

Dice Project Report



"Java, but worse"

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

CONTENTS

1	Introduction	9
	Background	9
	Related Work	9
	Goals	10
	Cross-Platform	10
	Flexibility	10
	Transparency	10
	Familiarity	10
2	Language Tutorial	11
	Environment Setup	11
	Using the Compiler	11
	The basics	13
	Primitives	13
	Arrays	13
	Operators	14
	Control Flow	14
	Branching	14
	Loops	15
	Defining methods	16
	Classes and Inheritance	16
3	Language Reference Manual	19
	Introduction	19
	Types	19
	Primitive Types and Values	20
	Non-Primitive Types	20
	Casting	21
	Lexical Conventions	21
	Identifiers	21
	Keywords	21
	Literals	21
	Separators	22
	Operators	23
	White Space	23
	Comments	23
	Expressions and Operators	23
	Syntax Notation	23
	Primary Expressions	24
	Unary Operations	24
	Dynamic Memory Management	25

Multiplicative Operations	25
Additive Operations	25
Relational Operations	26
Equality Operations	26
Logical Operations	26
Assignment Operation	26
Statements	26
Control Flow Statements	26
File Inclusion	28
Declaration Statements	28
Program Structure and Scope	29
Program Structure	29
Classes	30
Class Declaration	30
Inheritance	30
Access Modifiers	30
Referencing instances	30
Grammar	30
4 Project Plan	36
Planning Process	36
Specification Process	36
Development Process	37
Testing Process	37
Team Responsibilities	37
Github Usernames	37
Project Log	37
Git Commit History	42
Software Development Environment	97
Programming Style Guide	97
5 Architecture	98
The Compiler	98
The Scanner	98
The Parser	99
The Semantic Analyzer	99
The Code Generator	100
The Utilities	102
Supplementary Code	108
The Standard Library	108
String	108
File	110
Integer	110
Built-in Functions	111
Functions Implemented in C	112
Declarations	112

6	Test Plan	113
	Testing Phases	113
	Unit Testing	113
	Integration Testing	113
	Automation	113
	Test Suites	114
	Dice to LL IR	114
	Hello World Example	114
	Class Extends Example	127
	For Loop Test	141
7	Lessons Learned	154
	David	154
	Emily	154
	Khaled	154
	Phillip	155
8	Code Listing	156
	_tags	156
	analyzer.ml	157
	ast.ml	183
	bindings.c	185
	codegen.ml	188
	conf.ml	208
	dice.ml	209
	exceptions.ml	215
	filepath.ml	217
	Makefile	221
	parser.mly	222
	processor.ml	228
	sast.ml	229
	scanner.mll	231
	stdlibe.dice	234
	utils.ml	242
	Test Suite Code	255
	tester.sh	255
	test-var1.dice.out	261
	test-stdlib-stringclass.dice.out	262
	test-stdlib-integerclass1.dice	263
	test-ConstructorInherited.dice	264
	test-ifEmptyBlock2.dice.out	265
	test-global1.dice.out	266
	test-if7.dice	267
	test-var3.dice	268
	test-classFunctionOverload1.dice.out	269
	test-applicative.dice	270
	test-forEmptyBlock2.dice	271
	test-if1.dice	272
	test-func5.dice	273
	test-arith5.dice	274

test-bool5.dice	275
test-constructor2.dice	276
test-arithSigned2.dice.out	277
test-classExtends2.dice	278
test-arithSigned1.dice.out	279
test-forEmptyBlock.dice	280
test-func5.dice.out	281
test-float.dice.out	282
test-stdlib-integerclass1.dice.out	283
test-for2.dice.out	284
test-if4.dice	285
test-arith7.dice	286
test-if5.dice	287
test-arithSigned3.dice	288
test-if7.dice.out	289
test-classGetter.dice.out	290
test-stdlib-compare.dice	291
test-class.dice.out	292
test-for1.dice.out	293
test-classInheritanceArgument.dice	294
test-whileBreak.dice	295
test-while1.dice	296
test-fileio.dice.out	297
test-classExtends2.dice.out	298
test-forContinue.dice.out	299
test-fib.dice	300
test-bool1.dice	301
test-forBreak.dice	302
test-bool6.dice	303
test-bool4.dice.out	304
test-stdlib-stringclassContains2.dice	305
test-classGetter.dice	306
test-var3.dice.out	307
test-forContinue.dice	308
test-stdlib-stringclassReverse.dice.out	309
test-while1.dice.out	310
test-float.dice	311
test-arith5.dice.out	312
test-array4.dice	313
test-arithSigned1.dice	314
test-if2.dice.out	315
test-stdlib-concat.dice	316
test-classReturnObjects.dice.out	317
test-if8.dice	318
test-stmts1.dice	319
test-if6.dice.out	320
test-classExtendsGetter.dice.out	321
test-ops1.dice.out	322
test-arith4.dice	323

test-func3.dice.out	324
test-class.dice	325
test-bool9.dice.out	326
test-whileContinue.dice.out	327
test-stdlib-copy.dice.out	328
test-stdlib-integerclass2.dice.out	329
test-classExtends.dice	330
test-if3.dice	331
test-bool8.dice.out	332
test-scope.dice.out	333
test-constructor1.dice	334
test-stdlib-concat.dice.out	335
test-forEmptyBlock2.dice.out	336
test-if4.dice.out	337
test-array.dice.out	338
test-array2.dice.out	339
test-objectDeclarationInheritance.dice.out	340
test-if5.dice.out	341
test-forEmptyBlock.dice.out	342
test-var4.dice.out	343
test-whileContinue.dice	344
test-array3.dice	345
test-if3.dice.out	346
test-arith6.dice	347
test-helloTwice.dice.out	348
test-stdlib-stringclassLength.dice.out	349
test-bool3.dice.out	350
test-hello.dice	351
test-array.dice	352
test-exit.dice	353
test-helloTwice.dice	354
test-arithSigned2.dice	355
test-cyclicalIncludes2.dice	356
test-if2.dice	357
test-constructorDefault.dice	358
test-var1.dice	359
test-arithSigned4.dice.out	360
test-ifEmptyBlock.dice.out	361
test-stdlib-compare.dice.out	362
test-cyclicalIncludes.dice.out	363
test-bool7.dice.out	364
test-classSetter.dice.out	365
test-stdlib-stringclassReverse.dice	366
test-factorialRecursive.dice	367
test-classInheritanceArgument.dice.out	368
test-constructorInherited.dice.out	369
test-bool8.dice	370
test-classFunctionOverload.dice	371
test-stdlib-stringclassContains.dice.out	372

test-arith8.dice	373
test-array4.dice.out	374
test-stdlib-copy.dice	375
test-arith7.dice.out	376
test-classFunctionOverload1.dice	377
test-stdlib-stringclassContains.dice	378
test-factorialRecursive.dice.out	379
test-stdlib-integerclass2.dice	380
test-bool6.dice.out	381
test-cyclicalIncludes.dice	382
test-bool1.dice.out	383
test-stdlib-stringclass3.dice	384
test-arith3.dice	385
test-emptyBlock.dice	386
test-intOverflow.dice.out	387
test-classFunctionOverload.dice.out	388
test-exit.dice.out	389
test-if1.dice.out	390
test-stdlib-stringclass2.dice	391
test-arith6.dice.out	392
test-stdlib-stringclassLength.dice	393
test-stdlib-stringclassContains2.dice.out	394
test-ops1.dice	395
test-stdlib-stringclass.dice	396
test-arith2.dice	397
test-float-max.dice	398
test-arith1.dice	399
test-stdlib-stringclass3.dice.out	400
test-ifEmptyBlock2.dice	401
test-array3.dice.out	402
test-arithSigned4.dice	403
test-classSetter.dice	404
test-classExtendsSetter.dice	405
test-gcd.dice	406
test-bool7.dice	407
test-classExtendsGetter.dice	408
test-func4.dice.out	409
test-constructor1.dice.out	410
test-fib.dice.out	411
test-forBreak.dice.out	412
test-func3.dice	413
test-scope.dice	414
test-objectDeclarationInheritance.dice	415
test-bool9.dice	416
test-if8.dice.out	417
test-hello.dice.out	418
test-fileio.dice	419
test-arith3.dice.out	420
test-float-max.dice.out	421

test-var4.dice	422
test-cyclicalIncludes2.dice.out	423
test-classExtendsSetter.dice.out	424
test-bool4.dice	425
test-bool2.dice	426
test-classExtends.dice.out	427
test-gcd.dice.out	428
test-bool2.dice.out	429
test-func4.dice	430
test-emptyBlock.dice.out	431
test-constructor2.dice.out	432
test-for2.dice	433
test-array2.dice	434
test-constructorDefault.dice.out	435
test-applicative.dice.out	436
test-stmts1.dice.out	437
test-global1.dice	438
test-intOverflow.dice	439
test-if6.dice	440
test-ifEmptyBlock.dice	441
test-arith1.dice.out	442
test-arith4.dice.out	443
test-whileBreak.dice.out	444
test-classReturnObjects.dice	445
test-stdlib-stringclass2.dice.out	446
test-intMax.dice.out	447
test-arith2.dice.out	448
test-for1.dice	449
test-bool3.dice	450
test-arithSigned3.dice.out	451
test-arith8.dice.out	452
test-intMax.dice	453
test-bool5.dice.out	454
test-args.dice.out	455
test-args.dice	456
E-test-cyclicalIncludesDuplicate.dice.out	457
E-test-objectCreation2.dice.out	458
E-test-scope3.dice	459
E-test-objectCreation2.dice	460
E-test-objectAssignMismatch.dice.out	461
E-test-cyclicalIncludes.dice.out	462
E-test-scope1.dice.out	463
E-test-objectCreation1.dice.out	464
E-test-scope2.dice.out	465
E-test-assignMismatch.dice.out	466
E-test-duplicate.dice	467
E-test-scope3.dice.out	468
E-test-objectCreation4.dice	469
E-test-constructor.dice	470

E-test-scope2.dice	471
E-test-constructor.dice.out	472
E-test-noReturn.dice	473
E-test-cyclicalIncludesDuplicate2.dice.out	474
E-test-objectCreation1.dice	475
E-test-cyclicalIncludes.dice	476
E-test-undefinedClass2.dice	477
E-test-mainClassNotDefined.dice	478
E-test-privateFieldsAccess.dice	479
E-test-duplicate.dice.out	480
E-test-stdlib-overload.dice.out	481
E-test-noReturn.dice.out	482
E-test-undefinedClass.dice	483
E-test-objectAssignMismatch.dice	484
E-test-privateFunctionAccess.dice.out	485
E-test-objectCreation3.dice.out	486
E-test-objectCreation3.dice	487
E-test-privateFieldsAccess.dice.out	488
E-test-assignMismatch2.dice.out	489
E-test-scope1.dice	490
E-test-stdlib-overload.dice	491
E-test-objectCreation4.dice.out	492
E-test-cyclicalIncludesDuplicate.dice	493
E-test-undefinedClass2.dice.out	494
E-test-cyclicalIncludesDuplicate2.dice	495
E-test-assignMismatch2.dice	496
E-test-privateFunctionAccess.dice	497
E-test-constructor1.dice	498
E-test-assignMismatch.dice	499
E-test-mainClassNotDefined.dice.out	500
E-test-undefinedClass.dice.out	501
E-test-constructor1.dice.out	502
test_pretty.dice	503
test.dice	504
primitives.dice	505
test_pretty.dice.ManualTokens	506
primitives.dice.ManualTokens	507
test.dice.ManualTokens	508
Demo_Animals.dice	509

9 References

514

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 [here](http://www.ubuntu.com/)¹. This is followed by downloading virtualbox and following the corresponding tutorial for setting up a custom Ubuntu VM [here](http://www.wikihow.com/Install-Ubuntu-on-VirtualBox)². 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 ($\langle \text{filename} \rangle . \langle \text{ext} \rangle \rightarrow \langle \text{filename} \rangle . \text{ll}$)

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
13        ↪ elements *)
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
12        ↪ true *)
13        f and g; not f; (* evaluate to false *)
14
15        c = -a;      (* c is now -3 *)
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
5          ↪ for loop *)
6          for (i = 0 ; i < 5 ; i = i + 1) {
7              a = a + 2;
8          }
9          (* a is now set to 10 *)
10
11          int b = 0;
12          int j;
13          for (j = 0 ; j < 5 ; j = j + 1) {
14              a = a + 2;
15              if(a >= 14){
16                  break;                (* will break out of the parent for
17                  ↪ loop *)
18              }
19              else { continue; } (* will skip the remaining code and start the
20              ↪ next iteration *)
21              b = b + 10;
22          }
23          (* b is still zero, a is 14 *)
24
25          while(b<5){
26              b = b + 1;
27          }
28      }
29  }
```



```
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         ↪ *)                    ↪ *)
27         this.yCoord = 0;
28         this.radius = 0;
29     }
30
31     constructor(int x, int y, int r){
32         this.xCoord = x;
33         this.yCoord = y;
34         this.radius = r;
35     }
36
37     public void myAction(){ (* This method overrides the one defined in parent
38     ↪ class *)                ↪ *)
39         print("circle\n");
40         print(this.radius);
41     }
42 }
43
44 class example8 {
45     public void main(char[][] args) {
46         class circle a = new circle(1, 2, 3);
47         class circle[] b = new class circle[10];
48         b[0] = a;
49         print(b[0].radius, "\n");
```

```
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

The keyword **this** is used in the body of method and constructor declarations to reference the instance of the object that the method or constructor will bind to 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     |         name
80
81 array_type:
82     type_tag LBRACKET brackets RBRACKET
83
84 datatype:
85     type_tag
86     |         array_type
87
88 brackets:
89     /* nothing */
90     |         brackets RBRACKET LBRACKET
91
92 stmt_list:
93     /* nothing */
94     |         stmt_list stmt
95
96 stmt:
97     expr SEMI
98     |         RETURN expr SEMI
99     |         RETURN SEMI
100    |         LBRACE stmt_list RBRACE
101    |         IF LPAREN expr RPAREN stmt
102    |         IF LPAREN expr RPAREN stmt ELSE stmt
103    |         FOR LPAREN expr_opt SEMI expr_opt SEMI expr_opt RPAREN stmt
104    |         WHILE LPAREN expr RPAREN stmt
105    |         BREAK SEMI
106    |         CONTINUE SEMI
107    |         datatype ID SEMI
108    |         datatype ID ASSIGN expr SEMI
109
110 expr_opt:
111     /* nothing */
112     |         expr
113
114 expr:
115     literals
116     |         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:

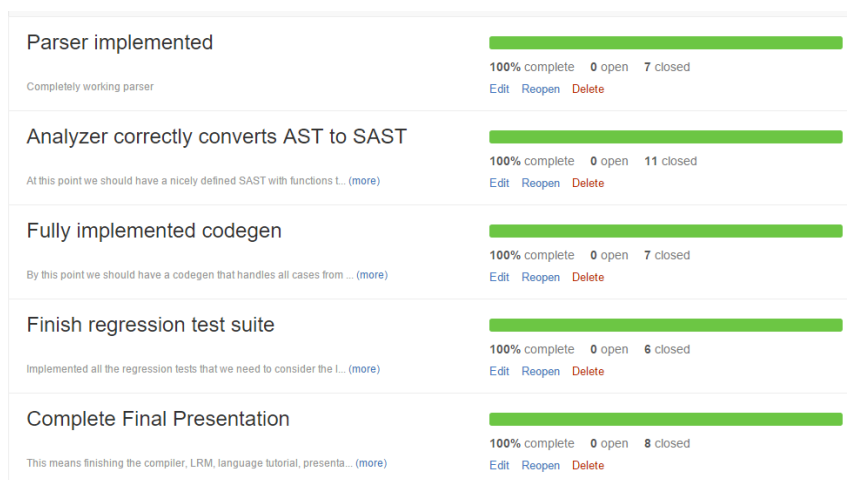


Figure 4.1: Milestoning on Github.

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
- Phillip Schiffrin - nethacker11

Project Log

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

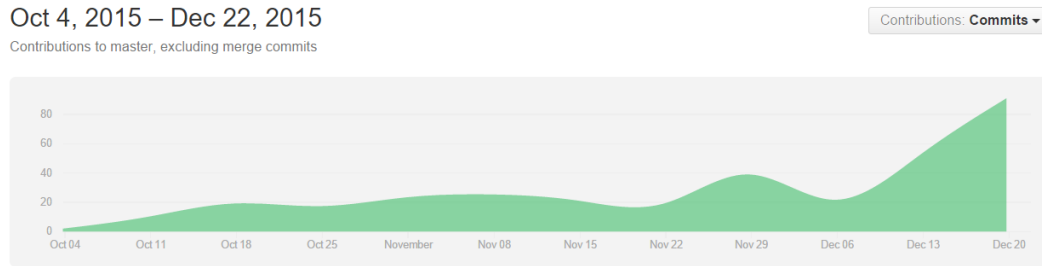


Figure 4.2: Commit timeline on Github.

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-ctor1.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 738b0558ddb9fe894a7611be0f1f9f590f38094a
2  Merge: 700e197 df6915a
3  Author: David Watkins <djrival7@gmail.com>
4  Date:   Tue Dec 22 20:45:33 2015 -0500
5
6      Merge branch 'master' of https://github.com/DavidWatkins/DiceLanguage
7
8  commit 700e1979474d977ffb3496c7435f4f9dbace09e2
9  Author: David Watkins <djrival7@gmail.com>
10 Date:   Tue Dec 22 20:39:19 2015 -0500
11
12      Added changes to standard library description
13
14  commit df6915a7d7a802e673daca3a6b364060b024035b
15  Merge: 8ac36b8 dbf27d2
16  Author: nethacker11 <philip.schiffrin@gmail.com>
17  Date:   Tue Dec 22 20:34:58 2015 -0500
18
19      Merge branch 'master' of https://github.com/DavidWatkins/Dice
20
21  commit 8ac36b8a6d9714d1f096f8c8e990da9bd971afe7
22  Author: nethacker11 <philip.schiffrin@gmail.com>
23  Date:   Tue Dec 22 20:34:51 2015 -0500
24
25      CFuncs.tex added
26
```

```
27 commit dbf27d2d940c93f0de61a210f98167faefeae014
28 Author: David Watkins <djrival7@gmail.com>
29 Date: Tue Dec 22 20:28:22 2015 -0500
30
31 Added grammar and small changes to lrm
32
33 commit 421588dcb8b30f42134c880143492e4822dbba2e
34 Author: nethacker11 <philip.schiffrin@gmail.com>
35 Date: Tue Dec 22 20:23:33 2015 -0500
36
37 added Builtin.tex
38
39 commit 0ec68907641d9be8a992b3dd7b023ec8e4f48afc
40 Merge: ef75162 ea3b98f
41 Author: David Watkins <djrival7@gmail.com>
42 Date: Tue Dec 22 19:46:19 2015 -0500
43
44 Merge branch 'master' of https://github.com/DavidWatkins/DiceLanguage
45
46 commit ef75162bd113e48a2ba794aa7ce002b613eeae3c
47 Author: David Watkins <djrival7@gmail.com>
48 Date: Tue Dec 22 19:45:53 2015 -0500
49
50 Added additional code for test plan in final report
51
52 commit ea3b98f6be0be4f8a66158b94d8391cd7b719948
53 Merge: 8524dfd 6378550
54 Author: nethacker11 <philip.schiffrin@gmail.com>
55 Date: Tue Dec 22 19:45:41 2015 -0500
56
57 Merge branch 'master' of https://github.com/DavidWatkins/Dice
58
59 commit 8524dfd397ddf421dcf7c7bd948649a825c355f5
60 Author: nethacker11 <philip.schiffrin@gmail.com>
61 Date: Tue Dec 22 19:45:33 2015 -0500
62
63 updated standard library in Library.tex
64
65 commit 63785501516be9117a65f4bc0908396b5496058c
66 Merge: 48d7e07 035c054
67 Author: David Watkins <djrival7@gmail.com>
68 Date: Tue Dec 22 19:12:36 2015 -0500
69
70 Merge branch 'master' of https://github.com/DavidWatkins/DiceLanguage
71
72 commit 48d7e0772b5cfe8e899efecb622041b198199497
73 Author: David Watkins <djrival7@gmail.com>
74 Date: Tue Dec 22 19:12:14 2015 -0500
75
```

```
76     Added additional stuff to proposal and tutorial
77
78     commit 035c054a00bf0ccc1b2b8d2dc1809f6fbab4dc08
79     Author: Khaled Atef <kaa2168@columbia.edu>
80     Date:    Tue Dec 22 19:11:12 2015 -0500
81
82     Added Test Plan and Khal lessons learned to Final Report directory
83
84     commit 39a768eca63505299bfacc07eef0322753a4de64
85     Author: nethacker11 <philip.schiffrin@gmail.com>
86     Date:    Tue Dec 22 18:59:25 2015 -0500
87
88     updated Syntax.tex for final report
89
90     commit 41e9106396bb0b2e693dd35bfd131151c7c1b641
91     Author: David Watkins <djw2146@columbia.edu>
92     Date:    Tue Dec 22 18:28:54 2015 -0500
93
94     ADedd more stuf
95
96     commit c58d595f376df552bb65e1fdb33ec05a674eb8cd
97     Author: David Watkins <davidw@tkins.me>
98     Date:    Tue Dec 22 18:23:32 2015 -0500
99
100    Added Demo_Animals to tex file
101
102    commit afa84191ecd40be39d14295c6c1f3fa25e7be6f6
103    Author: David Watkins <davidw@tkins.me>
104    Date:    Tue Dec 22 18:19:54 2015 -0500
105
106    Fixed hello world demo breaking tests
107
108    commit 7e2a1b9e07040cb9929b5dc971a297c83b0a9fe1
109    Author: David Watkins <davidw@tkins.me>
110    Date:    Tue Dec 22 18:14:31 2015 -0500
111
112    iejsiu
113
114    commit ab07735004b3f480677e269e65e9f009e9f10bdb
115    Author: David Watkins <davidw@tkins.me>
116    Date:    Tue Dec 22 18:12:52 2015 -0500
117
118    ijij
119
120    commit b21f0885522047bf0a62afb4da5edb292958ade4
121    Author: David Watkins <davidw@tkins.me>
122    Date:    Tue Dec 22 18:11:09 2015 -0500
123
124    Maybe this works?
```

```
125
126 commit 5dd98b2548b8718ebf8342ef3460bfa740a6ffad
127 Author: David Watkins <davidw@tkins.me>
128 Date: Tue Dec 22 15:20:54 2015 -0500
129
130     Fixed another bug
131
132 commit d6b49aae775f433bc4c733ef539f9d5b84605c6f
133 Author: David Watkins <djwt2146@columbia.edu>
134 Date: Tue Dec 22 15:19:31 2015 -0500
135
136     updated code.texY
137
138 commit d92d49c30ebe2da66203d0eb28db41d78b0d9ec5
139 Author: David Watkins <davidw@tkins.me>
140 Date: Tue Dec 22 15:17:11 2015 -0500
141
142     Fixed tests
143
144 commit cfebb0d5104705df4e358b35879645c6f5190439
145 Author: David Watkins <davidw@tkins.me>
146 Date: Tue Dec 22 15:09:13 2015 -0500
147
148     Fixed section title on tests
149
150 commit b8e048f0a326c3fc4fccee0a99928aa0564f8233
151 Merge: e94920a 07ee0b6
152 Author: David Watkins <davidw@tkins.me>
153 Date: Tue Dec 22 15:06:33 2015 -0500
154
155     Merge branch 'master' of https://github.com/DavidWatkins/Dice
156
157 commit e94920ae5f16643a0f5ae85393d7cae7e8dc58f5
158 Author: David Watkins <davidw@tkins.me>
159 Date: Tue Dec 22 15:06:16 2015 -0500
160
161     Added code for adding tests to final report
162
163 commit 07ee0b6cc870f6a7c171d159a85f8c142807f6f7
164 Author: David Watkins <DavidWatkins@users.noreply.github.com>
165 Date: Tue Dec 22 13:59:47 2015 -0500
166
167     Update README.md
168
169 commit a16003fbdec97727c492c857335bc93478a50a70
170 Author: David Watkins <djwt2146@columbia.edu>
171 Date: Tue Dec 22 05:15:01 2015 -0500
172
173     Added basis for final project report
```

```
174
175 commit f3e5fe83dae72565f2950c096c6ff0efecb1b567
176 Author: David Watkins <davidw@tkins.me>
177 Date: Tue Dec 22 04:46:38 2015 -0500
178
179     Need to fixed error tests
180
181 commit e16dc0448ac4444fe75f7fee46b10825fda2ba6d
182 Author: David Watkins <djrival7@gmail.com>
183 Date: Mon Dec 21 20:14:33 2015 -0500
184
185     Added presentation
186
187 commit 0bc2d56336f2bed25b1715a1b5c632a49147eea8
188 Author: Khaled Atef <kaa2168@columbia.edu>
189 Date: Mon Dec 21 15:25:43 2015 -0500
190
191     Logo modified
192
193 commit d39a5d9feb9ba50426b6caa3c32668ab57c410c5
194 Author: David Watkins <davidw@tkins.me>
195 Date: Mon Dec 21 14:23:01 2015 -0500
196
197     Finished demo code
198
199 commit c0ccf162f43b88ef2c732de15acd419250e5db5c
200 Author: David Watkins <davidw@tkins.me>
201 Date: Mon Dec 21 14:21:42 2015 -0500
202
203     Removed unnecessary files
204
205 commit a03afb187f7b93c8c05874e1357975d3edf69fac
206 Author: David Watkins <davidw@tkins.me>
207 Date: Mon Dec 21 13:55:16 2015 -0500
208
209     Fixed the demo
210
211 commit eec6e6f7989d4022ac261cc453bb7646e84e0a69
212 Author: Khaled Atef <kaa2168@columbia.edu>
213 Date: Mon Dec 21 07:31:49 2015 -0500
214
215     input/output coordinated
216
217 commit ca6abe8eeda764edfa1c2abd2bce730619ee53c9
218 Author: Khaled Atef <kaa2168@columbia.edu>
219 Date: Mon Dec 21 07:02:23 2015 -0500
220
221     basics implemented for demo
222
```

```
223 commit 8d2eda8d25c81a0294b3cc52c285c76314600870
224 Author: Khaled Atef <kaa2168@columbia.edu>
225 Date: Mon Dec 21 06:23:25 2015 -0500
226
227     modified ascii art for demo
228
229 commit 0a3a0c3958e224b1883714e99bd317624dd5514b
230 Merge: 96d30dd 2437414
231 Author: Khaled Atef <kaa2168@columbia.edu>
232 Date: Mon Dec 21 06:18:42 2015 -0500
233
234     Merge branch 'master' of https://github.com/DavidWatkins/Dice
235
236 commit 96d30ddc28576c7013902f157a5435315967ddd1
237 Author: Khaled Atef <kaa2168@columbia.edu>
238 Date: Mon Dec 21 06:18:36 2015 -0500
239
240     file for demo
241
242 commit 24374142973e158c61ea3955ac8d963599a2b75d
243 Author: Khaled Atef <kaa2168@columbia.edu>
244 Date: Mon Dec 21 05:57:03 2015 -0500
245
246     Othello still broken after many compiler errors
247
248 commit b5fbea0a2101e0c18d6bc476f0e7dfc18539c356
249 Merge: bff1792 502eff9
250 Author: Emily Chen <emchennyc@gmail.com>
251 Date: Mon Dec 21 02:39:56 2015 -0500
252
253     Merge branch 'master' of https://github.com/DavidWatkins/Dice
254
255 commit bff17927857bd562451279c9109ba57f01469829
256 Author: Emily Chen <emchennyc@gmail.com>
257 Date: Mon Dec 21 02:39:00 2015 -0500
258
259     halfway through translating OthelloGame
260
261 commit 502eff9a39c369dcd131c4b36220018c0e16fbc4
262 Merge: 4da809d 79744e6
263 Author: nethacker11 <philip.schiffrin@gmail.com>
264 Date: Mon Dec 21 02:35:23 2015 -0500
265
266     Merge branch 'master' of https://github.com/DavidWatkins/Dice
267
268 commit 4da809d3964870b705e10f8126e77e80c152474f
269 Author: nethacker11 <philip.schiffrin@gmail.com>
270 Date: Mon Dec 21 02:34:47 2015 -0500
271
```



```
272     updated humanplayer, doesn't work
273
274     commit 79744e6e61a16d7e049323d5af621e6be2049bb6
275     Merge: 1086a20 76df32a
276     Author: Khaled Atef <kaa2168@columbia.edu>
277     Date:   Mon Dec 21 02:10:20 2015 -0500
278
279     Merge branch 'master' of https://github.com/DavidWatkins/Dice
280
281     commit 1086a2003fcf4604b4b799b3c3e18cbb05901b48
282     Author: Khaled Atef <kaa2168@columbia.edu>
283     Date:   Mon Dec 21 02:10:11 2015 -0500
284
285     First round of edits to parserScanner regex rules
286
287     commit 76df32ae8b70759eeddb134f57b8e3f6403e2e5f
288     Merge: 8a75b65 fb0a776
289     Author: Emily Chen <emchennyc@gmail.com>
290     Date:   Mon Dec 21 02:08:18 2015 -0500
291
292     Merge branch 'master' of https://github.com/DavidWatkins/Dice
293
294     commit 8a75b65ddc464749d36e7998dcd243e8ef47b241
295     Author: Emily Chen <emchennyc@gmail.com>
296     Date:   Mon Dec 21 02:07:45 2015 -0500
297
298     includes classes HumanPlayer, Player, LocationObj
299
300     commit fb0a7763290ca205303a36e595792cab8bda14b
301     Author: nethacker11 <philip.schiffrin@gmail.com>
302     Date:   Mon Dec 21 02:04:24 2015 -0500
303
304     updated demo files
305
306     commit a7e0a84173eee4c06f0413a7b8bde8c3a3ee1844
307     Author: nethacker11 <philip.schiffrin@gmail.com>
308     Date:   Mon Dec 21 01:10:57 2015 -0500
309
310     updated demo stuff
311
312     commit c5882be1259eee843e06004c347cc1d047c79851
313     Merge: e91324a 15fe681
314     Author: nethacker11 <philip.schiffrin@gmail.com>
315     Date:   Sun Dec 20 23:38:05 2015 -0500
316
317     Merge branch 'master' of https://github.com/DavidWatkins/Dice
318
319     commit e91324aef67a7876f967e35b4f4a6ca323af95f7
320     Author: nethacker11 <philip.schiffrin@gmail.com>
```

```
321 Date:    Sun Dec 20 23:35:45 2015 -0500
322
323     added toInteger in stdlib
324
325 commit 15fe681f3b48135f96cfcf0c191bd6989b76fad9
326 Author: Khaled Atef <kaa2168@columbia.edu>
327 Date:    Sun Dec 20 22:19:03 2015 -0500
328
329     125 tests working!
330
331 commit 9dc00916011d9c69d13ff247268e615c2b0ac122
332 Author: David Watkins <davidw@tkins.me>
333 Date:    Sun Dec 20 21:50:00 2015 -0500
334
335     OthelloRunner Basic working
336
337 commit 9451871b5f68a79f41c4c463894b0cb6cf802b1f
338 Merge: e6007de bed598a
339 Author: Khaled Atef <kaa2168@columbia.edu>
340 Date:    Sun Dec 20 21:21:19 2015 -0500
341
342     Merge branch 'master' of https://github.com/DavidWatkins/Dice
343
344 commit e6007de0f670b43d7ff183860c77b95e0d381b99
345 Author: Khaled Atef <kaa2168@columbia.edu>
346 Date:    Sun Dec 20 21:21:04 2015 -0500
347
348     first draft Othello
349
350 commit bed598a8d60c21c69228029a024e7a5c3526c77d
351 Author: David Watkins <davidw@tkins.me>
352 Date:    Sun Dec 20 21:09:58 2015 -0500
353
354     Got object access working
355
356 commit d82e1a593479bd9dd04454014feedfa7dab7f0b4
357 Author: Khaled Atef <kaa2168@columbia.edu>
358 Date:    Sun Dec 20 20:53:29 2015 -0500
359
360     fileio test output works!
361
362 commit f000aa8d545bb8450340105b070501e9c242bcf1
363 Author: Khaled Atef <kaa2168@columbia.edu>
364 Date:    Sun Dec 20 20:50:32 2015 -0500
365
366     removed delete test
367
368 commit 9a1f7cde27e9c688ec84ad76385e27ffd1e7dcb1
369 Merge: 4c82a21 41949c7
```

```
370 Author: David Watkins <davidw@tkins.me>
371 Date: Sun Dec 20 20:45:44 2015 -0500
372
373 Merge branch 'master' of https://github.com/DavidWatkins/Dice
374
375 commit 41949c76776af134beb6de2a473e3e869403a2d5
376 Author: Khaled Atef <kaa2168@columbia.edu>
377 Date: Sun Dec 20 20:45:29 2015 -0500
378
379 Modified output to match test
380
381 commit 4c82a21756ba8abf9aa149d16f9b949e4b3f80c4
382 Author: David Watkins <davidw@tkins.me>
383 Date: Sun Dec 20 20:45:17 2015 -0500
384
385 test-fileio now prints and writes itself
386
387 commit f86d9cb3250e36ac60bcd42d65fce9d63bfda90
388 Merge: 39fea6b 0d28a10
389 Author: Khaled Atef <kaa2168@columbia.edu>
390 Date: Sun Dec 20 20:40:33 2015 -0500
391
392 Merge branch 'master' of https://github.com/DavidWatkins/Dice
393
394 commit 0d28a10d1ae9333877cdadd0f7eb7c99a587d561
395 Author: David Watkins <davidw@tkins.me>
396 Date: Sun Dec 20 20:39:55 2015 -0500
397
398 Fixed file io
399
400 commit 39fea6ba072e0eb973deadf72c28dc70140432c3
401 Author: Khaled Atef <kaa2168@columbia.edu>
402 Date: Sun Dec 20 20:23:11 2015 -0500
403
404 new tests
405
406 commit f989f8fcd03394dd759d65be5fa93406e7300fe8
407 Merge: ea1fc65 83d8ac3
408 Author: David Watkins <DavidWatkins@users.noreply.github.com>
409 Date: Sun Dec 20 19:17:14 2015 -0500
410
411 Merge pull request #135 from DavidWatkins/fix-overrides-check
412
413 check for overridden methods takes ret type into account
414
415 commit ea1fc652a4bdde559280c96e38a01cd5ac165783
416 Merge: 0c7039c 3163d40
417 Author: David Watkins <DavidWatkins@users.noreply.github.com>
418 Date: Sun Dec 20 19:16:39 2015 -0500
```

```
419
420     Merge pull request #134 from DavidWatkins/subclass_assignment
421
422     Subclass assignment
423
424     commit 0c7039c8d05f1a359ce8af67ed3fc0c581770539
425     Author: David Watkins <davidw@tkins.me>
426     Date:    Sun Dec 20 19:11:32 2015 -0500
427
428     Fixed assignment of obj_access problem
429
430     commit 6aeaa4c8a0d3fe6852c80263c918334a0d22dc06
431     Author: David Watkins <davidw@tkins.me>
432     Date:    Sun Dec 20 18:51:03 2015 -0500
433
434     Fixed stringClassReverse
435
436     commit 37ac35175eb27c39665b4bf77ee71d4a566bab4a
437     Author: David Watkins <davidw@tkins.me>
438     Date:    Sun Dec 20 18:26:16 2015 -0500
439
440     Added array access on obj_access
441
442     commit 83d8ac3fa9a130f8667cd6cf82691e8738bc94d4
443     Author: Emily Chen <emchennyc@gmail.com>
444     Date:    Sun Dec 20 18:12:23 2015 -0500
445
446     check for overridden methods takes ret type into account
447
448     commit 15d429843e5c9a584fa4914936df1ba3783b212f
449     Author: David Watkins <davidw@tkins.me>
450     Date:    Sun Dec 20 18:05:48 2015 -0500
451
452     Fixed array create initialize
453
454     commit f2390b94a80cfff1c217b533cac61d954bdfac3
455     Author: Khaled Atef <kaa2168@columbia.edu>
456     Date:    Sun Dec 20 17:49:22 2015 -0500
457
458     tests...
459
460     commit 3163d400ace38ecdc60f41b643a27b9fa60dcd26
461     Author: Emily Chen <emchennyc@gmail.com>
462     Date:    Sun Dec 20 17:44:45 2015 -0500
463
464     fixed formatting
465
466     commit ab4a07e9e55a5ce2db8f30782faa018b0762a53a
467     Author: David Watkins <davidw@tkins.me>
```

```
468 Date:    Sun Dec 20 17:39:11 2015 -0500
469
470     Changed function naming collision schema
471
472 commit e91e642ad5fbfd8a64bea0b5e2295aaeb3ff4145
473 Author: Emily Chen <emchennyc@gmail.com>
474 Date:    Sun Dec 20 17:20:20 2015 -0500
475
476     fixed subclass assignment not to raise exception with reg object creation
477
478 commit dba6456b40bf8fc2c032b34984c210c27352a4e2
479 Author: Emily Chen <emchennyc@gmail.com>
480 Date:    Sun Dec 20 16:48:52 2015 -0500
481
482     checks subclass assignment
483
484 commit 0b512528037bec86727f7e721a08d636759ef845
485 Author: Khaled Atef <kaa2168@columbia.edu>
486 Date:    Sun Dec 20 16:45:20 2015 -0500
487
488     more tests and fixes
489
490 commit dc3d893e18172bfa7fdb9733fb9990b22f26a3dc
491 Author: Khaled Atef <kaa2168@columbia.edu>
492 Date:    Sun Dec 20 16:12:12 2015 -0500
493
494     cyclical inheritance test added
495
496 commit 00009886c90714b113bd2e9066df7c0314fe99be
497 Author: Khaled Atef <kaa2168@columbia.edu>
498 Date:    Sun Dec 20 15:52:35 2015 -0500
499
500     inheritance object passed in arg test
501
502 commit 79585bfacf986d5b013396ecdea2c4ce1f078edd
503 Merge: ae4bcc4 b5d6640
504 Author: David Watkins <davidw@tkins.me>
505 Date:    Sun Dec 20 15:41:22 2015 -0500
506
507     Merge branch 'master' of https://github.com/DavidWatkins/Dice
508
509 commit ae4bcc4ec6860484529e4431d96531ce245a3823
510 Author: David Watkins <davidw@tkins.me>
511 Date:    Sun Dec 20 15:40:50 2015 -0500
512
513     Fixed way accessing inherited methods checker thing grammar english pls
514
515 commit b5d6640ecfe55fa20bc69d109be8ef38cb2df82a
516 Merge: 777db46 da9452f
```

```
517 Author: Khaled Atef <kaa2168@columbia.edu>
518 Date: Sun Dec 20 15:30:29 2015 -0500
519
520 Merge branch 'master' of https://github.com/DavidWatkins/Dice
521
522 commit 777db465f5de4f9ade562b56254806d86f884f88
523 Author: Khaled Atef <kaa2168@columbia.edu>
524 Date: Sun Dec 20 15:30:18 2015 -0500
525
526     more tests
527
528 commit b15dd23dd09a127b4b45eeef83bc8f284c86f3de
529 Author: Khaled Atef <kaa2168@columbia.edu>
530 Date: Sun Dec 20 15:02:14 2015 -0500
531
532     tests =0
533
534 commit da9452feecda712b24ae53419fc3858db4f7ffbb
535 Author: David Watkins <davidw@tkins.me>
536 Date: Sun Dec 20 15:00:04 2015 -0500
537
538     Fixed empty main problem
539
540 commit 7d23e2a16c131048d43fafa146b577ca5f18a8fb
541 Author: Khaled Atef <kaa2168@columbia.edu>
542 Date: Sun Dec 20 14:52:01 2015 -0500
543
544     fixed tests
545
546 commit dddd825bf32500fdd232c563c41b77a3e4426c44
547 Merge: 6b689f2 46d105a
548 Author: David Watkins <davidw@tkins.me>
549 Date: Sun Dec 20 14:51:10 2015 -0500
550
551 Merge branch 'master' of https://github.com/DavidWatkins/Dice
552
553 commit 6b689f2c8446921678637a0d876c4411bbaa360b
554 Author: David Watkins <davidw@tkins.me>
555 Date: Sun Dec 20 14:50:51 2015 -0500
556
557     Added casting to subtypes
558
559 commit 46d105aef7000673550854485f86d0359b0c8b00
560 Author: Khaled Atef <kaa2168@columbia.edu>
561 Date: Sun Dec 20 14:39:13 2015 -0500
562
563     more tests including cyclical includes
564
565 commit 81392df3b88074c974fe897d35ee65b3cfe026d4
```

```
566 Merge: 9ace750 9301a8c
567 Author: nethacker11 <philip.schiffrin@gmail.com>
568 Date: Sun Dec 20 14:06:38 2015 -0500
569
570 Merge branch 'master' of https://github.com/DavidWatkins/Dice
571
572 commit 9ace75050be810f9e0e460d47c409e972aaaa990
573 Author: nethacker11 <philip.schiffrin@gmail.com>
574 Date: Sun Dec 20 14:06:23 2015 -0500
575
576 added 2 tests
577
578 commit 9301a8c8bebadeb4cf67f4199b1084c9d25107b3
579 Merge: f9503b9 20c6b6c
580 Author: David Watkins <davidw@tkins.me>
581 Date: Sun Dec 20 14:05:44 2015 -0500
582
583 Merge branch 'master' of https://github.com/DavidWatkins/Dice
584
585 commit f9503b95b010f8c9516093fe1b9cac3f6e8a7f3c
586 Merge: df64b34 f17b85f
587 Author: David Watkins <davidw@tkins.me>
588 Date: Sun Dec 20 14:05:29 2015 -0500
589
590 Merge branch 'master' of https://github.com/DavidWatkins/Dice
591
592 commit 20c6b6c16425120bbe1da4d355178c054b384698
593 Author: Khaled Atef <kaa2168@columbia.edu>
594 Date: Sun Dec 20 14:01:26 2015 -0500
595
596 more tests passing
597
598 commit df64b347fd6e07abb2d4f0834da862231ff35cba
599 Author: David Watkins <davidw@tkins.me>
600 Date: Sun Dec 20 13:49:00 2015 -0500
601
602 Added some broken stuff
603
604 commit f17b85fedaf22ff07158c044185229a9d96f4f13
605 Author: nethacker11 <philip.schiffrin@gmail.com>
606 Date: Sun Dec 20 13:46:57 2015 -0500
607
608 added getchar()
609
610 commit 034b4a4e8a56c49e0de21385534708706f88f3af
611 Author: David Watkins <davidw@tkins.me>
612 Date: Sun Dec 20 12:58:20 2015 -0500
613
614 Functions now have working private scope
```

```
615
616 commit fef6f2a5139dd5dda3d0d00cb349898d584ac0da
617 Author: David Watkins <davidw@tkins.me>
618 Date: Sun Dec 20 12:32:55 2015 -0500
619
620     main args is now working
621
622 commit 47a6d182878aa980a372554b5eb7bd331cf60e7f
623 Author: David Watkins <davidw@tkins.me>
624 Date: Sun Dec 20 11:26:54 2015 -0500
625
626     Break and continue now work
627
628 commit a9be4f6c34ee4230620875dc92bd7f7489d66c5f
629 Author: David Watkins <davidw@tkins.me>
630 Date: Sun Dec 20 10:01:28 2015 -0500
631
632     Added code for checking if break or continue is valid
633
634 commit 795773d726798b0b7d698e35293f4ee76c2acdf4
635 Author: David Watkins <davidw@tkins.me>
636 Date: Sun Dec 20 09:37:20 2015 -0500
637
638     Added basic private checking, not working for inheritance
639
640 commit 2e1c681369eb3397f0de724572cdf413988efbaa
641 Author: David Watkins <davidw@tkins.me>
642 Date: Sun Dec 20 08:54:08 2015 -0500
643
644     Added casting at the beginning of overridden function
645
646 commit ca425b48bfa72b4f26d4f2be8bc92f69a4cb4fdf
647 Author: David Watkins <davidw@tkins.me>
648 Date: Sun Dec 20 08:35:36 2015 -0500
649
650     Added default constructor
651
652 commit 98e3f63c3121a86e40c4445ff4bdd7f7dff36893
653 Author: David Watkins <davidw@tkins.me>
654 Date: Sun Dec 20 08:06:45 2015 -0500
655
656     Virtual function resolution works
657
658 commit 145101c510c43fb8809e5fe2ccdd7de2e8ece722
659 Author: David Watkins <davidw@tkins.me>
660 Date: Sun Dec 20 06:56:25 2015 -0500
661
662     Added working vtbl
663
```



```
664 commit 21f7e5cc757e7f94f3d41e71c95590188119a15b
665 Author: David Watkins <davidw@tkins.me>
666 Date: Sun Dec 20 05:26:16 2015 -0500
667
668     Cleaned up use of types in SAST
669
670 commit 064f098e6ced5aa733a3beabf8edd3dda5173db3
671 Author: David Watkins <davidw@tkins.me>
672 Date: Sun Dec 20 05:12:03 2015 -0500
673
674     Added unused integer to all scalls
675
676 commit 9ee2d0ef828eff03f3acd0ed117610481d012135
677 Merge: 2042484 76746fd
678 Author: David Watkins <davidw@tkins.me>
679 Date: Sun Dec 20 05:01:45 2015 -0500
680
681     Merge branch 'master' of https://github.com/DavidWatkins/Dice
682
683 commit 2042484a2a9e8778eb1c4a86c00cb0ba8e5e0625
684 Author: David Watkins <davidw@tkins.me>
685 Date: Sun Dec 20 05:01:23 2015 -0500
686
687     Incorporated Emily's changes to Analyzer
688
689 commit 76746fdb001845cb72dd757f870fc985b4f2261a
690 Merge: fa8e2ee c0eedeb
691 Author: Khaled Atef <kaa2168@columbia.edu>
692 Date: Sun Dec 20 03:20:05 2015 -0500
693
694     Merge branch 'master' of https://github.com/DavidWatkins/Dice
695
696 commit fa8e2eea360f9b10e068fa1937317cafb003df12
697 Author: Khaled Atef <kaa2168@columbia.edu>
698 Date: Sun Dec 20 03:19:53 2015 -0500
699
700     more tests
701
702 commit c0eedebd7866f602cd79bf581ba5030f5a9a53e4
703 Author: David Watkins <davidw@tkins.me>
704 Date: Sun Dec 20 03:15:15 2015 -0500
705
706     Reformatted some code, fixed exit bug
707
708 commit 0a275a096762f01c506384a281c827a0689e8ab5
709 Author: Khaled Atef <kaa2168@columbia.edu>
710 Date: Sun Dec 20 02:20:27 2015 -0500
711
712     modified dice.ml to pass exceptions
```

```
713
714 commit d61f20801707ee4ac695135909823b3ee4b09073
715 Author: nethacker11 <philip.schiffrin@gmail.com>
716 Date: Sun Dec 20 00:18:20 2015 -0500
717
718     took out print stmt in stdlib
719
720 commit e1bc841aa24a9ef597e94232e04735c26c4276cd
721 Author: Khaled Atef <kaa2168@columbia.edu>
722 Date: Sun Dec 20 00:06:16 2015 -0500
723
724     More tests =)
725
726 commit 60a80460f04a4ffe25d7bbe319734bab7c8ebc82
727 Author: nethacker11 <philip.schiffrin@gmail.com>
728 Date: Sat Dec 19 22:45:15 2015 -0500
729
730     fixed concat in stdlib
731
732 commit 7ad7480ee90a8759271b0961507d7f084990a162
733 Author: Khaled Atef <kaa2168@columbia.edu>
734 Date: Sat Dec 19 21:25:12 2015 -0500
735
736     Added more tests and modified dice.ml to account for an exception to make the test script work
737
738 commit 1eeea68662d793173c0dd4587cd244eb379e3176
739 Merge: 3529056 50a7529
740 Author: David Watkins <davidw@tkins.me>
741 Date: Sat Dec 19 17:20:15 2015 -0500
742
743     Merge branch 'master' of https://github.com/DavidWatkins/Dice
744
745 commit 3529056aae15850c8e3ce00eb314e0393d5a1ff3
746 Author: David Watkins <davidw@tkins.me>
747 Date: Sat Dec 19 17:19:43 2015 -0500
748
749     Added changes to allow for exit
750
751 commit 50a7529746b3b7488fb038d75817983dcef56713
752 Merge: d2b04d3 3fd9fbf
753 Author: nethacker11 <philip.schiffrin@gmail.com>
754 Date: Sat Dec 19 17:16:14 2015 -0500
755
756     Merge branch 'master' of https://github.com/DavidWatkins/Dice
757
758 commit d2b04d339c239b0000ccfdde4b93ee3bfe13a878
759 Author: nethacker11 <philip.schiffrin@gmail.com>
760 Date: Sat Dec 19 17:15:51 2015 -0500
761
```

```
762     updated stdlib to include Integer and String has reverse()
763
764     commit 3fd9fbf47a382b0c8bc02e6f13e32c810f7f9807
765     Merge: 8ac9eed 14e1b19
766     Author: Khaled Atef <kaa2168@columbia.edu>
767     Date:   Sat Dec 19 16:46:38 2015 -0500
768
769     Merge branch 'master' of https://github.com/DavidWatkins/Dice
770
771     commit 8ac9eed3f00065424b59350a074749246d411869
772     Author: Khaled Atef <kaa2168@columbia.edu>
773     Date:   Sat Dec 19 16:46:20 2015 -0500
774
775     more tweaks to tests and script
776
777     commit 14e1b190bfb5972a1a0394a23be43f178eef971b
778     Author: David Watkins <davidw@tkins.me>
779     Date:   Sat Dec 19 16:31:22 2015 -0500
780
781     Fixed codegen for char_lits to i8_t
782
783     commit d984aff231ee6eb90e5921994f5d6fd14e044a79
784     Author: nethacker11 <philip.schiffrin@gmail.com>
785     Date:   Sat Dec 19 16:19:40 2015 -0500
786
787     added test cases and updated stdlib
788
789     commit ff79fff82264ba8743377b3515514df0a988d7fc
790     Merge: b336d0a 602dc41
791     Author: David Watkins <davidw@tkins.me>
792     Date:   Sat Dec 19 15:57:18 2015 -0500
793
794     Merge branch 'master' of https://github.com/DavidWatkins/Dice
795
796     commit b336d0a633387906a3a63d44541953e1c6a4616
797     Author: David Watkins <davidw@tkins.me>
798     Date:   Sat Dec 19 15:57:02 2015 -0500
799
800     Added modulo
801
802     commit 602dc4179efe1a87778934fb87428ddd5ee72d90
803     Author: Khaled Atef <kaa2168@columbia.edu>
804     Date:   Sat Dec 19 15:55:55 2015 -0500
805
806     corrected tester script to account for errors from exception tests
807
808     commit cedf61d44d4d5a1faf2424eb50cf983df9df22f3
809     Author: David Watkins <davidw@tkins.me>
810     Date:   Sat Dec 19 15:24:25 2015 -0500
```

```
811
812     Fixed function element access
813
814 commit 664bef08cd785fcd8f862874acfce3ced40bc5d2
815 Author: nethacker11 <philip.schiffrin@gmail.com>
816 Date:   Sat Dec 19 15:16:10 2015 -0500
817
818     added stdlib2 test and updated stdlib
819
820 commit f4a81c401d29969e5b341c99f2e68e003318bb2e
821 Merge: 285aa85 3b7465c
822 Author: nethacker11 <philip.schiffrin@gmail.com>
823 Date:   Sat Dec 19 15:14:14 2015 -0500
824
825     Merge branch 'master' of https://github.com/DavidWatkins/Dice
826
827 commit 3b7465cf892745766ce5a4bf08e4fdbdb28468eb
828 Author: David Watkins <davidw@tkins.me>
829 Date:   Sat Dec 19 15:13:45 2015 -0500
830
831     This time for sure!
832
833 commit 285aa8594fe6d3b1fea5e2983e5599cc19bec253
834 Merge: 3425edc d7ed17e
835 Author: nethacker11 <philip.schiffrin@gmail.com>
836 Date:   Sat Dec 19 15:10:53 2015 -0500
837
838     Merge branch 'master' of https://github.com/DavidWatkins/Dice
839
840 commit d7ed17e991fa0322eac0a63f0b55c84d4e2c1115
841 Author: David Watkins <davidw@tkins.me>
842 Date:   Sat Dec 19 15:10:09 2015 -0500
843
844     Fixed function param passing bug
845
846 commit 3425edceaa05209d8f57da67867dac753d0ea0bc
847 Merge: 41afbc1 97de937
848 Author: nethacker11 <philip.schiffrin@gmail.com>
849 Date:   Sat Dec 19 15:02:04 2015 -0500
850
851     Merge branch 'master' of https://github.com/DavidWatkins/Dice
852
853 commit 41afbc17bf2426ee44dc27bd254b550edbd78245
854 Author: nethacker11 <philip.schiffrin@gmail.com>
855 Date:   Sat Dec 19 15:02:02 2015 -0500
856
857     updated codegen for lseek
858
859 commit 97de93788f1701bcc7e334d8061fe58fca6a5d35
```

```
860 Author: David Watkins <davidw@tkins.me>
861 Date: Sat Dec 19 15:01:22 2015 -0500
862
863 Fixed codegen_call for lseek
864
865 commit 3d58076d10a7060f85dfb363ec5cbce759038257
866 Merge: 7413f9b 464fc4c
867 Author: nethacker11 <philip.schiffrin@gmail.com>
868 Date: Sat Dec 19 14:57:34 2015 -0500
869
870 Merge branch 'master' of https://github.com/DavidWatkins/Dice
871
872 commit 464fc4c5d6119034866a6118cce83e59a56b3520
873 Merge: 87f4d52 7a63abf
874 Author: David Watkins <davidw@tkins.me>
875 Date: Sat Dec 19 14:55:19 2015 -0500
876
877 Merge branch 'master' of https://github.com/DavidWatkins/Dice
878
879 commit 87f4d52f2e6d185d46bc29b8635c6f98d7eb7853
880 Author: David Watkins <davidw@tkins.me>
881 Date: Sat Dec 19 14:55:02 2015 -0500
882
883 Added lseek syntax to analyzer
884
885 commit 7413f9b0e14a20d00314556df6e6c4890fd243f3
886 Merge: 3c1c15b 7a63abf
887 Author: nethacker11 <philip.schiffrin@gmail.com>
888 Date: Sat Dec 19 14:22:33 2015 -0500
889
890 Merge branch 'master' of https://github.com/DavidWatkins/Dice
891
892 commit 7a63abffd7edafb87ecf82df2225e7ea2148eeb8
893 Merge: c482260 afae098
894 Author: Khaled Atef <kaa2168@columbia.edu>
895 Date: Sat Dec 19 14:22:02 2015 -0500
896
897 Merge branch 'master' of https://github.com/DavidWatkins/Dice
898
899 commit c48226078ee263889c6de75ff8d0f6f572a6a7ee
900 Author: Khaled Atef <kaa2168@columbia.edu>
901 Date: Sat Dec 19 14:21:41 2015 -0500
902
903 added stdlib string
904
905 commit 3c1c15b99113b0e570fa6625ba4a2a0ee1c917e5
906 Merge: 480dc4d afae098
907 Author: nethacker11 <philip.schiffrin@gmail.com>
908 Date: Sat Dec 19 14:19:44 2015 -0500
```

```
909
910     Merge branch 'master' of https://github.com/DavidWatkins/Dice
911
912     commit 480dc4d0c15a9c5cd5bccfb7c8d05aebb423b9e7
913     Author: nethacker11 <philip.schiffrin@gmail.com>
914     Date:   Sat Dec 19 14:18:18 2015 -0500
915
916         changed stdlib
917
918     commit afae098e32e66e69b0349e9809ce6d237f451179
919     Merge: acbea61 404c6df
920     Author: David Watkins <davidw@tkins.me>
921     Date:   Sat Dec 19 14:17:52 2015 -0500
922
923     Merge branch 'master' of https://github.com/DavidWatkins/Dice
924
925     commit acbea6113ddccfa59ce06c8288a4bfe81b134f6f
926     Author: David Watkins <davidw@tkins.me>
927     Date:   Sat Dec 19 14:17:31 2015 -0500
928
929         Fixed right associativity of parser
930
931     commit 404c6df62cc80b61ceffed8cc666f9591757d5e0
932     Merge: 782ca3f 3e4e5e6
933     Author: Khaled Atef <kaa2168@columbia.edu>
934     Date:   Sat Dec 19 14:15:07 2015 -0500
935
936     Merge branch 'master' of https://github.com/DavidWatkins/Dice
937
938     commit 17c1362a3d24b7edf544491948a259fb816524a4
939     Merge: c248f39 3e4e5e6
940     Author: nethacker11 <philip.schiffrin@gmail.com>
941     Date:   Sat Dec 19 14:15:07 2015 -0500
942
943     Merge branch 'master' of https://github.com/DavidWatkins/Dice
944
945     commit c248f394794dc1a26b0052db05a4a2abffe5ba89
946     Author: nethacker11 <philip.schiffrin@gmail.com>
947     Date:   Sat Dec 19 14:15:05 2015 -0500
948
949         updated stdlib
950
951     commit 782ca3fa5c903b0c87d7402724934c25a3cf3a30
952     Author: Khaled Atef <kaa2168@columbia.edu>
953     Date:   Sat Dec 19 14:14:49 2015 -0500
954
955         modified tests
956
957     commit 3e4e5e6b27248dbe9de6af579040dbc991f2b5be
```

```
958 Author: David Watkins <davidw@tkins.me>
959 Date: Sat Dec 19 14:13:16 2015 -0500
960
961 Fixed array access for chars
962
963 commit cbcdf6c41b458da3355bf3aecb58a5d3549752e
964 Merge: 3ca5e39 0c9870c
965 Author: David Watkins <davidw@tkins.me>
966 Date: Sat Dec 19 13:54:44 2015 -0500
967
968 Merge branch 'master' of https://github.com/DavidWatkins/Dice
969
970 commit 3ca5e39a56c7a6c239d38e9c58eabd03304f1526
971 Author: David Watkins <davidw@tkins.me>
972 Date: Sat Dec 19 13:54:14 2015 -0500
973
974 Fixed array acces for strings
975
976 commit 0c9870c3948b1e193926f363ec551830d8aae9ae
977 Author: Khaled Atef <kaa2168@columbia.edu>
978 Date: Sat Dec 19 13:54:04 2015 -0500
979
980 added more tests
981
982 commit 91c9bc47dff55afd6269202ad1654145cf55b5da
983 Author: David Watkins <davidw@tkins.me>
984 Date: Sat Dec 19 05:07:25 2015 -0500
985
986 Fixed stdlib
987
988 commit c603715b9036aa50daa30a423ee6e0b30fd9e8ce
989 Author: David Watkins <davidw@tkins.me>
990 Date: Sat Dec 19 04:08:52 2015 -0500
991
992 While loops work
993
994 commit 27b53ff8e9131b2e686ed29755d54690936a2131
995 Author: David Watkins <davidw@tkins.me>
996 Date: Sat Dec 19 04:02:09 2015 -0500
997
998 Fixed bug with array length
999
1000 commit 64b72feeb55e71b92c1fd7810e5ccb82ae736f41
1001 Author: David Watkins <davidw@tkins.me>
1002 Date: Sat Dec 19 03:39:57 2015 -0500
1003
1004 Fixed odd incorrect ordering bug
1005
1006 commit 170e4fd2e2285c0d7f106426651199a48c5b20e6
```

```
1007 Author: David Watkins <davidw@tkins.me>
1008 Date: Sat Dec 19 03:34:00 2015 -0500
1009
1010 Fixed includes bug, fixed char array assignment of int length
1011
1012 commit 7c8d274ea55d5118e70db8f3d11dd5cff42d36e4
1013 Author: David Watkins <davidw@tkins.me>
1014 Date: Sat Dec 19 01:36:29 2015 -0500
1015
1016 Migrated files and folders to appropriate place for new makefile schema
1017
1018 commit a1ae8ffbc1d1fe84c755abf98a44392680a63c20
1019 Author: nethacker11 <philip.schiffrin@gmail.com>
1020 Date: Fri Dec 18 22:57:30 2015 -0500
1021
1022 updated stdlib and analyzer and codegen for built in functions
1023
1024 commit 1a5244813f0c299c673096a48a09dad022133599
1025 Author: David Watkins <davidw@tkins.me>
1026 Date: Fri Dec 18 20:01:52 2015 -0500
1027
1028 Fixed \0, its now \000
1029
1030 commit a2d07124a44c96af5b158996c049fead07644dc5
1031 Author: nethacker11 <philip.schiffrin@gmail.com>
1032 Date: Fri Dec 18 20:02:58 2015 -0500
1033
1034 updated stdlib.dice
1035
1036 commit e9c8d476beb76ebd9a4f4d1a23f5cf722d741744
1037 Author: David Watkins <davidw@tkins.me>
1038 Date: Fri Dec 18 19:47:00 2015 -0500
1039
1040 backslash zero yo
1041
1042 commit d6be8f34690274401b8123cf491254274e8030b9
1043 Author: David Watkins <davidw@tkins.me>
1044 Date: Fri Dec 18 19:33:09 2015 -0500
1045
1046 works now?
1047
1048 commit 8ad670e00d5b7cf8020581861306cf89ab17b8a6
1049 Merge: aec396d c6aflee
1050 Author: David Watkins <davidw@tkins.me>
1051 Date: Fri Dec 18 19:17:00 2015 -0500
1052
1053 Merge branch 'master' of https://github.com/DavidWatkins/Dice
1054
1055 commit aec396db7c9a6714ce6e5de976596b42c1d03c8e
```



```
1056 Author: David Watkins <davidw@tkins.me>
1057 Date:   Fri Dec 18 19:16:41 2015 -0500
1058
1059     Works *crosses fingers*
1060
1061 commit c6af1eecd3362b57591589d61493c14707c11479
1062 Author: nethacker11 <philip.schiffrin@gmail.com>
1063 Date:   Fri Dec 18 19:13:08 2015 -0500
1064
1065     updated stdlib.dice
1066
1067 commit b0e033a148286f9de9c2cef0b37c799fb5ec36d0
1068 Author: David Watkins <davidw@tkins.me>
1069 Date:   Fri Dec 18 18:43:07 2015 -0500
1070
1071     So uh, nested comments are a thing
1072
1073 commit 0e91f6aca66d2804747918f460114f356842befd
1074 Author: Khaled Atef <kaa2168@columbia.edu>
1075 Date:   Fri Dec 18 17:37:31 2015 -0500
1076
1077     Exceptions folder created, need to add more tests here
1078
1079 commit 643197852baaf3fff864761ab7376bf32e6bacf0
1080 Author: nethacker11 <philip.schiffrin@gmail.com>
1081 Date:   Fri Dec 18 17:12:04 2015 -0500
1082
1083     added stdlib.dice, passes analyzer but not tested
1084
1085 commit b9c354db5a56e4d8e9543a1c00147260283e5d51
1086 Merge: 75cb0da 5ae669c
1087 Author: Khaled Atef <kaa2168@columbia.edu>
1088 Date:   Fri Dec 18 03:46:41 2015 -0500
1089
1090     Merge branch 'master' of https://github.com/DavidWatkins/Dice
1091
1092 commit 75cb0daf1e69f062f9cb1e6c66079639ededd3e0
1093 Author: Khaled Atef <kaa2168@columbia.edu>
1094 Date:   Fri Dec 18 03:41:15 2015 -0500
1095
1096     modified test script
1097
1098 commit 5ae669cf25734ab2bdfb6c989bfd933b98bdebb9
1099 Author: David Watkins <davidw@tkins.me>
1100 Date:   Thu Dec 17 19:26:41 2015 -0500
1101
1102     Works?
1103
1104 commit 1cfe2ae2cf20eb203f45617097d9daa93abf3793
```

```
1105 Author: nethacker11 <philip.schiffrin@gmail.com>
1106 Date: Thu Dec 17 19:24:59 2015 -0500
1107
1108     added write function
1109
1110 commit 013f06fe8fcbf6d8db7dcf2cd32af311d47b7f2c
1111 Merge: 4554586 b0dcfe9
1112 Author: nethacker11 <philip.schiffrin@gmail.com>
1113 Date: Thu Dec 17 18:59:31 2015 -0500
1114
1115     Merge branch 'master' of https://github.com/DavidWatkins/Dice
1116
1117 commit 4554586421327badd3daf2acb2c212fb98303a53
1118 Author: nethacker11 <philip.schiffrin@gmail.com>
1119 Date: Thu Dec 17 18:59:29 2015 -0500
1120
1121     added more build in function declarations
1122
1123 commit b0dcfe9708793c4946a123dbf45afea8c305027e
1124 Author: David Watkins <davidw@tkins.me>
1125 Date: Thu Dec 17 18:58:19 2015 -0500
1126
1127     Fixed shift/reduce, added linking of c functions
1128
1129 commit 9c7a140e1e036a70bb4af3159d21265a2799bcaf
1130 Author: nethacker11 <philip.schiffrin@gmail.com>
1131 Date: Thu Dec 17 18:06:58 2015 -0500
1132
1133     added c function declarations in codegen.ml under built in functions
1134
1135 commit d04c2b99e7c467914839a1b6429d7284d8c78725
1136 Author: nethacker11 <philip.schiffrin@gmail.com>
1137 Date: Thu Dec 17 17:37:41 2015 -0500
1138
1139     added folder for c library extensions for .bc files to be linked in dice.ml
1140
1141 commit d058e9c00fc86b609da0dce4a906b10718ca3430
1142 Author: David Watkins <davidw@tkins.me>
1143 Date: Wed Dec 16 16:55:34 2015 -0500
1144
1145     Added delete command to free memory
1146
1147 commit 9414ee274b553debcc02a052fff0fd34e46e14e8
1148 Author: David Watkins <davidw@tkins.me>
1149 Date: Wed Dec 16 16:29:17 2015 -0500
1150
1151     Added multi-dimensional c code
1152
1153 commit a08e96f67a96dd181abf6c67b769d371c326fa03
```

```
1154 Author: David Watkins <davidw@tkins.me>
1155 Date:   Wed Dec 16 16:28:52 2015 -0500
1156
1157     Array length working, also added multi-dimensional c code
1158
1159 commit 59e4b9b012b92b799cdac22849c667130731163a
1160 Author: David Watkins <davidw@tkins.me>
1161 Date:   Wed Dec 16 01:41:37 2015 -0500
1162
1163     Array primitives work
1164
1165 commit 3ab1e0ff494e1bdc57460aa3d930b5f15fe3c0a6
1166 Author: David Watkins <davidw@tkins.me>
1167 Date:   Tue Dec 15 23:45:42 2015 -0500
1168
1169     Fixed single dimension arrays
1170
1171 commit f4ccfe7371bdd8c051db4735db872885a0578f42
1172 Author: nethacker11 <philip.schiffrin@gmail.com>
1173 Date:   Tue Dec 15 22:20:45 2015 -0500
1174
1175     build_array_malloc in progress
1176
1177 commit 3e27ec7a42f5d91620f8c16390cf53a82f9e858f
1178 Author: nethacker11 <philip.schiffrin@gmail.com>
1179 Date:   Tue Dec 15 19:20:37 2015 -0500
1180
1181     changing to single dimensional arrays, compiles but looks like arraycreate is not accessed again
1182
1183 commit 10e87f3b9c82258c06247f972144d63f582dbc4c
1184 Author: David Watkins <davidw@tkins.me>
1185 Date:   Tue Dec 15 18:44:34 2015 -0500
1186
1187     Working status
1188
1189 commit c71bfa88710ef0a7c39f98fb4ece382a6dbb877c
1190 Author: David Watkins <davidw@tkins.me>
1191 Date:   Sat Dec 12 19:04:56 2015 -0500
1192
1193     ArrayCreate doesn't work, added code for array deref
1194
1195 commit b9ed042660504b766617f397d21c8858756f4f95
1196 Author: David Watkins <davidw@tkins.me>
1197 Date:   Sat Dec 12 18:57:28 2015 -0500
1198
1199     Added basic array methods
1200
1201 commit cdc675d5c824d42a7e82749a2472bb1da8726008
1202 Author: David Watkins <davidw@tkins.me>
```

```
1203 Date:   Fri Dec 11 15:28:10 2015 -0500
1204
1205     Fixed bug where constructors weren't being checked by name
1206
1207 commit 8346a0009480db6799587fa8a1b3ab0178c5ea43
1208 Author: David Watkins <DavidWatkins@users.noreply.github.com>
1209 Date:   Thu Dec 10 18:18:39 2015 -0500
1210
1211     Update README.md
1212
1213 commit b33b3a318fc92dbcdf434c72888f0c2399b3173f
1214 Author: David Watkins <davidw@tkins.me>
1215 Date:   Tue Dec 8 17:23:53 2015 -0500
1216
1217     Added help printing to compiler with no arguments
1218
1219 commit ec57d8062f137244729246260d34a7cd47641525
1220 Merge: ae65af0 bb7a89b
1221 Author: David Watkins <DavidWatkins@users.noreply.github.com>
1222 Date:   Sun Dec 6 17:21:44 2015 -0500
1223
1224     Merge pull request #79 from DavidWatkins/Kappa
1225
1226     Kappa
1227
1228 commit bb7a89b50cd1045cd8c0b711288d7bead3f8af20
1229 Merge: ae65af0 43e4e3b
1230 Author: David Watkins <davidw@tkins.me>
1231 Date:   Sun Dec 6 17:21:21 2015 -0500
1232
1233     Merge branch 'Kappa' of https://github.com/DavidWatkins/Dice into Kappa
1234
1235 commit ae65af04ea8c138768db9f1e25249d4c564d9882
1236 Merge: 914b15a df7d695
1237 Author: David Watkins <DavidWatkins@users.noreply.github.com>
1238 Date:   Sat Dec 5 21:31:11 2015 -0500
1239
1240     Merge pull request #77 from DavidWatkins/ObjAccess
1241
1242     Obj access
1243
1244 commit df7d695d9f61cb7709ac5bd24422a23691d969dc
1245 Author: David Watkins <davidw@tkins.me>
1246 Date:   Sat Dec 5 21:29:47 2015 -0500
1247
1248     Classes are now working, fixed tests to match up with new rules
1249
1250 commit 3547bd54ce8e66a8d984ecac37ef478f43d1d773
1251 Author: David Watkins <davidw@tkins.me>
```

```
1252 Date:   Fri Dec 4 15:39:07 2015 -0500
1253
1254     Sigh
1255
1256 commit 914b15a3301e9de97ff5b9fcbf57f7731fbd90a0
1257 Merge: bc5da4f b474701
1258 Author: Khaled Atef <kaa2168@columbia.edu>
1259 Date:   Fri Dec 4 01:27:57 2015 -0500
1260
1261     Merge branch 'master' of https://github.com/DavidWatkins/Dice
1262
1263 commit bc5da4f925a2a6995b3d79cff92fff0f87f0384d
1264 Author: Khaled Atef <kaa2168@columbia.edu>
1265 Date:   Fri Dec 4 01:27:04 2015 -0500
1266
1267     added else if tests
1268
1269 commit 43e4e3bf1d4a64e5fa71b3642a21250f37bb7334
1270 Author: Khaled Atef <kaa2168@columbia.edu>
1271 Date:   Fri Dec 4 01:14:26 2015 -0500
1272
1273     unop working
1274
1275 commit 2fedba447dd85d89582b3aad84c0a470db87de7c
1276 Author: David Watkins <davidw@tkins.me>
1277 Date:   Wed Dec 2 17:14:26 2015 -0500
1278
1279     Still WIP
1280
1281 commit a0c3cbf70c80847b0892ef61c9bf34c109ca1f49
1282 Author: David Watkins <davidw@tkins.me>
1283 Date:   Wed Dec 2 15:56:52 2015 -0500
1284
1285     Added sample test script
1286
1287 commit a639719f7a7d885a4008be87ac94cbe5ec170695
1288 Author: David Watkins <davidw@tkins.me>
1289 Date:   Wed Dec 2 15:56:03 2015 -0500
1290
1291     WIP
1292
1293 commit b47470171b10bbe3b8f7bcc9f7f0e52bf73a01e1
1294 Author: David Watkins <DavidWatkins@users.noreply.github.com>
1295 Date:   Wed Dec 2 07:33:33 2015 -0500
1296
1297     Update README.md
1298
1299 commit 15e55374aea051650f8f627205dbcc8160544a75
1300 Author: David Watkins <davidw@tkins.me>
```

```
1301 Date:   Wed Dec 2 06:48:25 2015 -0500
1302
1303     Function parameters are working
1304
1305 commit 0ff181573ba6a1fea1105ecc4b72f8fb269db965
1306 Author: David Watkins <davidw@tkins.me>
1307 Date:   Wed Dec 2 06:00:30 2015 -0500
1308
1309     Added basic function calls to compiler
1310
1311 commit d99e2cc2f5b17ce3826ffe4aa0c6bc39e8297465
1312 Author: Khaled Atef <kaa2168@columbia.edu>
1313 Date:   Wed Dec 2 04:26:03 2015 -0500
1314
1315     unop implemented, but not working. All tests are failing.
1316
1317 commit aa1368f6872e5c20d669f38614fd431e3b21c65
1318 Author: David Watkins <davidw@tkins.me>
1319 Date:   Wed Dec 2 03:48:01 2015 -0500
1320
1321     Added lazy evaluation and fixed error with function names
1322
1323 commit d0fa8223f546f315afc023d637f240af34329e36
1324 Author: David Watkins <davidw@tkins.me>
1325 Date:   Wed Dec 2 03:06:35 2015 -0500
1326
1327     Changed wording in helper
1328
1329 commit 74059d062fdbbdc1679dac574052c05459751c08
1330 Author: David Watkins <davidw@tkins.me>
1331 Date:   Wed Dec 2 03:04:36 2015 -0500
1332
1333     Added the ability to compile to a file
1334
1335 commit 2078c5fdb94b2cc6d265617c7d12d81e507c7e57
1336 Author: Khaled Atef <kaa2168@columbia.edu>
1337 Date:   Wed Dec 2 02:15:27 2015 -0500
1338
1339     corrected test-bool4.dice
1340
1341 commit c0d5caee65fb50c2aa083309957bd1b20dba1c1c
1342 Author: David Watkins <davidw@tkins.me>
1343 Date:   Wed Dec 2 01:58:56 2015 -0500
1344
1345     Float comparison expressions now evaluate properly
1346
1347 commit c6bb01085947ef3f51cbdc885238c3964039b708
1348 Author: Khaled Atef <kaa2168@columbia.edu>
1349 Date:   Wed Dec 2 01:29:37 2015 -0500
```

```
1350
1351     fixed tests and added more for bools
1352
1353 commit 0d9c3a0dfe3b893c500f75090b5f04c89bb4401c
1354 Merge: 63fdb09 a2300ae
1355 Author: David Watkins <davidw@tkins.me>
1356 Date:   Wed Dec 2 01:25:58 2015 -0500
1357
1358     Merge branch 'master' of https://github.com/DavidWatkins/Dice
1359
1360 commit 63fdb093f7254bc4934fdde8b564b1d97463eada
1361 Author: David Watkins <davidw@tkins.me>
1362 Date:   Wed Dec 2 01:25:27 2015 -0500
1363
1364     Added printing string representations of boolean values to codgen
1365
1366 commit a2300aedb2fe11c3fa612e1ae8f7d60537fb3019
1367 Author: Khaled Atef <kaa2168@columbia.edu>
1368 Date:   Wed Dec 2 00:36:29 2015 -0500
1369
1370     Fixed syntax error
1371
1372 commit 861aee2ddb899d888a13e2a42e5a81c0a1528cd4
1373 Merge: e7494e3 b0ab4a8
1374 Author: Khaled Atef <kaa2168@columbia.edu>
1375 Date:   Wed Dec 2 00:16:32 2015 -0500
1376
1377     Merge branch 'master' of https://github.com/DavidWatkins/Dice
1378
1379 commit b0ab4a8e92319f72c3d1bb2376475b424cbf1887
1380 Author: David Watkins <davidw@tkins.me>
1381 Date:   Wed Dec 2 00:16:10 2015 -0500
1382
1383     Reverted change to printing floats
1384
1385 commit e7494e3b6488dc49d28bb3bcad6e77f7ea42d265
1386 Merge: d969ca2 21ac0fa
1387 Author: Khaled Atef <kaa2168@columbia.edu>
1388 Date:   Wed Dec 2 00:11:39 2015 -0500
1389
1390     wMerge branch 'master' of https://github.com/DavidWatkins/Dice
1391
1392 commit d969ca2fc12093e18f946e893328b3cdb788ff43
1393 Author: Khaled Atef <kaa2168@columbia.edu>
1394 Date:   Wed Dec 2 00:11:08 2015 -0500
1395
1396     nested if tests added with boolean tests of logical operators
1397
1398 commit 21ac0fa10db8c24347a0a56ed39cfc1b92e7ae19
```

```
1399 Author: David Watkins <davidw@tkins.me>
1400 Date:   Wed Dec 2 00:07:50 2015 -0500
1401
1402     Fixed printing of floats
1403
1404 commit 4ca9ff15d8016c5fe78f81a23eb2b5bc19a443de
1405 Author: David Watkins <davidw@tkins.me>
1406 Date:   Tue Dec 1 23:52:18 2015 -0500
1407
1408     Added exception for invalid integer operation in codegen
1409
1410 commit 9c25e446d76918a3b11be98a9d0aef72f2345e57
1411 Merge: c45b5f8 72718b2
1412 Author: David Watkins <DavidWatkins@users.noreply.github.com>
1413 Date:   Tue Dec 1 23:50:17 2015 -0500
1414
1415     Merge pull request #68 from DavidWatkins/Kappa
1416
1417     Kappa
1418
1419 commit 72718b24c9c77818965340c2642b9746452517f9
1420 Merge: 2031096 c45b5f8
1421 Author: David Watkins <davidw@tkins.me>
1422 Date:   Tue Dec 1 23:49:47 2015 -0500
1423
1424     Merge branch 'master' into Kappa
1425
1426 commit 203109635a92704afcaf6ba8f7686e4bc56ee463
1427 Author: Khaled Atef <kaa2168@columbia.edu>
1428 Date:   Tue Dec 1 22:40:47 2015 -0500
1429
1430     fixed unused match warnings but matching AST type instead of llvalue. David determined that the Oc
1431
1432 commit c45b5f88281cfa8c5989fbc883bbe97230bac8c2
1433 Merge: 3707602 7630cb1
1434 Author: David Watkins <DavidWatkins@users.noreply.github.com>
1435 Date:   Tue Dec 1 21:27:59 2015 -0500
1436
1437     Merge pull request #66 from DavidWatkins/emily
1438
1439     Emily
1440
1441 commit 7630cb139714b189697761faeb495d9a6d8055ad
1442 Author: Emily Chen <emchennyc@gmail.com>
1443 Date:   Tue Dec 1 21:26:16 2015 -0500
1444
1445     raised wrong exception when trying to instantiate undefined class
1446
1447 commit 9d0040a4aa5506d46024e2c870dee099527cb6db
```



```
1448 Author: Emily Chen <emchennyc@gmail.com>
1449 Date: Tue Dec 1 21:13:34 2015 -0500
1450
1451     threw wrong exception for UndefinedClass case
1452
1453 commit 63765ae27d235ca0664cb329e72324475f80d6c0
1454 Author: Emily Chen <emchennyc@gmail.com>
1455 Date: Tue Dec 1 20:26:28 2015 -0500
1456
1457     object creation flags when actuals don't match any existing constructor
1458
1459 commit 1d3c59c8bf8ce3d1c621697025a9c529f0285a2c
1460 Author: Emily Chen <emchennyc@gmail.com>
1461 Date: Tue Dec 1 17:26:18 2015 -0500
1462
1463     types of actuals printed in same order as types of formals
1464
1465 commit 045fc2aa1cd78c1c93f58ce4c4412ccceada0b39
1466 Author: Emily Chen <emchennyc@gmail.com>
1467 Date: Tue Dec 1 16:52:13 2015 -0500
1468
1469     can print types of formals and actuals
1470
1471 commit 5e2ea6f7870c893d0e8fa6df422f3be55a240555
1472 Author: Khaled Atef <kaa2168@columbia.edu>
1473 Date: Tue Dec 1 16:06:31 2015 -0500
1474
1475     Test cases for arith negation added and build_global_stringptr modified for debugging
1476
1477 commit 4913954cb8166998c6aae53a2c1f733c06473890
1478 Author: Khaled Atef <kaa2168@columbia.edu>
1479 Date: Tue Dec 1 08:43:55 2015 -0500
1480
1481     added cast test (float+int)
1482
1483 commit b4f2afc359fb3ad8dcf1e658d428480c84c183a7
1484 Author: Khaled Atef <kaa2168@columbia.edu>
1485 Date: Tue Dec 1 07:25:26 2015 -0500
1486
1487     Compiles git add codegen.ml !
1488
1489 commit 3ea9139620f9b937d7c6892cf1be2209cad34635
1490 Author: Emily Chen <emchennyc@gmail.com>
1491 Date: Tue Dec 1 03:13:43 2015 -0500
1492
1493     check_object_creation raises exception if instantiating unknown class
1494
1495 commit d98122c25680d734687c5e95d67832d620830d84
1496 Author: Emily Chen <emchennyc@gmail.com>
```

```
1497 Date: Tue Dec 1 02:41:32 2015 -0500
1498
1499 checks object decl to see if the class is available
1500
1501 commit 3bd51afa9cf5b2041543025837ed50d93ffe7d52
1502 Author: Khaled Atef <kaa2168@columbia.edu>
1503 Date: Tue Dec 1 01:48:21 2015 -0500
1504
1505 fought through several rounds of compilation errors.
1506
1507 commit 4494d69fc57cdac1a944a92ea7660f899a906a9f
1508 Author: Khaled Atef <kaa2168@columbia.edu>
1509 Date: Tue Dec 1 01:22:02 2015 -0500
1510
1511 Rough draft of handle_binop implemented. Still need to compile it, but pushing to access on VM. I h
1512
1513 commit 37076028c622a12be9c222ca2331f265c99ac625
1514 Author: David Watkins <DavidWatkins@users.noreply.github.com>
1515 Date: Mon Nov 30 23:49:19 2015 -0500
1516
1517 Update README.md
1518
1519 commit 34586a39bbe449392a730dcbcf3e85dd2b70941c
1520 Author: David Watkins <davidw@tkins.me>
1521 Date: Mon Nov 30 23:46:51 2015 -0500
1522
1523 Merged Emily's changes to master
1524
1525 commit a2446010f6af0c06f465581a0b09bde85d4f1a3c
1526 Author: David Watkins <davidw@tkins.me>
1527 Date: Mon Nov 30 23:41:58 2015 -0500
1528
1529 Fixed pretty printer and loops
1530
1531 commit 9b8880077317b5dafd319becca52528d2fa8a393
1532 Merge: 889c3b7 6f8b207
1533 Author: David Watkins <DavidWatkins@users.noreply.github.com>
1534 Date: Mon Nov 30 23:35:44 2015 -0500
1535
1536 Merge pull request #56 from DavidWatkins/emily
1537
1538 Emily
1539
1540 commit 6f8b20749beb30748ed3912f631ad30cbfdf9ab0
1541 Author: David Watkins <davidw@tkins.me>
1542 Date: Mon Nov 30 23:34:52 2015 -0500
1543
1544 Added primitive variables
1545
```

```
1546 commit 1bfb88e3e588de2b2d097d9efa76cee25753129d
1547 Author: Emily Chen <emchennyc@gmail.com>
1548 Date: Mon Nov 30 23:27:52 2015 -0500
1549
1550     remove debugging statements
1551
1552 commit 0fe96a727e2fbd895fdc20e4bc3b4b423fcec17f
1553 Merge: ef16286 889c3b7
1554 Author: Emily Chen <emchennyc@gmail.com>
1555 Date: Mon Nov 30 22:41:57 2015 -0500
1556
1557     Merge branch 'master' of https://github.com/DavidWatkins/Dice into emily
1558
1559 commit ef1628630066d0d7d20112a5def6a221fc38827c
1560 Author: Emily Chen <emchennyc@gmail.com>
1561 Date: Mon Nov 30 22:41:07 2015 -0500
1562
1563     converting local to slocal works for primitive types
1564
1565 commit 16491e2e0c6a513271acd0519f77b97818555344
1566 Author: Emily Chen <emchennyc@gmail.com>
1567 Date: Mon Nov 30 22:01:09 2015 -0500
1568
1569     local var decls are tracked even without assignment expr
1570
1571 commit 2adbb32da2aa5ccc60561594cb402d00b2e9c7bf
1572 Author: Emily Chen <emchennyc@gmail.com>
1573 Date: Mon Nov 30 21:48:09 2015 -0500
1574
1575     local var decl is added to env when statement includes nonempty expr
1576
1577 commit 889c3b715b50b63391a454a75a5d9d7dfbdd2657
1578 Merge: 3064464 3d3154c
1579 Author: David Watkins <davidw@tkins.me>
1580 Date: Mon Nov 30 19:49:10 2015 -0500
1581
1582     Merge branch 'master' of https://github.com/DavidWatkins/Dice
1583
1584 commit 306446425bd29b931a6325b47b3da0cc3b84e04f
1585 Author: David Watkins <davidw@tkins.me>
1586 Date: Mon Nov 30 19:48:49 2015 -0500
1587
1588     Added pretty printing of sast and ast in JSON
1589
1590 commit 3d3154cea0c622561c7a63946e5e85ec8eb07e8d
1591 Author: David Watkins <DavidWatkins@users.noreply.github.com>
1592 Date: Mon Nov 30 16:18:51 2015 -0500
1593
1594     Update README.md
```

1595

1596 commit 64d255b692d1d3f156a210253bb4db2b1bd123ba

1597 Author: Khaled Atef <kaa2168@columbia.edu>

1598 Date: Mon Nov 30 13:26:06 2015 -0500

1599

1600 modified test script to perform automatic compilation of Dice Executable at the beginning of each s

1601

1602 commit f4312c13faa500a2601e08b1fbd53568750df70b

1603 Author: Khaled Atef <kaa2168@columbia.edu>

1604 Date: Mon Nov 30 12:14:07 2015 -0500

1605

1606 corrected syntax error

1607

1608 commit b562f21c0f9e17dc946ddf2d1faa206346607e5d

1609 Merge: 338553e db99c23

1610 Author: David Watkins <davidw@tkins.me>

1611 Date: Mon Nov 30 08:10:05 2015 -0500

1612

1613 Merge branch 'emily'

1614

1615 commit db99c2314ba0bdf2bba05501e971dd379e1a0bbc

1616 Merge: 9f1d6c7 338553e

1617 Author: David Watkins <davidw@tkins.me>

1618 Date: Mon Nov 30 08:09:54 2015 -0500

1619

1620 Merge branch 'master' into emily

1621

1622 commit 338553e016ab991c9a5b278eb4e6fccb3e632121

1623 Author: David Watkins <davidw@tkins.me>

1624 Date: Mon Nov 30 03:18:29 2015 -0500

1625

1626 Added code for building for loops

1627

1628 commit ed35422de1e3530e82a4ecdb99a07c01774707cb

1629 Merge: ac53ca3 7fe5c5c

1630 Author: David Watkins <davidw@tkins.me>

1631 Date: Mon Nov 30 02:35:32 2015 -0500

1632

1633 Merge branch 'master' of <https://github.com/DavidWatkins/Dice>

1634

1635 commit ac53ca3084d06312dabbe3058ab51694774c6bc3

1636 Author: David Watkins <davidw@tkins.me>

1637 Date: Mon Nov 30 02:35:07 2015 -0500

1638

1639 Fixed elseless if problem

1640

1641 commit 7fe5c5ca653e8a5973228953b681f431833a16bd

1642 Merge: 9b5f7d1 50d5298

1643 Author: Khaled Atef <kaa2168@columbia.edu>

```
1644 Date:   Mon Nov 30 02:15:10 2015 -0500
1645
1646     Merge branch 'master' of https://github.com/DavidWatkins/Dice
1647
1648 commit 9b5f7d18f8f895d4979b9a6bc32164262cb9bc31
1649 Author: Khaled Atef <kaa2168@columbia.edu>
1650 Date:   Mon Nov 30 02:14:35 2015 -0500
1651
1652     basic inhertiance test added
1653
1654 commit 50d52984ad083eac3722a630cd027a0714537459
1655 Merge: 75a41c3 1932754
1656 Author: David Watkins <davidw@tkins.me>
1657 Date:   Mon Nov 30 01:39:36 2015 -0500
1658
1659     Merge branch 'master' of https://github.com/DavidWatkins/Dice
1660
1661 commit 75a41c3cf0da048bb6b76875f97d428cf84d8e41
1662 Author: David Watkins <davidw@tkins.me>
1663 Date:   Mon Nov 30 01:39:06 2015 -0500
1664
1665     Ifs semi-implemented, multi-line programs work now
1666
1667 commit 9f1d6c75a4454794ac10e636bb6dd07291cbd642
1668 Merge: a874f50 1932754
1669 Author: Emily Chen <emchennyc@gmail.com>
1670 Date:   Mon Nov 30 01:38:40 2015 -0500
1671
1672     Merge branch 'master' of https://github.com/DavidWatkins/Dice into emily
1673
1674 commit a874f50a43d711086c8038dc92c06014bf11a39c
1675 Author: Emily Chen <emchennyc@gmail.com>
1676 Date:   Mon Nov 30 01:37:41 2015 -0500
1677
1678     check_binop succeeds when only literal operands; doesn't handle IDs yet
1679
1680 commit 19327541aec867c8b3617ce7a55c2b8c30afc56a
1681 Author: Khaled Atef <kaa2168@columbia.edu>
1682 Date:   Mon Nov 30 01:35:32 2015 -0500
1683
1684     array tests added for single and multidimensional arrays.
1685
1686 commit d1a88f6cc7a112e34869a03c36f0fbb8c95dea73
1687 Author: Khaled Atef <kaa2168@columbia.edu>
1688 Date:   Sun Nov 29 23:56:36 2015 -0500
1689
1690     mroe tests
1691
1692 commit 01c53a3905d6dda85bd7d407ce1024c049401f3f
```

```
1693 Merge: 0052631 96c9f21
1694 Author: Emily Chen <emchennyc@gmail.com>
1695 Date: Sat Nov 28 16:48:45 2015 -0500
1696
1697 Merge pull request #50 from DavidWatkins/epsilon
1698
1699 Epsilon
1700
1701 commit 96c9f21a876921f9fe7b54c7c5520d2684926080
1702 Merge: 0828f97 0052631
1703 Author: Emily Chen <ec2805@columbia.edu>
1704 Date: Sat Nov 28 16:43:56 2015 -0500
1705
1706 Merge branch 'master' of https://github.com/DavidWatkins/Dice into epsilon
1707
1708 commit 0828f97195361fcde1f315f76c0b9b40602fdfa6
1709 Author: Emily Chen <ec2805@columbia.edu>
1710 Date: Sat Nov 28 16:43:40 2015 -0500
1711
1712 current state of LRM, WIP
1713
1714 commit 00526316b382f1fcdbf7e20dd8116d76f3c0af49
1715 Author: David Watkins <davidw@tkins.me>
1716 Date: Thu Nov 26 03:53:13 2015 -0500
1717
1718 Added environments as return types for expressions and statements
1719
1720 commit 7bd0f08fd5735207d23ef0282f75571779c17032
1721 Author: David Watkins <davidw@tkins.me>
1722 Date: Thu Nov 26 03:28:16 2015 -0500
1723
1724 Added assignment type checking
1725
1726 commit 91f50320126a774977e963b002952cffcdaf8c0b
1727 Author: David Watkins <davidw@tkins.me>
1728 Date: Thu Nov 26 03:18:53 2015 -0500
1729
1730 Reorganized analyser unop
1731
1732 commit eb1e72d42ffccfa994b21db18c5ee7594b3086cb
1733 Author: David Watkins <davidw@tkins.me>
1734 Date: Thu Nov 26 03:09:06 2015 -0500
1735
1736 Print will now accept variable number of arguments and print integers
1737
1738 commit 2697f7d36eee3267d4d08acd72ee36218cfe885f
1739 Author: David Watkins <davidw@tkins.me>
1740 Date: Thu Nov 26 02:43:40 2015 -0500
1741
```

```
1742     Added reserved functions to analyzer
1743
1744     commit f016c356c05016b220b3503f7ef331c0cc6fe9e9
1745     Author: David Watkins <davidw@tkins.me>
1746     Date:   Wed Nov 25 23:14:40 2015 -0500
1747
1748     Analyzer now uses SExpr instead of expr
1749
1750     commit 405feab53aeb98996d924e1d0c054b2c057893b8
1751     Author: David Watkins <davidw@tkins.me>
1752     Date:   Wed Nov 25 20:46:37 2015 -0500
1753
1754     Added test ocaml code to produce llvm
1755
1756     commit fb92dc93387bc04a842ce20414642a8e0d6be079
1757     Merge:  a70917b d3bfd36
1758     Author:  Emily Chen <ec2805@columbia.edu>
1759     Date:    Wed Nov 25 14:19:20 2015 -0500
1760
1761     Merge branch 'master' of https://github.com/DavidWatkins/Dice into epsilon
1762
1763     commit d3bfd36c6a493a1c4b768bcc86049b5245975fdc
1764     Author:  David Watkins <davidw@tkins.me>
1765     Date:    Mon Nov 23 03:55:35 2015 -0500
1766
1767     Added a lot
1768
1769     commit 18c53d74b916b57cf79523da4bb5532408f0d623
1770     Merge:  e714714 c3635ab
1771     Author:  David Watkins <DavidWatkins@users.noreply.github.com>
1772     Date:    Sat Nov 21 22:30:11 2015 -0500
1773
1774     Merge pull request #46 from DavidWatkins/Kreygasm
1775
1776     Kreygasm
1777
1778     commit c3635ab4859a46182ea3be101ffe08c80567da83
1779     Author:  nethacker11 <philip.schiffrin@gmail.com>
1780     Date:    Sat Nov 21 22:28:57 2015 -0500
1781
1782     duplicates checked in stringmaps
1783
1784     commit 347bb718b69bba5cedcf8d920e81fb46a9857602
1785     Author:  David Watkins <davidw@tkins.me>
1786     Date:    Sat Nov 21 21:58:42 2015 -0500
1787
1788     fubic
1789
1790     commit 796dad808edb739bb79e140ae8284affc416ba8e
```

```
1791 Author: nethacker11 <philip.schiffrin@gmail.com>
1792 Date: Sat Nov 21 20:30:52 2015 -0500
1793
1794 analyzer broken
1795
1796 commit 83453a96dc22e2b0cb3c0b5fadda6c38cd34f84d
1797 Author: nethacker11 <philip.schiffrin@gmail.com>
1798 Date: Fri Nov 20 15:34:00 2015 -0500
1799
1800 updated analyzer for global table
1801
1802 commit a70917bebd8b122fc1456a0de6d647f27e124378
1803 Author: Emily Chen <ec2805@columbia.edu>
1804 Date: Tue Nov 17 05:39:53 2015 -0500
1805
1806 specify wraparound behavior for char overflow during addition operation
1807
1808 commit 5ef7d4040ae2a3f384dc8e49108df9f911da54e8
1809 Author: Emily Chen <ec2805@columbia.edu>
1810 Date: Tue Nov 17 05:32:45 2015 -0500
1811
1812 fixed typos in Type section
1813
1814 commit 519ecd38eb0ff36345e404500a58799ab6e6f22e
1815 Author: Emily Chen <ec2805@columbia.edu>
1816 Date: Tue Nov 17 05:32:15 2015 -0500
1817
1818 fixed typos in Type section
1819
1820 commit 1dbea9cdd7ce99e8afb045d825c10cf7b61da1e6
1821 Author: Emily Chen <ec2805@columbia.edu>
1822 Date: Tue Nov 17 05:28:27 2015 -0500
1823
1824 lrm pdf
1825
1826 commit 218cdd226af54fc7a12aa54174686808b9c0c080
1827 Author: Emily Chen <ec2805@columbia.edu>
1828 Date: Tue Nov 17 05:27:01 2015 -0500
1829
1830 expressions emulate K&R reference
1831
1832 commit a23065b93cfa8ea563b2e5cafe47e4001364329f
1833 Author: Emily Chen <ec2805@columbia.edu>
1834 Date: Tue Nov 17 04:14:47 2015 -0500
1835
1836 remove examples from Types section
1837
1838 commit e71471403e598ff74fff7e1c18b6c26f84db7c4e
1839 Author: Emily Chen <ec2805@columbia.edu>
```



```
1840 Date:   Tue Nov 17 01:26:58 2015 -0500
1841
1842     update regex for int, float
1843
1844 commit 01369938d06a83b7a411e97ea7f3105355ecb1c7
1845 Author: David Watkins <davidw@tkins.me>
1846 Date:   Mon Nov 16 21:09:30 2015 -0500
1847
1848     Added new keyword, fixed pretty printing, allowed varied variable declaration
1849
1850 commit 90ac3e878efdb7d8471a49ac07b2717d568394ec
1851 Author: David Watkins <davidw@tkins.me>
1852 Date:   Mon Nov 16 05:00:47 2015 -0500
1853
1854     Hello world demo code
1855
1856 commit 4a37b8b8e8fe6d695354db10b78f4273584ece35
1857 Author: David Watkins <davidw@tkins.me>
1858 Date:   Mon Nov 16 04:55:33 2015 -0500
1859
1860     Added escape characters to string literals
1861
1862 commit cc898068bdd320886cbf6d6d950edc00a5cb8afe
1863 Merge: be88ee9 d8ed6e5
1864 Author: David Watkins <DavidWatkins@users.noreply.github.com>
1865 Date:   Sun Nov 15 15:24:08 2015 -0500
1866
1867     Merge pull request #7 from DavidWatkins/Delta
1868
1869     Delta
1870
1871 commit be88ee9635bd7a60d134eac92cd8516dc08ccd06
1872 Author: David Watkins <davidw@tkins.me>
1873 Date:   Sun Nov 15 03:00:33 2015 -0500
1874
1875     Removed bindings.c
1876
1877 commit b35203029ea05992df4d7356c556d8250379ec3e
1878 Merge: 1a79286 bdfc46f
1879 Author: David Watkins <davidw@tkins.me>
1880 Date:   Sun Nov 15 02:55:59 2015 -0500
1881
1882     Merge branch 'Delta' of https://github.com/DavidWatkins/Dice into HEAD
1883
1884 commit d8ed6e5ac77c66a1e28e43a91d1c7c90d10d096c
1885 Merge: 819c652 bdfc46f
1886 Author: David Watkins <davidw@tkins.me>
1887 Date:   Sun Nov 15 02:49:47 2015 -0500
1888
```

```
1889     Merge branch 'Delta' of https://github.com/DavidWatkins/Dice into Delta
1890
1891     commit bdfc46f92ab376ea29d6c672efa1c95cdc547f78
1892     Author: Khaled Atef <kaa2168@columbia.edu>
1893     Date:    Sun Nov 15 02:49:23 2015 -0500
1894
1895         fixed cleaning up of temp files
1896
1897     commit 819c652f8af6058175b61339d75612f798b7f446
1898     Author: David Watkins <davidw@tkins.me>
1899     Date:    Sun Nov 15 02:43:59 2015 -0500
1900
1901         Added unary minus
1902
1903     commit b79952cc5f486e080c31a7ae00b8977fa6812aa2
1904     Author: David Watkins <davidw@tkins.me>
1905     Date:    Sun Nov 15 02:37:18 2015 -0500
1906
1907         Removed - from int and float literals
1908
1909     commit b7e306b4eb8dbb8859a79a7242064713f923605d
1910     Author: David Watkins <davidw@tkins.me>
1911     Date:    Sun Nov 15 02:29:54 2015 -0500
1912
1913         Fixed rule with return
1914
1915     commit 30877b0821d29c59f8e86dbe8a0d4437d63dc6bc
1916     Author: Khaled Atef <kaa2168@columbia.edu>
1917     Date:    Sun Nov 15 02:21:01 2015 -0500
1918
1919         tester corrected to work with lli
1920
1921     commit f037b3b5dc89409d59d55b5be4ed2a816b317be6
1922     Merge: e843f0e ae1d756
1923     Author: Khaled Atef <kaa2168@columbia.edu>
1924     Date:    Sun Nov 15 02:19:51 2015 -0500
1925
1926     Merge branch 'Delta' of https://github.com/DavidWatkins/Dice into Delta
1927
1928     commit e843f0ef31580d6846586a8ca49c2840222276c9
1929     Author: Khaled Atef <kaa2168@columbia.edu>
1930     Date:    Sun Nov 15 02:19:34 2015 -0500
1931
1932         Corrected syntax errors in test case code
1933
1934     commit ae1d7560e16dd7eecdbc34a291d8f4e41a97eeeb
1935     Author: David Watkins <DavidWatkins@users.noreply.github.com>
1936     Date:    Sun Nov 15 02:10:16 2015 -0500
1937
```

```
1938     Update README.md
1939
1940     commit 026fd5026bf957515f9b0902aa2d278b48197fe0
1941     Author: David Watkins <DavidWatkins@users.noreply.github.com>
1942     Date:    Sun Nov 15 01:55:44 2015 -0500
1943
1944     Update README.md
1945
1946     commit 5ca0b69612e326f2523c5c7542f1ae5ea02d6f24
1947     Author: David Watkins <davidw@tkins.me>
1948     Date:    Sat Nov 14 20:14:41 2015 -0500
1949
1950     Small edit to readme
1951
1952     commit 1c2547eef86a517507a6cbec1655f38df7875290
1953     Author: David Watkins <davidw@tkins.me>
1954     Date:    Sat Nov 14 20:12:53 2015 -0500
1955
1956     Small changes
1957
1958     commit 54d7539d119aa459278dca7b3bcb68c248054948
1959     Author: David Watkins <davidw@tkins.me>
1960     Date:    Sat Nov 14 20:09:16 2015 -0500
1961
1962     Added to README
1963
1964     commit d88306a8e86159b467a5da701bd315ce8e713d5a
1965     Author: David Watkins <davidw@tkins.me>
1966     Date:    Sat Nov 14 19:57:11 2015 -0500
1967
1968     Compiler works, run build.sh
1969
1970     commit 32d87fa0fd8a81778ddd2d61936d64f5ef6aebc6
1971     Author: Khaled Atef <kaa2168@columbia.edu>
1972     Date:    Sat Nov 14 17:40:47 2015 -0500
1973
1974     modified test script to use lli
1975
1976     commit 0db46dad8c7f8f97385be5602b4340efb7485c44
1977     Merge: 8aa9ed2 2287265
1978     Author: David Watkins <davidw@tkins.me>
1979     Date:    Sat Nov 14 16:38:18 2015 -0500
1980
1981     Merge branch 'Delta' of https://github.com/DavidWatkins/Dice into Delta
1982
1983     commit 8aa9ed21512f946604e9824e72d0f32f48460cb9
1984     Author: David Watkins <davidw@tkins.me>
1985     Date:    Sat Nov 14 16:37:59 2015 -0500
1986
```

```
1987     Works!!!!!!
1988
1989     commit 228726581d0b46dd87aeeacfe2fc66b276ecd434
1990     Merge: 01e738d 65f6ba6
1991     Author: Khaled Atef <kaa2168@columbia.edu>
1992     Date:   Sat Nov 14 16:19:25 2015 -0500
1993
1994     Merge branch 'Delta' of https://github.com/DavidWatkins/Dice into Delta
1995
1996     commit 01e738dae2e637fd7a2265aa16bc3135172b24e2
1997     Author: Khaled Atef <kaa2168@columbia.edu>
1998     Date:   Sat Nov 14 16:11:58 2015 -0500
1999
2000     testing script and basic test cases
2001
2002     commit 65f6ba6989f389b3db5b95cb4f0eaf0438f10157
2003     Author: David <davidw@tkins.me>
2004     Date:   Sat Nov 14 13:41:36 2015 -0500
2005
2006     Made changes yo
2007
2008     commit 7dbe37512674db6a5f911bcc336878c45c5c1aca
2009     Author: David <davidw@tkins.me>
2010     Date:   Sat Nov 14 03:38:45 2015 -0500
2011
2012     I give up for now
2013
2014     commit 04c4053bf43276ac073c4141ce1d2f79ddbc3452
2015     Author: David <davidw@tkins.me>
2016     Date:   Sat Nov 14 03:31:54 2015 -0500
2017
2018     iWhatever
2019
2020     commit 1bfde45790ff5c37a06fed8a732d95343d6b09fe
2021     Author: David Watkins <djwt146@columbia.edu>
2022     Date:   Fri Nov 13 23:51:52 2015 -0500
2023
2024     Again WIP
2025
2026     commit 6bc13cfadb5458a77b2d521390683981502d3cd1
2027     Author: David Watkins <djwt146@columbia.edu>
2028     Date:   Fri Nov 13 16:25:05 2015 -0500
2029
2030     Added new way to make, figuring out layout for code based on tutorial
2031
2032     commit 1a79286feb4a6b21d8ded437dda312143a485f9b
2033     Author: David Watkins <davidw@tkins.me>
2034     Date:   Thu Nov 12 20:37:37 2015 -0400
2035
```

```
2036      Wrong rule for utils
2037
2038      commit 550cd68be9b44ca6b4bb57dd3ca6539ae4ee04ca
2039      Author: David Watkins <djrival7@gmail.com>
2040      Date:   Wed Nov 11 15:44:05 2015 -0500
2041
2042      Created base code for compiler and improved processinclude
2043
2044      commit e37f596018482de09bad4e87b85a9b346c33b374
2045      Author: David Watkins <dju2146@columbia.edu>
2046      Date:   Wed Nov 11 02:27:24 2015 -0500
2047
2048      Added more descriptive error messages to dice files with incorrect syntax
2049
2050      commit 2c93957d1cf2401c8eae282123bcaa397290c194
2051      Author: David Watkins <dju2146@columbia.edu>
2052      Date:   Wed Nov 11 01:14:35 2015 -0500
2053
2054      Changed primitive arrays to support inclusion of expressions and fixed
2055      escaped char literals
2056
2057      commit bfe58d3de921dd7428b6fb18d2ba2240336a0164
2058      Merge: 13e70b1 77193cc
2059      Author: Khaled Atef <kaa2168@columbia.edu>
2060      Date:   Mon Nov 9 02:28:11 2015 -0500
2061
2062      Merge branch 'master' of https://github.com/DavidWatkins/Dice
2063
2064      commit 13e70b1a603b87eb5c59dd96a7908a68e2b4286e
2065      Author: Khaled Atef <kaa2168@columbia.edu>
2066      Date:   Mon Nov 9 02:27:48 2015 -0500
2067
2068      testing script implemented for Scanner tokenizer with some basic test cases. More to follow soon
2069
2070      commit 77193cc351212bbe8fd9ff02c73f71c2a958cf91
2071      Author: Emily Chen <ec2805@columbia.edu>
2072      Date:   Mon Nov 9 05:16:18 2015 +0000
2073
2074      updated roles, re-rendered LRM pdf
2075
2076      commit 833154914d227be8b3d75ff2266cb5b1d016f185
2077      Author: Emily Chen <ec2805@columbia.edu>
2078      Date:   Mon Nov 9 05:13:39 2015 +0000
2079
2080      updated roles
2081
2082      commit 48b5419853a7dad44ae3ef590ddb48f9f0fc40fb
2083      Author: Emily Chen <ec2805@columbia.edu>
2084      Date:   Mon Nov 9 05:08:07 2015 +0000
```

```
2085
2086     updated LRM pdf
2087
2088 commit 62673cf7dcffe60a07ba5711e4df02c9d4542589
2089 Author: Emily Chen <ec2805@columbia.edu>
2090 Date:   Sun Nov 8 02:53:38 2015 +0000
2091
2092     add description for logical operators and member access operator
2093
2094 commit 650993c85b05f1415aa1fedf4f995358d5e94f6b
2095 Author: Emily Chen <ec2805@columbia.edu>
2096 Date:   Sun Nov 8 02:24:12 2015 +0000
2097
2098     add example of inheritance using "extends" kw
2099
2100 commit eb8488828a35dfe92828ff403eb3fa60b734eaf2
2101 Author: Emily Chen <ec2805@columbia.edu>
2102 Date:   Sun Nov 8 02:05:59 2015 +0000
2103
2104     updated examples so they don't declare and initialize in same statement
2105
2106 commit ac91d9956899f8a7246bef381942afcf505f95b6
2107 Author: Emily Chen <ec2805@columbia.edu>
2108 Date:   Sun Nov 8 01:06:22 2015 +0000
2109
2110     no class name collisions within module or between modules
2111
2112 commit 662d69a4aa9be43b54f85db25ec23ced249629b6
2113 Author: Emily Chen <ec2805@columbia.edu>
2114 Date:   Sun Nov 8 00:49:20 2015 +0000
2115
2116     change .di to .dice; specify that recursive includes are not supported
2117
2118 commit 61afc126141b1ceaa5618ca4af55e1b6229a9f2f
2119 Author: Emily Chen <ec2805@columbia.edu>
2120 Date:   Sun Nov 8 00:39:14 2015 +0000
2121
2122     describe Include statement
2123
2124 commit 9d04c148421a6793a71f17543fba74b0179b5a9e
2125 Author: Emily Chen <ec2805@columbia.edu>
2126 Date:   Sat Nov 7 23:59:35 2015 +0000
2127
2128     updated array declaration, initialization, access in LRM
2129
2130 commit c73e469da24918a33fbc32fadf46eef0bb34a83d
2131 Merge: 51f4c2e 18a9ca2
2132 Author: David Watkins <djrival7@gmail.com>
2133 Date:   Sat Nov 7 13:55:09 2015 -0500
```

```
2134
2135     Merge branch 'master' of https://github.com/DavidWatkins/DiceLanguage
2136
2137     commit 51f4c2e5e49ea49f1b9d7fdc84b116f79da7ef74
2138     Author: David Watkins <djrival7@gmail.com>
2139     Date:   Sat Nov 7 12:16:59 2015 -0500
2140
2141     Added nested primitive arrays to parser
2142
2143     commit 18a9ca288d017dffec5eb5240f6ed4e6551f4484
2144     Author: Emily Chen <ec2805@columbia.edu>
2145     Date:   Fri Nov 6 23:24:59 2015 +0000
2146
2147     updated operator precedence in parser
2148
2149     commit 5a8cef865cec749060472a2da08f68e5f66ab603
2150     Merge: 8a3a33b 233b9ae
2151     Author: David Watkins <DavidWatkins@users.noreply.github.com>
2152     Date:   Thu Nov 5 01:55:34 2015 -0500
2153
2154     Merge pull request #3 from DavidWatkins/DavidFix
2155
2156     David fix
2157
2158     commit 233b9ae4d2ad329217da5f89a02208e65f0f1056
2159     Merge: 2339f25 8a3a33b
2160     Author: David Watkins <DavidWatkins@users.noreply.github.com>
2161     Date:   Thu Nov 5 01:51:37 2015 -0500
2162
2163     Merge pull request #2 from DavidWatkins/master
2164
2165     Merge pull request #1 from DavidWatkins/DavidFix
2166
2167     commit 2339f25231f1ee437ed6703545d6c7b010ec99a7
2168     Author: David Watkins <djwt146@columbia.edu>
2169     Date:   Thu Nov 5 01:47:31 2015 -0500
2170
2171     Added AST printing method by using menhir inside ocaml
2172
2173     commit 8a3a33b299306b2322b4c87df2c42f559bb0f612
2174     Merge: 9507d54 d095d57
2175     Author: David Watkins <DavidWatkins@users.noreply.github.com>
2176     Date:   Thu Nov 5 00:57:10 2015 -0500
2177
2178     Merge pull request #1 from DavidWatkins/DavidFix
2179
2180     David fix
2181
2182     commit d095d57af5fb83805fcbd5d70ff133facad7f64a
```

```
2183 Author: David Watkins <djw2146@columbia.edu>
2184 Date: Thu Nov 5 00:37:34 2015 -0500
2185
2186 Pretty printer bug fixed, tokenizer now prints line numbers
2187
2188 commit 7c800f61455e0a09d421afb4d384e1d0a07f5283
2189 Author: David Watkins <djw2146@columbia.edu>
2190 Date: Wed Nov 4 22:12:26 2015 -0500
2191
2192 Made changes to front end, pretty print and tokenizer work
2193
2194 commit b2a0bbd31c3484a2a914a1fb01cd8e05ecbc074f
2195 Author: David Watkins <djw2146@columbia.edu>
2196 Date: Wed Nov 4 20:41:00 2015 -0500
2197
2198 Fixed makefile?
2199
2200 commit 14f5ace95044a80e3d8d2a49e5e6a645f704a6ae
2201 Author: David Watkins <djw2146@columbia.edu>
2202 Date: Wed Nov 4 20:35:55 2015 -0500
2203
2204 This is just a test run of additional useful files, needs to be compiled
2205 on a unix system
2206
2207 commit a87e9b9a3706cb33fb79f260391b786bf53c6a40
2208 Author: David Watkins <djrival7@gmail.com>
2209 Date: Wed Nov 4 15:57:22 2015 -0500
2210
2211 Fixed parser with class keyword, removed array keyword
2212
2213 commit 41825d291a910847c3aa0d5d67f5c60f3698fcef
2214 Author: David Watkins <djrival7@gmail.com>
2215 Date: Wed Nov 4 15:50:46 2015 -0500
2216
2217 Revert "Fixed operator precedence"
2218
2219 This reverts commit d132fdb8d21ba69a8d9d1c71c0ab71af5231eac0.
2220
2221 commit d132fdb8d21ba69a8d9d1c71c0ab71af5231eac0
2222 Author: David Watkins <djrival7@gmail.com>
2223 Date: Wed Nov 4 15:24:56 2015 -0500
2224
2225 Fixed operator precedence
2226
2227 commit 0726d908374df6c447d82290e06a83b84d0fdd0a
2228 Author: David Watkins <djrival7@gmail.com>
2229 Date: Wed Nov 4 15:23:35 2015 -0500
2230
2231 Fixed bucket_args to refer to general expr list
```



```
2232
2233 commit ffec7d32cba2126885009518d79f072feea88628
2234 Author: David Watkins <djrival7@gmail.com>
2235 Date: Wed Nov 4 15:21:37 2015 -0500
2236
2237 Added datatypes to primitive arrays
2238
2239 commit 096f5a8bd45b44f23a42abaf837ad1f55597bce3
2240 Author: David Watkins <djrival7@gmail.com>
2241 Date: Wed Nov 4 15:18:01 2015 -0500
2242
2243 Fixed bug, apparently no issues wot
2244
2245 commit f30730a1cd041a3df6010c1ee62a59b17842e68f
2246 Author: David Watkins <djw2146@columbia.edu>
2247 Date: Wed Nov 4 14:43:20 2015 -0500
2248
2249 Fixed Menhir errors
2250
2251 commit 89a3de8e2e78735910eb385a3b51f4795e115c62
2252 Author: David Watkins <djrival7@gmail.com>
2253 Date: Wed Nov 4 13:38:03 2015 -0500
2254
2255 Removed extraneous files
2256
2257 commit c6d4db34bcca5aa6cc86a4e6f670a858c2f0b6bc
2258 Author: David Watkins <djrival7@gmail.com>
2259 Date: Wed Nov 4 13:36:38 2015 -0500
2260
2261 Cleaned up git directory
2262
2263 commit 4d157027e77211c94f295f34cef0d03a19c7f102
2264 Author: David Watkins <djrival7@gmail.com>
2265 Date: Wed Nov 4 13:20:24 2015 -0500
2266
2267 Found a more elegant solution to array problem
2268
2269 commit d7f28aa1dad4d1e788ab7b2aaab962372dfe1e71
2270 Author: David Watkins <djrival7@gmail.com>
2271 Date: Wed Nov 4 00:06:05 2015 -0500
2272
2273 No shift reduce but not ideal
2274
2275 commit d29a030e18489e489f0f8941cfbc70cc85c81a03
2276 Author: David Watkins <djrival7@gmail.com>
2277 Date: Tue Nov 3 23:45:34 2015 -0500
2278
2279 Super close, just ambiguity surroundign array access
2280
```

```
2281 commit 5473e62f147ec34eaccd4289930c4b2144d7c968
2282 Author: David Watkins <djrival7@gmail.com>
2283 Date: Tue Nov 3 18:04:37 2015 -0500
2284
2285     More stuff
2286
2287 commit dc6ad10dc89c389534ec225082c64da35a512268
2288 Author: David Watkins <djrival7@gmail.com>
2289 Date: Tue Nov 3 14:10:00 2015 -0500
2290
2291     Shift/Reduce down to 2, fixed layout of cdecl and cbody
2292
2293 commit 9507d5426cd130cc927941a4620383040c895717
2294 Author: Khaled Atef <kaa2168@columbia.edu>
2295 Date: Mon Nov 2 13:07:36 2015 -0500
2296
2297     BIBLETHUMP delimiter still in this version. AST modified to remove actions not used
2298
2299 commit 5419ec530d153c46c7ee4a3c12765a73c0fcb0c0
2300 Author: Khaled Atef <kaa2168@columbia.edu>
2301 Date: Mon Nov 2 01:32:27 2015 -0500
2302
2303     Corrected the array access production to account for multidimensional arrays
2304
2305 commit ce9ed98c205d39871c544a3640d92a6e1c77c31f
2306 Author: Khaled Atef <kaa2168@columbia.edu>
2307 Date: Mon Nov 2 01:29:07 2015 -0500
2308
2309     Parser compiles w/o any errors, but not tested yet. Multidimensional arrays implemented
2310
2311 commit 8ee5b859b0fea8a91154fe05694cd875c05fbd9e
2312 Author: Philip Schiffrin <philip.schiffrin@gmail.com>
2313 Date: Thu Oct 29 22:58:33 2015 -0400
2314
2315     hacked a solution for the last shift/reduce by adding token FUN at beginning of fdecls
2316
2317 commit bd6ecf78c93229b46957e67a50056e975ddcc072
2318 Author: Philip Schiffrin <philip.schiffrin@gmail.com>
2319 Date: Thu Oct 29 21:29:48 2015 -0400
2320
2321     fixed all reduce/reduce and most shift/reduce errors in the parser, lost most of our logic
2322
2323 commit d8cdf93ba6240b597307acb59407165c5b8eeff2
2324 Merge: df1979e 5a7f132
2325 Author: David Watkins <dju2146@columbia.edu>
2326 Date: Mon Oct 26 20:12:34 2015 -0400
2327
2328     Merge branch 'master' of https://github.com/DavidWatkins/DiceLanguage
2329
```

```
2330 commit df1979e32572f5bb0e3e5e4c95d22ff99ae77c53
2331 Author: David Watkins <djw2146@columbia.edu>
2332 Date: Mon Oct 26 20:12:23 2015 -0400
2333
2334     Fixed language to dice again
2335
2336 commit 5a7f13237ad7c32bc9609653e30b3f71b9c3aaf1
2337 Author: David Watkins <DavidWatkins@users.noreply.github.com>
2338 Date: Mon Oct 26 16:30:45 2015 -0400
2339
2340     Update README.md
2341
2342 commit f88282699ce5b7e3ea3c672c51e1ac1a69527687
2343 Author: David Watkins <djw2146@columbia.edu>
2344 Date: Mon Oct 26 02:57:24 2015 -0400
2345
2346     Fixed with edits
2347
2348 commit 6778395705ceea2284b4f1e13c022ddfd2c45f64
2349 Author: David Watkins <djw2146@columbia.edu>
2350 Date: Mon Oct 26 02:38:35 2015 -0400
2351
2352     Fixed intro
2353
2354 commit 3905c2317321473a3e481e2b52902461ba03d5bb
2355 Merge: 764eeb4 f207e7c
2356 Author: David Watkins <djw2146@columbia.edu>
2357 Date: Mon Oct 26 02:28:08 2015 -0400
2358
2359     Merge branch 'master' of https://github.com/DavidWatkins/DiceLanguage
2360
2361 commit 764eeb44f8ffa599a2451b037b67078683d6eef9
2362 Author: David Watkins <djw2146@columbia.edu>
2363 Date: Mon Oct 26 02:27:49 2015 -0400
2364
2365     Finished final draft of LRM
2366
2367 commit f207e7c0a231b852c21103503bbec8fa6edb483a
2368 Merge: d37bf92 0c85801
2369 Author: Khaled Atef <kaa2168@columbia.edu>
2370 Date: Mon Oct 26 00:34:51 2015 -0400
2371
2372     Merge branch 'master' of https://github.com/DavidWatkins/JFlat
2373
2374 commit d37bf92df1077fd3f612827f14c34cb7e6095af4
2375 Author: Khaled Atef <kaa2168@columbia.edu>
2376 Date: Mon Oct 26 00:34:34 2015 -0400
2377
2378     Parser compiles, but produces 457 reduce/reduce errors.
```

```
2379
2380 commit 0c858019e2b58ae63d8ec581a185953a558a464f
2381 Author: David Watkins <djwt146@columbia.edu>
2382 Date: Sun Oct 25 22:18:37 2015 -0400
2383
2384     Removed extraneous file
2385
2386 commit fe8bd9d0a8abf0f521a6ff44da6d7d615fa259b1
2387 Author: Emily Chen <ec2805@columbia.edu>
2388 Date: Sun Oct 25 04:29:11 2015 +0000
2389
2390     adding content for statements section; TODO include statements
2391
2392 commit 35b69e9bdf8fb8623f19d7d75790c7197c188c74
2393 Author: Emily Chen <ec2805@columbia.edu>
2394 Date: Sun Oct 25 03:30:32 2015 +0000
2395
2396     remove elseif keyword
2397
2398 commit 71bfd4fc327bcce124455bea1a731351a884ea25
2399 Author: Emily Chen <ec2805@columbia.edu>
2400 Date: Sun Oct 25 02:33:32 2015 +0000
2401
2402     update Statements sections
2403
2404 commit c991aa20fcad8e2a7ddab6f0552c105cd943e752
2405 Author: David Watkins <djrival7@gmail.com>
2406 Date: Sat Oct 24 22:23:32 2015 -0400
2407
2408     Whatevs
2409
2410 commit b4a0296539e07421a7436d7530154bbd8208158d
2411 Author: David Watkins <djrival7@gmail.com>
2412 Date: Sat Oct 24 21:34:36 2015 -0400
2413
2414     More
2415
2416 commit b6d7bdda712a8bbc77c923861fa2dd1161f383e9
2417 Merge: 00c7c8b 8b447ba
2418 Author: David Watkins <djrival7@gmail.com>
2419 Date: Sat Oct 24 21:28:34 2015 -0400
2420
2421     Merge branch 'master' of https://github.com/DavidWatkins/DiceLanguage
2422
2423 commit 00c7c8bff49163684020efa60cccd3ae70e481c4
2424 Author: David Watkins <djrival7@gmail.com>
2425 Date: Sat Oct 24 21:27:44 2015 -0400
2426
2427     Stuff
```

```
2428
2429 commit 8b447ba1a5c2cbfdb18ec5edf26527a926f28e0c
2430 Author: Emily Chen <ec2805@columbia.edu>
2431 Date: Sun Oct 25 01:01:54 2015 +0000
2432
2433     updated constructor definition
2434
2435 commit 00225f13ceeb3ac4ba51811a308cf069046b58af
2436 Author: Emily Chen <ec2805@columbia.edu>
2437 Date: Sun Oct 25 00:21:34 2015 +0000
2438
2439     method names cannot be same as class name
2440
2441 commit 02cc3c7206ed0626bf93ba93bb7924f9896431ea
2442 Merge: fed5273 1335b3e
2443 Author: Emily Chen <ec2805@columbia.edu>
2444 Date: Sat Oct 24 23:16:24 2015 +0000
2445
2446     Merge branch 'master' of https://github.com/DavidWatkins/JFlat
2447
2448 commit 1335b3e88c32dbbcdfe8e0954bb9fd0b89ce4903
2449 Merge: 45adcd4 d56fbb6
2450 Author: Khaled Atef <kaa2168@columbia.edu>
2451 Date: Sat Oct 24 19:24:12 2015 -0400
2452
2453     Merge branch 'master' of https://github.com/DavidWatkins/JFlat
2454
2455 commit d56fbb6510545503ea3aled8eeb0dd38a20cd42c
2456 Author: Philip Schiffrin <philip.schiffrin@gmail.com>
2457 Date: Sat Oct 24 19:23:49 2015 -0400
2458
2459     changed doubles to floats
2460
2461 commit 45adcd4da21201da272a7807a47ebde99fbb1e77
2462 Author: Khaled Atef <kaa2168@columbia.edu>
2463 Date: Sat Oct 24 19:23:04 2015 -0400
2464
2465     changed double to float
2466
2467 commit be271ea2be45684e4cc07fb2723a75343509bf6a
2468 Author: David Watkins <djrival7@gmail.com>
2469 Date: Sat Oct 24 19:18:21 2015 -0400
2470
2471     LOL more stuff
2472
2473 commit fed5273dc9e7adae462996c0a7fadc4bdfb91bdd
2474 Merge: 119f1a5 d56fbb6
2475 Author: Emily Chen <ec2805@columbia.edu>
2476 Date: Sat Oct 24 23:15:30 2015 +0000
```

```
2477
2478     Merge branch 'master' of https://github.com/DavidWatkins/JFlat
2479
2480 commit 119f1a57fef847539f3ab76ba3af1bbbe0ef91b2
2481 Author: Emily Chen <ec2805@columbia.edu>
2482 Date:   Sat Oct 24 23:13:31 2015 +0000
2483
2484     update lexical elements to replace double w float
2485
2486 commit a793510d204f60c87c8d44ec5f03c6ea7713c1f0
2487 Author: David Watkins <djrival7@gmail.com>
2488 Date:   Sat Oct 24 18:49:37 2015 -0400
2489
2490     Whatever
2491
2492 commit 4ef1e1b02c3fbaace4d0c4471e4bd048b5c7e8e1
2493 Author: David Watkins <djwt146@columbia.edu>
2494 Date:   Sat Oct 24 18:20:07 2015 -0400
2495
2496     Added array and object creation to parser
2497
2498 commit 923b6e6ee2e37705f88441be373b71e478475326
2499 Author: David Watkins <djwt146@columbia.edu>
2500 Date:   Sat Oct 24 17:35:10 2015 -0400
2501
2502     Added Program def
2503
2504 commit 0873e88184cb95d37ce0e6ceb7a320fda04f3dbe
2505 Author: David Watkins <djwt146@columbia.edu>
2506 Date:   Sat Oct 24 16:52:25 2015 -0400
2507
2508     Added more info to the parser
2509
2510 commit 049a058293023065afb6f90def5b215c298e5511
2511 Author: Khaled Atef <kaa2168@columbia.edu>
2512 Date:   Sat Oct 24 14:36:56 2015 -0400
2513
2514     Added negative doubles in scanner
2515
2516 commit b10a719cae082c8add0a509266264312d528a7df
2517 Author: Khaled Atef <kaa2168@columbia.edu>
2518 Date:   Sat Oct 24 13:42:59 2015 -0400
2519
2520     Corrected precedence chart by removing modulo reference
2521
2522 commit 32497722d8a7efbba2baaaa8ab80ddf899b8f429
2523 Merge: 1ca3359 388cada
2524 Author: Emily Chen <ec2805@columbia.edu>
2525 Date:   Sat Oct 24 05:46:04 2015 +0000
```

```
2526
2527     Merge branch 'master' of https://github.com/DavidWatkins/JFlat
2528
2529     commit 1ca3359bdd54bca5247219fddcd6d7de94dcfaa8
2530     Author: Emily Chen <ec2805@columbia.edu>
2531     Date:   Sat Oct 24 05:45:12 2015 +0000
2532
2533         renamed lexical elements LRM
2534
2535     commit 5c18e1c07073971ecd666ee761db3e757693d631
2536     Author: Emily Chen <ec2805@columbia.edu>
2537     Date:   Sat Oct 24 05:44:06 2015 +0000
2538
2539         classes LRM
2540
2541     commit 388cada09bd0c3b9d43c052780c708b636c958ba
2542     Merge: 7489892 865accd
2543     Author: Khaled Atef <kaa2168@columbia.edu>
2544     Date:   Sat Oct 24 00:00:54 2015 -0400
2545
2546     Merge branch 'master' of https://github.com/DavidWatkins/JFlat
2547
2548     commit 865accd230bbb198523fac3a74aadf14a0e157ff
2549     Author: Philip Schiffrin <philip.schiffrin@gmail.com>
2550     Date:   Fri Oct 23 23:54:06 2015 -0400
2551
2552         updated LRM_Phil.txt with structure, scope, and arrays
2553
2554     commit 74898925d7d70a982dd70e3ff2f14b73c7d26f0e
2555     Author: Khaled Atef <kaa2168@columbia.edu>
2556     Date:   Fri Oct 23 23:52:04 2015 -0400
2557
2558         Expressions/Operators portion of LRM
2559
2560     commit 3b92c848a16d0e7a8f9315d680c13bfb5f17888
2561     Author: Emily Chen <ec2805@columbia.edu>
2562     Date:   Sat Oct 24 01:36:09 2015 +0000
2563
2564         add 'this' keyword
2565
2566     commit 1aecbbdfefb15eee2c260c3703274b9dba43777b
2567     Merge: 3650409 42a62d0
2568     Author: Emily Chen <ec2805@columbia.edu>
2569     Date:   Sat Oct 24 01:12:16 2015 +0000
2570
2571     Merge branch 'master' of https://github.com/DavidWatkins/JFlat
2572
2573     commit 3650409be97c9ed8cd94802464950aaced1993c5
2574     Author: Emily Chen <ec2805@columbia.edu>
```

```
2575 Date:   Sat Oct 24 01:11:34 2015 +0000
2576
2577     lexical elements section for LRM
2578
2579 commit 42a62d0bf9d983cd3a0f15d4b8f1cb09bf41598e
2580 Author: David Watkins <djrival7@gmail.com>
2581 Date:   Fri Oct 23 19:14:43 2015 -0400
2582
2583     More stuff
2584
2585 commit 8e3398b458b45005129ab57c82ff6a13f9abafcb
2586 Author: Philip Schiffrin <philip.schiffrin@gmail.com>
2587 Date:   Fri Oct 23 16:54:05 2015 -0400
2588
2589     added lrm text for data types
2590
2591 commit fe46f41b37d368df2a0c5b552e7166f47c6b227d
2592 Author: David Watkins <DavidWatkins@users.noreply.github.com>
2593 Date:   Fri Oct 23 16:45:12 2015 -0400
2594
2595     Update README.md
2596
2597 commit ac7ee22f9b3ea22922761752306918e49ce148c8
2598 Merge: 8456847 8aa535a
2599 Author: David Watkins <djrival7@gmail.com>
2600 Date:   Fri Oct 23 16:40:21 2015 -0400
2601
2602     Merge branch 'master' of https://github.com/DavidWatkins/DiceLanguage
2603
2604 commit 84568474c4b9868e460decac0c99e730fecfc92d
2605 Author: David Watkins <djrival7@gmail.com>
2606 Date:   Fri Oct 23 16:39:44 2015 -0400
2607
2608     Added stuff
2609
2610 commit 8aa535ad4d802f7205c73c896b93b13dd8180239
2611 Author: David Watkins <davidw@tkins.me>
2612 Date:   Mon Oct 12 11:32:44 2015 -0400
2613
2614     Added isprime ll
2615
2616 commit 4812638376210accb89e6774c0442a918daa4b2f
2617 Author: David Watkins <davidw@tkins.me>
2618 Date:   Mon Oct 12 11:09:09 2015 -0400
2619
2620     Code working with llvm helloworld example, but does not return result
2621
2622 commit 4cc504c6d2d3732f2a96bc999ef2a28098a320a1
2623 Author: Emily Chen <ec2805@columbia.edu>
```



```
2624 Date:   Mon Oct 12 03:03:44 2015 -0400
2625
2626     remove obsolete TODO
2627
2628 commit 491935e26c6e27de4c14fb38f5d911eab53e0127
2629 Author: Emily Chen <ec2805@columbia.edu>
2630 Date:   Mon Oct 12 02:52:38 2015 -0400
2631
2632     host first sends the number of expected bytes it's sending
2633
2634 commit 546776b3750cfa4dfbe9bdb2388ce16ced7b83b4
2635 Author: Emily Chen <ec2805@columbia.edu>
2636 Date:   Mon Oct 12 02:19:24 2015 -0400
2637
2638     child proc successfully executes command and writes results to file
2639
2640 commit a129ab12519bfcdc1d5805b31ce6a966eeaaef52b
2641 Author: Emily Chen <ec2805@columbia.edu>
2642 Date:   Mon Oct 12 01:45:24 2015 -0400
2643
2644     worker no longer blocks after host is done sending all file data
2645
2646 commit 3c7dd62f15c82bf9a8e75c4b9e3a8b5b006a8d60
2647 Author: David Watkins <davidw@tkins.me>
2648 Date:   Sun Oct 11 20:02:24 2015 -0400
2649
2650     Code now compiles and added test.py
2651
2652 commit 89b213ca24cacf788cf48f740a2beaecfa58bb50
2653 Merge: 682c38f bb4d7c3
2654 Author: David Watkins <djrival7@gmail.com>
2655 Date:   Sun Oct 11 19:25:53 2015 -0400
2656
2657     Merge branch 'master' of https://github.com/DavidWatkins/DiceLanguage
2658
2659 commit 682c38fb01a5ccd93c73c1e1e050d3763fcf5e03
2660 Author: David Watkins <djrival7@gmail.com>
2661 Date:   Sun Oct 11 19:25:30 2015 -0400
2662
2663     Added new code for worker and Makefile
2664
2665 commit bb4d7c386dd1f202183ca3a39fcc2ff184b28fb9
2666 Author: Philip Schiffrin <philip.schiffrin@gmail.com>
2667 Date:   Sun Oct 11 16:37:52 2015 -0400
2668
2669     added isprime for llvm test - llvm code generated by llvm from isprime.c
2670
2671 commit 6ad9e088cdb74598d64d6ce3d1c55d8064295342
2672 Author: David Watkins <djrival7@gmail.com>
```

```
2673 Date:    Fri Oct 9 23:18:24 2015 -0400
2674
2675     Added initial C code
2676
2677 commit 87f6d1a0d5c756bce028ee1e0622b51957665906
2678 Author: David Watkins <DavidWatkins@users.noreply.github.com>
2679 Date:    Fri Oct 9 23:17:27 2015 -0400
2680
2681     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:

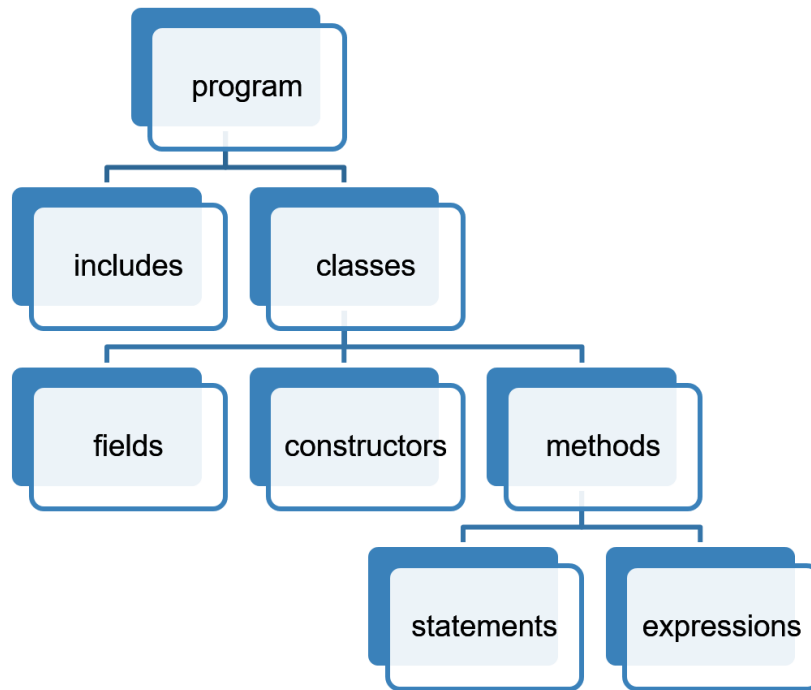


Figure 5.1: AST program representation.

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

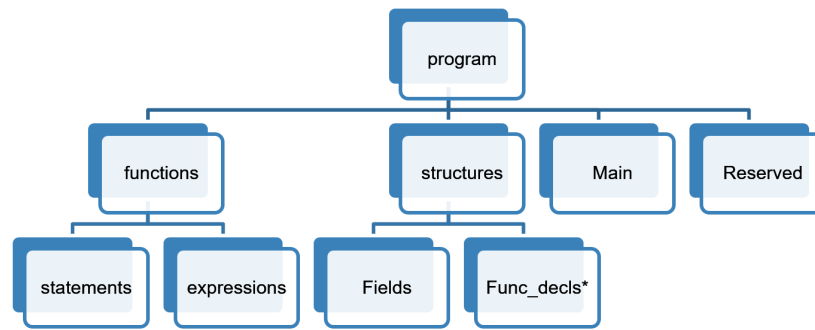


Figure 5.2: SAST representation.

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.

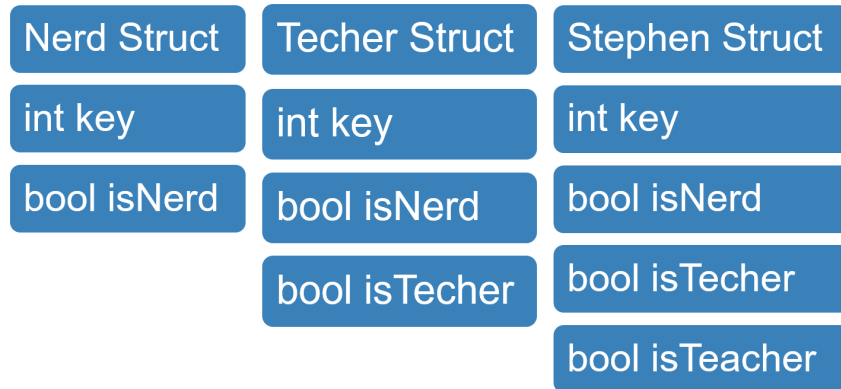


Figure 5.3: Structs example with inheritance.

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 }
```

```
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":
60                                 ↪ "this",
61                                 ↪ "datatype":
62                                 ↪ "class test"
63                                 ↪ }
64                         },
65                         "op": ".",
66                         "rhs": {
67                             "id": { "name":
68                                 ↪ ".key",
69                                 ↪ "datatype":
70                                 ↪ "int" }
71                         },
72                         "datatype": "int"
73                     }
74                 },
75                 "op": "=",
76                 "rhs": { "int_lit": { "val": 0,
77                     ↪ "datatype": "int" } },
78                 "datatype": "int"
79             }
80         },
81         "datatype": "int"
82     }
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                                                                     ↪ "ignore",
118                                                                 "datatype":
119                                                                     ↪ "class
120                                                                     ↪ test"
121                                                             }
122                                                         }
123                                                     ],
124                                                 }
125                                             ],
126                                         "index": 0,
127                                         "datatype": "int"
128                                     }
129                                 }
130                             ],
131                         "index": 0,
132                         "datatype": "char[]"
133                     }
134                 }
135             }
136         ]
137     }
138 }

```

```
130         ],
131         "index": 0,
132         "datatype": "class test"
133     }
134 }
135 }
136 },
137 {
138     "sexpr": {
139     "expr": {
140         "assign": {
141         "lhs": {
142             "objaccess": {
143             "lhs": {
144                 "id": { "name": "this",
145                     ↪ "datatype": "class test" }
146             },
147             "op": ".",
148             "rhs": { "id": { "name": ".key",
149                 ↪ "datatype": "int" } },
150             "datatype": "int"
151         }
152         },
153         "op": "=",
154         "rhs": { "int_lit": { "val": 0, "datatype": "int" } },
155         "datatype": "int"
156     }
157 },
158 "datatype": "int"
159 }
160 },
161 {
162     "sexpr": {
163     "expr": {
164         "call": {
165         "name": "print",
166         "params": [
167             {
168                 "string_lit": {
169                 "val": "Hello, World!",
170                 "datatype": "char[]"
171             }
172         ]
173         },
174         "index": 0,
175         "datatype": "void"
176     }
177 },
178 "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.

Dice to LL IR

The following code examples are dice source files that compile to an associated LLVM IR file.

Hello World Example

The following "Hello, World!" program is the first program we got running in our language.

test-hello.dice

```
1  class test {
2      public void main(char[] [] args) {
3          print("Hello, World!");
4      }
5  }
```

test-hello.ll

```
1  ; ModuleID = 'Dice Codegen'
2  target datalayout = "e-m:e-i64:64-f80:128-n8:16:32:64-S128"
3  target triple = "x86_64-pc-linux-gnu"
4
5  %test = type <{ i32 }>
6
7  @tmp = private unnamed_addr constant [14 x i8] c"Hello, World!\00"
8  @tmp.1 = private unnamed_addr constant [3 x i8] c"%s\00"
9
10 declare i32 @printf(i8*, ...)
11
12 declare noalias i8* @malloc(i32)
13
14 declare i32 @open(i8*, i32)
15
16 declare i32 @close(i32)
17
18 declare i32 @read(i32, i8*, i32)
19
20 declare i32 @write(i32, i8*, i32)
21
22 declare i32 @lseek(i32, i32, i32)
23
24 declare void @exit(i32)
25
26 declare i8* @realloc(i8*, i32)
27
```

```

28 declare i32 @getchar()
29
30 define i64* @lookup(i32 %c_index, i32 %f_index) {
31 entry:
32     %tmp = alloca i64**
33     %tmp1 = alloca i64*, i32 0
34     %tmp2 = getelementptr i64**, i64*** %tmp, i32 0
35     store i64** %tmp1, i64*** %tmp2
36     ret i64* null
37 }
38
39 define %test* @test.constructor() {
40 entry:
41     %this = alloca %test
42     %tmp = call i8* @malloc(i32 ptrtoint (i1** getelementptr (i1*, i1** null, i32 1) to
43         ↪ i32))
44     %tmp1 = bitcast i8* %tmp to %test*
45     %tmp2 = load %test, %test* %tmp1
46     store %test %tmp2, %test* %this
47     %key = getelementptr inbounds %test, %test* %this, i32 0, i32 0
48     store i32 0, i32* %key
49     ret %test* %this
50 }
51
52 define i32 @main(i32 %argc, i8** %argv) {
53 entry:
54     %arr_size = add i32 %argc, 1
55     %mallocsize = mul i32 %arr_size, ptrtoint (i1** getelementptr (i1*, i1** null, i32 1)
56         ↪ to i32)
57     %malloccall = tail call i8* @malloc(i32 %mallocsize)
58     %args = bitcast i8* %malloccall to i8***
59     %args1 = bitcast i8*** %args to i8**
60     %argc_len = bitcast i8** %args1 to i32*
61     %arr_1 = getelementptr i8*, i8** %args1, i32 1
62     store i32 %argc, i32* %argc_len
63     br label %args.cond
64
65 args.cond:                                     ; preds = %args.init, %entry
66     %counter = phi i32 [ 0, %entry ], [ %tmp, %args.init ]
67     %tmp = add i32 %counter, 1
68     %tmp2 = icmp slt i32 %counter, %argc
69     br i1 %tmp2, label %args.init, label %args.done
70
71 args.init:                                     ; preds = %args.cond
72     %tmp3 = getelementptr i8*, i8** %arr_1, i32 %counter
73     %tmp4 = getelementptr i8*, i8** %argv, i32 %counter
74     %tmp5 = load i8*, i8** %tmp4
75     store i8* %tmp5, i8** %tmp3
76     br label %args.cond

```

```

75
76 args.done:                                ; preds = %args.cond
77   %this = alloca %test
78   %tmp6 = call i8* @malloc(i32 ptrtoint (i1** getelementptr (i1*, i1** null, i32 1) to
    ↪ i32))
79   %tmp7 = bitcast i8* %tmp6 to %test*
80   %tmp8 = load %test, %test* %tmp7
81   store %test %tmp8, %test* %this
82   %.key = getelementptr inbounds %test, %test* %this, i32 0, i32 0
83   store i32 0, i32* %.key
84   %tmp9 = call i32 (i8*, ...) @printf(i8* getelementptr inbounds ([3 x i8], [3 x i8]*
    ↪ @tmp.1, i32 0, i32 0), i8* getelementptr inbounds ([14 x i8], [14 x i8]* @tmp, i32
    ↪ 0, i32 0))
85   ret i32 0
86 }
87
88 ; Function Attrs: nounwind uwtable
89 define i8* @input() #0 {
90   %initial_size = alloca i32, align 4
91   %str = alloca i8*, align 8
92   %index = alloca i32, align 4
93   %tmp = alloca i8, align 1
94   store i32 100, i32* %initial_size, align 4
95   %1 = load i32, i32* %initial_size, align 4
96   %2 = sext i32 %1 to i64
97   %3 = call noalias i8* bitcast (i8* (i32)* @malloc to i8* (i64*)(i64 %2) #1
98   store i8* %3, i8** %str, align 8
99   store i32 0, i32* %index, align 4
100  store i8 48, i8* %tmp, align 1
101  br label %4
102
103 ; <label>:4                                ; preds = %20, %0
104   %5 = call i32 @getchar()
105   %6 = trunc i32 %5 to i8
106   store i8 %6, i8* %tmp, align 1
107   %7 = sext i8 %6 to i32
108   %8 = icmp ne i32 %7, 10
109   br i1 %8, label %9, label %27
110
111 ; <label>:9                                ; preds = %4
112   %10 = load i32, i32* %index, align 4
113   %11 = load i32, i32* %initial_size, align 4
114   %12 = sub nsw i32 %11, 1
115   %13 = icmp sge i32 %10, %12
116   br i1 %13, label %14, label %20
117
118 ; <label>:14                                ; preds = %9
119   %15 = load i8*, i8** %str, align 8
120   %16 = load i32, i32* %initial_size, align 4

```

```

121  %17 = mul nsw i32 %16, 2
122  store i32 %17, i32* %initial_size, align 4
123  %18 = sext i32 %17 to i64
124  %19 = call i8* @bitcast (i8* (i8*, i32)* @realloc to i8* (i8*, i64)*) (i8* %15, i64 %18)
    ↪ #1
125  store i8* %19, i8** %str, align 8
126  br label %20
127
128  ; <label>:20                                ; preds = %14, %9
129  %21 = load i8, i8* %tmp, align 1
130  %22 = load i32, i32* %index, align 4
131  %23 = add nsw i32 %22, 1
132  store i32 %23, i32* %index, align 4
133  %24 = sext i32 %22 to i64
134  %25 = load i8*, i8** %str, align 8
135  %26 = getelementptr inbounds i8, i8* %25, i64 %24
136  store i8 %21, i8* %26, align 1
137  br label %4
138
139  ; <label>:27                                ; preds = %4
140  %28 = load i32, i32* %index, align 4
141  %29 = sext i32 %28 to i64
142  %30 = load i8*, i8** %str, align 8
143  %31 = getelementptr inbounds i8, i8* %30, i64 %29
144  store i8 0, i8* %31, align 1
145  %32 = load i8*, i8** %str, align 8
146  ret i8* %32
147 }
148
149 ; Function Attrs: nounwind uwtable
150 define void @rec_init(i64* %arr, i32 %curr_offset, i32* %static_offsets, i32* %indexes,
    ↪ i32* %dims, i32 %dimc, i32 %dim_curr) #0 {
151  %1 = alloca i64*, align 8
152  %2 = alloca i32, align 4
153  %3 = alloca i32*, align 8
154  %4 = alloca i32*, align 8
155  %5 = alloca i32*, align 8
156  %6 = alloca i32, align 4
157  %7 = alloca i32, align 4
158  %static_offset = alloca i32, align 4
159  %dynamic_offset = alloca i32, align 4
160  %i = alloca i32, align 4
161  %tmp = alloca i32, align 4
162  %j = alloca i32, align 4
163  %i1 = alloca i32, align 4
164  %offset = alloca i32, align 4
165  %sub = alloca i64*, align 8
166  store i64* %arr, i64** %1, align 8
167  store i32 %curr_offset, i32* %2, align 4

```

```

168     store i32* %static_offsets, i32** %3, align 8
169     store i32* %indexes, i32** %4, align 8
170     store i32* %dims, i32** %5, align 8
171     store i32 %dimc, i32* %6, align 4
172     store i32 %dim_curr, i32* %7, align 4
173     %8 = load i32, i32* %7, align 4
174     %9 = sext i32 %8 to i64
175     %10 = load i32*, i32** %5, align 8
176     %11 = getelementptr inbounds i32, i32* %10, i64 %9
177     %12 = load i32, i32* %11, align 4
178     %13 = sext i32 %12 to i64
179     %14 = load i32, i32* %2, align 4
180     %15 = sext i32 %14 to i64
181     %16 = load i64*, i64** %1, align 8
182     %17 = getelementptr inbounds i64, i64* %16, i64 %15
183     store i64 %13, i64* %17, align 8
184     %18 = load i32, i32* %7, align 4
185     %19 = add nsw i32 %18, 1
186     %20 = load i32, i32* %6, align 4
187     %21 = icmp sge i32 %19, %20
188     br i1 %21, label %22, label %23
189
190 ; <label>:22                                ; preds = %0
191     br label %115
192
193 ; <label>:23                                ; preds = %0
194     %24 = load i32, i32* %7, align 4
195     %25 = sext i32 %24 to i64
196     %26 = load i32*, i32** %3, align 8
197     %27 = getelementptr inbounds i32, i32* %26, i64 %25
198     %28 = load i32, i32* %27, align 4
199     store i32 %28, i32* %static_offset, align 4
200     store i32 0, i32* %dynamic_offset, align 4
201     store i32 0, i32* %i, align 4
202     br label %29
203
204 ; <label>:29                                ; preds = %60, %23
205     %30 = load i32, i32* %i, align 4
206     %31 = load i32, i32* %7, align 4
207     %32 = icmp slt i32 %30, %31
208     br i1 %32, label %33, label %63
209
210 ; <label>:33                                ; preds = %29
211     %34 = load i32, i32* %i, align 4
212     %35 = sext i32 %34 to i64
213     %36 = load i32*, i32** %4, align 8
214     %37 = getelementptr inbounds i32, i32* %36, i64 %35
215     %38 = load i32, i32* %37, align 4
216     store i32 %38, i32* %tmp, align 4

```

```
217 %39 = load i32, i32* %i, align 4
218 %40 = add nsw i32 %39, 1
219 store i32 %40, i32* %j, align 4
220 br label %41
221
222 ; <label>:41                                ; preds = %53, %33
223 %42 = load i32, i32* %j, align 4
224 %43 = load i32, i32* %7, align 4
225 %44 = icmp sle i32 %42, %43
226 br i1 %44, label %45, label %56
227
228 ; <label>:45                                ; preds = %41
229 %46 = load i32, i32* %j, align 4
230 %47 = sext i32 %46 to i64
231 %48 = load i32*, i32** %5, align 8
232 %49 = getelementptr inbounds i32, i32* %48, i64 %47
233 %50 = load i32, i32* %49, align 4
234 %51 = load i32, i32* %tmp, align 4
235 %52 = mul nsw i32 %51, %50
236 store i32 %52, i32* %tmp, align 4
237 br label %53
238
239 ; <label>:53                                ; preds = %45
240 %54 = load i32, i32* %j, align 4
241 %55 = add nsw i32 %54, 1
242 store i32 %55, i32* %j, align 4
243 br label %41
244
245 ; <label>:56                                ; preds = %41
246 %57 = load i32, i32* %tmp, align 4
247 %58 = load i32, i32* %dynamic_offset, align 4
248 %59 = add nsw i32 %58, %57
249 store i32 %59, i32* %dynamic_offset, align 4
250 br label %60
251
252 ; <label>:60                                ; preds = %56
253 %61 = load i32, i32* %i, align 4
254 %62 = add nsw i32 %61, 1
255 store i32 %62, i32* %i, align 4
256 br label %29
257
258 ; <label>:63                                ; preds = %29
259 store i32 0, i32* %i1, align 4
260 br label %64
261
262 ; <label>:64                                ; preds = %112, %63
263 %65 = load i32, i32* %i1, align 4
264 %66 = load i32, i32* %7, align 4
265 %67 = sext i32 %66 to i64
```



```

266 %68 = load i32*, i32** %5, align 8
267 %69 = getelementptr inbounds i32, i32* %68, i64 %67
268 %70 = load i32, i32* %69, align 4
269 %71 = icmp slt i32 %65, %70
270 br i1 %71, label %72, label %115
271
272 ; <label>:72                                ; preds = %64
273 %73 = load i32, i32* %static_offset, align 4
274 %74 = load i32, i32* %dynamic_offset, align 4
275 %75 = load i32, i32* %i1, align 4
276 %76 = add nsw i32 %74, %75
277 %77 = load i32, i32* %7, align 4
278 %78 = add nsw i32 %77, 1
279 %79 = sext i32 %78 to i64
280 %80 = load i32*, i32** %5, align 8
281 %81 = getelementptr inbounds i32, i32* %80, i64 %79
282 %82 = load i32, i32* %81, align 4
283 %83 = add nsw i32 %82, 1
284 %84 = mul nsw i32 %76, %83
285 %85 = add nsw i32 %73, %84
286 store i32 %85, i32* %offset, align 4
287 %86 = load i64*, i64** %1, align 8
288 %87 = load i32, i32* %offset, align 4
289 %88 = sext i32 %87 to i64
290 %89 = getelementptr inbounds i64, i64* %86, i64 %88
291 store i64* %89, i64** %sub, align 8
292 %90 = load i64*, i64** %sub, align 8
293 %91 = ptrtoint i64* %90 to i64
294 %92 = load i32, i32* %2, align 4
295 %93 = add nsw i32 %92, 1
296 %94 = load i32, i32* %i1, align 4
297 %95 = add nsw i32 %93, %94
298 %96 = sext i32 %95 to i64
299 %97 = load i64*, i64** %1, align 8
300 %98 = getelementptr inbounds i64, i64* %97, i64 %96
301 store i64 %91, i64* %98, align 8
302 %99 = load i32, i32* %i1, align 4
303 %100 = load i32, i32* %7, align 4
304 %101 = sext i32 %100 to i64
305 %102 = load i32*, i32** %4, align 8
306 %103 = getelementptr inbounds i32, i32* %102, i64 %101
307 store i32 %99, i32* %103, align 4
308 %104 = load i64*, i64** %1, align 8
309 %105 = load i32, i32* %offset, align 4
310 %106 = load i32*, i32** %3, align 8
311 %107 = load i32*, i32** %4, align 8
312 %108 = load i32*, i32** %5, align 8
313 %109 = load i32, i32* %6, align 4
314 %110 = load i32, i32* %7, align 4

```

```

315     %111 = add nsw i32 %110, 1
316     call void @rec_init(i64* %104, i32 %105, i32* %106, i32* %107, i32* %108, i32 %109, i32
    ↪ %111)
317     br label %112
318
319 ; <label>:112                                ; preds = %72
320     %113 = load i32, i32* %i1, align 4
321     %114 = add nsw i32 %113, 1
322     store i32 %114, i32* %i1, align 4
323     br label %64
324
325 ; <label>:115                                ; preds = %22, %64
326     ret void
327 }
328
329 ; Function Attrs: nounwind uwtable
330 define i64* @init_arr(i32* %dims, i32 %dimc) #0 {
331     %1 = alloca i32*, align 8
332     %2 = alloca i32, align 4
333     %3 = alloca i8*
334     %total = alloca i32, align 4
335     %i = alloca i32, align 4
336     %j = alloca i32, align 4
337     %i1 = alloca i32, align 4
338     %length = alloca i32, align 4
339     %i2 = alloca i32, align 4
340     %tmp = alloca i32, align 4
341     %j3 = alloca i32, align 4
342     %arr = alloca i64*, align 8
343     %i4 = alloca i32, align 4
344     store i32* %dims, i32** %1, align 8
345     store i32 %dimc, i32* %2, align 4
346     %4 = load i32, i32* %2, align 4
347     %5 = zext i32 %4 to i64
348     %6 = call i8* @llvm.stacksave()
349     store i8* %6, i8** %3
350     %7 = alloca i32, i64 %5, align 16
351     store i32 0, i32* %total, align 4
352     store i32 0, i32* %i, align 4
353     br label %8
354
355 ; <label>:8                                    ; preds = %56, %0
356     %9 = load i32, i32* %i, align 4
357     %10 = load i32, i32* %2, align 4
358     %11 = icmp slt i32 %9, %10
359     br i1 %11, label %12, label %59
360
361 ; <label>:12                                    ; preds = %8
362     %13 = load i32, i32* %i, align 4

```

```

363 %14 = sext i32 %13 to i64
364 %15 = getelementptr inbounds i32, i32* %7, i64 %14
365 store i32 1, i32* %15, align 4
366 store i32 0, i32* %j, align 4
367 br label %16
368
369 ; <label>:16                                ; preds = %31, %12
370 %17 = load i32, i32* %j, align 4
371 %18 = load i32, i32* %i, align 4
372 %19 = icmp slt i32 %17, %18
373 br i1 %19, label %20, label %34
374
375 ; <label>:20                                ; preds = %16
376 %21 = load i32, i32* %j, align 4
377 %22 = sext i32 %21 to i64
378 %23 = load i32*, i32** %1, align 8
379 %24 = getelementptr inbounds i32, i32* %23, i64 %22
380 %25 = load i32, i32* %24, align 4
381 %26 = load i32, i32* %i, align 4
382 %27 = sext i32 %26 to i64
383 %28 = getelementptr inbounds i32, i32* %7, i64 %27
384 %29 = load i32, i32* %28, align 4
385 %30 = mul nsw i32 %29, %25
386 store i32 %30, i32* %28, align 4
387 br label %31
388
389 ; <label>:31                                ; preds = %20
390 %32 = load i32, i32* %j, align 4
391 %33 = add nsw i32 %32, 1
392 store i32 %33, i32* %j, align 4
393 br label %16
394
395 ; <label>:34                                ; preds = %16
396 %35 = load i32, i32* %i, align 4
397 %36 = sext i32 %35 to i64
398 %37 = load i32*, i32** %1, align 8
399 %38 = getelementptr inbounds i32, i32* %37, i64 %36
400 %39 = load i32, i32* %38, align 4
401 %40 = add nsw i32 %39, 1
402 %41 = load i32, i32* %i, align 4
403 %42 = sext i32 %41 to i64
404 %43 = getelementptr inbounds i32, i32* %7, i64 %42
405 %44 = load i32, i32* %43, align 4
406 %45 = mul nsw i32 %44, %40
407 store i32 %45, i32* %43, align 4
408 %46 = load i32, i32* %total, align 4
409 %47 = load i32, i32* %i, align 4
410 %48 = sext i32 %47 to i64
411 %49 = getelementptr inbounds i32, i32* %7, i64 %48

```

```

412 %50 = load i32, i32* %49, align 4
413 %51 = add nsw i32 %50, %46
414 store i32 %51, i32* %49, align 4
415 %52 = load i32, i32* %i, align 4
416 %53 = sext i32 %52 to i64
417 %54 = getelementptr inbounds i32, i32* %7, i64 %53
418 %55 = load i32, i32* %54, align 4
419 store i32 %55, i32* %total, align 4
420 br label %56
421
422 ; <label>:56                                ; preds = %34
423 %57 = load i32, i32* %i, align 4
424 %58 = add nsw i32 %57, 1
425 store i32 %58, i32* %i, align 4
426 br label %8
427
428 ; <label>:59                                ; preds = %8
429 %60 = load i32, i32* %2, align 4
430 %61 = zext i32 %60 to i64
431 %62 = alloca i32, i64 %61, align 16
432 store i32 0, i32* %i1, align 4
433 br label %63
434
435 ; <label>:63                                ; preds = %71, %59
436 %64 = load i32, i32* %i1, align 4
437 %65 = load i32, i32* %2, align 4
438 %66 = icmp slt i32 %64, %65
439 br i1 %66, label %67, label %74
440
441 ; <label>:67                                ; preds = %63
442 %68 = load i32, i32* %i1, align 4
443 %69 = sext i32 %68 to i64
444 %70 = getelementptr inbounds i32, i32* %62, i64 %69
445 store i32 0, i32* %70, align 4
446 br label %71
447
448 ; <label>:71                                ; preds = %67
449 %72 = load i32, i32* %i1, align 4
450 %73 = add nsw i32 %72, 1
451 store i32 %73, i32* %i1, align 4
452 br label %63
453
454 ; <label>:74                                ; preds = %63
455 store i32 0, i32* %length, align 4
456 store i32 0, i32* %i2, align 4
457 br label %75
458
459 ; <label>:75                                ; preds = %108, %74
460 %76 = load i32, i32* %i2, align 4

```

```
461 %77 = load i32, i32* %2, align 4
462 %78 = icmp slt i32 %76, %77
463 br i1 %78, label %79, label %111
464
465 ; <label>:79                                ; preds = %75
466 store i32 1, i32* %tmp, align 4
467 %80 = load i32, i32* %i2, align 4
468 %81 = sub nsw i32 %80, 1
469 store i32 %81, i32* %j3, align 4
470 br label %82
471
472 ; <label>:82                                ; preds = %93, %79
473 %83 = load i32, i32* %j3, align 4
474 %84 = icmp sge i32 %83, 0
475 br i1 %84, label %85, label %96
476
477 ; <label>:85                                ; preds = %82
478 %86 = load i32, i32* %j3, align 4
479 %87 = sext i32 %86 to i64
480 %88 = load i32*, i32** %1, align 8
481 %89 = getelementptr inbounds i32, i32* %88, i64 %87
482 %90 = load i32, i32* %89, align 4
483 %91 = load i32, i32* %tmp, align 4
484 %92 = mul nsw i32 %91, %90
485 store i32 %92, i32* %tmp, align 4
486 br label %93
487
488 ; <label>:93                                ; preds = %85
489 %94 = load i32, i32* %j3, align 4
490 %95 = add nsw i32 %94, -1
491 store i32 %95, i32* %j3, align 4
492 br label %82
493
494 ; <label>:96                                ; preds = %82
495 %97 = load i32, i32* %i2, align 4
496 %98 = sext i32 %97 to i64
497 %99 = load i32*, i32** %1, align 8
498 %100 = getelementptr inbounds i32, i32* %99, i64 %98
499 %101 = load i32, i32* %100, align 4
500 %102 = add nsw i32 %101, 1
501 %103 = load i32, i32* %tmp, align 4
502 %104 = mul nsw i32 %103, %102
503 store i32 %104, i32* %tmp, align 4
504 %105 = load i32, i32* %tmp, align 4
505 %106 = load i32, i32* %length, align 4
506 %107 = add nsw i32 %106, %105
507 store i32 %107, i32* %length, align 4
508 br label %108
509
```

```

510 ; <label>:108                                ; preds = %96
511   %109 = load i32, i32* %i2, align 4
512   %110 = add nsw i32 %109, 1
513   store i32 %110, i32* %i2, align 4
514   br label %75
515
516 ; <label>:111                                ; preds = %75
517   %112 = load i32, i32* %length, align 4
518   %113 = sext i32 %112 to i64
519   %114 = call noalias i8* @bitcast (i8* (i32)* @malloc to i8* (i64*)) (i64 %113) #1
520   %115 = bitcast i8* %114 to i64*
521   store i64* %115, i64** %arr, align 8
522   store i32 0, i32* %i4, align 4
523   br label %116
524
525 ; <label>:116                                ; preds = %125, %111
526   %117 = load i32, i32* %i4, align 4
527   %118 = load i32, i32* %length, align 4
528   %119 = icmp slt i32 %117, %118
529   br i1 %119, label %120, label %128
530
531 ; <label>:120                                ; preds = %116
532   %121 = load i32, i32* %i4, align 4
533   %122 = sext i32 %121 to i64
534   %123 = load i64*, i64** %arr, align 8
535   %124 = getelementptr inbounds i64, i64* %123, i64 %122
536   store i64 0, i64* %124, align 8
537   br label %125
538
539 ; <label>:125                                ; preds = %120
540   %126 = load i32, i32* %i4, align 4
541   %127 = add nsw i32 %126, 1
542   store i32 %127, i32* %i4, align 4
543   br label %116
544
545 ; <label>:128                                ; preds = %116
546   %129 = load i64*, i64** %arr, align 8
547   %130 = load i32*, i32** %1, align 8
548   %131 = load i32, i32* %2, align 4
549   call void @rec_init(i64* %129, i32 0, i32* %7, i32* %62, i32* %130, i32 %131, i32 0)
550   %132 = load i64*, i64** %arr, align 8
551   %133 = load i8*, i8** %3
552   call void @llvm.stackrestore(i8* %133)
553   ret i64* %132
554 }
555
556 ; Function Attrs: nounwind
557 declare i8* @llvm.stacksave() #1
558

```

```
559 ; Function Attrs: nounwind
560 declare void @llvm.stackrestore(i8*) #1
561
562 attributes #0 = { nounwind uwtable "disable-tail-calls"="false"
↪ "less-precise-fpmad"="false" "no-frame-pointer-elim"="true"
↪ "no-frame-pointer-elim-non-leaf" "no-infs-fp-math"="false" "no-nans-fp-math"="false"
↪ "stack-protector-buffer-size"="8" "target-cpu"="x86-64"
↪ "target-features"="+sse,+sse2" "unsafe-fp-math"="false" "use-soft-float"="false" }
563 attributes #1 = { nounwind }
564
565 !llvm.ident = !{!0}
566
567 !0 = !{"Ubuntu clang version 3.7.0-2ubuntu1 (tags/RELEASE_370/final) (based on LLVM
↪ 3.7.0)"}

```

Class Extends Example

The following test checks if a child class inherits the parent's fields:

test-classExtends.dice

```

1  class shape {
2      public float xCoord;
3      public float yCoord;
4  }
5
6  class circle extends shape {
7      public float radius;
8  }
9
10 class test {
11     public void main(char[] [] args) {
12         class circle a = new circle();
13         a.xCoord = 1.5;
14         print(a.xCoord);
15     }
16 }
```

test-classExtends.ll

```

1      ; ModuleID = 'Dice Codegen'
2      target datalayout = "e-m:e-i64:64-f80:128-n8:16:32:64-S128"
3      target triple = "x86_64-pc-linux-gnu"
4
5      %test = type <{ i32 }>
6      %circle = type <{ i32, double, double, double }>
7      %shape = type <{ i32, double, double }>
8
9      @tmp = private unnamed_addr constant [3 x i8] c"%f\00"
10
11      declare i32 @printf(i8*, ...)
12
13      declare noalias i8* @malloc(i32)
14
15      declare i32 @open(i8*, i32)
16
17      declare i32 @close(i32)
18
19      declare i32 @read(i32, i8*, i32)
20
21      declare i32 @write(i32, i8*, i32)
22
23      declare i32 @lseek(i32, i32, i32)
24
25      declare void @exit(i32)
```



```

26
27     declare i8* @realloc(i8*, i32)
28
29     declare i32 @getchar()
30
31     define i64* @lookup(i32 %c_index, i32 %f_index) {
32     entry:
33         %tmp = alloca i64**, i32 3
34         %tmp1 = alloca i64*, i32 0
35         %tmp2 = getelementptr i64**, i64*** %tmp, i32 2
36         store i64** %tmp1, i64*** %tmp2
37         %tmp3 = alloca i64*, i32 0
38         %tmp4 = getelementptr i64**, i64*** %tmp, i32 1
39         store i64** %tmp3, i64*** %tmp4
40         %tmp5 = alloca i64*, i32 0
41         %tmp6 = getelementptr i64**, i64*** %tmp, i32 0
42         store i64** %tmp5, i64*** %tmp6
43         ret i64* null
44     }
45
46     define %test* @test.constructor() {
47     entry:
48         %this = alloca %test
49         %tmp = call i8* @malloc(i32 ptrtoint (i1** getelementptr (i1*, i1** null, i32 1) to i32))
50         %tmp1 = bitcast i8* %tmp to %test*
51         %tmp2 = load %test, %test* %tmp1
52         store %test %tmp2, %test* %this
53         %.key = getelementptr inbounds %test, %test* %this, i32 0, i32 0
54         store i32 2, i32* %.key
55         ret %test* %this
56     }
57
58     define %circle* @circle.constructor() {
59     entry:
60         %this = alloca %circle
61         %tmp = call i8* @malloc(i32 ptrtoint (i1** getelementptr (i1*, i1** null, i32 1) to i32))
62         %tmp1 = bitcast i8* %tmp to %circle*
63         %tmp2 = load %circle, %circle* %tmp1
64         store %circle %tmp2, %circle* %this
65         %.key = getelementptr inbounds %circle, %circle* %this, i32 0, i32 0
66         store i32 1, i32* %.key
67         ret %circle* %this
68     }
69
70     define %shape* @shape.constructor() {
71     entry:
72         %this = alloca %shape
73         %tmp = call i8* @malloc(i32 ptrtoint (i1** getelementptr (i1*, i1** null, i32 1) to i32))
74         %tmp1 = bitcast i8* %tmp to %shape*

```

```

75 %tmp2 = load %shape, %shape* %tmp1
76 store %shape %tmp2, %shape* %this
77 %.key = getelementptr inbounds %shape, %shape* %this, i32 0, i32 0
78 store i32 0, i32* %.key
79 ret %shape* %this
80 }
81
82 define i32 @main(i32 %argc, i8** %argv) {
83 entry:
84 %arr_size = add i32 %argc, 1
85 %mallocsize = mul i32 %arr_size, ptrtoint (i1** getelementptr (i1*, i1** null, i32 1) to
    ↪ i32)
86 %malloccall = tail call i8* @malloc(i32 %mallocsize)
87 %args = bitcast i8* %malloccall to i8***
88 %args1 = bitcast i8*** %args to i8**
89 %argc_len = bitcast i8** %args1 to i32*
90 %arr_1 = getelementptr i8*, i8** %args1, i32 1
91 store i32 %argc, i32* %argc_len
92 br label %args.cond
93
94 args.cond:                                     ; preds = %args.init, %entry
95 %counter = phi i32 [ 0, %entry ], [ %tmp, %args.init ]
96 %tmp = add i32 %counter, 1
97 %tmp2 = icmp slt i32 %counter, %argc
98 br i1 %tmp2, label %args.init, label %args.done
99
100 args.init:                                     ; preds = %args.cond
101 %tmp3 = getelementptr i8*, i8** %arr_1, i32 %counter
102 %tmp4 = getelementptr i8*, i8** %argv, i32 %counter
103 %tmp5 = load i8*, i8** %tmp4
104 store i8* %tmp5, i8** %tmp3
105 br label %args.cond
106
107 args.done:                                     ; preds = %args.cond
108 %this = alloca %test
109 %tmp6 = call i8* @malloc(i32 ptrtoint (i1** getelementptr (i1*, i1** null, i32 1) to
    ↪ i32))
110 %tmp7 = bitcast i8* %tmp6 to %test*
111 %tmp8 = load %test, %test* %tmp7
112 store %test %tmp8, %test* %this
113 %.key = getelementptr inbounds %test, %test* %this, i32 0, i32 0
114 store i32 2, i32* %.key
115 %a = alloca %circle
116 %tmp9 = call %circle* @circle.constructor()
117 %tmp10 = load %circle, %circle* %tmp9
118 store %circle %tmp10, %circle* %a
119 %xCoord = getelementptr inbounds %circle, %circle* %a, i32 0, i32 2
120 store double 1.500000e+00, double* %xCoord
121 %xCoord11 = getelementptr inbounds %circle, %circle* %a, i32 0, i32 2

```

```

122 %xCoord12 = load double, double* %xCoord11
123 %tmp13 = call i32 (i8*, ...) @printf(i8* getelementptr inbounds ([3 x i8], [3 x i8]*
    ↪ @tmp, i32 0, i32 0), double %xCoord12)
124 ret i32 0
125 }
126
127 ; Function Attrs: nounwind uwtable
128 define i8* @input() #0 {
129     %initial_size = alloca i32, align 4
130     %str = alloca i8*, align 8
131     %index = alloca i32, align 4
132     %tmp = alloca i8, align 1
133     store i32 100, i32* %initial_size, align 4
134     %1 = load i32, i32* %initial_size, align 4
135     %2 = sext i32 %1 to i64
136     %3 = call noalias i8* bitcast (i8* (i32)* @malloc to i8* (i64)*) (i64 %2) #1
137     store i8* %3, i8** %str, align 8
138     store i32 0, i32* %index, align 4
139     store i8 48, i8* %tmp, align 1
140     br label %4
141
142     ; <label>:4                                ; preds = %20, %0
143     %5 = call i32 @getchar()
144     %6 = trunc i32 %5 to i8
145     store i8 %6, i8* %tmp, align 1
146     %7 = sext i8 %6 to i32
147     %8 = icmp ne i32 %7, 10
148     br i1 %8, label %9, label %27
149
150     ; <label>:9                                ; preds = %4
151     %10 = load i32, i32* %index, align 4
152     %11 = load i32, i32* %initial_size, align 4
153     %12 = sub nsw i32 %11, 1
154     %13 = icmp sge i32 %10, %12
155     br i1 %13, label %14, label %20
156
157     ; <label>:14                               ; preds = %9
158     %15 = load i8*, i8** %str, align 8
159     %16 = load i32, i32* %initial_size, align 4
160     %17 = mul nsw i32 %16, 2
161     store i32 %17, i32* %initial_size, align 4
162     %18 = sext i32 %17 to i64
163     %19 = call i8* bitcast (i8* (i8*, i32)* @realloc to i8* (i8*, i64)*) (i8* %15, i64
    ↪ %18) #1
164     store i8* %19, i8** %str, align 8
165     br label %20
166
167     ; <label>:20                               ; preds = %14, %9
168     %21 = load i8, i8* %tmp, align 1

```

```

169     %22 = load i32, i32* %index, align 4
170     %23 = add nsw i32 %22, 1
171     store i32 %23, i32* %index, align 4
172     %24 = sext i32 %22 to i64
173     %25 = load i8*, i8** %str, align 8
174     %26 = getelementptr inbounds i8, i8* %25, i64 %24
175     store i8 %21, i8* %26, align 1
176     br label %4
177
178     ; <label>:27                                ; preds = %4
179     %28 = load i32, i32* %index, align 4
180     %29 = sext i32 %28 to i64
181     %30 = load i8*, i8** %str, align 8
182     %31 = getelementptr inbounds i8, i8* %30, i64 %29
183     store i8 0, i8* %31, align 1
184     %32 = load i8*, i8** %str, align 8
185     ret i8* %32
186 }
187
188 ; Function Attrs: nounwind uwtable
189 define void @rec_init(i64* %arr, i32 %curr_offset, i32* %static_offsets, i32* %indexes,
190     ↪ i32* %dims, i32 %dimc, i32 %dim_curr) #0 {
191     %1 = alloca i64*, align 8
192     %2 = alloca i32, align 4
193     %3 = alloca i32*, align 8
194     %4 = alloca i32*, align 8
195     %5 = alloca i32*, align 8
196     %6 = alloca i32, align 4
197     %7 = alloca i32, align 4
198     %static_offset = alloca i32, align 4
199     %dynamic_offset = alloca i32, align 4
200     %i = alloca i32, align 4
201     %tmp = alloca i32, align 4
202     %j = alloca i32, align 4
203     %i1 = alloca i32, align 4
204     %offset = alloca i32, align 4
205     %sub = alloca i64*, align 8
206     store i64* %arr, i64** %1, align 8
207     store i32 %curr_offset, i32* %2, align 4
208     store i32* %static_offsets, i32** %3, align 8
209     store i32* %indexes, i32** %4, align 8
210     store i32* %dims, i32** %5, align 8
211     store i32 %dimc, i32* %6, align 4
212     store i32 %dim_curr, i32* %7, align 4
213     %8 = load i32, i32* %7, align 4
214     %9 = sext i32 %8 to i64
215     %10 = load i32*, i32** %5, align 8
216     %11 = getelementptr inbounds i32, i32* %10, i64 %9
217     %12 = load i32, i32* %11, align 4

```

```

217 %13 = sext i32 %12 to i64
218 %14 = load i32, i32* %2, align 4
219 %15 = sext i32 %14 to i64
220 %16 = load i64*, i64** %1, align 8
221 %17 = getelementptr inbounds i64, i64* %16, i64 %15
222 store i64 %13, i64* %17, align 8
223 %18 = load i32, i32* %7, align 4
224 %19 = add nsw i32 %18, 1
225 %20 = load i32, i32* %6, align 4
226 %21 = icmp sge i32 %19, %20
227 br i1 %21, label %22, label %23
228
229 ; <label>:22                                ; preds = %0
230 br label %115
231
232 ; <label>:23                                ; preds = %0
233 %24 = load i32, i32* %7, align 4
234 %25 = sext i32 %24 to i64
235 %26 = load i32*, i32** %3, align 8
236 %27 = getelementptr inbounds i32, i32* %26, i64 %25
237 %28 = load i32, i32* %27, align 4
238 store i32 %28, i32* %static_offset, align 4
239 store i32 0, i32* %dynamic_offset, align 4
240 store i32 0, i32* %i, align 4
241 br label %29
242
243 ; <label>:29                                ; preds = %60, %23
244 %30 = load i32, i32* %i, align 4
245 %31 = load i32, i32* %7, align 4
246 %32 = icmp slt i32 %30, %31
247 br i1 %32, label %33, label %63
248
249 ; <label>:33                                ; preds = %29
250 %34 = load i32, i32* %i, align 4
251 %35 = sext i32 %34 to i64
252 %36 = load i32*, i32** %4, align 8
253 %37 = getelementptr inbounds i32, i32* %36, i64 %35
254 %38 = load i32, i32* %37, align 4
255 store i32 %38, i32* %tmp, align 4
256 %39 = load i32, i32* %i, align 4
257 %40 = add nsw i32 %39, 1
258 store i32 %40, i32* %j, align 4
259 br label %41
260
261 ; <label>:41                                ; preds = %53, %33
262 %42 = load i32, i32* %j, align 4
263 %43 = load i32, i32* %7, align 4
264 %44 = icmp sle i32 %42, %43
265 br i1 %44, label %45, label %56

```

```
266
267 ; <label>:45                                ; preds = %41
268 %46 = load i32, i32* %j, align 4
269 %47 = sext i32 %46 to i64
270 %48 = load i32*, i32** %5, align 8
271 %49 = getelementptr inbounds i32, i32* %48, i64 %47
272 %50 = load i32, i32* %49, align 4
273 %51 = load i32, i32* %tmp, align 4
274 %52 = mul nsw i32 %51, %50
275 store i32 %52, i32* %tmp, align 4
276 br label %53
277
278 ; <label>:53                                ; preds = %45
279 %54 = load i32, i32* %j, align 4
280 %55 = add nsw i32 %54, 1
281 store i32 %55, i32* %j, align 4
282 br label %41
283
284 ; <label>:56                                ; preds = %41
285 %57 = load i32, i32* %tmp, align 4
286 %58 = load i32, i32* %dynamic_offset, align 4
287 %59 = add nsw i32 %58, %57
288 store i32 %59, i32* %dynamic_offset, align 4
289 br label %60
290
291 ; <label>:60                                ; preds = %56
292 %61 = load i32, i32* %i, align 4
293 %62 = add nsw i32 %61, 1
294 store i32 %62, i32* %i, align 4
295 br label %29
296
297 ; <label>:63                                ; preds = %29
298 store i32 0, i32* %i1, align 4
299 br label %64
300
301 ; <label>:64                                ; preds = %112, %63
302 %65 = load i32, i32* %i1, align 4
303 %66 = load i32, i32* %7, align 4
304 %67 = sext i32 %66 to i64
305 %68 = load i32*, i32** %5, align 8
306 %69 = getelementptr inbounds i32, i32* %68, i64 %67
307 %70 = load i32, i32* %69, align 4
308 %71 = icmp slt i32 %65, %70
309 br i1 %71, label %72, label %115
310
311 ; <label>:72                                ; preds = %64
312 %73 = load i32, i32* %static_offset, align 4
313 %74 = load i32, i32* %dynamic_offset, align 4
314 %75 = load i32, i32* %i1, align 4
```

```

315 %76 = add nsw i32 %74, %75
316 %77 = load i32, i32* %7, align 4
317 %78 = add nsw i32 %77, 1
318 %79 = sext i32 %78 to i64
319 %80 = load i32*, i32** %5, align 8
320 %81 = getelementptr inbounds i32, i32* %80, i64 %79
321 %82 = load i32, i32* %81, align 4
322 %83 = add nsw i32 %82, 1
323 %84 = mul nsw i32 %76, %83
324 %85 = add nsw i32 %73, %84
325 store i32 %85, i32* %offset, align 4
326 %86 = load i64*, i64** %1, align 8
327 %87 = load i32, i32* %offset, align 4
328 %88 = sext i32 %87 to i64
329 %89 = getelementptr inbounds i64, i64* %86, i64 %88
330 store i64* %89, i64** %sub, align 8
331 %90 = load i64*, i64** %sub, align 8
332 %91 = ptrtoint i64* %90 to i64
333 %92 = load i32, i32* %2, align 4
334 %93 = add nsw i32 %92, 1
335 %94 = load i32, i32* %i1, align 4
336 %95 = add nsw i32 %93, %94
337 %96 = sext i32 %95 to i64
338 %97 = load i64*, i64** %1, align 8
339 %98 = getelementptr inbounds i64, i64* %97, i64 %96
340 store i64 %91, i64* %98, align 8
341 %99 = load i32, i32* %i1, align 4
342 %100 = load i32, i32* %7, align 4
343 %101 = sext i32 %100 to i64
344 %102 = load i32*, i32** %4, align 8
345 %103 = getelementptr inbounds i32, i32* %102, i64 %101
346 store i32 %99, i32* %103, align 4
347 %104 = load i64*, i64** %1, align 8
348 %105 = load i32, i32* %offset, align 4
349 %106 = load i32*, i32** %3, align 8
350 %107 = load i32*, i32** %4, align 8
351 %108 = load i32*, i32** %5, align 8
352 %109 = load i32, i32* %6, align 4
353 %110 = load i32, i32* %7, align 4
354 %111 = add nsw i32 %110, 1
355 call void @rec_init(i64* %104, i32 %105, i32* %106, i32* %107, i32* %108, i32 %109, i32
    ↪ %111)
356 br label %112
357
358 ; <label>:112                                ; preds = %72
359 %113 = load i32, i32* %i1, align 4
360 %114 = add nsw i32 %113, 1
361 store i32 %114, i32* %i1, align 4
362 br label %64

```

```

363
364 ; <label>:115                                ; preds = %22, %64
365 ret void
366 }
367
368 ; Function Attrs: nounwind uwtable
369 define i64* @init_arr(i32* %dims, i32 %dimc) #0 {
370 %1 = alloca i32*, align 8
371 %2 = alloca i32, align 4
372 %3 = alloca i8*
373 %total = alloca i32, align 4
374 %i = alloca i32, align 4
375 %j = alloca i32, align 4
376 %i1 = alloca i32, align 4
377 %length = alloca i32, align 4
378 %i2 = alloca i32, align 4
379 %tmp = alloca i32, align 4
380 %j3 = alloca i32, align 4
381 %arr = alloca i64*, align 8
382 %i4 = alloca i32, align 4
383 store i32* %dims, i32** %1, align 8
384 store i32 %dimc, i32* %2, align 4
385 %4 = load i32, i32* %2, align 4
386 %5 = zext i32 %4 to i64
387 %6 = call i8* @llvm.stacksave()
388 store i8* %6, i8** %3
389 %7 = alloca i32, i64 %5, align 16
390 store i32 0, i32* %total, align 4
391 store i32 0, i32* %i, align 4
392 br label %8
393
394 ; <label>:8                                    ; preds = %56, %0
395 %9 = load i32, i32* %i, align 4
396 %10 = load i32, i32* %2, align 4
397 %11 = icmp slt i32 %9, %10
398 br i1 %11, label %12, label %59
399
400 ; <label>:12                                    ; preds = %8
401 %13 = load i32, i32* %i, align 4
402 %14 = sext i32 %13 to i64
403 %15 = getelementptr inbounds i32, i32* %7, i64 %14
404 store i32 1, i32* %15, align 4
405 store i32 0, i32* %j, align 4
406 br label %16
407
408 ; <label>:16                                    ; preds = %31, %12
409 %17 = load i32, i32* %j, align 4
410 %18 = load i32, i32* %i, align 4
411 %19 = icmp slt i32 %17, %18

```



```
412 br i1 %19, label %20, label %34
413
414 ; <label>:20                                ; preds = %16
415 %21 = load i32, i32* %j, align 4
416 %22 = sext i32 %21 to i64
417 %23 = load i32*, i32** %1, align 8
418 %24 = getelementptr inbounds i32, i32* %23, i64 %22
419 %25 = load i32, i32* %24, align 4
420 %26 = load i32, i32* %i, align 4
421 %27 = sext i32 %26 to i64
422 %28 = getelementptr inbounds i32, i32* %7, i64 %27
423 %29 = load i32, i32* %28, align 4
424 %30 = mul nsw i32 %29, %25
425 store i32 %30, i32* %28, align 4
426 br label %31
427
428 ; <label>:31                                ; preds = %20
429 %32 = load i32, i32* %j, align 4
430 %33 = add nsw i32 %32, 1
431 store i32 %33, i32* %j, align 4
432 br label %16
433
434 ; <label>:34                                ; preds = %16
435 %35 = load i32, i32* %i, align 4
436 %36 = sext i32 %35 to i64
437 %37 = load i32*, i32** %1, align 8
438 %38 = getelementptr inbounds i32, i32* %37, i64 %36
439 %39 = load i32, i32* %38, align 4
440 %40 = add nsw i32 %39, 1
441 %41 = load i32, i32* %i, align 4
442 %42 = sext i32 %41 to i64
443 %43 = getelementptr inbounds i32, i32* %7, i64 %42
444 %44 = load i32, i32* %43, align 4
445 %45 = mul nsw i32 %44, %40
446 store i32 %45, i32* %43, align 4
447 %46 = load i32, i32* %total, align 4
448 %47 = load i32, i32* %i, align 4
449 %48 = sext i32 %47 to i64
450 %49 = getelementptr inbounds i32, i32* %7, i64 %48
451 %50 = load i32, i32* %49, align 4
452 %51 = add nsw i32 %50, %46
453 store i32 %51, i32* %49, align 4
454 %52 = load i32, i32* %i, align 4
455 %53 = sext i32 %52 to i64
456 %54 = getelementptr inbounds i32, i32* %7, i64 %53
457 %55 = load i32, i32* %54, align 4
458 store i32 %55, i32* %total, align 4
459 br label %56
460
```

```
461 ; <label>:56 ; preds = %34
462 %57 = load i32, i32* %i, align 4
463 %58 = add nsw i32 %57, 1
464 store i32 %58, i32* %i, align 4
465 br label %8
466
467 ; <label>:59 ; preds = %8
468 %60 = load i32, i32* %2, align 4
469 %61 = zext i32 %60 to i64
470 %62 = alloca i32, i64 %61, align 16
471 store i32 0, i32* %i1, align 4
472 br label %63
473
474 ; <label>:63 ; preds = %71, %59
475 %64 = load i32, i32* %i1, align 4
476 %65 = load i32, i32* %2, align 4
477 %66 = icmp slt i32 %64, %65
478 br i1 %66, label %67, label %74
479
480 ; <label>:67 ; preds = %63
481 %68 = load i32, i32* %i1, align 4
482 %69 = sext i32 %68 to i64
483 %70 = getelementptr inbounds i32, i32* %62, i64 %69
484 store i32 0, i32* %70, align 4
485 br label %71
486
487 ; <label>:71 ; preds = %67
488 %72 = load i32, i32* %i1, align 4
489 %73 = add nsw i32 %72, 1
490 store i32 %73, i32* %i1, align 4
491 br label %63
492
493 ; <label>:74 ; preds = %63
494 store i32 0, i32* %length, align 4
495 store i32 0, i32* %i2, align 4
496 br label %75
497
498 ; <label>:75 ; preds = %108, %74
499 %76 = load i32, i32* %i2, align 4
500 %77 = load i32, i32* %2, align 4
501 %78 = icmp slt i32 %76, %77
502 br i1 %78, label %79, label %111
503
504 ; <label>:79 ; preds = %75
505 store i32 1, i32* %tmp, align 4
506 %80 = load i32, i32* %i2, align 4
507 %81 = sub nsw i32 %80, 1
508 store i32 %81, i32* %j3, align 4
509 br label %82
```

```

510
511 ; <label>:82                                ; preds = %93, %79
512 %83 = load i32, i32* %j3, align 4
513 %84 = icmp sge i32 %83, 0
514 br i1 %84, label %85, label %96
515
516 ; <label>:85                                ; preds = %82
517 %86 = load i32, i32* %j3, align 4
518 %87 = sext i32 %86 to i64
519 %88 = load i32*, i32** %1, align 8
520 %89 = getelementptr inbounds i32, i32* %88, i64 %87
521 %90 = load i32, i32* %89, align 4
522 %91 = load i32, i32* %tmp, align 4
523 %92 = mul nsw i32 %91, %90
524 store i32 %92, i32* %tmp, align 4
525 br label %93
526
527 ; <label>:93                                ; preds = %85
528 %94 = load i32, i32* %j3, align 4
529 %95 = add nsw i32 %94, -1
530 store i32 %95, i32* %j3, align 4
531 br label %82
532
533 ; <label>:96                                ; preds = %82
534 %97 = load i32, i32* %i2, align 4
535 %98 = sext i32 %97 to i64
536 %99 = load i32*, i32** %1, align 8
537 %100 = getelementptr inbounds i32, i32* %99, i64 %98
538 %101 = load i32, i32* %100, align 4
539 %102 = add nsw i32 %101, 1
540 %103 = load i32, i32* %tmp, align 4
541 %104 = mul nsw i32 %103, %102
542 store i32 %104, i32* %tmp, align 4
543 %105 = load i32, i32* %tmp, align 4
544 %106 = load i32, i32* %length, align 4
545 %107 = add nsw i32 %106, %105
546 store i32 %107, i32* %length, align 4
547 br label %108
548
549 ; <label>:108                               ; preds = %96
550 %109 = load i32, i32* %i2, align 4
551 %110 = add nsw i32 %109, 1
552 store i32 %110, i32* %i2, align 4
553 br label %75
554
555 ; <label>:111                               ; preds = %75
556 %112 = load i32, i32* %length, align 4
557 %113 = sext i32 %112 to i64
558 %114 = call noalias i8* @bitcast (i8* (i32)* @malloc to i8* (i64*)(i64 %113)) #1

```

```

559 %115 = bitcast i8* %114 to i64*
560 store i64* %115, i64** %arr, align 8
561 store i32 0, i32* %i4, align 4
562 br label %116
563
564 ; <label>:116                                ; preds = %125, %111
565 %117 = load i32, i32* %i4, align 4
566 %118 = load i32, i32* %length, align 4
567 %119 = icmp slt i32 %117, %118
568 br i1 %119, label %120, label %128
569
570 ; <label>:120                                ; preds = %116
571 %121 = load i32, i32* %i4, align 4
572 %122 = sext i32 %121 to i64
573 %123 = load i64*, i64** %arr, align 8
574 %124 = getelementptr inbounds i64, i64* %123, i64 %122
575 store i64 0, i64* %124, align 8
576 br label %125
577
578 ; <label>:125                                ; preds = %120
579 %126 = load i32, i32* %i4, align 4
580 %127 = add nsw i32 %126, 1
581 store i32 %127, i32* %i4, align 4
582 br label %116
583
584 ; <label>:128                                ; preds = %116
585 %129 = load i64*, i64** %arr, align 8
586 %130 = load i32*, i32** %1, align 8
587 %131 = load i32, i32* %2, align 4
588 call void @rec_init(i64* %129, i32 0, i32* %7, i32* %62, i32* %130, i32 %131, i32 0)
589 %132 = load i64*, i64** %arr, align 8
590 %133 = load i8*, i8** %3
591 call void @llvm.stackrestore(i8* %133)
592 ret i64* %132
593 }
594
595 ; Function Attrs: nounwind
596 declare i8* @llvm.stacksave() #1
597
598 ; Function Attrs: nounwind
599 declare void @llvm.stackrestore(i8*) #1
600
601 attributes #0 = { nounwind uwtable "disable-tail-calls"="false"
  ↳ "less-precise-fpmad"="false" "no-frame-pointer-elim"="true"
  ↳ "no-frame-pointer-elim-non-leaf" "no-infs-fp-math"="false" "no-nans-fp-math"="false"
  ↳ "stack-protector-buffer-size"="8" "target-cpu"="x86-64"
  ↳ "target-features"="+sse,+sse2" "unsafe-fp-math"="false" "use-soft-float"="false" }
602 attributes #1 = { nounwind }
603

```

```
604  !llvm.ident = !{!0}
605
606  !0 = !{"Ubuntu clang version 3.7.0-2ubuntu1 (tags/RELEASE_370/final) (based on LLVM
↪    3.7.0)"}
```

For Loop Test

The following test is the first of several for loop checks. This one ensures that the correct amount of iterations are complete for the specified block within the curly braces:

test-for1.dice

```

1  class test {
2      public void main(char[] [] args) {
3          int i;
4          for (i = 0 ; i < 5 ; i = i + 1) {
5              print(i);
6          }
7          print(42);
8      }
9  }
```

test-for1.ll

```

1      ; ModuleID = 'Dice Codegen'
2      target datalayout = "e-m:e-i64:64-f80:128-n8:16:32:64-S128"
3      target triple = "x86_64-pc-linux-gnu"
4
5      %test = type <{ i32 }>
6
7      @tmp = private unnamed_addr constant [3 x i8] c"%d\00"
8      @tmp.1 = private unnamed_addr constant [3 x i8] c"%d\00"
9
10     declare i32 @printf(i8*, ...)
11
12     declare noalias i8* @malloc(i32)
13
14     declare i32 @open(i8*, i32)
15
16     declare i32 @close(i32)
17
18     declare i32 @read(i32, i8*, i32)
19
20     declare i32 @write(i32, i8*, i32)
21
22     declare i32 @lseek(i32, i32, i32)
23
24     declare void @exit(i32)
25
26     declare i8* @realloc(i8*, i32)
27
28     declare i32 @getchar()
29
30     define i64* @lookup(i32 %c_index, i32 %f_index) {
31     entry:
```

```

32     %tmp = alloca i64**
33     %tmp1 = alloca i64*, i32 0
34     %tmp2 = getelementptr i64**, i64*** %tmp, i32 0
35     store i64** %tmp1, i64*** %tmp2
36     ret i64* null
37 }
38
39 define %test* @test.constructor() {
40 entry:
41     %this = alloca %test
42     %tmp = call i8* @malloc(i32 ptrtoint (i1** getelementptr (i1*, i1** null, i32 1) to i32))
43     %tmp1 = bitcast i8* %tmp to %test*
44     %tmp2 = load %test, %test* %tmp1
45     store %test %tmp2, %test* %this
46     %.key = getelementptr inbounds %test, %test* %this, i32 0, i32 0
47     store i32 0, i32* %.key
48     ret %test* %this
49 }
50
51 define i32 @main(i32 %argc, i8** %argv) {
52 entry:
53     %arr_size = add i32 %argc, 1
54     %mallocsize = mul i32 %arr_size, ptrtoint (i1** getelementptr (i1*, i1** null, i32 1) to
55     ↪ i32)
56     %malloccall = tail call i8* @malloc(i32 %mallocsize)
57     %args = bitcast i8* %malloccall to i8***
58     %args1 = bitcast i8*** %args to i8**
59     %argc_len = bitcast i8** %args1 to i32*
60     store i32 %argc, i32* %argc_len
61     br label %args.cond
62
63 args.cond:                                     ; preds = %args.init, %entry
64     %counter = phi i32 [ 0, %entry ], [ %tmp, %args.init ]
65     %tmp = add i32 %counter, 1
66     %tmp2 = icmp slt i32 %counter, %argc
67     br i1 %tmp2, label %args.init, label %args.done
68
69 args.init:                                     ; preds = %args.cond
70     %tmp3 = getelementptr i8*, i8** %arr_1, i32 %counter
71     %tmp4 = getelementptr i8*, i8** %argv, i32 %counter
72     %tmp5 = load i8*, i8** %tmp4
73     store i8* %tmp5, i8** %tmp3
74     br label %args.cond
75
76 args.done:                                     ; preds = %args.cond
77     %this = alloca %test
78     %tmp6 = call i8* @malloc(i32 ptrtoint (i1** getelementptr (i1*, i1** null, i32 1) to
79     ↪ i32))

```

```

79 %tmp7 = bitcast i8* %tmp6 to %test*
80 %tmp8 = load %test, %test* %tmp7
81 store %test %tmp8, %test* %this
82 %.key = getelementptr inbounds %test, %test* %this, i32 0, i32 0
83 store i32 0, i32* %.key
84 %i = alloca i32
85 store i32 0, i32* %i
86 br label %cond
87
88 loop:                                     ; preds = %cond
89 %i9 = load i32, i32* %i
90 %tmp10 = call i32 @i8*, ... @printf(i8* getelementptr inbounds ([3 x i8], [3 x i8]*
    ↪ @tmp, i32 0, i32 0), i32 %i9)
91 br label %inc
92
93 inc:                                     ; preds = %loop
94 %i11 = load i32, i32* %i
95 %addtmp = add i32 %i11, 1
96 store i32 %addtmp, i32* %i
97 br label %cond
98
99 cond:                                   ; preds = %inc, %args.done
100 %i12 = load i32, i32* %i
101 %lesstmp = icmp slt i32 %i12, 5
102 br i1 %lesstmp, label %loop, label %afterloop
103
104 afterloop:                             ; preds = %cond
105 %tmp13 = call i32 @i8*, ... @printf(i8* getelementptr inbounds ([3 x i8], [3 x i8]*
    ↪ @tmp.1, i32 0, i32 0), i32 42)
106 ret i32 0
107 }
108
109 ; Function Attrs: nounwind uwtable
110 define i8* @input() #0 {
111     %initial_size = alloca i32, align 4
112     %str = alloca i8*, align 8
113     %index = alloca i32, align 4
114     %tmp = alloca i8, align 1
115     store i32 100, i32* %initial_size, align 4
116     %1 = load i32, i32* %initial_size, align 4
117     %2 = sext i32 %1 to i64
118     %3 = call noalias i8* bitcast (i8* (i32)* @malloc to i8* (i64*)(i64 %2)) #1
119     store i8* %3, i8** %str, align 8
120     store i32 0, i32* %index, align 4
121     store i8 48, i8* %tmp, align 1
122     br label %4
123
124     ; <label>:4                             ; preds = %20, %0
125     %5 = call i32 @getchar()

```



```

126     %6 = trunc i32 %5 to i8
127     store i8 %6, i8* %tmp, align 1
128     %7 = sext i8 %6 to i32
129     %8 = icmp ne i32 %7, 10
130     br i1 %8, label %9, label %27
131
132     ; <label>:9                                ; preds = %4
133     %10 = load i32, i32* %index, align 4
134     %11 = load i32, i32* %initial_size, align 4
135     %12 = sub nsw i32 %11, 1
136     %13 = icmp sge i32 %10, %12
137     br i1 %13, label %14, label %20
138
139     ; <label>:14                                ; preds = %9
140     %15 = load i8*, i8** %str, align 8
141     %16 = load i32, i32* %initial_size, align 4
142     %17 = mul nsw i32 %16, 2
143     store i32 %17, i32* %initial_size, align 4
144     %18 = sext i32 %17 to i64
145     %19 = call i8* @bitcast (i8* (i8*, i32)* @realloc to i8* (i8*, i64)*) (i8* %15, i64
        ↪ %18) #1
146     store i8* %19, i8** %str, align 8
147     br label %20
148
149     ; <label>:20                                ; preds = %14, %9
150     %21 = load i8, i8* %tmp, align 1
151     %22 = load i32, i32* %index, align 4
152     %23 = add nsw i32 %22, 1
153     store i32 %23, i32* %index, align 4
154     %24 = sext i32 %22 to i64
155     %25 = load i8*, i8** %str, align 8
156     %26 = getelementptr inbounds i8, i8* %25, i64 %24
157     store i8 %21, i8* %26, align 1
158     br label %4
159
160     ; <label>:27                                ; preds = %4
161     %28 = load i32, i32* %index, align 4
162     %29 = sext i32 %28 to i64
163     %30 = load i8*, i8** %str, align 8
164     %31 = getelementptr inbounds i8, i8* %30, i64 %29
165     store i8 0, i8* %31, align 1
166     %32 = load i8*, i8** %str, align 8
167     ret i8* %32
168 }
169
170 ; Function Attrs: nounwind uwtable
171 define void @rec_init(i64* %arr, i32 %curr_offset, i32* %static_offsets, i32* %indexes,
        ↪ i32* %dims, i32 %dimc, i32 %dim_curr) #0 {
172     %1 = alloca i64*, align 8

```

```

173 %2 = alloca i32, align 4
174 %3 = alloca i32*, align 8
175 %4 = alloca i32*, align 8
176 %5 = alloca i32*, align 8
177 %6 = alloca i32, align 4
178 %7 = alloca i32, align 4
179 %static_offset = alloca i32, align 4
180 %dynamic_offset = alloca i32, align 4
181 %i = alloca i32, align 4
182 %tmp = alloca i32, align 4
183 %j = alloca i32, align 4
184 %i1 = alloca i32, align 4
185 %offset = alloca i32, align 4
186 %sub = alloca i64*, align 8
187 store i64* %arr, i64** %1, align 8
188 store i32 %curr_offset, i32* %2, align 4
189 store i32* %static_offsets, i32** %3, align 8
190 store i32* %indexes, i32** %4, align 8
191 store i32* %dims, i32** %5, align 8
192 store i32 %dimc, i32* %6, align 4
193 store i32 %dim_curr, i32* %7, align 4
194 %8 = load i32, i32* %7, align 4
195 %9 = sext i32 %8 to i64
196 %10 = load i32*, i32** %5, align 8
197 %11 = getelementptr inbounds i32, i32* %10, i64 %9
198 %12 = load i32, i32* %11, align 4
199 %13 = sext i32 %12 to i64
200 %14 = load i32, i32* %2, align 4
201 %15 = sext i32 %14 to i64
202 %16 = load i64*, i64** %1, align 8
203 %17 = getelementptr inbounds i64, i64* %16, i64 %15
204 store i64 %13, i64* %17, align 8
205 %18 = load i32, i32* %7, align 4
206 %19 = add nsw i32 %18, 1
207 %20 = load i32, i32* %6, align 4
208 %21 = icmp sge i32 %19, %20
209 br i1 %21, label %22, label %23
210
211 ; <label>:22                                ; preds = %0
212 br label %115
213
214 ; <label>:23                                ; preds = %0
215 %24 = load i32, i32* %7, align 4
216 %25 = sext i32 %24 to i64
217 %26 = load i32*, i32** %3, align 8
218 %27 = getelementptr inbounds i32, i32* %26, i64 %25
219 %28 = load i32, i32* %27, align 4
220 store i32 %28, i32* %static_offset, align 4
221 store i32 0, i32* %dynamic_offset, align 4

```

```

222 store i32 0, i32* %i, align 4
223 br label %29
224
225 ; <label>:29                                ; preds = %60, %23
226 %30 = load i32, i32* %i, align 4
227 %31 = load i32, i32* %7, align 4
228 %32 = icmp slt i32 %30, %31
229 br i1 %32, label %33, label %63
230
231 ; <label>:33                                ; preds = %29
232 %34 = load i32, i32* %i, align 4
233 %35 = sext i32 %34 to i64
234 %36 = load i32*, i32** %4, align 8
235 %37 = getelementptr inbounds i32, i32* %36, i64 %35
236 %38 = load i32, i32* %37, align 4
237 store i32 %38, i32* %tmp, align 4
238 %39 = load i32, i32* %i, align 4
239 %40 = add nsw i32 %39, 1
240 store i32 %40, i32* %j, align 4
241 br label %41
242
243 ; <label>:41                                ; preds = %53, %33
244 %42 = load i32, i32* %j, align 4
245 %43 = load i32, i32* %7, align 4
246 %44 = icmp sle i32 %42, %43
247 br i1 %44, label %45, label %56
248
249 ; <label>:45                                ; preds = %41
250 %46 = load i32, i32* %j, align 4
251 %47 = sext i32 %46 to i64
252 %48 = load i32*, i32** %5, align 8
253 %49 = getelementptr inbounds i32, i32* %48, i64 %47
254 %50 = load i32, i32* %49, align 4
255 %51 = load i32, i32* %tmp, align 4
256 %52 = mul nsw i32 %51, %50
257 store i32 %52, i32* %tmp, align 4
258 br label %53
259
260 ; <label>:53                                ; preds = %45
261 %54 = load i32, i32* %j, align 4
262 %55 = add nsw i32 %54, 1
263 store i32 %55, i32* %j, align 4
264 br label %41
265
266 ; <label>:56                                ; preds = %41
267 %57 = load i32, i32* %tmp, align 4
268 %58 = load i32, i32* %dynamic_offset, align 4
269 %59 = add nsw i32 %58, %57
270 store i32 %59, i32* %dynamic_offset, align 4

```

```
271 br label %60
272
273 ; <label>:60                                ; preds = %56
274 %61 = load i32, i32* %i, align 4
275 %62 = add nsw i32 %61, 1
276 store i32 %62, i32* %i, align 4
277 br label %29
278
279 ; <label>:63                                ; preds = %29
280 store i32 0, i32* %i1, align 4
281 br label %64
282
283 ; <label>:64                                ; preds = %112, %63
284 %65 = load i32, i32* %i1, align 4
285 %66 = load i32, i32* %7, align 4
286 %67 = sext i32 %66 to i64
287 %68 = load i32*, i32** %5, align 8
288 %69 = getelementptr inbounds i32, i32* %68, i64 %67
289 %70 = load i32, i32* %69, align 4
290 %71 = icmp slt i32 %65, %70
291 br i1 %71, label %72, label %115
292
293 ; <label>:72                                ; preds = %64
294 %73 = load i32, i32* %static_offset, align 4
295 %74 = load i32, i32* %dynamic_offset, align 4
296 %75 = load i32, i32* %i1, align 4
297 %76 = add nsw i32 %74, %75
298 %77 = load i32, i32* %7, align 4
299 %78 = add nsw i32 %77, 1
300 %79 = sext i32 %78 to i64
301 %80 = load i32*, i32** %5, align 8
302 %81 = getelementptr inbounds i32, i32* %80, i64 %79
303 %82 = load i32, i32* %81, align 4
304 %83 = add nsw i32 %82, 1
305 %84 = mul nsw i32 %76, %83
306 %85 = add nsw i32 %73, %84
307 store i32 %85, i32* %offset, align 4
308 %86 = load i64*, i64** %1, align 8
309 %87 = load i32, i32* %offset, align 4
310 %88 = sext i32 %87 to i64
311 %89 = getelementptr inbounds i64, i64* %86, i64 %88
312 store i64* %89, i64** %sub, align 8
313 %90 = load i64*, i64** %sub, align 8
314 %91 = ptrtoint i64* %90 to i64
315 %92 = load i32, i32* %2, align 4
316 %93 = add nsw i32 %92, 1
317 %94 = load i32, i32* %i1, align 4
318 %95 = add nsw i32 %93, %94
319 %96 = sext i32 %95 to i64
```

```

320 %97 = load i64*, i64** %1, align 8
321 %98 = getelementptr inbounds i64, i64* %97, i64 %96
322 store i64 %91, i64* %98, align 8
323 %99 = load i32, i32* %i1, align 4
324 %100 = load i32, i32* %7, align 4
325 %101 = sext i32 %100 to i64
326 %102 = load i32*, i32** %4, align 8
327 %103 = getelementptr inbounds i32, i32* %102, i64 %101
328 store i32 %99, i32* %103, align 4
329 %104 = load i64*, i64** %1, align 8
330 %105 = load i32, i32* %offset, align 4
331 %106 = load i32*, i32** %3, align 8
332 %107 = load i32*, i32** %4, align 8
333 %108 = load i32*, i32** %5, align 8
334 %109 = load i32, i32* %6, align 4
335 %110 = load i32, i32* %7, align 4
336 %111 = add nsw i32 %110, 1
337 call void @rec_init(i64* %104, i32 %105, i32* %106, i32* %107, i32* %108, i32 %109, i32
    ↪ %111)
338 br label %112
339
340 ; <label>:112                                ; preds = %72
341 %113 = load i32, i32* %i1, align 4
342 %114 = add nsw i32 %113, 1
343 store i32 %114, i32* %i1, align 4
344 br label %64
345
346 ; <label>:115                                ; preds = %22, %64
347 ret void
348 }
349
350 ; Function Attrs: nounwind uwtable
351 define i64* @init_arr(i32* %dims, i32 %dimc) #0 {
352 %1 = alloca i32*, align 8
353 %2 = alloca i32, align 4
354 %3 = alloca i8*
355 %total = alloca i32, align 4
356 %i = alloca i32, align 4
357 %j = alloca i32, align 4
358 %i1 = alloca i32, align 4
359 %length = alloca i32, align 4
360 %i2 = alloca i32, align 4
361 %tmp = alloca i32, align 4
362 %j3 = alloca i32, align 4
363 %arr = alloca i64*, align 8
364 %i4 = alloca i32, align 4
365 store i32* %dims, i32** %1, align 8
366 store i32 %dimc, i32* %2, align 4
367 %4 = load i32, i32* %2, align 4

```

```

368 %5 = zext i32 %4 to i64
369 %6 = call i8* @llvm.stacksave()
370 store i8* %6, i8** %3
371 %7 = alloca i32, i64 %5, align 16
372 store i32 0, i32* %total, align 4
373 store i32 0, i32* %i, align 4
374 br label %8
375
376 ; <label>:8                                ; preds = %56, %0
377 %9 = load i32, i32* %i, align 4
378 %10 = load i32, i32* %2, align 4
379 %11 = icmp slt i32 %9, %10
380 br i1 %11, label %12, label %59
381
382 ; <label>:12                                ; preds = %8
383 %13 = load i32, i32* %i, align 4
384 %14 = sext i32 %13 to i64
385 %15 = getelementptr inbounds i32, i32* %7, i64 %14
386 store i32 1, i32* %15, align 4
387 store i32 0, i32* %j, align 4
388 br label %16
389
390 ; <label>:16                                ; preds = %31, %12
391 %17 = load i32, i32* %j, align 4
392 %18 = load i32, i32* %i, align 4
393 %19 = icmp slt i32 %17, %18
394 br i1 %19, label %20, label %34
395
396 ; <label>:20                                ; preds = %16
397 %21 = load i32, i32* %j, align 4
398 %22 = sext i32 %21 to i64
399 %23 = load i32*, i32** %1, align 8
400 %24 = getelementptr inbounds i32, i32* %23, i64 %22
401 %25 = load i32, i32* %24, align 4
402 %26 = load i32, i32* %i, align 4
403 %27 = sext i32 %26 to i64
404 %28 = getelementptr inbounds i32, i32* %7, i64 %27
405 %29 = load i32, i32* %28, align 4
406 %30 = mul nsw i32 %29, %25
407 store i32 %30, i32* %28, align 4
408 br label %31
409
410 ; <label>:31                                ; preds = %20
411 %32 = load i32, i32* %j, align 4
412 %33 = add nsw i32 %32, 1
413 store i32 %33, i32* %j, align 4
414 br label %16
415
416 ; <label>:34                                ; preds = %16

```

```

417 %35 = load i32, i32* %i, align 4
418 %36 = sext i32 %35 to i64
419 %37 = load i32*, i32** %1, align 8
420 %38 = getelementptr inbounds i32, i32* %37, i64 %36
421 %39 = load i32, i32* %38, align 4
422 %40 = add nsw i32 %39, 1
423 %41 = load i32, i32* %i, align 4
424 %42 = sext i32 %41 to i64
425 %43 = getelementptr inbounds i32, i32* %7, i64 %42
426 %44 = load i32, i32* %43, align 4
427 %45 = mul nsw i32 %44, %40
428 store i32 %45, i32* %43, align 4
429 %46 = load i32, i32* %total, align 4
430 %47 = load i32, i32* %i, align 4
431 %48 = sext i32 %47 to i64
432 %49 = getelementptr inbounds i32, i32* %7, i64 %48
433 %50 = load i32, i32* %49, align 4
434 %51 = add nsw i32 %50, %46
435 store i32 %51, i32* %49, align 4
436 %52 = load i32, i32* %i, align 4
437 %53 = sext i32 %52 to i64
438 %54 = getelementptr inbounds i32, i32* %7, i64 %53
439 %55 = load i32, i32* %54, align 4
440 store i32 %55, i32* %total, align 4
441 br label %56
442
443 ; <label>:56                                ; preds = %34
444 %57 = load i32, i32* %i, align 4
445 %58 = add nsw i32 %57, 1
446 store i32 %58, i32* %i, align 4
447 br label %8
448
449 ; <label>:59                                ; preds = %8
450 %60 = load i32, i32* %2, align 4
451 %61 = zext i32 %60 to i64
452 %62 = alloca i32, i64 %61, align 16
453 store i32 0, i32* %i1, align 4
454 br label %63
455
456 ; <label>:63                                ; preds = %71, %59
457 %64 = load i32, i32* %i1, align 4
458 %65 = load i32, i32* %2, align 4
459 %66 = icmp slt i32 %64, %65
460 br i1 %66, label %67, label %74
461
462 ; <label>:67                                ; preds = %63
463 %68 = load i32, i32* %i1, align 4
464 %69 = sext i32 %68 to i64
465 %70 = getelementptr inbounds i32, i32* %62, i64 %69

```

```

466 store i32 0, i32* %70, align 4
467 br label %71
468
469 ; <label>:71                                ; preds = %67
470 %72 = load i32, i32* %i1, align 4
471 %73 = add nsw i32 %72, 1
472 store i32 %73, i32* %i1, align 4
473 br label %63
474
475 ; <label>:74                                ; preds = %63
476 store i32 0, i32* %length, align 4
477 store i32 0, i32* %i2, align 4
478 br label %75
479
480 ; <label>:75                                ; preds = %108, %74
481 %76 = load i32, i32* %i2, align 4
482 %77 = load i32, i32* %2, align 4
483 %78 = icmp slt i32 %76, %77
484 br i1 %78, label %79, label %111
485
486 ; <label>:79                                ; preds = %75
487 store i32 1, i32* %tmp, align 4
488 %80 = load i32, i32* %i2, align 4
489 %81 = sub nsw i32 %80, 1
490 store i32 %81, i32* %j3, align 4
491 br label %82
492
493 ; <label>:82                                ; preds = %93, %79
494 %83 = load i32, i32* %j3, align 4
495 %84 = icmp sge i32 %83, 0
496 br i1 %84, label %85, label %96
497
498 ; <label>:85                                ; preds = %82
499 %86 = load i32, i32* %j3, align 4
500 %87 = sext i32 %86 to i64
501 %88 = load i32*, i32** %1, align 8
502 %89 = getelementptr inbounds i32, i32* %88, i64 %87
503 %90 = load i32, i32* %89, align 4
504 %91 = load i32, i32* %tmp, align 4
505 %92 = mul nsw i32 %91, %90
506 store i32 %92, i32* %tmp, align 4
507 br label %93
508
509 ; <label>:93                                ; preds = %85
510 %94 = load i32, i32* %j3, align 4
511 %95 = add nsw i32 %94, -1
512 store i32 %95, i32* %j3, align 4
513 br label %82
514

```



```

515 ; <label>:96                                ; preds = %82
516 %97 = load i32, i32* %i2, align 4
517 %98 = sext i32 %97 to i64
518 %99 = load i32*, i32** %1, align 8
519 %100 = getelementptr inbounds i32, i32* %99, i64 %98
520 %101 = load i32, i32* %100, align 4
521 %102 = add nsw i32 %101, 1
522 %103 = load i32, i32* %tmp, align 4
523 %104 = mul nsw i32 %103, %102
524 store i32 %104, i32* %tmp, align 4
525 %105 = load i32, i32* %tmp, align 4
526 %106 = load i32, i32* %length, align 4
527 %107 = add nsw i32 %106, %105
528 store i32 %107, i32* %length, align 4
529 br label %108
530
531 ; <label>:108                                ; preds = %96
532 %109 = load i32, i32* %i2, align 4
533 %110 = add nsw i32 %109, 1
534 store i32 %110, i32* %i2, align 4
535 br label %75
536
537 ; <label>:111                                ; preds = %75
538 %112 = load i32, i32* %length, align 4
539 %113 = sext i32 %112 to i64
540 %114 = call noalias i8* @bitcast (i8* (i32)* @malloc to i8* (i64*)) (i64 %113) #1
541 %115 = bitcast i8* %114 to i64*
542 store i64* %115, i64** %arr, align 8
543 store i32 0, i32* %i4, align 4
544 br label %116
545
546 ; <label>:116                                ; preds = %125, %111
547 %117 = load i32, i32* %i4, align 4
548 %118 = load i32, i32* %length, align 4
549 %119 = icmp slt i32 %117, %118
550 br i1 %119, label %120, label %128
551
552 ; <label>:120                                ; preds = %116
553 %121 = load i32, i32* %i4, align 4
554 %122 = sext i32 %121 to i64
555 %123 = load i64*, i64** %arr, align 8
556 %124 = getelementptr inbounds i64, i64* %123, i64 %122
557 store i64 0, i64* %124, align 8
558 br label %125
559
560 ; <label>:125                                ; preds = %120
561 %126 = load i32, i32* %i4, align 4
562 %127 = add nsw i32 %126, 1
563 store i32 %127, i32* %i4, align 4

```

```
564 br label %116
565
566 ; <label>:128                                ; preds = %116
567 %129 = load i64*, i64** %arr, align 8
568 %130 = load i32*, i32** %1, align 8
569 %131 = load i32, i32* %2, align 4
570 call void @rec_init(i64* %129, i32 0, i32* %7, i32* %62, i32* %130, i32 %131, i32 0)
571 %132 = load i64*, i64** %arr, align 8
572 %133 = load i8*, i8** %3
573 call void @llvm.stackrestore(i8* %133)
574 ret i64* %132
575 }
576
577 ; Function Attrs: nounwind
578 declare i8* @llvm.stacksave() #1
579
580 ; Function Attrs: nounwind
581 declare void @llvm.stackrestore(i8*) #1
582
583 attributes #0 = { nounwind uwtable "disable-tail-calls"="false"
  ↳ "less-precise-fpmad"="false" "no-frame-pointer-elim"="true"
  ↳ "no-frame-pointer-elim-non-leaf" "no-infs-fp-math"="false" "no-nans-fp-math"="false"
  ↳ "stack-protector-buffer-size"="8" "target-cpu"="x86-64"
  ↳ "target-features"="+sse,+sse2" "unsafe-fp-math"="false" "use-soft-float"="false" }
584 attributes #1 = { nounwind }
585
586 !llvm.ident = !{!0}
587
588 !0 = !{"Ubuntu clang version 3.7.0-2ubuntu1 (tags/RELEASE_370/final) (based on LLVM
  ↳ 3.7.0)"}

```

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.

Phillip

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

`_tags`

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

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(      "cast",
73 [SCall("malloc",
74 [
75 SCall("sizeof", [SId("ignore", ret_type)], Datatype(Int_t), 0)
76 ],
77 Arraytype(Char_t, 1), 0)
78 ],
79 ret_type,
80 0
81 )
82 );
83 SExpr(
84 SAssign(
85 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 m) (i @ t)
158 in
159 iterate_includes classes (StringMap.add (Filepath.realpath filename) 1 StringMap.empty)
  ↪ includes
160
161 let get_name cname fdecl =
162 (* We use '.' to separate types so llvm will recognize the function name and it won't
  ↪ conflict *)
163 (* let params = List.fold_left (fun s -> (function Formal(t, _) -> s ^ "." ^
  ↪ Utils.string_of_datatype t | _ -> "" )) "" fdecl.formals in *)
164 let name = Utils.string_of_fname fdecl.fname in
165 if name = "main"
166 then "main"
167 else cname ^ "." ^ name(* ^ params *)
168
169 let get_constructor_name cname fdecl =
170 let params = List.fold_left (fun s -> (function Formal(t, _) -> s ^ "." ^
  ↪ Utils.string_of_datatype t | _ -> "" )) "" fdecl.formals in
171 let name = Utils.string_of_fname fdecl.fname in
172 cname ^ "." ^ name ^ params
173
174 let get_name_without_class fdecl =
175 (* We use '.' to separate types so llvm will recognize the function name and it won't
  ↪ conflict *)
176 let params = List.fold_left (fun s -> (function Formal(t, _) -> s ^ "." ^
  ↪ 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 (Utils.string_of_fname
  ↪ f.sfname) f m) StringMap.empty reserved in
184 let helper m (cdecl:Ast.class_decl) =
185 let fieldfun = (fun m -> (function Field(s, d, n) -> if (StringMap.mem (n) m) then
  ↪ raise(Exceptions.DuplicateField) else (StringMap.add n (Field(s, d, n)) m))) in
186 let funcname = get_name cdecl.cname in

```

```

187 let funcfun m fdecl =
188   if (StringMap.mem (funcname fdecl) m)
189   then raise(Exceptions.DuplicateFunction(funcname fdecl))
190   else if (StringMap.mem (Utils.string_of_fname fdecl.fname) reserved_map)
191   then raise(Exceptions.CannotUseReservedFuncName(Utils.string_of_fname fdecl.fname))
192   else (StringMap.add (funcname fdecl) fdecl m)
193   in
194   let constructor_name = get_constructor_name cdecl.cname in
195   let constructorfun m fdecl =
196     if fdecl.formals = [] then m
197     else if StringMap.mem (constructor_name fdecl) m
198     then raise(Exceptions.DuplicateConstructor)
199     else (StringMap.add (constructor_name fdecl) fdecl m)
200     in
201     let default_c = default_c cdecl.cname in
202     let constructor_map = StringMap.add (get_constructor_name cdecl.cname default_c)
203       ↪ default_c StringMap.empty in
204     (if (StringMap.mem cdecl.cname m) then raise (Exceptions.DuplicateClassName(cdecl.cname))
205      ↪ else
206       StringMap.add cdecl.cname
207       {
208         field_map = List.fold_left fieldfun StringMap.empty cdecl.cbody.fields;
209         func_map = List.fold_left funcfun StringMap.empty cdecl.cbody.methods;
210         constructor_map = List.fold_left constructorfun constructor_map
211           ↪ cdecl.cbody.constructors;
212         reserved_map = reserved_map;
213         cdecl = cdecl }
214       m) in
215     List.fold_left helper StringMap.empty cdecls
216   in
217   let rec get_all_descendants cname accum =
218     if Hashtbl.mem predecessors cname then
219       let direct_descendants = Hashtbl.find predecessors cname in
220       let add_childs_descendants desc_set direct_descendant =
221         get_all_descendants direct_descendant (StringSet.add direct_descendant desc_set)
222       in
223       List.fold_left add_childs_descendants accum direct_descendants
224     else accum
225   in
226   let inherited potential_predec potential_child =
227     match potential_predec, potential_child with
228     | Datatype(Objecttype(predec_cname)), Datatype(Objecttype(child_cname)) ->
229       let descendants = get_all_descendants predec_cname StringSet.empty in
230       if (predec_cname = child_cname) || (StringSet.mem child_cname descendants) then true
231       else raise (Exceptions.LocalAssignTypeMismatch(predec_cname, child_cname))
232     | _ , _ -> false
233   in
234   let get_equality_binop_type type1 type2 se1 se2 op =
235     (* Equality op not supported for float operands. The correct way to test floats
236     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 in
291 let se, _ = expr_to_sexpr env (List.hd el) in
292 let el = List.tl el in
293 let se_t = get_type_from_sexpr se in
294 let sel, t = iter se_t ([se]) el in
295 let se_t = match t with
296 Datatype(x) -> Arraytype(x, 1)
297 | Arraytype(x, n) -> Arraytype(x, n+1)
298 | _ as t -> raise (Exceptions.InvalidArrayPrimitiveType(Utils.string_of_datatype
  ↪ t))
299 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 | _ as t ->
315 let error_msg = Utils.string_of_datatype t in
316 raise (Exceptions.ArrayInitTypeInvalid(error_msg))
317 in
318 let sexpr_type = convert_d_to_arraytype d in
319 let sel = List.map check_elem_type el in
320 SArrayCreate(d, sel, sexpr_type)

```

```

321
322 and check_array_access env e el =
323   (* Get dimensions of array, ex: foo[10][4][2] is dimen=3 *)
324   let array_dimensions = List.length el in
325   (* Check every e in el is of type Datatype(Int_t). Ensure all indices are ints *)
326   let check_elem_type arg =
327     let sexpr, _ = expr_to_sexpr env arg in
328     let sexpr_type = get_type_from_sexpr sexpr in
329     if sexpr_type = Datatype(Int_t)
330     then sexpr
331     else raise (Exceptions.MustPassIntegerTypeToArrayAccess)
332   in
333   (* converting e to se also checks if the array id has been declared *)
334   let se, _ = expr_to_sexpr env e in
335   let se_type = get_type_from_sexpr se in
336
337   (* Check that e has enough dimens as e's in el. Return overall datatype of access*)
338   let check_array_dim_vs_params num_params = function
339     Arraytype(t, n) ->
340     if num_params < n then
341       Arraytype(t, (n-num_params))
342     else if num_params = n then
343       Datatype(t)
344     else
345       raise (Exceptions.ArrayAccessInvalidParamLength(string_of_int num_params, string_of_int
346         ↪ n))
347   | _ as t ->
348   let error_msg = Utils.string_of_datatype t in
349   raise (Exceptions.ArrayAccessExpressionNotArray(error_msg))
350   in
351   let sexpr_type = check_array_dim_vs_params array_dimensions se_type in
352   let sel = List.map check_elem_type el in
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
361       ↪ (Utils.string_of_expr e))
362   in
363   let ptype_name parent_type = match parent_type with
364     Datatype(Objecttype(name)) -> name
365     | _ as d -> raise
366       ↪ (Exceptions.ObjAccessMustHaveObjectType (Utils.string_of_datatype d))
367   in
368   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(n, env.env_name,
375 ↪ tlenv.env_name))
376 else d
377 in
378 try match_field (StringMap.find id cmap.field_map)
379 with | Not_found -> raise (Exceptions.UnknownIdentifierForClass(id, cname))
380 in
381 function
382 (* Check fields in parent *)
383 Id s                                -> SId(s, (get_id_type_from_object env s pt_name
384 ↪ top_level_env)), env
385 (* Check functions in parent *)
386 | Call(fname, e1)                    ->
387 let env = update_env_name env pt_name in
388 check_call_type top_level_env true env fname e1, env
389 (* Set parent, check if base is field *)
390 | ObjAccess(e1, e2)                  ->
391 let old_env = env in
392 let lhs, env = check_rhs env parent_type top_level_env e1 in
393 let lhs_type = get_type_from_sexpr lhs in
394
395 let pt_name = ptype_name lhs_type in
396 let lhs_env = update_env_name env pt_name in
397
398 let rhs, env = check_rhs lhs_env lhs_type top_level_env e2 in
399 let rhs_type = get_type_from_sexpr rhs in
400 SObjAccess(lhs, rhs, rhs_type), old_env
401 | _ as e                                -> raise (Exceptions.InvalidAccessLHS
402 ↪ (Utils.string_of_expr e))
403 in
404 let arr_lhs, _ = expr_to_sexpr env lhs in
405 let arr_lhs_type = get_type_from_sexpr arr_lhs in
406 match arr_lhs_type with
407 Arraytype(Char_t, 1) -> raise (Exceptions.CannotAccessLengthOfCharArray)
408 | Arraytype(_, _) ->
409 let rhs = match rhs with
410 Id("length") -> SId("length", Datatype(Int_t))
411 | _ -> raise (Exceptions.CanOnlyAccessLengthOfArray)
412 in
413 SObjAccess(arr_lhs, rhs, Datatype(Int_t))
414 | _ ->
415 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 el =
423 let sel, env = expr1_to_sexpr1 env el 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 -> raise (Exceptions.CannotPassNonInheritedClassesInPlaceOfOthers(f, p))
439 in
440 let rt = Datatype(Objecttype(f)) in
441 SCall("cast", [param; SId("ignore", rt)], rt, 0)
442 with | Not_found -> raise (Exceptions.ClassIsNotExtendedBy(f, p))
443 else param
444 | _ -> if fty = pty then param else
445   ↪ raise (Exceptions.IncorrectTypePassedToFunction(fname, Utils.string_of_datatype pty))
446 in
447
448 let index fdecl fname =
449 let cdecl = cmap.cdecl in
450 (* Have to update this with all parent classes checks *)
451 let _ =
452 if fdecl.scope = Ast.Private && top_level_env.env_name <> env.env_name then
453 raise (Exceptions.CannotAccessPrivateFunctionInNonProperScope(get_name env.env_name fdecl,
454   ↪ env.env_name, top_level_env.env_name))
455 in
456 (* Not exactly sure why there needs to be a list.rev *)
457 let fns = List.rev cdecl.cbody.methods in
458 let rec find x lst =
459 match lst with
460 | [] -> raise (Failure ("Could not find " ^ fname))
461 | fdecl :: t ->

```

```

460 let search_name = (get_name env.env_name fdecl) in
461 if x = search_name then 0
462 else if search_name = "main" then find x t
463 else 1 + find x t
464 in
465 find fname fns
466 in
467
468 let handle_params (formals) params =
469 match formals, params with
470 [Many(Any)], _ -> params
471 | [], [] -> []
472 | [], _
473 | _, [] -> raise (Exceptions.IncorrectTypePassedToFunction(fname,
  ↳ Utils.string_of_datatype (Datatype(Void_t))))
474 | _ ->
475 let len1 = List.length formals in
476 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 el =
494 let sel, env = expr1_to_sexpr1 env el 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, se1, 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(Utls.string_of_datatype type1,
529 ↪ Utls.string_of_datatype type2))
529 | _ ->
530 if type1 = type2
531 then SAssign(se1, se2, type1)
532 else raise (Exceptions.AssignmentTypeMismatch(Utls.string_of_datatype type1,
533 ↪ Utls.string_of_datatype type2))
534
535 and check_unop env op e =
536 let check_num_unop t = function
537 Sub      -> t
538 |      _      -> raise(Exceptions.InvalidUnaryOperation)
539 in
540 let check_bool_unop = function
541 Not      -> Datatype(Bool_t)
542 |      _      -> raise(Exceptions.InvalidUnaryOperation)
543 in
544 let se, env = expr_to_sexpr env e in
545 let t = get_type_from_sexpr se in
546 match t with
547 Datatype(Int_t)
548 |      Datatype(Float_t)      -> SUnop(op, se, check_num_unop t op)
549 |      Datatype(Bool_t)      -> SUnop(op, se, check_bool_unop op)
550 |      _ -> raise(Exceptions.InvalidUnaryOperation)
551
552 and check_binop env e1 op e2 =
553 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 ((Utils.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, el) -> check_object_constructor env s el, env
583 | Call(s, el) -> check_call_type env false env s el, env
584
585 | ArrayCreate(d, el) -> check_array_init env d el, env
586 | ArrayAccess(e, el) -> check_array_access env e el, env
587 | ArrayPrimitive el -> check_array_primitive env el, 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 el =
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 el), !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 sl env = match sl with
660 [] -> SBlock([SExpr(SNoexpr, Datatype(Void_t))]), env
661 | _ ->
662 let sl, _ = convert_stmt_list_to_sstmt_list env sl in
663 SBlock(sl), 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

```

```

697 let forbody, _ = parse_stmt env s in
698 let conditional = get_type_from_sexpr se2 in
699 let sfor =
700   if (conditional = Datatype(Bool_t) || conditional = Datatype(Void_t))
701   then SFor(se1, se2, se3, forbody)
702   else raise Exceptions.InvalidForStatementType
703   in
704
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", ret_type)], Datatype(Int_t), 0)
770 ],
771 Arraytype(Char_t, 1), 0)
772 ],
773 ret_type, 0
774 )
775 );
776 SExpr(
777 SAssign(
778 SObjAccess(
779 SId("this", ret_type),
780 SId(".key", Datatype(Int_t)),
781 Datatype(Int_t)
782 ),
783 SInt_Lit(key),
784 Datatype(Int_t)
785 ),
786 Datatype(Int_t)
787 )
788 in
789 init_this @ fbody
790
791 let convert_constructor_to_sfdecl class_maps reserved class_map cname constructor =

```

```

792 let env = {
793     env_class_maps      = class_maps;
794     env_name            = cname;
795     env_cmap            = class_map;
796     env_locals          = StringMap.empty;
797     env_parameters      = List.fold_left (fun m f -> match f with Formal(d, s) ->
798         ↪ (StringMap.add s f m) | _ -> m) StringMap.empty constructor.formals;
799     env_returnType      = Datatype(Objecttype(cname));
800     env_in_for          = false;
801     env_in_while        = false;
802     env_reserved        = reserved;
803 } in
804 let fbody = fst (convert_stmt_list_to_sstmt_list env constructor.body) in
805 {
806     sfname              = Ast.FName (get_constructor_name cname constructor);
807     sreturnType         = Datatype(Objecttype(cname));
808     sformals            = constructor.formals;
809     sbody               = append_code_to_constructor fbody cname
810         ↪ (Datatype(Objecttype(cname)));
811     func_type          = Sast.User;
812     overrides          = false;
813     source              = "NA";
814 }
815
816 let check_fbody fname fbody returnType =
817 let len = List.length fbody in
818 if len = 0 then () else
819 let final_stmt = List.hd (List.rev fbody) in
820 match returnType, final_stmt with
821 Datatype(Void_t), _ -> ()
822 | _, SReturn(_, _) -> ()
823 | _ -> raise(Exceptions.AllNonVoidFunctionsMustEndWithReturn(fname))
824
825 let convert_fdecl_to_sfdecl class_maps reserved class_map cname fdecl =
826 let root_cname = match fdecl.root_cname with
827 Some(x) -> x
828 | None -> cname
829 in
830 let class_formal =
831 if fdecl.overrides then
832 Ast.Formal(Datatype(Objecttype(root_cname)), "this")
833 else
834 Ast.Formal(Datatype(Objecttype(cname)), "this")
835 in
836 let env_param_helper m fname = match fname with
837 Formal(d, s) -> (StringMap.add s fname m)
838 | _ -> m
839 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
926         ↪ child_cdecl.cbody.methods
927 }
928 in
929 {
930     cname = child_cdecl.cname;
931     extends = child_cdecl.extends;
932     cbody = child_cbody
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
1052     ↪ "nbyte"]);
1052     reserved_stub "write"     (i32_t)        ([mf i32_t "fd"; mf str_t "buf"; mf i32_t
1053     ↪ "nbyte"]);
1053     reserved_stub "lseek"     (i32_t)        ([mf i32_t "fd"; mf i32_t "offset"; mf
1054     ↪ 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 cmap_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 in
1121 let merged = merge_maps predec_field_map desc_field_map in
1122 let updated = update_class_maps "field_map" merged descendant accum in

```

```

1123 if (StringMap.mem descendant predecessors) then
1124 let descendants_of_descendant = StringMap.find descendant predecessors in
1125 add_inherited_fields descendant descendants_of_descendant updated
1126 else updated
1127 in
1128 List.fold_left cmap_inherit cmap_to_update desc
1129 (* end of add_inherited_fields *)
1130 in
1131 let result = StringSet.fold (fun x a -> add_inherited_fields x (StringMap.find x
1132   ↪ predecessors) a) roots cmap_inherit
1133 (*in let _ = print_map result*)
1134 in result
1135
1136 (* TODO Check that this actually works *)
1137 let check_cyclical_inheritance cdecls predecessors =
1138 let handle_predecessor cdecl parent predecessor =
1139 if cdecl.cname = predecessor then
1140 raise (Exceptions.CyclicalDependencyBetween(cdecl.cname, parent))
1141 in
1142 let handle_cdecl cdecl =
1143 if StringMap.mem cdecl.cname predecessors
1144 then
1145 let pred_list = StringMap.find cdecl.cname predecessors in
1146 List.iter (handle_predecessor cdecl (List.hd pred_list)) pred_list
1147 else ()
1148 in
1149 List.iter handle_cdecl cdecls
1150
1151 let build_func_map_inherited_lookup cdecls_inherited =
1152 let build_func_map cdecl =
1153 let add_func m fdecl = StringMap.add (get_name cdecl.cname fdecl) fdecl m in
1154 List.fold_left add_func StringMap.empty cdecl.cbody.methods
1155 in
1156 let add_class_func_map m cdecl = StringMap.add cdecl.cname (build_func_map cdecl) m in
1157 List.fold_left add_class_func_map StringMap.empty cdecls_inherited
1158
1159 let add_inherited_methods cmap cdecls func_maps_inherited =
1160 let find_cdecl cname =
1161 try List.find (fun cdecl -> cdecl.cname = cname) cdecls
1162 with | Not_found -> raise Not_found
1163 in
1164 let update_with_inherited_methods cname cmap =
1165 let fmap = StringMap.find cname func_maps_inherited in
1166 let cdecl = find_cdecl cname in
1167 {
1168     field_map = cmap.field_map;
1169     func_map = fmap;
1170     constructor_map = cmap.constructor_map;
1171     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
1178     let handle_inheritance cdecls class_maps =
1179     let predecessors = build_inheritance_forest cdecls class_maps in
1180     let cdecls_inherited = inherit_fields_cdecls cdecls predecessors in
1181     let func_maps_inherited = build_func_map_inherited_lookup cdecls_inherited in
1182     ignore(check_cyclical_inheritance cdecls predecessors);
1183     let cmaps_with_inherited_fields = inherit_fields class_maps predecessors in
1184     let cmaps_inherited = add_inherited_methods cmaps_with_inherited_fields cdecls_inherited
1185         ↪ func_maps_inherited in
1186     cmaps_inherited, cdecls_inherited
1187
1188     let generate_struct_indexes cdecls =
1189     let cdecl_handler index cdecl =
1190     Hashtbl.add struct_indexes cdecl.cname index
1191     in
1192     List.iteri cdecl_handler cdecls
1193
1194     (* Main method for analyzer *)
1195     let analyze filename program = match program with
1196     Program(includes, classes) ->
1197     (* Include code from external files *)
1198     let cdecls = process_includes filename includes classes in
1199     ignore(generate_struct_indexes cdecls);
1200
1201     (* Add built-in functions *)
1202     let reserved = add_reserved_functions in
1203     (* Generate the class_maps for look up in checking functions *)
1204     let class_maps = build_class_maps reserved cdecls in
1205     let class_maps, cdecls = handle_inheritance cdecls class_maps in
1206     let sast = convert_cdecls_to_sast class_maps reserved cdecls in
1207     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 |
13   Boolean_Lit of bool
14 |
15   Float_Lit of float
16 |
17   String_Lit of string
18 |
19   Char_Lit of char
20 |
21   This
22 |
23   Id of string
24 |
25   Binop of expr * op * expr
26 |
27   Assign of expr * expr
28 |
29   Noexpr
30 |
31   ArrayCreate of datatype * expr list
32 |
33   ArrayAccess of expr * expr list
34 |
35   ObjAccess of expr * expr
36 |
37   Call of string * expr list
38 |
39   ObjectCreate of string * expr list
40 |
41   ArrayPrimitive of expr list
42 |
43   Unop of op * expr
44 |
45   Null
46 |
47   Delete of expr
48
49 type stmt =
50 Block of stmt list
51 |
52   Expr of expr
53 |
54   Return of expr
55 |
56   If of expr * stmt * stmt
57 |
58   For of expr * expr * expr * stmt
59 |
60   While of expr * stmt
61 |
62   Break
63 |
64   Continue
65 |
66   Local of datatype * string * expr
67
68 type field = Field of scope * datatype * string
69 type include_stmt = Include of string
70
71 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]
49             ↪ + 1));
50
51         long* sub = arr + offset;
52         arr[curr_offset + 1 + i] = (long) sub;
53
54         indexes[dim_curr] = i;
55         rec_init(arr, offset, static_offsets, indexes, dims, dimc, dim_curr + 1);
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 +
123     //                 ↪ i);
124     //             //             }
125     //             printf("\n");
126     // }
```

```
* =====
```

```
* Code Generation
```

```
*=====*)
```

```
open Llvmlvm  
open Ast  
open Sast  
open Analyzer  
open Exceptions  
open Batteries  
open Hashtbl  
open Conf
```

```
open Llvm.MemoryBuffer  
open Llvm_bitreader
```

```
let context = global_context ()  
let the_module = create_module context "Dice Codegen"  
let builder = builder context  
let named_values:(string, llvalue) Hashtbl.t = Hashtbl.create 50  
let named_params:(string, llvalue) Hashtbl.t = Hashtbl.create 50  
let struct_types:(string, lltype) Hashtbl.t = Hashtbl.create 10  
let struct_field_indexes:(string, int) Hashtbl.t = Hashtbl.create 50
```

```
let i32_t = i32_type context;;  
let i8_t = i8_type context;;  
let f_t = double_type context;;  
let i1_t = i1_type context;;  
let str_t = pointer_type i8_t;;  
let i64_t = i64_type context;;  
let void_t = void_type context;;
```

```
let str_type = Arraytype(Char_t, 1)
```

```
let (br_block) = ref (block_of_value (const_int i32_t 0))  
let (cont_block) = ref (block_of_value (const_int i32_t 0))  
let is_loop = ref false
```

```
let debug = fun s ->  
print_endline ("^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^s");  
dump_module the_module;  
print_endline ("^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^s");  
( )
```

```
let rec get_ptr_type datatype = match datatype with  
Arraytype(t, 0) -> get_type (Datatype(t))  
|          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),
72   ↪ Datatype(Int_t)
73 |     (Datatype(Int_t), Datatype(Char_t))                ->
74   ↪ (build_uitofp lhs i8_t "tmp" llbuilder, rhs), Datatype(Char_t)
75 (* /           (Datatype(Int_t), Datatype(Bool_t))                ->
76   ↪ (lhs, const_zext rhs i32_t) *)
77 |     (Datatype(Int_t), Datatype(Float_t))                -> (build_sitofp lhs f_t
78   ↪ "tmp" llbuilder, rhs), Datatype(Float_t)
79
80 (* char to, __ ) ( using uitofp since char isn't signed *)
81 |     (Datatype(Char_t), Datatype(Int_t))                -> (lhs, build_uitofp rhs
82   ↪ i8_t "tmp" llbuilder), Datatype(Char_t)
83 |     (Datatype(Char_t), Datatype(Char_t))                -> (lhs, rhs),
84   ↪ Datatype(Char_t)
85 (* /           (Datatype(Char_t), Datatype(Bool_t))                -> (lhs,
86   ↪ const_zext rhs i8_t) *)
87 (* /           (Datatype(Char_t), Datatype(Float_t))                ->
88   ↪ (const_uitofp lhs f_t, rhs) *)
89
90 (* bool to, __ ) ( zext fills the empty bits with zeros, zero extension *)
91 (* /           (Datatype(Bool_t), Datatype(Int_t))                -> (const_zext
92   ↪ lhs i32_t, rhs) *)
93 (* /           (Datatype(Bool_t), Datatype(Char_t))                -> (const_zext
94   ↪ lhs i8_t, rhs) *)
95 |     (Datatype(Bool_t), Datatype(Bool_t))                -> (lhs, rhs),
96   ↪ 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_multmp" 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_sgtmp" 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 "multmp" 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(Utils.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" llbuilder
179   | (Sub, Datatype(Float_t))      -> build_fneg e "flt_unoptmp" llbuilder
180   | (Not, Datatype(Bool_t))       -> build_not e "bool_unoptmp" llbuilder
181   | _                             -> raise Exceptions.UnopNotSupported in
182
183   let unop_type_handler d = match d with
184   Datatype(Float_t)
185   | Datatype(Int_t)
186   | Datatype(Bool_t)      -> unops op eType e
187   | _ -> raise Exceptions.InvalidUnopEvaluationType
188   in
189
190   unop_type_handler d
191
192   and func_lookup fname =
193   match (lookup_function fname the_module) with
194   None          -> raise (Exceptions.LLVMFunctionNotFound fname)
195   | Some f      -> f
196
197   and codegen_print el llbuilder =
198   let printf = func_lookup "printf" in
199   let tmp_count = ref 0 in
200   let incr_tmp = fun x -> incr tmp_count in
201
202   let map_expr_to_printfexpr expr =
203   let exprType = Analyzer.get_type_from_sexpr expr in
204   match exprType with
205   Datatype(Bool_t) ->
206   incr_tmp ();
207   let tmp_var = "tmp" ^ (string_of_int !tmp_count) in
208   let trueStr = SString_Lit("true") in
209   let falseStr = SString_Lit("false") in
210   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(id, trueStr, str_type), str_type),
214   SExpr(SAssign(id, 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(Utills.string_of_datatype
    ↪ currType, Utills.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 el 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
352 ↪ (Utils.string_of_sexpr se))
353 in
354 (* Needs to be changed *)
355 let rec check_rhs isLHS parent_expr parent_type =
356 let parent_str = Utils.string_of_object parent_type in
357 function
358 (* Check fields in parent *)
359 SId(field, d) ->
360 let search_term = (parent_str ^ "." ^ field) in
361 let field_index = Hashtbl.find struct_field_indexes search_term in
362 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, e1, d) ->
375
376 let ce = check_rhs false parent_expr parent_type e in
377 let index = codegen_sexpr llbuilder (List.hd e1) 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" llbuilder in
391 let c_index = build_load c_index "cindex" llbuilder in
392 let lookup = func_lookup "lookup" in
393 let fptr = build_call lookup [| c_index; index |] "fptr" llbuilder in
394 let fptr2 = func_lookup fname in
395 let f_ty = type_of fptr2 in
396 let param1 = param fptr2 0 in
397 let param_ty = type_of param1 in
398 let fptr = build_pointercast fptr f_ty fname llbuilder in
399 let ret = codegen_func_call param_ty fptr parent_expr e1 d llbuilder in
400 let ret = ret
401 (* if not isLHS && not isAssign then
402 build_load ret "tmp" llbuilder
403 else
404 ret *)
405 in
406 ret
407 (* Set parent, check if base is field *)
408 |          SObjAccess(e1, e2, d)          ->
409 let e1_type = Analyzer.get_type_from_sexpr e1 in
410 let e1 = check_rhs true parent_expr parent_type e1 in

```

```

411 let e2 = check_rhs true e1 e1_type e2 in
412 e2
413 | _ as e -> raise (Exceptions.InvalidAccessLHS (Utils.string_of_sexpr e))
414 in
415 let lhs_type = Analyzer.get_type_from_sexpr lhs in
416 match lhs_type with
417 | Arraytype(_, _) ->
418   let lhs = codegen_sexpr llbuilder lhs in
419   let _ = match rhs with
420   | SId("length", _) -> "length"
421   | _ -> raise (Exceptions.CanOnlyAccessLengthOfArray)
422   in
423   let _val = build_gep lhs [| (const_int i32_t 0) |] "tmp" llbuilder in
424   build_load _val "tmp" llbuilder
425 | _ ->
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 e1 d llbuilder =
431   let f = func_lookup fname in
432   let params = List.map (codegen_sexpr llbuilder) e1 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 e1 d llbuilder =
442   let index = codegen_sexpr llbuilder (List.hd e1) 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 bbbbody = 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), bbbbody) counter;
468 let cmp = build_icmp Icmp.Slt counter arr_len "tmp" llbuilder in
469 ignore (build_cond_br cmp bbbbody bbdone llbuilder);
470 position_at_end bbbbody 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 in
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" llbuilder in
504
505 (* Store length at this position *)
506 ignore(build_store size_real arr_len_ptr llbuilder);
507 initialise_array arr_len_ptr size_real (const_int i32_t 0) 0 llbuilder;
508 arr

```

```

509
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 llbuilder));
  ↪ (* Store length at this position *)
519 (* initialise_array arr size_real (const_int i32_t 0) 1 llbuilder; *)
520
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)) [] "tmp" llbuilder in
524 ignore(build_store llval arr_ptr llbuilder); ) llvalues;
525 arr
526
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 const_int
  ↪ i1_t 0
537 | SFloat_Lit(f) -> const_float f_t f
538 | SString_Lit(s) -> codegen_string_lit s llbuilder
539 | SChar_Lit(c) -> const_int i8_t (Char.code c)
540 | SId(id, d) -> codegen_id true false id d llbuilder
541 | SBinop(e1, op, e2, d) -> handle_binop e1 op e2 d llbuilder
542 | SAssign(e1, e2, d) -> codegen_assign e1 e2 d llbuilder
543 | SNoexpr -> build_add (const_int i32_t 0) (const_int i32_t 0)
  ↪ "nop" llbuilder
544 | SArrayCreate(t, el, d) -> codegen_array_create llbuilder t d el
545 | SArrayAccess(e, el, d) -> codegen_array_access false e el d llbuilder
546 | SObjAccess(e1, e2, d) -> codegen_obj_access true e1 e2 d llbuilder
547 | SCall(fname, el, d, _) -> codegen_call llbuilder d el fname
548 | SObjectCreate(id, el, d) -> codegen_obj_create id el d llbuilder
549 | SArrayPrimitive(el, d) -> codegen_array_prim d el llbuilder
550 | SUnop(op, e, d) -> handle_unop op e d llbuilder
551 | SNull -> const_null i32_t
552 | SDelete e -> codegen_delete e
  ↪ llbuilder
553

```



```

554 and codegen_if_stmt exp then_ (else_:Sast.sstmt) llbuilder =
555 let cond_val = codegen_sexpr llbuilder exp in
556
557 (* Grab the first block so that we might later add the conditional branch
558 * to it at the end of the function. *)
559 let start_bb = insertion_block llbuilder in
560 let the_function = block_parent start_bb in
561
562 let then_bb = append_block context "then" the_function in
563
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_alloca 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 alloca = build_alloca t var_name llbuilder in
675 Hashtbl.add named_values var_name alloca;
676 let lhs = SId(var_name, datatype) in
677 match expr with
678 SNoexpr -> alloca
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 name d llbuilder)
686   ↳ llbuilder
687 | _ -> build_ret (codegen_id true true name d llbuilder) llbuilder)
688 | SObjAccess(e1, e2, d) -> build_ret (codegen_obj_access true e1 e2 d llbuilder)
689   ↳ llbuilder
690 | SNoexpr -> build_ret_void llbuilder
691 | _ -> build_ret (codegen_sexpr llbuilder expr) llbuilder
692
693 and codegen_break llbuilder =
694 let block = fun () -> !br_block in
695 build_br (block ()) llbuilder
696
697 and codegen_continue llbuilder =
698 let block = fun () -> !cont_block in
699 build_br (block ()) llbuilder
700
701

```

```

699 and codegen_stmt llbuilder = function
700 SBlock s1                                -> List.hd(List.map (codegen_stmt llbuilder) s1)
701 | SExpr(e, d)                             -> codegen_sexpr llbuilder e
702 | SReturn(e, d)                           -> codegen_ret d e llbuilder
703 | SIf (e, s1, s2)                          -> codegen_if_stmt e s1 s2 llbuilder
704 | SFor (e1, e2, e3, s)                     -> codegen_for e1 e2 e3 s llbuilder
705 | SWhile (e, s)                            -> codegen_while e s llbuilder
706 | 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, _) -> get_type t :: l
714   ↪ | _ -> is_var_arg := true; l )) [] sfdecl.sformals) in
714 let fty = if !is_var_arg
715 then var_arg_function_type (get_type sfdecl.sreturnType) (Array.of_list params)
716 else function_type (get_type sfdecl.sreturnType) (Array.of_list params)
717 in
718 define_function fname fty the_module
719
720 let init_params f formals =
721 let formals = Array.of_list (formals) in
722 Array.iteri (fun i a ->
723 let n = formals.(i) in
724 let n = Utils.string_of_formal_name n in
725 set_value_name n a;
726 Hashtbl.add named_params n a;
727 ) (params f)
728
729 let codegen_func sfdecl =
730 Hashtbl.clear named_values;
731 Hashtbl.clear named_params;
732 let fname = (Utils.string_of_fname sfdecl.sfname) in
733 let f = func_lookup fname in
734 let llbuilder = builder_at_end context (entry_block f) in
735 let _ = init_params f sfdecl.sformals in
736 let _ = if sfdecl.overrides then
737 let this_param = Hashtbl.find named_params "this" in
738 let source = Datatype(Objecttype(sfdecl.source)) in
739 let casted_param = build_pointercast this_param (get_type source) "casted" llbuilder in
740 Hashtbl.replace named_params "this" casted_param;
741 in
742 let _ = codegen_stmt llbuilder (SBlock (sfdecl.sbody)) in
743 if sfdecl.sreturnType = Datatype(Void_t)
744 then ignore(build_ret_void llbuilder);
745 ()
746

```

```

747 let codegen_vtbl sdecls =
748 let rt = pointer_type i64_t in
749 let void_pt = pointer_type i64_t in
750 let void_ppt = pointer_type void_pt in
751
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_ppt (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) counter;
863 let cmp = build_icmp Icmp.Slt counter argc "tmp" llbuilder in
864 ignore (build_cond_br cmp bbbody bbdone llbuilder);
865 position_at_end bbbody llbuilder;
866
867 (* Assign array position to init_val *)
868 let arr_ptr = build_gep args [| counter |] "tmp" llbuilder in
869 let argv_val = build_gep argv [| counter |] "tmp" llbuilder in
870 let argv_val = build_load argv_val "tmp" llbuilder in
871 ignore (build_store argv_val arr_ptr llbuilder);
872 ignore (build_br bbcond llbuilder);
873 position_at_end bbdone llbuilder
874
875 let construct_args argc argv llbuilder =
876 let str_pt = pointer_type str_t in
877 let size_real = build_add argc (const_int i32_t 1) "arr_size" llbuilder in
878
879 let arr = build_array_malloc str_pt size_real "args" llbuilder in
880 let arr = build_pointercast arr str_pt "args" llbuilder in
881 let arr_len_ptr = build_pointercast arr (pointer_type i32_t) "argc_len" llbuilder in
882 let arr_1 = build_gep arr [| const_int i32_t 1 |] "arr_1" llbuilder in
883
884 (* Store length at this position *)
885 ignore (build_store argc arr_len_ptr llbuilder);
886 ignore (init_args argv arr_1 argc llbuilder);
887 arr
888
889 let codegen_main main =
890 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 Llvml
2  open Llvml_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 _ = Llm_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
77   ↪ (token_list ()))
78 | TokenEndl -> print_string (Utils.token_list_to_string_endl
79   ↪ (token_list ()))
80 | Ast -> print_string (pretty_to_string
81   ↪ (Utils.print_tree (program ())))
82 | Sast -> print_string (pretty_to_string
83   ↪ (Utils.map_sprogram_to_json (sprogram ())))
84 | PrettyPrint -> print_string (Utils.string_of_program (program ()))
85 | Compile -> dump_module (llm ())
86 | CompileToFile -> print_module (dice_name filename) (llm ())
87 with
88 Exceptions.IllegalCharacter(filename, c, ln) ->
89 print_string
90 (
91 "In \"^ filename ^ "\", Illegal Character, '\" ^
92 Char.escaped c ^ '\", line \" ^ string_of_int ln ^ "\n"
93 )
94 | Exceptions.UnmatchedQuotation(ln) -> print_endline("Unmatched
95   ↪ 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 ReturnTypeError 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!" context
15 in
16 raise ex
17 in Printf.ksprintf do_raise fmt
18
19 module StringMap = struct
20 include Map.Make(String)
21 let find_nf = find
22 let find_safe key map = try find key map with Not_found -> raise_safe "BUG: Key '%s' not
    ↳ found in StringMap!" key
23 let find key map = try Some (find key map) with Not_found -> None
24 let map_bindings fn map = fold (fun key value acc -> fn key value :: acc) map []
25 end
26
27 type path_component =
28 | Filename of string    (* foo/ *)
29 | ParentDir    (* ../ *)
30 | CurrentDir    (* ./ *)
31 | EmptyComponent (* / *)
32
33 type filepath = string
34
35
36 let on_windows = Filename.dir_sep <> "/"
37
38 let path_is_absolute path = not (Filename.is_relative path)
39
40 let string_tail s i =
41 let len = String.length s in
42 if i > len then failwith ("String '\" ^ s ^ \"' too short to split at " ^ (string_of_int
    ↳ i))
43 else String.sub s i (len - i)
44
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 < 1 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 < 1 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 +/ 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
```

Makefile

```
1 TARGET=src/dice
2 LIBS=-I,/usr/lib/ocaml/
3 FLAGS= -j 0 -r -use-ocamlfind -pkg
  ↪ yojson,llvm,llvm.analysis,llvm.bitwriter,llvm.bitreader,llvm.linker,llvm.target,batteries
4 OCAMLBUILD=ocamlbuild
5 OPAM=opam config env
6 CLIBEXT=_includes
7
8
9 all: native
10     @clang-3.7 -c -emit-llvm src/bindings.c
11     @mkdir -p $(CLIBEXT)
12     @mv bindings.bc $(CLIBEXT)/bindings.bc
13     @cp src/stdlib.dice $(CLIBEXT)/stdlib.dice
14     @mv dice.native dice
15     @echo Compilation Complete
16
17 clean:
18     @cd src
19     $(OCAMLBUILD) -clean
20     @cd ..
21     @rm -rf $(CLIBEXT)
22     @echo cleaning complete
23
24 native:
25     @cd src
26     @eval `opam config env`
27     $(OCAMLBUILD) $(FLAGS) $(TARGET).native
28     @cd ..
29
30 byte:
31     $(OCAMLBUILD) $(FLAGS) $(TARGET).byte
32
33 depend:
34     echo "Not needed."
```

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 GT LEQ 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     forms_opt:
154     /* nothing */ { [] }
155     |          formal_list   { List.rev $1 }
156
157     formal_list:
158     formal                { [$1] }
159     |          formal_list COMMA formal { $3 :: $1 }
160
161     formal:
162     datatype ID { Formal($1, $2) }
163
164     actuals_opt:
165     /* nothing */ { [] }
166     |          actuals_list   { List.rev $1 }
167
168     actuals_list:
169     expr                { [$1] }
170     |          actuals_list COMMA expr { $3 :: $1 }
171
172
173     /*****
174     DATATYPES
175     *****/
176     primitive:
177     INT                { Int_t }
178     |          FLOAT                { Float_t }
179     |          CHAR                { Char_t }
180     |          BOOL                { Bool_t }
181     |          VOID                { Void_t }
182
183     name:
184     CLASS ID { Objecttype($2) }
185
186     type_tag:
187     primitive { $1 }
188     |          name                { $1 }
189
190     array_type:
191     type_tag LBRACKET brackets RBRACKET { Arraytype($1, $3) }
192
193     datatype:
194     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   ↪ }
232 |     expr MINUS expr { Binop($1, Sub, $3)
233   ↪ }
234 |     expr TIMES expr { Binop($1, Mult, $3)
235   ↪ }
236 |     expr DIVIDE expr { Binop($1, Div, $3)
237   ↪ }
238 |     expr EQ expr { Binop($1, Equal, $3)
239   ↪ }
240 |     expr NEQ expr { Binop($1, Neq, $3)
241   ↪ }
242 |     expr LT expr { Binop($1, Less, $3)
243   ↪ }

```

```

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 lexbuf
24             ↪ token_list)
25     in helper 0 0 lexbuf []
26
27 let parser filen token_list =
28     let token_list = ref(token_list) in
29     let tokenizer _ =
30         match !token_list with
31         | (head, curr) :: tail ->
32             filename := filen;
33             line_number := curr.lineno;
34             char_num := curr.cnum;
35             last_token := head;
36             token_list := tail;
37         | [] -> raise (Exceptions.MissingEOF)
38     in
39     let program = Parser.program tokenizer (Lexing.from_string "") in
40     program
```

sast.ml

```

1  open Ast
2
3  type sexpr =
4  SInt_Lit of int
5  |
6    SBoolean_Lit of bool
7  |
8    SFloat_Lit of float
9  |
10   SString_Lit of string
11 |
12   SChar_Lit of char
13 |
14   SId of string * datatype
15 |
16   SBinop of sexpr * op * sexpr * datatype
17 |
18   SAssign of sexpr * sexpr * datatype
19 |
20   SNoexpr
21 |
22   SArrayCreate of datatype * sexpr list * datatype
23 |
24   SArrayAccess of sexpr * sexpr list * datatype
25 |
26   SObjAccess of sexpr * sexpr * datatype
27 |
28   SCall of string * sexpr list * datatype * int
29 |
30   SObjectCreate of string * sexpr list * datatype
31 |
32   SArrayPrimitive of sexpr list * datatype
33 |
34   SUNop of op * sexpr * datatype
35 |
36   SNull
37 |
38   SDelete of sexpr
39
40 type sstmt =
41 SBlock of sstmt list
42 |
43   SExpr of sexpr * datatype
44 |
45   SReturn of sexpr * datatype
46 |
47   SIf of sexpr * sstmt * sstmt
48 |
49   SFor of sexpr * sexpr * sexpr * sstmt
50 |
51   SWhile of sexpr * sstmt
52 |
53   SBreak
54 |
55   SContinue
56 |
57   SLocal of datatype * string * sexpr
58
59 type func_type = User | Reserved
60
61 type sfunc_decl = {
62   sfname : fname;
63   sreturnType : datatype;
64   sformals : formal list;
65   sbody : sstmt list;
66   func_type : func_type;
67   source : string;
68   overrides : bool;
69 }
70
71 type sclass_decl = {
72   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, e1)          -> "new " ^ s ^ "(" ^ String.concat ", " (List.map
   ↪   string_of_expr e1) ^ ")"
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 (indent+1)) stmts) ^
133     indent_string ^ "}\n"
134
135 | Expr(expr)                                ->
136 indent_string ^ string_of_expr expr ^ ";\n";
137
138 | Return(expr)                                ->
139 indent_string ^ "return " ^ string_of_expr expr ^ ";\n";
140
141 | If(e, s, Block([Expr(Noexpr)]))            ->
142 indent_string ^ "if (" ^ string_of_expr e ^ ")\n" ^
143 (string_of_stmt (indent+1) s)
144
145 | If(e, s1, s2)                                ->
146 indent_string ^ "if (" ^ string_of_expr e ^ ")\n" ^
147 string_of_stmt (indent+1) s1 ^
148 indent_string ^ "else\n" ^
149 string_of_stmt (indent+1) s2
150
151 | For(e1, e2, e3, s)                            ->
152 indent_string ^ "for (" ^ string_of_expr e1 ^ " ; " ^ string_of_expr e2 ^ " ; " ^
153     ^ string_of_expr e3 ^ ")\n" ^
154 string_of_stmt (indent) s
155
156 | While(e, s)                                    ->
157 indent_string ^ "while (" ^ string_of_expr e ^ ")\n" ^
158 string_of_stmt (indent) s
159
160 | Break                                            -> indent_string ^ "break;\n"
161 | Continue                                        -> indent_string ^ "continue;\n"
162 | Local(d, s, e)                                -> indent_string ^ string_of_datatype d ^ " "
163     ^ s ^ string_of_local_expr e ^ ";\n"
164 in get_stmt_string
165
166 let string_of_local_sexpr = function
167 SNoexpr                -> ""
168 | e                    -> " = " ^ string_of_sexpr e
169
170 let rec string_of_sstmt indent =
171 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 (indent+1)) stmts) ^
175     indent_string ^ "}\n"
176
177   | SExpr(expr, _)                             ->
178   indent_string ^ string_of_sexpr expr ^ ";\n";
179
180   | SReturn(expr, _)                           ->
181   indent_string ^ "return " ^ string_of_sexpr expr ^ ";\n";
182
183   | SIf(e, s, SBlock([SExpr(SNoexpr, _)]))      ->
184   indent_string ^ "if (" ^ string_of_sexpr e ^ ")\n" ^
185   (string_of_sstmt (indent+1) s)
186
187   | SIf(e, s1, s2)                             ->
188   indent_string ^ "if (" ^ string_of_sexpr e ^ ")\n" ^
189   string_of_sstmt (indent+1) s1 ^
190   indent_string ^ "else\n" ^
191   string_of_sstmt (indent+1) s2
192
193   | SFor(e1, e2, e3, s)                         ->
194   indent_string ^ "for (" ^ string_of_sexpr e1 ^ " ; " ^ string_of_sexpr e2 ^ " ; " ^
195   ↪ string_of_sexpr e3 ^ ")\n" ^
196   string_of_sstmt (indent) s
197
198   | SWhile(e, s)                               ->
199   indent_string ^ "while (" ^ string_of_sexpr e ^ ")\n" ^
200   string_of_sstmt (indent) s
201
202   | SBreak                                     -> indent_string ^ "break;\n"
203   | SContinue                                -> indent_string ^ "continue;\n"
204   | SLocal(d, s, e)                          -> indent_string ^ string_of_datatype d ^ " "
205   ↪ ^ s ^ string_of_local_sexpr e ^ ";\n"
206
207 in get_stmt_string
208
209 (* Print Function *)
210
211 let string_of_fname = function
212   Constructor -> "constructor"
213   | FName(s)   -> s
214
215 let string_of_formal = function
216   Formal(d, s) -> (string_of_datatype d) ^ " " ^ s
217   | _          -> ""
218
219 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) ^ ") {" ^
225   (* body *)
226   String.concat "" (List.map (string_of_stmt 2) fdecl.body) ^
227   "\t}\n\n"
228   (* Class Printing *)
229
230 let string_of_extends = function
231   NoParent      -> ""
232 |   Parent(s)   -> "extends " ^ s ^ " "
233 let string_of_field = function
234   Field(s, d, id) -> (string_of_scope s) ^ " " ^ (string_of_datatype d) ^ " " ^ id ^ ";\n"
235
236 let string_of_cbody cbody =
237   String.concat "" (List.map (fun s -> "\t" ^ s) (List.map string_of_field cbody.fields)) ^
238   String.concat "" (List.map (fun s -> "\t" ^ s) (List.map string_of_func_decl
239   ↪ cbody.constructors)) ^
240   String.concat "" (List.map (fun s -> "\t" ^ s) (List.map string_of_func_decl
241   ↪ cbody.methods))
242
243 let string_of_class_decl cdecl =
244   "class " ^ cdecl.cname ^ " " ^ (string_of_extends cdecl.extends) ^ "{\n" ^
245   (string_of_cbody cdecl.cbody) ^
246   "}\n"
247
248 (* Include Printing *)
249
250 let rec string_of_include = function
251   Include(s) -> "include(" ^ s ^ ");\n"
252
253 (* Print whole program *)
254
255 let string_of_program = function
256   Program(includes, cdecls) ->
257   String.concat "" (List.map string_of_include includes) ^ "\n" ^
258   String.concat "\n" (List.map string_of_class_decl cdecls)
259
260 (* Print AST tree representation *)
261
262 let includes_tree includes =
263   'List (List.map (function Include s -> 'String s) includes)

```



```

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     ("name", 'String s);
274     ("datatype", 'String (string_of_datatype d));
275   ]
276   | Many d -> 'Assoc [("Many", 'String (string_of_datatype d));]
277   ) formals)
278
279 let rec map_expr_to_json = function
280   Int_Lit(i) -> 'Assoc [("int_lit", 'Int i)]
281   | Boolean_Lit(b) -> 'Assoc [("bool_lit", 'Bool b)]
282   | Float_Lit(f) -> 'Assoc [("float_lit", 'Float f)]
283   | String_Lit(s) -> 'Assoc [("string_lit", 'String s)]
284   | Char_Lit(c) -> 'Assoc [("char_lit", 'String
    ↪ (Char.escaped c))]
285   | This -> 'String "this"
286   | Id(s) -> 'Assoc [("id", 'String s)]
287   | Binop(e1, o, e2) -> 'Assoc [("binop", 'Assoc [("lhs",
    ↪ map_expr_to_json e1); ("op", 'String (string_of_op o)); ("rhs", map_expr_to_json
    ↪ e2)]]]
288   | Assign(e1, e2) -> 'Assoc [("assign", 'Assoc [("lhs",
    ↪ map_expr_to_json e1); ("op", 'String "="); ("rhs", map_expr_to_json e2)]]]
289   | 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 scdecl.scname);
400 ("sfields", map_fields_to_json scdecl.sfields);
401 ("sfuncs", map_sfdecls_to_json scdecl.sfuncs);
402 ])]
403 ])
404 scdecls)
405
406 let map_sprogram_to_json sprogram =
407 'Assoc [("sprogram", 'Assoc [

```

```

408 ("classes", map_scdecls_to_json sprogram.classes);
409 ("functions", map_sfdecls_to_json sprogram.functions);
410 ("main", map_sfdecl_to_json sprogram.main);
411 ("reserved", map_sfdecls_to_json sprogram.reserved);
412 ])]
413
414 (* Print tokens *)
415
416 let string_of_token = function
417 LPAREN                                -> "LPAREN"
418 |      RPAREN                          -> "RPAREN"
419 |      LBRACE                          -> "LBRACE"
420 |      RBRACE                          -> "RBRACE"
421 |      SEMI                            -> "SEMI"
422 |      COMMA                           -> "COMMA"
423 |      PLUS                            -> "PLUS"
424 |      MINUS                           -> "MINUS"
425 |      TIMES                           -> "TIMES"
426 |      DIVIDE                          -> "DIVIDE"
427 |      ASSIGN                          -> "ASSIGN"
428 |      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

```

Test Suite Code

tester.sh

```
1  #!/bin/bash
2  # This script must reside in the "Test Suite" directory of the project
3  # Make sure the "dice" executable is in the "Compiler" directory
4
5  diceExecPath=./dice
6  testOption=$1 #stores the test flag since functions can't see the £1
7  vFlag=$2 #stores the -v flag since functions can't see it with £2
8  pass=0
9  fail=0
10 RED='\033[0;31m'
11 GREEN='\033[0;32m'
12 CYAN='\033[0;36m'
13 NC='\033[0m'
14 errorFile=errors.log
15 excpTestFlag=0
16
17 # Set time limit for all operations
18 ulimit -t 30
19
20 usage(){
21     echo "Usage: $0 [test flag] [other]";
22     echo "";
23     echo "[test flag] = -c    Test Compiler (default if test flag not selected)";
24     echo "                -d    Test Compiler and display Dice Compiler messages";
25     echo "                -s    Test Scanner";
26     echo "                -m    Run script without compiling Dice executable";
27     echo "[other]      = -v    Verbose (prints log results)";
28     exit 1;
29 }
30
31 confirmation(){
32     #£? is the exit code for diff, if 0, then test output matched!
33     if [ $? -eq 0 ];
34     then
35         echo -e "${GREEN}$filename passed!${NC}" >> session_file
36         echo -e "${GREEN}$filename passed!${NC}"
37         ((pass++))
38     else
39
40         echo -e "${RED}$filename FAILED${NC}" >> session_file
41         echo -e "${RED}$filename FAILED${NC}"
42
43         #print out expected output and result
44         echo "Expected Output:" >> session_file
45
```



```

46         if [ $excpTestFlag -eq 0 ];           then
47             cat "$testPath"$filename$testExtension >> session_file
48         else
49             cat "$testExceptionsPath"$filename$testExtension >>
               ↪ session_file
50         fi
51         echo "" >> session_file
52         echo "Generated Output:" >> session_file
53         cat temp_Dice_Tester >> session_file
54         echo "" >> session_file
55         ((fail++))
56     fi
57 }
58
59 header(){
60     echo ""
61     echo "*****" >> session_file
62     echo "Dice Test Script Results:" >> session_file
63     date >> session_file
64     echo "" >> session_file
65 }
66
67 test_function(){
68     header #func
69
70     for testFile in "$testPath"*.dice; do
71
72         filename=$(basename "$testFile")
73
74         echo "=====" >> session_file
75         echo "Testing: $filename" >> session_file
76
77         if [ "$testOption" == "-s" ]; then
78             #Create file to be tested (with tokens)
79             $diceExecPath $diceOption "$testFile" > temp_Dice_Tester
80             #Test output differences use the diff command and neglect screen
               ↪ output
81             diff temp_Dice_Tester "$testPath"$filename$testExtension >
               ↪ /dev/null
82             confirmation #function
83         else #Only other option is -c or -d which perform the same function
               ↪ except where noted below
84             #extract filename without extension for executable
85             name=$(echo $filename | cut -f 1 -d '.')
86
87             if [ "$testOption" == "-d" ]; then
88                 #run the executable and port output (stderr) to temp test
               ↪ file
89                 #port stdout (compiler msgs) to screen with color

```

```

90         echo -e -n "${CYAN}"
91         $diceExecPath $diceOption "$testFile" 2> temp.ll
92         echo -e -n "${NC}"
93         echo ""
94
95     else
96         #Create header for any messages coming from Dice compiler
97
98         echo -e "${CYAN}Dice Compiler Messages (if any):" >>
99         ↪ session_file
100        #run the executable and port output (stderr) to temp test
101        ↪ file
102        #port stdout (compiler msgs) to log file
103        $diceExecPath $diceOption "$testFile" 2> temp.ll 1>>
104        ↪ session_file
105        echo -e "${NC}">> session_file
106        echo "" >> session_file
107    fi
108
109    #Run the llvm executable and port output to temp test file
110    lli temp.ll > temp_Dice_Tester
111
112    #Send all error messages this script generates (if any) to error
113    ↪ log file
114    exec 2> $errorFile
115
116    #Perform comparison of outputs
117    diff temp_Dice_Tester "$testPath"$filename$testExtension >
118    ↪ /dev/null
119    confirmation #function
120
121    fi
122done
123
124#The following portion is only to test compiler errors
125if [ "$testOption" == "-c" ] || [ "$testOption" == "-d" ] || [ "$testOption" ==
126↪ "-m" ] || [ $# -eq 0 ]; then
127
128    #set flag to prevent
129    excpTestFlag=1
130    for testFile in "$testExceptionsPath"*.dice; do
131
132        filename=$(basename "$testFile")
133
134        echo "===== " >> session_file
135        echo "Testing: $filename" >> session_file
136
137        #Only other option is -c or -d which perform the same function
138        ↪ except where noted below
139        #extract filename without extension for executable

```

```
132         name=$(echo $filename | cut -f 1 -d '.')
133
134         #run the executable and port error output (stdout) to temp test
135         ↪ file
136         #port stdout (compiler msgs) to log file
137         $diceExecPath $diceOption "$testFile" 1> temp_Dice_Tester
138         ↪ 2>/dev/null
139
140         #Perform comparison of outputs
141         diff temp_Dice_Tester
142         ↪ "$testExceptionsPath"$filename$testExtension >> /dev/null
143         confirmation #function
144     done
145
146     #Test if our executable can take in command line arguments:
147     filename=test-args.dice
148     $diceExecPath $diceOption "$argsPath"test-args.dice 2>temp.ll
149     lli temp.ll david emily phil > tempArgs
150     diff tempArgs "$argsPath"test-args.dice.out >/dev/null
151     confirmation
152     rm tempArgs
153
154 fi
155 echo "" >> session_file
156
157 #Verbose flag actuated
158 if [ "$vFlag" == "-v" ]; then
159     cat session_file
160 fi
161
162 #Copy session output to historical log
163 cat session_file >> "$logFile"
164
165 #Test status output
166 echo ""
167 echo -e "${GREEN}Tests Passed: $pass ${NC}"
168 echo -e "${RED}Tests Failed: $fail ${NC}"
169 echo "View $logFile for more information"
170
171 #Clean up temp files
172 rm temp_Dice_Tester;
173 rm session_file;
174 }
175
176 createDice(){
177     echo "Compiling dice executable"
178     cd ..
179     make clean 2>&1 > /dev/null
180     make
```

```
178     #cp dice ../Test\ Suite/Hello_World_Demo/dice
179     # cd Test\ Suite
180     echo "Compilation of dice executable complete"
181 }
182
183 #-----Script starts flag checking here -----
184 if [ "$testOption" == "-s" ]; then
185     echo "Scanner Test Started"
186     createDice
187     logFile=Test\ Suite/scanner_tests.log
188     testPath=Test\ Suite/Scanner\ Test\ Suite/
189     diceOption=-tendl
190     testExtension=.ManualTokens
191     test_function
192
193 elif [ "$testOption" == "-c" ] || [ "$testOption" == "-d" ] || [ "$testOption" == "-m" ]
194 ↪ || [ $# -eq 0 ]; then
195     echo "Compiler Test Started"
196
197     if [ "$testOption" == "-m" ]; then
198         if [ -f ../dice ]; then
199             echo "Skipping Dice recompilation"
200             cd ..
201         else
202             createDice
203         fi
204     else
205         createDice
206     fi
207
208     logFile=Test\ Suite/compiler_tests.log
209     testPath=Test\ Suite/Compiler_Test_Suite/
210     testExceptionsPath=Test\ Suite/Compiler_Test_Suite/Exceptions/
211     argsPath=Test\ Suite/Compiler_Test_Suite/Args/
212     diceOption=-c
213     testExtension=.out
214     test_function
215     rm temp.ll;
216
217 else
218     usage
219 fi
220
221 #Print out number of bash script errors and
222 if [ "$testOption" != "-s" ]; then
223     errorLines=$(cat $errorFile | wc -l)
224     mv $errorFile Test\ Suite/$errorFile
225     if [ $errorLines -ne 0 ]; then
```

```
226     echo "$errorLines lines of script errors reported. Please check $errorFile!"
227     else
228         mv Test\ Suite/$errorFile
229     fi
230 fi
231
232 exit 0
```

test-var1.dice.out

1 42

test-stdlib-stringclass.dice.out

```
1  hi
```

test-stdlib-integerclass1.dice

```
1  include("stdlib");
2
3  class Two {
4      public void main(char[] [] args) {
5          class Integer x = new Integer(128);
6          print(x.num(), "\n");
7      }
8  }
```


test-constructorInherited.dice

```
1  class shape {
2      public int xCoord;
3      public int yCoord;
4
5      constructor(){
6          this.xCoord = 0;
7          this.yCoord = 0;
8      }
9
10     constructor(int x, int y){
11         this.xCoord = x;
12         this.yCoord = y;
13     }
14 }
15
16 class circle extends shape {
17     public int radius;
18
19     constructor(){
20         this.radius = 0;
21     }
22     constructor(int r){
23         this.radius = r;
24     }
25     constructor(int x, int y, int r){
26         this.radius = r;
27         this.xCoord = x;
28         this.yCoord = y;
29     }
30 }
31
32 class test {
33     public void main(char[] [] args) {
34         class circle a = new circle(0,0,7);
35         print(a.xCoord);
36         print(a.yCoord);
37         print(a.radius);
38     }
39 }
```

test-ifEmptyBlock2.dice.out

1 17

test-global1.dice.out

1 42214322

test-if7.dice

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

test-var3.dice

```
1  class test {
2
3      public int a;
4
5      public void print2(int x, int y) {
6          print(x);
7          print(y);
8      }
9
10     public void main(char[] [] args) {
11         int b;
12         this.a = 42;
13         b = 57;
14         this.print2(this.a + b * 3, 77);
15     }
16 }
```

test-classFunctionOverload1.dice.out

1 10

test-applicative.dice

```
1  class test {
2
3      public int p(int i){
4          print(i);
5          return i;
6      }
7
8      public void q(int a, int b, int c){
9          int total = a ;
10         print(b);
11         total = total + c ;
12     }
13
14     public void main(char[] [] args) {
15         this.q( this.p(1), 2, this.p(3));
16     }
17 }
```

test-forEmptyBlock2.dice

```
1  class test {
2      public void main(char[] [] args) {
3          int i;
4          for (i = 0 ; i < 5 ; i = i + 1) {
5              (*empty block*) null;
6          }
7          print(1);
8      }
9  }
```


test-if1.dice

```
1  class test {  
2      public void main(char[] [] args) {  
3          if (true) print(42);  
4          print(17);  
5      }  
6  }
```

test-func5.dice

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

test-arith5.dice

```
1  class test {  
2      public void main(char[] [] args) {  
3          print(15-5);  
4      }  
5  }
```

test-bool5.dice

```
1  class test {  
2      public void main(char[] [] args) {  
3          print(1==2);  
4          print(1==1);  
5      }  
6  }
```

test-constructor2.dice

```
1  class shape {
2      public int xCoord;
3      public int yCoord;
4
5      constructor(int x, int y){
6          this.xCoord = x;
7          this.yCoord = y;
8      }
9
10     constructor(float x, float y){
11         this.xCoord = 0;
12         this.yCoord = 0;
13     }
14 }
15
16 class test {
17     public void main(char[][] args) {
18         class shape a = new shape(5,10);
19         print (a.xCoord);
20         print (a.yCoord);
21     }
22 }
```

test-arithSigned2.dice.out

1 -3-3-3.000000-3.000000

test-classExtends2.dice

```
1  class person {
2      public int ssn;
3  }
4
5  class worker extends person {
6      public int workid;
7  }
8
9  class programmer extends worker {
10     public int nerdCred;
11 }
12
13 class test {
14     public void main(char[] [] args) {
15         class programmer david = new programmer();
16         david.ssn = 123456789;
17         david.workid = 57;
18         david.nerdCred = 99;
19
20         print(david.ssn);
21         print(david.workid);
22         print(david.nerdCred);
23     }
24 }
```

test-arithSigned1.dice.out

1 -5-5-5.000000-5.000000

test-forEmptyBlock.dice

```
1  class test {  
2      public void main(char[] [] args) {  
3          int i;  
4          for (i = 0 ; i < 5 ; i = i + 1) {  
5              (*empty block*)  
6          }  
7          print(1);  
8      }  
9  }
```

test-func5.dice.out

1 113

test-float.dice.out

1 1.500000

test-stdlib-integerclass1.dice.out

1 128

test-for2.dice.out

1 5432142

test-if4.dice

```
1  class test {  
2      public void main(char[] [] args) {  
3          if (false)  
4              print(42);  
5          else  
6              print(8);  
7          print(17);  
8      }  
9  }
```

test-arith7.dice

```
1  class test {  
2      public void main(char[] [] args) {  
3          print(15/5);  
4      }  
5  }
```

test-if5.dice

```
1  class test {
2      public void main(char[] [] args) {
3          this.foo(3,5,6);
4      }
5
6      public void foo(int a, int b, int c) {
7          int d;
8          if (a == 3)
9              d = b;
10         else
11             d = c;
12         print(d);
13     }
14 }
```


test-arithSigned3.dice

```
1  class test {
2      public void main(char[] [] args) {
3          print(-1+3);
4          print(1+-3);
5          print(-1.0+3.0);
6          print(1.0+-3.0);
7      }
8  }
```

test-if7.dice.out

1 else

test-classGetter.dice.out

1 13

test-stdlib-compare.dice

```
1  include("stdlib");
2
3  class Two {
4      public void main(char[] [] args) {
5          class String b = new String("phil");
6          class String c = new String("khal");
7          class String d = c.copy(c);
8          print(b.string(), " == ", c.string(), " is ", b.compare(c));
9          print(c.string(), " == ", d.string(), " is ", c.compare(d));
10         }
11     }
```

test-class.dice.out

1 13

test-for1.dice.out

1 0123442

test-classInheritanceArgument.dice

```
1  class shape {
2      public int xCoord;
3      public int yCoord;
4  }
5
6  class circle extends shape {
7      public int radius;
8  }
9
10 class test {
11
12     public void main(char[] [] args) {
13         class circle a = new circle();
14         this.inheritanceTest(a);
15     }
16
17     public void inheritanceTest(class shape a){
18         print("pass");
19     }
20
21 }
```

test-whileBreak.dice

```
1  class test {
2      public void main(char[] [] args) {
3          int i;
4          i = 5;
5          while (i > 0) {
6              print(i);
7              if(i==3){
8                  break;
9              }
10             i = i - 1;
11         }
12     }
13 }
```


test-while1.dice

```
1  class test {
2      public void main(char[] [] args) {
3          int i;
4          i = 5;
5          while (i > 0) {
6              print(i);
7              i = i - 1;
8          }
9          print(42);
10     }
11 }
```

test-fileio.dice.out

```
1  include("stdlib");
2
3  class Two {
4
5      public void main(char[] [] args) {
6          class File a = new File("Test Suite/Compiler_Test_Suite/test-fileio.dice", true);
7          char[] buf = a.readfile(243);
8          a.closefile();
9          print(buf);
10         }
11     }
```

test-classExtends2.dice.out

1 1234567895799

test-forContinue.dice.out

1 23420

test-fib.dice

```
1  class test {
2
3      public int fib(int x) {
4          if (x < 2)
5              return 1;
6          return this.fib(x-1) + this.fib(x-2);
7      }
8
9      public void main(char[] [] args) {
10         print(this.fib(0));
11         print(this.fib(1));
12         print(this.fib(2));
13         print(this.fib(3));
14         print(this.fib(4));
15         print(this.fib(5));
16     }
17 }
```

test-bool1.dice

```
1  class test {  
2      public void main(char[] [] args) {  
3          print(1<2);  
4          print(1.0<2);  
5          print(1<2.0);  
6          print(1.0<2.0);  
7      }  
8  }
```

test-forBreak.dice

```
1  class test {
2      public void main(char[] [] args) {
3          int i;
4          for (i = 0 ; i < 5 ; i = i + 1) {
5              if(i==3){
6                  break;
7              }
8              print(i);
9          }
10         print(100);
11     }
12 }
```

test-bool6.dice

```
1  class test {
2      public void main(char[] [] args) {
3          print(1!=2);
4          print(1!=1);
5      }
6  }
```


test-bool4.dice.out

```
1  truetruetruefalse
```

test-stdlib-stringclassContains2.dice

```
1 include("stdlib");
2
3 class Two {
4     public void main(char[] [] args) {
5         class String b = new String("philkhal");
6         class String c = new String("butts");
7         print(b.contains(c));
8     }
9 }
```

test-classGetter.dice

```
1  class shape {
2      public int xCoord;
3      public int yCoord;
4
5      public int getX(){
6          return this.xCoord;
7      }
8      public int getY(){
9          return this.yCoord;
10     }
11
12 }
13
14 class test {
15     public void main(char[] [] args) {
16         class shape a = new shape();
17         a.xCoord = 1;
18         a.yCoord = 3;
19         print(a.getX());
20         print(a.getY());
21     }
22 }
```

test-var3.dice.out

1 21377

test-forContinue.dice

```
1  class test {
2      public void main(char[] [] args) {
3          int i;
4          for (i = 0 ; i < 5 ; i = i + 1) {
5              if(i<2){ continue; }
6              else{
7                  print(i);
8              }
9          }
10         print(20);
11     }
12 }
```

test-stdlib-stringclassReverse.dice.out

```
1  olleh
```

test-while1.dice.out

1 5432142

test-float.dice

```
1  class test {  
2      public void main(char[] [] args) {  
3          float a = 1.5;  
4          print(a);  
5  
6      }  
7  }
```


test-arith5.dice.out

1 10

test-array4.dice

```
1  class shape {
2      public int x;
3      public int y;
4
5      constructor(int a, int b){
6          this.x = a;
7          this.y = b;
8      }
9
10 }
11
12 class test {
13     public void main(char[] [] args) {
14         class shape[] a = new class shape[5];
15         class shape b = new shape(2,3);
16         a[1] = b;
17         print(a[1].x);
18     }
19 }
```

test-arithSigned1.dice

```
1  class test {
2      public void main(char[] [] args) {
3          print(-15/3);
4          print(15/-3);
5          print(-15.0/3.0);
6          print(15.0/-3.0);
7      }
8  }
```

test-if2.dice.out

1 4217

test-stdlib-concat.dice

```
1  include("stdlib");
2
3  class Two {
4      public void main(char[] [] args) {
5          class String b = new String("phil");
6          class String c = new String("khal");
7          class String a = b.concat(c);
8          print(b.string(), "\n");
9          print(c.string(), "\n");
10         print(a.string(), "\n");
11     }
12 }
```

test-classReturnObjects.dice.out

1 12

test-if8.dice

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

test-stmts1.dice

```
1  class test {
2      public void main(char[] [] args) {
3          print(this.foo(1,42));
4          print(this.foo(0,37));
5      }
6
7      public int foo(int a, int b) {
8          int i;
9          int j = b;
10         if ( a == 1)
11             return b + 3;
12         else
13             for (i = 0 ; i < 5 ; i = i + 1)
14                 j = j + 5;
15         return j;
16     }
17 }
```


test-if6.dice.out

1 42278

test-classExtendsGetter.dice.out

1 13

test-ops1.dice.out

```
1 3-125099false>true99true>false99true>false99true>true>false99false>true99false>true>true
```

test-arith4.dice

```
1  (* Test side-effect sequence in a series of statement *)
2
3  class test {
4      public int g;
5
6      public void main(char[] [] args) {
7
8          int l;
9          l = 1;
10         print(l);
11
12         this.g = 3;
13         print(this.g);
14
15         l = 5;
16         print(l+100);
17
18         this.g = 7;
19         print(this.g+100);
20     }
21 }
```

test-func3.dice.out

1 42171928

test-class.dice

```
1  class shape {
2      public int xCoord;
3      public int yCoord;
4
5      constructor (){
6      }
7  }
8
9  class test {
10     public void main(char[] [] args) {
11         class shape a = new shape();
12         a.xCoord = 1;
13         a.yCoord = 3;
14         print(a.xCoord);
15         print(a.yCoord);
16     }
17 }
```

test-bool9.dice.out

```
1  truetruetruefalsefalsefalse
```

test-whileContinue.dice.out

1 543

test-stdlib-copy.dice.out

1 philkhalkhal

test-stdlib-integerclass2.dice.out

1 128

test-classExtends.dice

```
1  class shape {
2      public float xCoord;
3      public float yCoord;
4  }
5
6  class circle extends shape {
7      public float radius;
8  }
9
10 class test {
11     public void main(char[] [] args) {
12         class circle a = new circle();
13         a.xCoord = 1.5;
14         print(a.xCoord);
15     }
16 }
```

test-if3.dice

```
1  class test {  
2      public void main(char[] [] args) {  
3          if (false)  
4              print(42);  
5          print(17);  
6      }  
7  }
```

test-bool8.dice.out

```
1  falsetrue
2  falsefalse
```

test-scope.dice.out

1 12321

test-constructor1.dice

```
1  class shape {
2      public int xCoord;
3      public int yCoord;
4
5      constructor(){
6          this.xCoord = 0;
7          this.yCoord = 0;
8      }
9
10     constructor(int x, int y){
11         this.xCoord = x;
12         this.yCoord = y;
13     }
14 }
15
16 class test {
17     public void main(char[][] args) {
18         class shape a = new shape();
19         class shape b = new shape(5,10);
20         print (a.xCoord);
21         print (a.yCoord);
22         print (b.xCoord);
23         print (b.yCoord);
24     }
25 }
```

test-stdlib-concat.dice.out

```
1  phil
2  khal
3  philkhal
```


test-forEmptyBlock2.dice.out

1 1

test-if4.dice.out

1 817

test-array.dice.out

1 04

test-array2.dice.out

1 1.5000004.500000

test-objectDeclarationInheritance.dice.out

1 pass

test-if5.dice.out

1 5

test-forEmptyBlock.dice.out

1 1

test-var4.dice.out

1 1242

test-whileContinue.dice

```
1  class test {
2      public void main(char[] [] args) {
3          int i;
4          i = 6;
5          while (i > 0) {
6              i = i - 1;
7
8              if(i<3){
9                  continue;
10             }
11
12             print(i);
13         }
14     }
15 }
```

test-array3.dice

```
1  class test {
2      public void main(char[] [] args) {
3          int[] a = new int[10];
4          a[0] = 1;
5          print(a[0]);
6          a[0] = 10;
7          print(a[0]);
8          a[9] = 2;
9          print(a[9]);
10     }
11 }
```

test-if3.dice.out

1 17

test-arith6.dice

```
1  class test {  
2      public void main(char[] [] args) {  
3          print(10*5);  
4      }  
5  }
```

test-helloTwice.dice.out

```
1 Hello, World!  
2 Professor Edwards favorite number is: 42!
```

test-stdlib-stringclassLength.dice.out

1 9

test-bool3.dice.out

```
1  falsetruefalsetrue
```

test-hello.dice

```
1  class test {  
2      public void main(char[] [] args) {  
3          print("Hello, World!");  
4      }  
5  }
```


test-array.dice

```
1  class test {  
2      public void main(char[] [] args) {  
3          int[] a = |0,1,2,3,4|;  
4          print(a[0]);  
5          print(a[4]);  
6      }  
7  }
```

test-exit.dice

```
1  class test {  
2      public void main(char[] [] args) {  
3  
4          print(1);  
5          exit(1);  
6          print(2);  
7  
8      }  
9  }
```

test-helloTwice.dice

```
1  class test {  
2      public void main(char[] [] args) {  
3          print("Hello, World!\n");  
4          print("Professor Edwards favorite number is: 42!\n");  
5      }  
6  }
```

test-arithSigned2.dice

```
1  class test {  
2      public void main(char[] [] args) {  
3          print(-1*3);  
4          print(1*-3);  
5          print(-1.0*3.0);  
6          print(1.0*-3.0);  
7      }  
8  }
```

test-cyclicalIncludes2.dice

```
1 include("Test Suite/Compiler_Test_Suite/test-cyclicalIncludes.dice");
2
3 class test2 {
4     constructor(){
5         this.output2();
6     }
7     public void output2(){
8         print("b");
9     }
10 }
```

test-if2.dice

```
1  class test {  
2      public void main(char[] [] args) {  
3          if (true) print(42);  
4          else print(8);  
5          print(17);  
6      }  
7  }
```

test-structorDefault.dice

```
1  class shape {
2      public int xCoord;
3      public int yCoord;
4
5  }
6
7  class test {
8      public void main(char[] [] args) {
9          class shape a = new shape();
10         a.xCoord = 5;
11         print (a.xCoord);
12     }
13 }
```

test-var1.dice

```
1  class test {  
2      public void main(char[] [] args) {  
3  
4          int a;  
5          a = 42;  
6          print(a);  
7      }  
8  }
```


test-arithSigned4.dice.out

1 -44-4.0000004.000000

test-ifEmptyBlock.dice.out

1 17

test-stdlib-compare.dice.out

```
1  phil == khal is falsekhal == khal is true
```

test-cyclicalIncludes.dice.out

1 ba

test-bool7.dice.out

```
1 truefalsefalsefalse
2 true true true false
```

test-classSetter.dice.out

1 13

test-stdlib-stringclassReverse.dice

```
1  include("stdlib");
2
3  class Test {
4      public void main(char[] [] args) {
5          class String a = new String("hello");
6          class String reverse = a.reverse();
7
8          print(reverse.string());
9      }
10 }
```

test-factorialRecursive.dice

```
1  class Factorial {
2
3      public void main(char[] [] args) {
4          print(this.factorial(5));
5      }
6
7      public int factorial(int n) {
8          int temp;
9          if(n <= 1) return 1;
10         temp = n * this.factorial(n - 1);
11         return temp;
12     }
13 }
```


test-classInheritanceArgument.dice.out

1 pass

test-ctorInherited.dice.out

1 007

test-bool8.dice

```
1  class test {
2      public void main(char[] [] args) {
3          print(not true);
4          print(not false);
5          print("\n");
6          print(not true and true);
7          print(not (true and true));
8      }
9  }
```

test-classFunctionOverload.dice

```
1  class shape {
2      public int xCoord;
3      public int yCoord;
4
5      constructor(){
6          this.xCoord = 0;
7          this.yCoord = 0;
8      }
9
10     constructor(int x, int y){
11         this.xCoord = x;
12         this.yCoord = y;
13     }
14
15     public int getArea(){
16         return 10;
17     }
18 }
19
20 class circle extends shape {
21     public int radius;
22
23     constructor(){
24         this.radius = 0;
25     }
26     constructor(int r){
27         this.radius = r;
28     }
29     constructor(int x, int y, int r){
30         this.radius = r;
31         this.xCoord = x;
32         this.yCoord = y;
33     }
34
35     public int getArea(){
36         return 3*this.radius*this.radius;
37     }
38 }
39
40 class test {
41     public void main(char[][] args) {
42         class circle a = new circle(0,0,2);
43         print(a.getArea());
44     }
45 }
```

test-stdlib-stringclassContains.dice.out

```
1 true
```

test-arith8.dice

```
1  class test {  
2      public void main(char[] [] args) {  
3          print(15+5.0);  
4          print("\n");  
5          print(1.5+1);  
6      }  
7  }
```

test-array4.dice.out

1 2

test-stdlib-copy.dice

```
1  include("stdlib");
2
3  class Two {
4      public void main(char[] [] args) {
5          class String b = new String("phil");
6          class String c = new String("khal");
7          class String d = c.copy(c);
8          print(b.string());
9          print(c.string());
10         print(d.string());
11     }
12 }
```


test-arith7.dice.out

1 3

test-classFunctionOverload1.dice

```
1  class shape {
2      public int xCoord;
3      public int yCoord;
4
5      constructor(){
6          this.xCoord = 0;
7          this.yCoord = 0;
8      }
9
10     constructor(int x, int y){
11         this.xCoord = x;
12         this.yCoord = y;
13     }
14
15     public int getArea(){
16         return 10;
17     }
18 }
19
20 class circle extends shape {
21     public int radius;
22
23     constructor(){
24         this.radius = 0;
25     }
26     constructor(int r){
27         this.radius = r;
28     }
29     constructor(int x, int y){
30         this.radius = 0;
31         this.xCoord = x;
32         this.yCoord = y;
33     }
34
35     public int getArea(){
36         return 3*this.radius*this.radius;
37     }
38 }
39
40 class test {
41     public void main(char[] [] args) {
42         class shape a = new shape(0,0);
43         print(a.getArea());
44     }
45 }
```

test-stdlib-stringclassContains.dice

```
1  include("stdlib");
2
3  class Two {
4      public void main(char[] [] args) {
5          class String b = new String("philkhal");
6          class String c = new String("khal");
7          print(b.contains(c));
8      }
9  }
```

test-factorialRecursive.dice.out

1 120

test-stdlib-integerclass2.dice

```
1 include("stdlib");
2
3 class Two {
4     public void main(char[] [] args) {
5         class Integer x = new Integer(128);
6         class String str = x.toString();
7         print(str.string(), "\n");
8     }
9 }
```

test-bool6.dice.out

```
1 truefalse
```

test-cyclicalIncludes.dice

```
1  include("Test Suite/Compiler_Test_Suite/test-cyclicalIncludes2.dice");
2
3  class test {
4      public void main(char[] [] args) {
5          class test2 a = new test2();
6          this.output();
7      }
8
9      public void output(){
10         print("a");
11     }
12 }
```


test-stdlib-stringclass3.dice

```
1  include("stdlib");
2
3  class test{
4
5      private class String x;
6
7      public void main(char[] [] args) {
8
9          class String a = new String("goodBye");
10         this.x = a;
11         print(this.x.string());
12
13     }
14 }
```

test-arith3.dice

```
1  (* Test left-to-right evaluation of expressions *)
2
3  class test {
4
5      public int a; (* Global variable *)
6
7      public int inca() {
8          this.a = this.a + 1;  (* Increment a; return its new value *)
9          return this.a;
10     }
11
12     public void main(char[][] args) {
13         this.a = 42;  (* Initialize a *)
14         print(this.inca() + this.a);
15     }
16 }
```

test-emptyBlock.dice

```
1  class test {
2      public void main(char[] [] args) {
3          {
4              (* Nothing in the following blocks*) {} {}
5          }
6          { null; }
7          print(1);
8      }
9  }
```

test-intOverflow.dice.out

1 passpass

test-classFunctionOverload.dice.out

1 12

test-exit.dice.out

1 1

test-if1.dice.out

1 4217

test-stdlib-stringclass2.dice

```
1  include("stdlib");
2
3  class test {
4      public void main(char[] [] args) {
5          class String s = new String("StringDoesn'tStartWithH");
6          print(s.string());
7      }
8  }
```


test-arith6.dice.out

1 50

test-stdlib-stringclassLength.dice

```
1  include("stdlib");
2
3  class Two {
4      public void main(char[] [] args) {
5          class String s = new String("123456789");
6          print(s.length());
7      }
8  }
```

test-stdlib-stringclassContains2.dice.out

1 false

test-ops1.dice

```
1  class test {
2      public void main(char[] [] args) {
3          print(1 + 2);
4          print(1 - 2);
5          print(1 * 2);
6          print(100 / 2);
7          print(99);
8          print(1 == 2);
9          print(1 == 1);
10         print(99);
11         print(1 != 2);
12         print(1 != 1);
13         print(99);
14         print(1 < 2);
15         print(2 < 1);
16         print(99);
17         print(1 <= 2);
18         print(1 <= 1);
19         print(2 <= 1);
20         print(99);
21         print(1 > 2);
22         print(2 > 1);
23         print(99);
24         print(1 >= 2);
25         print(1 >= 1);
26         print(2 >= 1);
27     }
28 }
```

test-stdlib-stringclass.dice

```
1  include("stdlib");
2
3  class Two {
4      public void main(char[] [] args) {
5          class String s = new String("hi");
6          print(s.string());
7      }
8  }
```

test-arith2.dice

```
1  class test {  
2      public void main(char[] [] args) {  
3          print(1 + 2 * 3 + 4);  
4      }  
5  }
```

test-float-max.dice

```
1  class test {  
2      public void main(char[] [] args) {  
3          float a = 0.01175494;  
4          float b = 1010123.45;  
5          print(a);  
6          print("\n");  
7          print(b);  
8      }  
9  }  
10 }
```

test-arith1.dice

```
1  class test {  
2      public void main(char[] [] args) {  
3          print(5+15);  
4      }  
5  }
```


test-stdlib-stringclass3.dice.out

```
1  goodBye
```

test-ifEmptyBlock2.dice

```
1  class test {  
2      public void main(char[] [] args) {  
3          if (false){}  
4          else {}  
5          print(17);  
6      }  
7  }
```

test-array3.dice.out

1 1102

test-arithSigned4.dice

```
1  class test {
2      public void main(char[] [] args) {
3          print(-1-3);
4          print(1--3);
5          print(-1.0-3.0);
6          print(1.0--3.0);
7      }
8  }
```

test-classSetter.dice

```
1  class shape {
2      public int xCoord;
3      public int yCoord;
4
5      public void setX(int x){
6          this.xCoord = x;
7      }
8      public void setY(int y){
9          this.yCoord = y;
10     }
11
12 }
13
14 class test {
15     public void main(char[] [] args) {
16         class shape a = new shape();
17         a.setX(1);
18         a.setY(3);
19         print(a.xCoord);
20         print(a.yCoord);
21     }
22 }
```

test-classExtendsSetter.dice

```
1  class shape {
2      public int xCoord;
3      public int yCoord;
4
5      public void setX(int x){
6          this.xCoord = x;
7      }
8      public void setY(int y){
9          this.yCoord = y;
10     }
11
12 }
13
14 class circle extends shape {
15     public int radius;
16
17 }
18
19 class test {
20     public void main(char[] [] args) {
21         class circle a = new circle();
22         a.setX(1);
23         a.setY(3);
24         print(a.xCoord);
25         print(a.yCoord);
26     }
27 }
```

test-gcd.dice

```
1  class test {
2
3      public void main(char[] [] args) {
4          print(this.gcd(2,14));
5          print(this.gcd(3,15));
6          print(this.gcd(99,121));
7      }
8
9      public int gcd(int x, int y){
10         int a = x;
11         int b = y;
12         while (a != b) {
13             if (a > b)
14                 a = a - b;
15             else
16                 b = b - a;
17         }
18         return a;
19     }
20 }
```

test-bool7.dice

```
1  class test {
2      public void main(char[] [] args) {
3          print(true and true);
4          print(false and true);
5          print(true and false);
6          print(false and false);
7          print("\n");
8          print(true or true);
9          print(false or true);
10         print(true or false);
11         print(false or false);
12     }
13 }
```


test-classExtendsGetter.dice

```
1  class shape {
2      public int xCoord;
3      public int yCoord;
4
5      public int getX(){
6          return this.xCoord;
7      }
8      public int getY(){
9          return this.yCoord;
10     }
11
12 }
13
14 class circle extends shape {
15     public int radius;
16
17 }
18
19 class test {
20     public void main(char[] [] args) {
21         class circle a = new circle();
22         a.xCoord = 1;
23         a.yCoord = 3;
24         print(a.getX());
25         print(a.getY());
26     }
27 }
```

test-func4.dice.out

1 371

test-constructor1.dice.out

1 00510

test-fib.dice.out

1 112358

test-forBreak.dice.out

1 012100

test-func3.dice

```
1  class test {
2      public void main(char[] [] args) {
3          this.printem(42,17,192,8);
4      }
5
6      public void printem(int a, int b, int c, int d) {
7          print(a);
8          print(b);
9          print(c);
10         print(d);
11     }
12 }
```

test-scope.dice

```
1  class test {
2      public void main(char[] [] args) {
3          int a;
4          a = 1;
5          {
6              int b = 2;
7              {
8                  int c = 3;
9                  print(a);
10                 print(b);
11                 print(c);
12             }
13             print(b);
14         }
15         print(a);
16     }
17 }
```

test-objectDeclarationInheritance.dice

```
1  class A {}
2  class B extends A {}
3  class C extends B {}
4
5  class test {
6
7      public void main(char[] [] args) {
8          class A myCObj = new C();
9          print("pass");
10     }
11 }
```


test-bool9.dice

```
1  class test {  
2      public void main(char[] [] args) {  
3          print(true, true, true, false, false, true, true, false, "\n");  
4      }  
5  }
```

test-if8.dice.out

```
1 elseif
```

test-hello.dice.out

```
1 Hello, World!
```

test-fileio.dice

```
1  include("stdlib");
2
3  class Two {
4
5      public void main(char[] [] args) {
6          class File a = new File("Test Suite/Compiler_Test_Suite/test-fileio.dice", true);
7          char[] buf = a.readfile(243);
8          a.closefile();
9          print(buf);
10         }
11     }
```

test-arith3.dice.out

1 86

test-float-max.dice.out

```
1 0.011755
2 1010123.450000
```

test-var4.dice

```
1  class test {
2      public int a;
3
4      public void foo(int b) {
5          int c;
6          c = this.a;
7          print(c);
8          this.a = b;
9          print(this.a);
10     }
11
12     public void main(char[][] args) {
13         this.a = 12;
14         this.foo(42);
15     }
16 }
```

test-cyclicalIncludes2.dice.out

1 ba

test-classExtendsSetter.dice.out

1 13

test-bool4.dice

```
1  class test {  
2      public void main(char[] [] args) {  
3          print(1<=2);  
4          print(1<=1);  
5          print(1<=2.0);  
6          print(2.1<=2.0);  
7      }  
8  }
```

test-bool2.dice

```
1  class test {  
2      public void main(char[] [] args) {  
3          print(1>2);  
4          print(1.0>2);  
5          print(1>2.0);  
6          print(1.0>2.0);  
7      }  
8  }
```

test-classExtends.dice.out

1 1.500000

test-gcd.dice.out

1 2311

test-bool2.dice.out

```
1 falsefalsefalsefalse
```

test-func4.dice

```
1  class test {
2      public int a;
3
4      constructor() {}
5
6      public int inca() {
7          this.a = 124;
8          return this.a + 124;
9      }
10
11     public int add2(int x, int y) {
12         return x + y;
13     }
14
15     public void main(char[] [] args) {
16         class test b = new test();
17         print(b.add2(b.inca(), 123));
18     }
19 }
```

test-emptyBlock.dice.out

1 1

test-constructor2.dice.out

1 510

test-for2.dice

```
1  class test {
2      public void main(char[] [] args) {
3          int i;
4          for ( i = 5 ; i > 0 ; i = i - 1 )
5              print(i);
6          print(42);
7      }
8  }
```

test-array2.dice

```
1  class test {  
2      public void main(char[] [] args) {  
3          float[] a = |1.0,1.5,2.5,3.5,4.5|;  
4          print(a[1]);  
5          print(a[4]);  
6      }  
7  }
```

test-**constructorDefault.dice.out**

1 5

test-applicative.dice.out

1 132

test-stmts1.dice.out

1 4562

test-global1.dice

```
1  class test {
2      public int a;
3      public int b;
4
5      public void printa(){
6          print(this.a);
7      }
8
9      public void printb(){
10         print(this.b);
11     }
12
13     public void incab(){
14         this.a = this.a + 1;
15         this.b = this.b + 1;
16     }
17
18     public void main(char[] [] args) {
19         this.a = 42;
20         this.b = 21;
21         this.printa();
22         this.printb();
23         this.incab();
24         this.printa();
25         this.printb();
26     }
27 }
```

test-intOverflow.dice

```
1  class test {
2      public void main(char[] [] args) {
3          int a = 2147483648; (*More than an int can hold should overflow*)
4          if(a<2147483647){
5              print("pass");
6          }
7          else{
8              print(a);
9          }
10
11         int b = -2147483649; (*More than an int can hold should overflow*)
12         if(b>-2147483648){
13             print("pass");
14         }
15         else{
16             print(b);
17         }
18     }
19 }
```


test-if6.dice

```
1  class test {
2      public void main(char[] [] args) {
3          if (true){
4              if(true)
5                  print(42);
6                  print(27);
7          }
8          else
9              print(8);
10
11         if (false){
12             if(true)
13                 print(42);
14                 print(27);
15         }
16         else
17             print(8);
18     }
19 }
20 }
```

test-ifEmptyBlock.dice

```
1  class test {  
2      public void main(char[] [] args) {  
3          if (true){}  
4          print(17);  
5      }  
6  }
```

test-arith1.dice.out

1 20

test-arith4.dice.out

1 13105107

test-whileBreak.dice.out

1 543

test-classReturnObjects.dice

```
1  class shape {
2      public int xCoord;
3      public int yCoord;
4
5      constructor (){
6          this.xCoord = 1;
7          this.yCoord = 2;
8      }
9
10 }
11
12 class test {
13     public void main(char[] [] args) {
14         class shape a = this.returnMe();
15         print(a.xCoord);
16         print(a.yCoord);
17     }
18
19     public class shape returnMe(){
20         class shape b = new shape();
21         return b;
22     }
23 }
```

test-stdlib-stringclass2.dice.out

```
1 StringDoesn'tStartWithH
```

test-intMax.dice.out

```
1 2147483647
2 -2147483648
```


test-arith2.dice.out

1 11

test-for1.dice

```
1  class test {
2      public void main(char[] [] args) {
3          int i;
4          for (i = 0 ; i < 5 ; i = i + 1) {
5              print(i);
6          }
7          print(42);
8      }
9  }
```

test-bool3.dice

```
1  class test {
2      public void main(char[] [] args) {
3          print(1>=2);
4          print(1>=1);
5          print(1>=2.0);
6          print(2.0>=2.0);
7      }
8  }
```

test-arithSigned3.dice.out

1 2-22.000000-2.000000

test-arith8.dice.out

```
1 20.000000
2 2.500000
```

test-intMax.dice

```
1  class test {
2      public void main(char[] [] args) {
3          int a = 2147483647;
4          int b = -2147483648;
5          print(a);
6          print("\n");
7          print(b);
8      }
9  }
```

test-bool5.dice.out

```
1  falsetrue
```

test-args.dice.out

1 davidemilyphil4

test-args.dice

```
1  class test {  
2      public void main(char[] [] args) {  
3          print(args[1]);  
4          print(args[2]);  
5          print(args[3]);  
6          print(args.length);  
7      }  
8  }
```

E-test-cyclicalIncludesDuplicate.dice.out

```
1  Class test not found
```

E-test-objectCreation2.dice.out

1

E-test-scope3.dice

```
1  class test {  
2  
3      public void main(char[] [] args) {  
4          int x;  
5          for(x = 0; x < 3; x = x+1){  
6              int y = 10;  
7              print(y);  
8          }  
9          print(y);  
10     }  
11 }
```

E-test-objectCreation2.dice

```
1  class Bar {
2  constructor(char c, float f) {}
3  }
4
5  class Foo {
6  constructor(bool b, char c, float f) {}
7  constructor(int a, bool b, char c, float f) {}
8  }
9
10 class test {
11 public void main(char[] [] args) {
12 char myc = 'z';
13 float myf = 4.5;
14 class Bar myb = new Bar(myc, myf);
15 class Foo myFooObj = new Foo(5, true, myc, myf);
16 }
17 }
```

E-test-objectAssignMismatch.dice.out

```
1 Invalid assignment of B to C
```

E-test-cyclicalIncludes.dice.out

```
1  Class test not found
```

E-test-scope1.dice.out

```
1 Undefined id x
```


E-test-objectCreation1.dice.out

1

E-test-scope2.dice.out

```
1 Undefined id x
```

E-test-assignMismatch.dice.out

```
1 Invalid assignment of float to int
```

E-test-duplicate.dice

```
1  class test {  
2  public void main(char[] [] args) {  
3  char myc = 'z';  
4  int myc = 2;  
5  float myf = 4.5;  
6  }  
7  }
```

E-test-scope3.dice.out

```
1 Undefined id y
```

E-test-objectCreation4.dice

```
1  class Bar {
2  constructor(char c, float f) {}
3  constructor(bool b, char c, float f) {}
4  }
5  class Foo {
6  constructor(int a, bool b, char c, float f) {}
7  }
8  class test {
9  public void main(char[] [] args) {
10 char myc = 'z';
11 float myf = 4.5;
12 class Bar myb = new Bar(myc, myf);
13 class Foo myFooObj = new Foo(5, true, myc, myf);
14 }
15 }
```

E-test-constructor.dice

```
1  class Foo {
2  constructor(char c, float f) {}
3  constructor(bool b, char c, float f) {}
4  }
5
6  class test {
7  public void main(char[] [] args) {
8  int mya = 2;
9  bool myb = false;
10 char myc = 'z';
11 float myf = 3.5;
12 class Foo myFooObj = new Foo(mya, myb, myc, myf);
13 }
14 }
```

E-test-scope2.dice

```
1  class test {  
2  
3      public void main(char[] [] args) {  
4          if(true){  
5              int x = 10;  
6              print(x);  
7          }  
8          print(x);  
9      }  
10 }
```


E-test-constructor.dice.out

```
1 ConstructorFoo.constructor.int.bool.char.float not found
```

E-test-noReturn.dice

```
1  class test {  
2  
3      public int increment(int x){  
4          x = x+1;  
5      }  
6      public void main(char[] [] args) {  
7          int x = this.increment(5);  
8      }  
9  }
```

E-test-cyclicalIncludesDuplicate2.dice.out

```
1  Class test not found
```

E-test-objectCreation1.dice

```
1  class Bar {
2  constructor(char c, float f) {}
3  constructor(bool b, char c, float f) {}
4  }
5
6  class Foo {
7  constructor(bool a, int b) {}
8  constructor(int a, bool b, char c, float f) {}
9  }
10
11  class test {
12  public void main(char[] [] args) {
13  int mya = 2;
14  bool myb = false;
15  char myc = 'z';
16  float myf = 3.5;
17  class Foo myFooObj = new Foo(mya, myb, myc, myf);
18  }
19  }
```

E-test-cyclicalIncludes.dice

```
1 include("Test Suite/Compiler_Test_Suite/test-cyclicalIncludes.dice");
2
3 class test {
4     public void main(char[] [] args) {
5         this.output();
6     }
7
8     public void output(){
9         print("a");
10    }
11 }
```

E-test-undefinedClass2.dice

```
1  class Foo {}
2
3  class Bar {}
4
5  class test {
6  public void main(char[] [] args) {
7  class Baz b;
8  }
9  }
```

E-test-mainClassNotDefined.dice

```
1  class test{  
2  
3  }
```

E-test-privateFieldsAccess.dice

```
1  class shape {
2      private int area;
3
4      constructor(){
5          this.area = 100;
6      }
7
8      public void setArea(int x){
9          this.area = x;
10     }
11
12     public int getArea(){
13         return this.area;
14     }
15 }
16
17
18 class test {
19     public void main(char[][] args) {
20         class shape a = new shape();
21         a.area = 50;
22
23     }
24 }
```


E-test-duplicate.dice.out

```
1 Duplicate local variable defined myc
```

E-test-stdlib-overload.dice.out

```
1  Cannot use name print because it is reserved
```

E-test-noReturn.dice.out

```
1 Non-void function test.increment does not end in return
```

E-test-undefinedClass.dice

```
1  class D {  
2      public void main(char[] [] args) {}  
3  }  
4  class A extends B {}  
5  class B extends C {}  
6  class C extends D {}  
7  class G extends H {}  
8  class I extends H {}
```

E-test-objectAssignMistmatch.dice

```
1  class A {}
2  class B extends A {}
3  class C {}
4  class test {
5  public void main(char[] [] args) {
6  class A myBObj = new B();
7    class B mySecondBObj = new C();
8  }
9  }
```

E-test-privateFunctionAccess.dice.out

```
1  Cannot access private function something.hi in scope something from object test
```

E-test-objectCreation3.dice.out

1

E-test-objectCreation3.dice

```
1  class Foo {}
2
3  class Baz {}
4
5  class test {
6  public void main(char[] [] args) {
7    class Baz b;
8  }
9  }
```


E-test-privateFieldsAccess.dice.out

```
1  Cannot access private field area in scope shape from object test
```

E-test-assignMismatch2.dice.out

```
1 Invalid assignment of int to float
```

E-test-scope1.dice

```
1  class test {  
2  
3      public void main(char[] [] args) {  
4          {  
5              int x = 10;  
6              print(x);  
7          }  
8          print(x);  
9      }  
10 }
```

E-test-stdlib-overload.dice

```
1  class test {  
2  
3      public void print(){  
4  
5      }  
6  
7  public void main(char[] [] args) {  
8  
9      }  
10 }
```

E-test-objectCreation4.dice.out

1

E-test-cyclicalIncludesDuplicate.dice

```
1  include("Test
   ↪ Suite/Compiler_Test_Suite/Exceptions/E-test-cyclicalIncludesDuplicate2.dice");
2
3  class test {
4      public void main(char[] [] args) {
5
6      }
7  }
```

E-test-undefinedClass2.dice.out

```
1 Undefined class Baz
```

E-test-cyclicalIncludesDuplicate2.dice

```
1  include("Test
   ↪ Suite/Compiler_Test_Suite/Exceptions/E-test-cyclicalIncludesDuplicate.dice");
2
3  class test {
4
5  }
```


E-test-assignMismatch2.dice

```
1  class test {  
2      public void main(char[] [] args) {  
3          int a;  
4          a = 1.0;  
5          print(a);  
6      }  
7  }
```

E-test-privateFunctionAccess.dice

```
1  class shape {
2  }
3
4  class something {
5
6      private void hi(){
7      }
8  }
9
10 class test {
11     public void main(char[] [] args) {
12         class something a = new something();
13         a.hi();
14     }
15 }
```

E-test-constructor1.dice

```
1  class shape {
2      public int xCoord;
3      public int yCoord;
4
5      constructor(int x, int y){
6          xCoord = 0;
7          yCoord = 0;
8      }
9
10     constructor(int x, int y){
11         xCoord = x;
12         yCoord = y;
13     }
14 }
15
16 class test {
17     public void main(char[][] args) {
18         (* Constructor clash *)
19     }
20 }
```

E-test-assignMismatch.dice

```
1  class test {  
2      public void main(char[] [] args) {  
3          float a;  
4          a = 1;  
5          print(a);  
6      }  
7  }
```

E-test-mainClassNotDefined.dice.out

```
1 Main not found in program
```

E-test-undefinedClass.dice.out

```
1 Undefined class H
```

E-test-constructor1.dice.out

1 Duplicate constructor found

test_pretty.dice

```
1  class test {
2      public void main (char[][] args) {
3          print("Hello World");
4      }
5  }
```


test.dice

```
1  class test {  
2      public void main(char[] [] args) {  
3          print("Hello World");  
4      }  
5  }
```

primitives.dice

```
1  class testPrims {
2      public int a;
3      public float b;
4      private char c;
5      private bool d;
6      public void main(char[] [] args) {
7          int e;
8          float f;
9          char g;
10         bool h;
11         a = -2147483648;
12         e = 2147483647;
13         b = 1.0;
14         f = 2.222222;
15         c = '0';
16         g = '\t';
17         d = true;
18         h = false;
19     }
20 }
```

test_pretty.dice.ManualTokens

```
1 1. CLASS ID(test) LBRACE
2 2. PUBLIC VOID ID(main) LPAREN CHAR LBRACKET RBRACKET LBRACKET RBRACKET ID(args) RPAREN
   ↪ LBRACE
3 3. ID(print) LPAREN STRING_LITERAL(Hello World) RPAREN SEMI
4 4. RBRACE
5 5. RBRACE
6 6. EOF
```

primitives.dice.ManualTokens

```
1 1. CLASS ID(testPrims) LBRACE
2 2. PUBLIC INT ID(a) SEMI
3 3. PUBLIC FLOAT ID(b) SEMI
4 4. PRIVATE CHAR ID(c) SEMI
5 5. PRIVATE BOOL ID(d) SEMI
6 6. PUBLIC VOID ID(main) LPAREN CHAR LBRACKET RBRACKET LBRACKET RBRACKET ID(args) RPAREN
   ↪ LBRACE
7 7. INT ID(e) SEMI
8 8. FLOAT ID(f) SEMI
9 9. CHAR ID(g) SEMI
10 10. BOOL ID(h) SEMI
11 11. ID(a) ASSIGN MINUS INT_LITERAL(2147483648) SEMI
12 12. ID(e) ASSIGN INT_LITERAL(2147483647) SEMI
13 13. ID(b) ASSIGN FLOAT_LITERAL(1.) SEMI
14 14. ID(f) ASSIGN FLOAT_LITERAL(2.222222) SEMI
15 15. ID(c) ASSIGN CHAR_LITERAL(0) SEMI
16 16. ID(g) ASSIGN CHAR_LITERAL(\t) SEMI
17 17. ID(d) ASSIGN TRUE SEMI
18 18. ID(h) ASSIGN FALSE SEMI
19 19. RBRACE
20 20. RBRACE EOF
```

test.dice.ManualTokens

```
1 1. CLASS ID(test) LBRACE
2 2. PUBLIC VOID ID(main) LPAREN CHAR LBRACKET RBRACKET LBRACKET RBRACKET ID(args) RPAREN
   ↪ LBRACE
3 3. ID(print) LPAREN STRING_LITERAL(Hello World) RPAREN SEMI
4 4. RBRACE
5 5. RBRACE EOF
```

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
190     ↪ or exit(by selecting a number):");
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 }
```

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.