(move.getHorizontal()*8)+move.getVertical()] = player;
457     }
458
459     (**
460     * Sets the location of a particular player (c)'s move. Returns false
461     * if move is invalid.
462     *
463     * @param move Location object containing horizontal and vertical
464     * coordinates
465     * @param c The type of player currently placing tile
466     * @return False if invalid move
467     *)
468
469
470     public bool setPlayerMove(class LocationObj move, int player) {
471         bool valid = false;
472         int yinc = 0;
473         int xinc = 0;
474         int[] board = this.board;
475
476         (* //If move is null or move space is taken up *)
477         if(move == null or board[(move.getHorizontal()*8)+move.getVertical()] != 2) {

```

```

478         return false;
479     }
480     int i;
481     for(i = 0; i < 8; i = i + 1){ (*/For the length of potential neighbors*)
482         (*/Linear Directions *)
483         if(i == 0){xinc = 1; yinc = 0;} (*/E *)
484         else if(i == 1){xinc = -1; yinc = 0;} (* //W *)
485         else if(i == 2){xinc = 0; yinc = 1;} (* //S *)
486         else if(i == 3){xinc = 0; yinc = -1;} (* //N *)
487
488         (*/Diagonals*)
489         else if(i == 4){xinc = 1; yinc = 1;}(*/SE*)
490         else if(i == 5){xinc = -1; yinc = 1;}(*/SW*)
491         else if(i == 6){xinc = 1; yinc = -1;}(*/NE*)
492         else if(i == 7){xinc = -1; yinc = -1;}(*/NW*)
493
494         (*/Change all potential old markers*)
495         if(this.checkDir(move, xinc, yinc, player, true)) {
496             valid = true;
497         }
498     }
499
500     if (valid) { (*/Valid move*)
501         this.setLoc(move, player);
502         return true;
503     }
504     return false; (*/Invalid move*)
505 }
506
507 (**
508  * Checks all possible directions to determine whether or not a move is valid
509  * Uses the private method checkDir to make sure that the move is valid
510  *
511  * @param move The current move to be checked
512  * @param kind The current type of player's move
513  * @return Whether or not the move was valid
514  *)
515 public bool isValid(class LocationObj move, int kind) {
516     int yinc = 0;
517     int xinc = 0;
518     int[] board = this.board;
519     (*/If move is null, within the boundaries of the array, or move space is taken
↪ up *)
520     if(move == null or
521         ((move.getHorizontal() > 0 and move.getHorizontal() < 8
522          and move.getVertical() > 0 and move.getVertical() < 8) == false) or
523
524         this.board[(move.getHorizontal()*8)+move.getVertical()] != 2) {
525         return false;

```

```

526     }
527     int i;
528     for(i = 0; i < 8; i = i + 1){ (//For the length of potential neighbors*)
529         (//Linear Directions*)
530         if(i == 0){xinc = 1; yinc = 0;}(//E*)
531         else if(i == 1){xinc = -1; yinc = 0;}(//W*)
532         else if(i == 2){xinc = 0; yinc = 1;}(//S*)
533         else if(i == 3){xinc = 0; yinc = -1;}(//N*)
534
535         (//Diagonals*)
536         else if(i == 4){xinc = 1; yinc = 1;}(//SE*)
537         else if(i == 5){xinc = -1; yinc = 1;}(//SW*)
538         else if(i == 6){xinc = 1; yinc = -1;}(//NE*)
539         else if(i == 7){xinc = -1; yinc = -1;}(//NW*)
540
541         (//Move is valid*)
542         if (this.checkDir(move, xinc, yinc, kind, false)) {
543             return true;
544         }
545     }
546     (//Move was invalid for all directions*)
547     return false;
548 }
549
550 (**
551  * Determines whether or not there is a valid move available for a particular player
552  *
553  * @param player The current player's color
554  * @return Whether or not the particular player can play
555  *)
556 public bool userMoveAvailable(int player) {
557     int i;
558     int j;
559     int[] board = this.board;
560     for (i = 0 ; i < 8 ; i=i+1) {
561         for (j = 0 ; j < 8 ; j=j+1) {
562             (//If the potential position is not occupied and is a valid move*)
563             class LocationObj l = new LocationObj(j,i);
564             if ((board[(j*8)+i] == 2) and this.isValid(l,player)) {
565                 return true;
566             }
567         }
568     }
569     return false;
570 }
571
572 }
573
574

```

```
575 (**
576  * OthelloGame implementation of Game that appropriately defines initialize and
↪   playGame()
577  * Utilizes a Board object for maintaining and managing the board and the pieces on it
578  *
579  *
580  * @author David Watkins
581  * UNI djw2146
582  *)
583
584  (*****)
585
586  class OthelloGame {
587
588      private class Player p1;
589      private class Player p2;
590      private class Board board;
591
592      (*
593       * Initializes the boards and the players to begin a new game.
594       * Also gives p1 and p2 the initial four positions
595       * @see Game#initialize(Player, Player)
596       *)
597      public void initialize(class Player p1, class Player p2) {
598          this.p1 = p1;
599          this.p2 = p2;
600          this.board = new Board();
601          class Board b = this.board;
602          b.initializeBoard();
603
604          class LocationObj move1 = new LocationObj(3, 3);
605          class LocationObj move2 = new LocationObj(4, 4); (White moves *)
606
607          class LocationObj move3 = new LocationObj(4, 3);
608          class LocationObj move4 = new LocationObj(3, 4); (Black moves *)
609
610
611          (Both p1 and p2 need to be initialized on initial state
612           //p1 initialization*)
613
614          class Player p1 = this.p1;
615          p1.setResult(move1);
616          p1.setResult(move2);
617          p1.setResult(move3);
618          p1.setResult(move4);
619
620          (p2 initialization*)
621          class Player p2 = this.p2;
```



```

623     p2.setResult(move1);
624     p2.setResult(move2);
625     p2.setResult(move3);
626     p2.setResult(move4);
627
628
629 }
630
631
632 (*
633  * Main playGame method for the Othello game
634  * Continuously calls the various placeTile methods until a winner is found
635  * Returns the winner of the game in type Player
636  *
637  * @see Game#playGame()
638  *)
639 (* @Override *)
640
641
642 public class Player playGame() {
643     int turn = 1; (//Black always goes first *)
644     (* ===== *)
645     int moveSkip = 0;
646     class Board b = this.board;
647     while( (b.thereIsWinner() == -1 and moveSkip != 2) ){ (* //No winner yet and two
↪ moves weren't skipped *)
648         bool retry = false;
649         class LocationObj move = null;
650
651         if (not (b.userMoveAvailable(turn))) { (*//No valid moves available for
↪ user*)
652             (* //Switch turn *)
653             (* ===== *)
654             if (turn == 1) {turn = 0;}
655             else {turn = 1;}
656
657             moveSkip = moveSkip + 1;
658         }
659
660     else if (turn == 1) { (*//p1 turn *)
661         while(not (b.isValid(move, turn))) { (*//Get p1 move*)
662             class Player temp = this.p1;
663             move = temp.placeTile(retry);
664             retry = true;
665         }
666
667         b.setPlayerMove(move, turn);
668
669         class Player temp2 = new Player();

```

```
670         (* temp2.setResult(move); *)
671         (******)
672         this.p1 = temp2;
673
674         turn = 0;
675
676         moveSkip = 0;
677     }
678
679     else{        (/*p2 turn */)
680         class Player temp3 = new Player();
681         while (not b.isValid(move, turn)) { (/*get p2 move */)
682             temp3 = this.p2;
683             move = temp3.placeTile(retry);
684
685             retry = true;
686         }
687
688         b.setPlayerMove(move, turn);
689         class Player temp4 = new Player();
690         temp4 = this.p1;
691         (* temp4.setResult(move); *)
692         (******)
693         this.p1 = temp4;
694         turn = 1;
695         moveSkip = 0;
696     }
697
698
699     b.setPlayerMove(move, turn);
700     retry = false;
701 }
702
703 (/*The winner of the game */)
704 int winner = b.whoHasMore();
705
706 (* //Return winner *)
707 if(winner == 1)
708     return this.p1;
709 else if(winner == 0)
710     return this.p2;
711 else (/*Tie */)
712     return null;
713
714 class Player toReturn = new Player();
715 return toReturn; (* Default return, should never get called *)
716 }
717
718
```

719 }

REFERENCES

- [1] <http://www.gnu.org/software/gnu-c-manual/gnu-c-manual.html> *The GNU C Reference Manual..* N.p., n.d. Web. 26 Oct. 2015.
- [2] <https://docs.oracle.com/javase/specs/jls/se8/html/index.html> *The Java Language Specification..* N.p., n.d. Web. 26 Oct. 2015.
- [3] Edwards, Stephen. "Programming Language and Translators." Lecture.
- [4] "Control Flow Statements." *The Java Tutorials Learning the Java Language Language Basics* N.p., n.d. Web. 26 Oct. 2015.
- [5] https://en.wikipedia.org/wiki/Dice_%28programming_language%29 *The Dice Wikipedia Page..* N.p., n.d. Web. 22 Dec. 2015.