Kazm
The Final Report

Katie Jooyoung Kim (`jk4534`) - Manager
Aapeli Vuorinen (`oav2108`) - Systems Architect
Zhonglin Yang (`zy2496`) - Language Guru
Molly McNutt (`mrm2234`) - Tester

"The easiest way to write a compiler is to compile it"
- Professor Stephen Edwards

29 December 2021

Authors listed in an order that makes their first names spell out 'Kazm'.

Contents

Introduction

The Kazm programming language is a statically and strongly typed programming language in the C-family. It is both a subset of the C programming language since it does not support all features of C, but also a superset in that it implements lightweight classes on top of structs through the concept of class methods. General purpose programming functionality such as input/output is provided through built-ins and the inclusion of existing C libraries.

The motivation to pursue Kazm was to bridge the gap between `C` and `C++`. As such, Kazm programmers will be alleviated from some of the headaches that are borne through legacy `C` conventions but will also avoid the complexities of `C++`.

Tutorial

Setting Up Your Environment

The easiest way to run the Kazm compiler is using Docker and the bundled helper. First clone the repository and enter the `kazm` directory, then compile the included `hello_world.kazm` file in the Kazm Docker container:

```
git clone git@github.com:aapeliv/kazm.git
cd kazm/
docker run --rm -it -v $(pwd):/home/kazm -w=/home/kazm aapeliv/plt
    ./compile.py --run hello_world.kazm
```

The helper can be run without the `--run` switch in which case the compiler will simply produce an executable but not run it.

Running The Test Script

When in the `Kazm` directory, you can run the test suite inside docker using the `test.py` test runner through the following command:

```
docker run --rm -it -v $(pwd):/home/kazm -w=/home/kazm aapeliv/plt
    ./test.py
```

This will run the comprehensive test suite and report the status of each individual test, as well as any deviations from the expected behavior. Included are tests that check both Kazm pass cases that should run as well as fail cases that should not compile. Failing cases are provided to ensure that the compiler catches errors such as when using an undeclared identifier or when there is a syntax error. The output generated on failing cases are compared to their complement .cerr file in `tests`. If there is no difference between, the case is considered to pass as we were able to generate the expected output.

Running A Single Test

A single test can be run by performing the following command. The example below will run on an example test file, `test-for.kazm`

```
docker run --rm -it -v $(pwd):/home/kazm -w=/home/kazm aapeliv/plt
    ./test.py --select test-for
```

The `--select` option can also be supplied a comma-separated list of test names in order to select multiple tests.

Cleaning Your Work Environment

If your `Kazm` directory gets cluttered with unnecessary files, we have provided a Makefile that you can run the following command to clean your workspace.

```
make clean
```

Viewing Intermediary Files

The following list of commands will show visibility into the LLVM IR and assembly files.

```
cd kazm
docker run --rm -it -v $(pwd):/home/kazm -w=/home/kazm aapeliv/plt
# now inside the container
opam config exec -- ocamlbuild -use-ocamlfind kazm.native
# run the kazm compiler to output LLVM IR
cat test-my_test.kazm | ./kazm.native > test-my_test.ll
# compile LLVM IR into assembly
llc --relocation-model=pic test-my_test.ll
```

Source File to Executable

The following list of commands will show visibility into the process from a Kazm source file to an executable.

```
cd kazm
docker run --rm -it -v $(pwd):/home/kazm -w=/home/kazm aapeliv/plt
# now inside the container
opam config exec -- ocamlbuild -use-ocamlfind kazm.native
# run the kazm compiler to output LLVM IR
cat test-my_test.kazm | ./kazm.native > test-my_test.ll
# compile LLVM IR into assembly
llc --relocation-model=pic test-my_test.ll
# compile builtins
cc -c builtins.c -o builtins.o
# compile test-my_test
```

```
cc -o test-my_test builtins.o test-my_test.s
# run!
./test-my_test
```

The Kazm Language Manual

Lexical Conventions

Kazm supports five types of tokens: identifiers, keywords, operators, literals, and delimiters. All white space, except for white space separating tokens or within strings, will be ignored. Comments will also be ignored.

Identifiers

A Kazm identifier is a sequence of one or more case sensitive ASCII letters, digits, or underscore '_' characters. Identifiers must not begin with a digit. Identifiers are used for class or variable types. Additionally, class types must begin with an uppercase alphabet character and variable names must begin with a lowercase alphabet character.

Keywords

The following are keywords in the language and cannot be used in other contexts: `int, double, void, bool, char, string, if, else, for, while, return, new, true, false, class, array, length`

Operators

Kazm supports the following operators:

- Arithmetic Operators: `+`, `-`, `*`, `/`
- Relational Operators: `==`, `!=`, `>`, `<`, `>=`, `<=`
- Logical Operators: `&&`, `||`, unary `!`
- Expression Operators: `[]`, `.`

Literals

Kazm literals represent a value of a primitive type. Kazm supports the following literals: int literal, double literal, bool literal, char literal, and string literal.

- `int` literals (32-bit): Sequence of digit characters 0 through 9 with an optional "`-`" in front to denote negative numbers. Ex: `100;` or `-100;`

- `double` literals (64-bit): Sequence of digit characters 0 through 9 which contain a decimal point and digits following the decimal point. Ex: `.9;` or `15.73;`
- `bool` literals (1-bit): denoted by the case sensitive constants: `true` and `false`
- char literals (8-bit): a character within single quotation marks (’). Ex: `’a’`
- `string` literal: a sequence of characters within double quotation marks (””). Ex: ”Hello, Kazm”

Delimiters

Comments: Single line comments will begin with // and multi-line comments begin with /* and end with */

White Space: All white space, except for white space separating tokens or within strings, will be ignored.

Delimiter Tokens: ( ) , ; { }

- Parentheses () will be used for mathematical expressions and delimit function arguments
- Curly Braces { } help with defining scope
- Semicolons (;) end statements as well as follow the right curly brace } after a class definition
- Commas (,) will separate function arguments and array elements

Types

Default Types

`bool, char, double, int, string`

- `bool` : A primitive data type that takes 1 byte of memory that takes either a `true` or `false` value. Can exist in the program as a boolean literal or as an identifier typed as `bool`. Examples: `true` or `bool a = false`;
- `char` : A primitive data type that takes 1 bit of memory. Can exist in the program as a character literal or as an identifier typed as char. Examples: `’a’` or `char a = ’a’`
- `double` : A primitive data type that takes 8 bytes of memory and follows standard double precision conventions. Can exist in the program as a double literal or as an identifier typed as double. Examples: `3.14` or `double x = 6.28`

- `int` : A primitive data type that takes 8 bytes of memory. Can exist in the program as an integer literal or as an identifier typed as int. Examples: `5` or `int a = 5`

- `string` : A built-in data type that represents a sequence of characters beginning and ending with double quotes (”). Can exist in the program as a string literal or as an identifier typed as `string`. Examples: `"Hi PLT"` or `string b = "Hello, World"`. The default string is ””.

The Array Data Structure

An array in Kazm is of fixed size. All elements in the array must be of the same type. An array declaration is formatted as follows:

```
array type[x] variable_name;
```

`type[]` indicates the type of the elements in the array. `x` is a place holder for an integer number which represents the length of the array. `variable_name` indicates the name of the array in the declaration.

Array Length

An array's length can be found using the dot `.` operator.

```
array int[5] foo = [1,2,3,4,5];
int x = foo.length;
int_println(x); // 5
```

Operators

Arithmetic Operators

Arithmetic Operators: `+`, `-`, `*`, `/`, %

Arithmetic operators are binary operators and are left to right associative. Addition and subtraction have lower precedence than multiplication and division.

The follow list will outline arithmetic operators and operands Kazm supports:

- `+` :
  - integer addition (returns `int`): `int + int`
  - double addition (returns `double`): `double + double`
- `-` :
  - integer subtraction (returns `int`): `int - int`
  - double subtraction (returns `double`): `double - double`

- `*` :
  - integer multiplication (returns `int`): `int * int`
  - double multiplication (returns `double`): `double * double`
- `/` :
  - integer division (returns `int`): `int / int`
  - double division (returns `double`): `double / double`
- % :
  - integer remainder (returns `int`): `int` % `int`

Relational Operators

Relational Operators: `==`, `!=`, `>`, `<`, `>=`, `<=`

Relational operators are left to right associative but have lower precedence than arithmetic operators. `==` and `!=` have a lower precedence than the other relational operators (which have the same precedence).

All of the relational operators are defined over integers and doubles.

Logical Operators

Logical Operators: `&&`, `||`, unary `!`

Logical operators have lower precedence than relational operators and can be binary (`&&` and `||`) operators or unary `!`.

`expr && expr` will return true if and only if both expressions operands are true, else false.

`expr || expr` will return true if at least one operand is true, else false.

`!expr` will return true if the resulting value of the `expr` is false itself, else false.

Expression Operators

Expression Operators: `[]`, `.`

These expression operators have highest precedence and are left associative.

The dot operator is used to access class member functions and variables.

The square bracket operator will allow users to index of an array.

Assignment Operator

Assignment Operator: `=`

`=` simply assigns an expression on the right to a variable on the left. If the type of the variable on the left and the expression on the right are not of the same type, an error will be thrown.

Expressions

Like C, Kazm expressions are values produced by a combination of operators and operands (variables, constants). An expression needs one or more operands and zero or more operators.

Declarations

All variables in Kazm must be declared before use. All declarations in Kazm associate a given identifier with a type. The type can be one of the default types or can be a user-defined type (i.e. class). While the type declaration is absolutely required, it is left optional for the user to initialize the identifier in the declaration. The value must be of the specified type. Each variable declaration must be preceded by the variable type. It is not possible to declare multiple variables in the same declaration.

Example declaration:

```
int a;
int b;
int c; // this is legal
char a, b, c; // this is not
```

Example declarations with and without initialization:

```
int a = 0;
array char[26] alphabet;
double pi = 3.14;
bool a = false;
```

Finally, users must remember to adhere to the lexical conventions outlined previously in the manual.

Statements

Kazm statements are Kazm code fragments that are executed in sequence.

Control Flow

if... else

In Kazm, the `if... else` statements are used to express decisions within the program. An `if` does not require an `else` but every `else` requires a parent `if`. There may also be multiple nested `if` statements. The form of a conditional is as follows:

`if` (expr) { statement } `else` { statement }

The expression is evaluated and if it is `true` then the statement following that expression is executed.

```
// nested if
if (true)
{
    if (true)
    {
        if (false) {
            println("false");
        } else {
            println("true");
        }
    }
}
```

for loops

The `for` statement allows a statement block to be executed a fixed number of times. The statement block must be enclosed within curly braces. Expr1 initializes the loop, expr2 is evaluated for each iteration of the loop, and expr3 is always executed at the end of each loop iteration. The loop runs while expr2 returns `true`. The syntax is as follows:

```
for (expr1; expr2; expr3) {
    statements
}
```

while loops

A `while` expression allows a statement block to be executed multiple times while a condition holds true. The statement block must be enclosed within curly braces. The syntax is as follows:

```
while (expr) {
    statement
}
```

Upon entering the while loop, expr is evaluated. If it is `true`, statement is executed and expr is reevaluated. This cycle continues until expr becomes `false` at which point execution resumes after the end of statement.

Note: Kazm does not support continue or break.

Functions

The execution of a program in Kazm requires one `main` function which takes no arguments and returns an int. The return of the `main` function may be omitted.

A Kazm file may contain additional functions which may return a type other than `int`. Functions that are not `main` may take in zero or more arguments. These arguments are passed by value.

Kazm does not support definitions of functions inside other functions.

Function Naming Convention and Definitions

Kazm function names, other than special member functions which will be discussed in the class section of the language reference manual, must begin with a lowercase alphabet character and can be followed by any number of lowercase or uppercase alphabet characters, numbers, and underscores _.

Functions will also have a return type, which precedes the name of the function. The return type can be `void`, any built-in types, or user-defined type. If no return is given, Kazm will supplement a return of the type's default value. Currently, Kazm does not support array and class return types.

Finally, a note on function parameters, all primitive types (`int, bool, char, double`

Built-In Functions

The following functions are Kazm built-in functions, implemented in C.

`void println(char *str)` print a string (newline)
`void print(char *str)` print a string (no newline)
`void int_println(int val)` print an int (newline)
`void int_print(int val)` print an int (no newline)
`void char_println(char val)` print a char (newline)
`void double_println(double val)` print a double (newline)
`void double_print(double val)` print a double (no newline)
`int next_int()` return an incremented global counter

Example Function: Add Two Integers

Example functions within an example program are as follows:

```
// add.kazm
int add(int x, int y) { // return type: int, name: add
  return x*y;
}

int main() {
    int a = 5;
    int b = add(a, 5);
    int_println(b); // prints 10
    return 0; // may be omitted
}
```

Scope

Kazm implements a sophisticated scoping mechanism that simplifies programming and tracks object lifetime.

A scope is generally defined by curly braces, and form a hierarchy from the largest scope (of a function or method) down to smaller scopes. The scope of a variable is the innermost scope where the variable was declared; and this is the largest scope in which the name can be used. Variables in smaller scopes cannot reuse the names of variables in parent scopes:

```
int main() {
  int a;
  {
    double a; // illegal
  }
}
```

Kazm tracks the lifetime of user-defined objects (see the next section for classes) using scope. When a class instance goes out of scope, it will be destroyed. This means that artificial scopes (those created by an extra pair of curly braces) can be used to manipulate the lifetime of objects:

```
class ScopeGreeter {
  ScopeGreeter() { println("Hi!"); }
  ~ScopeGreeter() { println("Bye!"); }
};

int main() {
  println("Top");
  {
    ScopeGreeter g;
    println("In scope");
```

```
    // g will be destroyed at the end of this scope, before the next
    println call
  }
  println("Bottom");
}
```

This program prints

```
Top
Hi!
In scope
Bye!
Bottom
```

Classes

A class is a user-defined structured object consisting of data members, class methods, as well as an optional constructor or destructor. This section of the report will incrementally walk through the definition of an example `class`: `Team_Project`

Class Definitions and Naming Convention

To define a `class`, Kazm utilizes syntax similar to `C++`. This requires that users enclose the class definition within curly braces { } and end their definition with a semicolon. Additionally, the name of the class must begin with an uppercase character. The rest of the characters in the name may be lowercase, uppercase, digit, or _ characters.

The following example declares an empty class:

```
class Team_Project { };
```

Data Members

A Kazm `class` may include zero or more data members, which are primitive types. Kazm does not yet support user-defiend class members. Class data members do no support access modifiers, all data members are public. Member functions as well as other parts of code outside of the class can manipulate these members. For the purpose of this example, our example class will include four data members (2 `int`s and 2 `string`s): `int num_of_members`, `double cap`, `string course_name`, and `string name`.

Using our example, we will update the class definition with data members as follows: `Team_Project` will be defined as:

```
class Team_Project {
  int num_of_members;
  double cap;
  string course_name;
  string name;
};
```

Class construction and `me`

During class initialization in Kazm, memory is allocated on the heap for a struct of the right size, and a pointer to it is stored in the variable name to be initialized. The contents of this allocated memory is undefined and the programmer ought to initialize all members variables before use.

To simplify this task, Kazm supports class constructors. A constructor of a class is a special class method whose job is to initialize and prepare the instance before it is used. The function definition of the constructor within the class has the same name (sensitive to cases) as the class itself and takes no arguments. If no constructor is defined for a class, then Kazm leaves the memory untouched, and the programmer must carefully initialize all members before use.

In Kazm, `me` is a keyword that allows referring to class members within class methods. `me` is syntactically equivalent to any other class instance.

Using our example, we will provide an example constructor. We will update the class definition with a default constructor as follows: `Team_Project` will be defined as:

```
class Team_Project {
  int num_of_members;
  double cap;
  string course_name;
  string name;

  Team_Project() {
    me.num_of_members = 5;
    me.cap = 1.0;
    me.course_name = "COMS4115: Programming Languages and
   Translators";
    me.name = "Aapeli";
  }
};
```

Class destruction

When a class goes out of scope, it is destroyed. This means that the memory allocated for the class is freed and the class will no longer be accessible. To simplify the management of classes, Kazm supports a destructor member function, which allows the programmer to cleanup any resources or perform cleanup actions for a class instance, knowing that they will be run before the memory is freed. Each class may have only one destructor. The destructor will be denoted by a ~ (tilde) in front of the name of the function, which will also be the case sensitive name of the class.

Using our example, we will update the class definition with a destructor as follows: `Team_Project` will be defined as:

```
class Team_Project {
  int num_of_members;
  double cap;
  string course_name;
  string name;

  Team_Project() {
    me.num_of_members = 5;
    me.cap = 1.0;
    me.course_name = "COMS4115: Programming Languages and
   Translators";
    me.name = "Aapeli";
  }

  ~Team_Project() {
    println("Goodbye from Team_Project");
  }
};
```

Class Methods

In addition to constructor and destructors, a class may consist of zero or more other member functions that may take zero or more arguments. The naming of these member functions follow the conventional variable and function naming conventions; namely, starting with a lowercase character.

For the purpose of this example, we will include a member function `get_info()` which will return a string. As such, we will update the class definition with a member function as follows: `Team_Project` will be defined as:

```
class Team_Project {
  int num_of_members;
  double cap;
```

```
  string course_name;
  string name;

  Team_Project() {
    me.num_of_members = 5;
    me.cap = 1.0;
    me.course_name = "COMS4115: Programming Languages and
   Translators";
    me.name = "Aapeli";
  }

  ~Team_Project() {
    println("Goodbye from Team_Project");
  }

  void print_info() {
    print(me.name);
    print(" is in the course: ");
    println(me.course_name);
  }
};
```

Using this in a program as follows:

```
int main() {
  Team_Project proj;
  proj.print_info();
}
```

Will print the following output:

```
Aapeli is in the course: COMS4115: Programming Languages and
   Translators
Goodbye from Team_Project
```

Class semantics

Classes in Kazm cannot be moved, reassigned or copied. We hoped to implement a deep-copy operation that would copy all data members recursively to allow assignment. We think this would work since there are no pointer types, all pointers are managed by us. In this way Kazm class variables would be similar to unique pointers in modern C++. However, we didn't have the time to complete this feature.

Classes can nonetheless be passed to functions as parameters (including methods of other classes), in which case they're "passed by reference" (implemented as pointers).

Project Plan: Management, Development, and Testing

Project Management

Throughout the semester, Kazm met once a week, either in-person or over zoom. During these meetings, group members discussed upcoming action items, scheduled timelines, and worked in a collaborative environment. All other communication regarding project related questions was conducted over Slack. Documents were shared on Overleaf for collaboration and presentation slides were shared over Google Drive. Kazm also met with John, our designated TA, to help understand best next steps to meet scheduled goals for the project.

Development

We used GitHub to facilitate development of Kazm. When developing, each member created a new branch which would later be merged with `develop` when that feature had finished being developed. This helped us avoid unnecessary conflicts and have clean work spaces throughout the semester.

Our team approached the project in an iterative manner. Our first deliverable was a first draft of Kazm's language reference manual. Next, our team focused on the creation of a simple 'Hello World' program in Kazm which required development on the scanner, parser, AST, and limited code generation. After these two milestones, our team divided the project based very loosely on our designated roles within the team but made sure to work closely together and facilitate smooth communication.

Development Tools

- Libraries and Languages : Python, C, OCaml
- Development Environments : Docker, vim, VS Code,

Testing

Each member contributed to creating test cases for Kazm, all of which can be found in the `tests` folder. We also used GitHub Actions to automatically run the tests on push to the GitHub repository. Our test cases include tests that are expected to pass as well as those expected to fail.

Individual Responsibilities

Due to extraneous priorities of each member throughout the semester, Kazm employed division of project responsibilities that transcended the official roles which were determined early on in the semester. At a high level, the project can be reduced to the following contributions:

- Aapeli: Codegen, Classes, Scopes, Test Runner, Docker, GitHub actions
- Zhonglin: Semantic analysis and SAST, Literals, strings
- Molly: Testing & Test Suite, Arrays (mostly support), Final Report
- Katie: Arrays, Final Report

Project Timeline of Major Milestones

- Sept 24 : Team Formation and Role Division
- Oct 31 : Language Reference Manual, Parser
- Nov 12 : Hello World
- Dec 29 : Final Report

The Purge

As the semester came to a close, we realized that we may have over promised certain features of the Kazm language. In order to work within our constraints, we decided to eliminate certain features in order to prioritize others. The first feature to go was tuples. Instead, we opted to implement arrays which was deemed to be easier to implement and also a more standard feature of a modern language, thus omission seemed inappropriate. Next to be cut were the following operators: `+=`, `-=`, `*=`, and `/=`. These were omitted from the language since our language offers a slightly more verbose syntax that achieves the same result. Many of these choices were made in order to prioritize Kazm classes, which proved to be more difficult to implement than expected.

Architecture Design

Scanner

The scanner (scanner.mll) takes a Kazm source file as input and generates a stream of tokens that are handled by the parser. At this stage, characters from the input file are compared to the recognized character(s) defined in the scanner. If any character from the input source file is invalid, the scanner will throw an error.

Parser

The parser (parser.mly) takes as input the stream of tokens produced by the scanner and creates an abstract syntax tree (AST) according to the context free grammar rules specified by the language. At this stage, if the program is syntactically incorrect, the parser will throw an error.

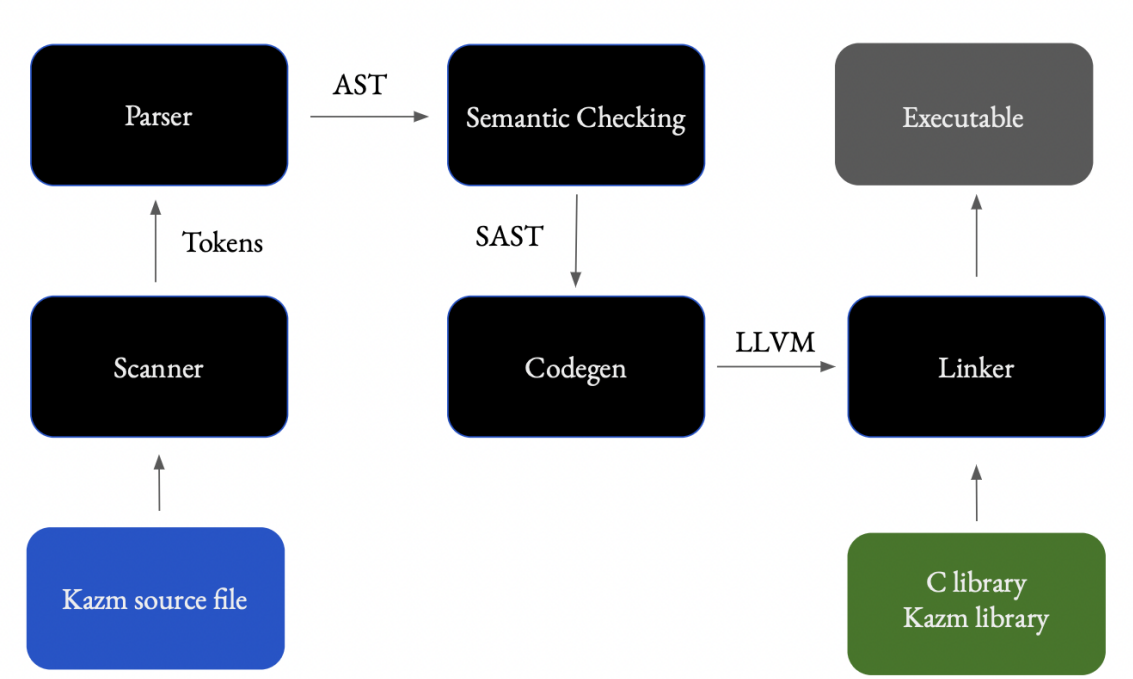

Figure 1: Diagram of Kazm's Translation Architecture

Semantic Checking

The semantic checker (checker.ml) checks the abstract syntax tree for semantic errors and creates a semantically checked abstract syntax tree (SAST). Such errors include incompatible type checking, functions with an incorrect function argument length, inappropriate scoping, and undefined classes. At this stage, if any of these errors occur, an exception error will be thrown and indicates what expression or statement causes the exception.

We opted to transform the AST into a SAST which includes the types of all expressions and statements as well as some other information. This allowed us flexibility in other parts of the compiler.

Code Generation

The code generator (codegen.ml) takes in the SAST and generates an LLVM IR module through the OCaml LLVM bindings. This IR is then dumped into a file and a helper script uses the LLVM compiler to compile it into an executable.

C and Kazm Libraries

To create the executable, we were able to utilize `C` functions through linking.

Testing Plan and Execution

The Kazm team used a python script to run all of our tests. This script creates a output log text file, loggy.txt, which logs the pass/fail status of each file; each file should pass. Loggy also has a accuracy rate at the bottom. Ideally, this would be 100.0%. The script also prints out each file, it's status, and the errors that are thrown on FAIL cases to terminal. Additionally, output is written (with pretty colors) to standard output.

Kazm's test cases are unit tests which helped us isolate the features of the language which were not working as expected. Test cases named test-*.kazm are cases that should compile and run as expected and fail-*.kazm cases are expected to fail.

The script expects that test-*.kazm files have a test-*.out file of the same name which contains the expected output of the program. If a file (fail-*.kazm) is expected to fail during compilation, the script expects a fail-*.cerr file of the same name which contains the error expected by the program. Similarly, a file which is expected to fail during runtime is expected to have a fail-*.err file of the same name which contains the expected error. The output of each test case is compared to the expected output (complementary .out file if the case is supposed to pass or .cerr file if it is supposed to fail). If there are no differences in the output and the expected .out or .cerr file, then the case is considered as passing. All of these files are within the tests directory.

There is also a tests.md file in the `PLT` directory which hosts an account of all of the tests that are within the `kazm/tests` directory. Here, we have divided the tests by whether they pass or fail, as well as which part of the language they are testing. In the tests.md file, a brief description for each test is included to clarify the intent of the source file (though, we wrote the test file names to be as descriptive as possible). The brief description from the tests.md file for each test matches the comment on the first line of the actual Kazm source file in `tests`.

Lessons Learned

Katie

I had an extremely high learning curve for the entirety of the contents of this course and the project (I changed majors). I learned everything from what is a compiler and what does it do to what the LLVM instructions mean and how to write OCaml. My previous development experience consisted of programs whose components could be considered logically separate and modified one by one. On the other hand, for this project, every addition of a feature required the modification and testing of all the related files at once. First ast.ml and sast.ml (to define what the feature being added is & aims to accomplish), then the parser.mly and the scanner.mll (to make the feature be recognized by the language), then checker.ml and codegen.ml (to actually accomplish the goal of the feature). While the learning curve for OCaml

itself made it very difficult for me to understand the errors initially, the testing suite with GitHub Actions made it much easier to address any issues the moment they appeared. Thanks to my teammates, I learned good software engineering practices as well as how to write a language.

Finally, I am extremely grateful of my wonderful teammates for being kind, fun, caring, intelligent, humble people who were always willing to help me understand and learn. I am very happy we got to know each other.

Aapeli

I really enjoyed the compiler part of the course as well as the functional programming part. With compilers, I found it really interesting to dig a bit into the LLVM project and understand how it works and how to make it useful, as well as thinking about how to implement common language features. I also found it quite revealing in that a language does not have to be super general and flexible to be very useful: OCaml is a bit quirky and has odd conventions but is still extremely powerful, similarly our language is meager but I can still see how it could be used for meaningful programs. After having done a project like this and making something somewhat useful, it's been interesting to see that even things as fundamental aren't that hard to create with modern tools and a bunch of thinking!

On the functional programming part, I really enjoyed finally writing some meaningful code in a functional language. I'd tinkered with FP before but this is the first time I really got to understand how to make it useful and leverage its strengths. Using OCaml's inferring type system was also a really interesting and educational experience. It took a while to get used to the (at first) cryptic error messages, but once you get used to it, it's very powerful and saves a lot of hassle. I even had a dream somewhere towards the end of the semester about constraint-based type systems. It's good stuff.

Finally, the project gave me a chance to think about team work, and how to use everyone's skills in a team to make a project happen; as well as how to deal with team frustrations and hopefully still have a fun atmosphere and get stuff done together!

Zhonglin

Through the course and the project, I have learned how to utilize OCaml language and other tools to write a new programming language. ocamllex and ocamlyacc made it easy for us to do the lexing and parsing part for our compiler. We didn't need to spend a lot time on figuring out how to handle shift-reduce conflicts when starting the project. And with OCaml LLVM bindings, we could write the compiler without knowing assembly instructions. The experience of writing this compiler was much different from my previous experience of writing a compiler which directly generate assembly codes. Other tools I learned were Docker, GitHub Actions, and test runners. My teammate set those up, and with these tools, we had a easy to development and testing environment.

Another thing I have learned is the importance of semantic checking for a compiler. I spent most of the time writing the semantic checker for our compiler. Two years ago, when I wrote my first compiler in Racket, I didn't know that some semantic checking could be done before generating assembly codes and how useful it could be. The semantic checking part seemed to be unnecessary for a compiler to work when I firstly tried to understand the MicroC compiler. However, when our group were developing our compiler, I found it was actually so handy. It could eliminate hidden bugs and help track where exceptions occurs.

Lastly, I have learned how team collaboration could make challenging tasks achievable. I am thankful of having three great teammates, Katie, Aapeli, and Molly. They always would like to help. With them, the large project was definitely less stressful. It has been full of enjoyment to work with them.

Molly

Throughout the semester, one of the more important technical paradigms that I gained exposure to is functional programming. I also realized that I do not really like it but I am thankful to have had the opportunity to learn it in this setting. In lecture, I liked learning more about compiler architecture as it is important to know but not something I would actively seek out on my own. Finally, I really enjoyed our discussion on parsing and believe it helped clarify the importance and relevance of context free grammars in programming that I had not picked up previously in CS Theory. Understanding the movement from scanner to codegen was also a really interesting aspect of the project and programming (and reprogramming) them was a good way to learn how design choices made previously would have downstream effects.

Regarding lessons that I learned throughout the course and project, I would say the most important takeaway I got from this project is that the people you work with are more important than the project itself. At the beginning of the semester, I think our team did a good job at trying to get to know more about each other in a non-academic setting. This was especially important because none of us had previously known each other. Socially, this would prove to be important later in the semester when competing priorities of each of us would temporarily distract us with work outside of Kazm. We were able to handle these sporadic absences well, using empathy and agility to keep the project on course.

Finally, as the only undergrad of the group, I learned that grad students are very smart.

Example Test Programs

Katie: Arrays

test-arr_all.kazm

```
int main()
{
    // array declaration without initialization
    array double[3] a;
    double_println(a[2]);

    // array declaration with initialization
    array int[5] b = [1, 2, 3, 4, 5];

    // array access and array assign
    int_println(b[0]);
    b[0] = b[4];
    int_println(b[0]);
    b[0] = 7;
    int_println(b[0]);

    // array length
    int_println(a.length);

    // output 0.000000 1 5 7 3
}
```

test-arr_all.ll

```
; ModuleID = 'kazm'
source_filename = "kazm"

declare void @print(i8*)

declare void @println(i8*)

declare void @int_print(i32)

declare void @int_println(i32)

declare void @double_print(double)

declare void @double_println(double)

declare void @char_println(i8)
```

```
declare i32 @next_int()

define i32 @main() {
entry:
  %malloccall = tail call i8* @malloc(i32 mul (i32 ptrtoint (i1**
   getelementptr (i1*, i1** null, i32 1) to i32), i32 3))
  %array_literal = bitcast i8* %malloccall to double**
  %array_ptr = bitcast double** %array_literal to double*
  %array_element = getelementptr inbounds double, double* %array_ptr
   , i32 0
  store double 0.000000e+00, double* %array_element
  %array_element1 = getelementptr inbounds double, double* %
   array_ptr, i32 1
  store double 0.000000e+00, double* %array_element1
  %array_element2 = getelementptr inbounds double, double* %
   array_ptr, i32 2
  store double 0.000000e+00, double* %array_element2
  %a = alloca double*
  store double* %array_ptr, double** %a
  %a__array = load double*, double** %a
  %a__element_ptr = getelementptr inbounds double, double* %a__array
   , i32 2
  %a__element = load double, double* %a__element_ptr
  call void @double_println(double %a__element)
  %malloccall3 = tail call i8* @malloc(i32 mul (i32 ptrtoint (i1**
   getelementptr (i1*, i1** null, i32 1) to i32), i32 5))
  %array_literal4 = bitcast i8* %malloccall3 to i32**
  %array_ptr5 = bitcast i32** %array_literal4 to i32*
  %array_element6 = getelementptr inbounds i32, i32* %array_ptr5,
   i32 0
  store i32 1, i32* %array_element6
  %array_element7 = getelementptr inbounds i32, i32* %array_ptr5,
   i32 1
  store i32 2, i32* %array_element7
  %array_element8 = getelementptr inbounds i32, i32* %array_ptr5,
   i32 2
  store i32 3, i32* %array_element8
  %array_element9 = getelementptr inbounds i32, i32* %array_ptr5,
   i32 3
  store i32 4, i32* %array_element9
  %array_element10 = getelementptr inbounds i32, i32* %array_ptr5,
   i32 4
  store i32 5, i32* %array_element10
  %b = alloca i32*
  store i32* %array_ptr5, i32** %b
```

```
  %b__array = load i32*, i32** %b
  %b__element_ptr = getelementptr inbounds i32, i32* %b__array, i32
   0
  %b__element = load i32, i32* %b__element_ptr
  call void @int_println(i32 %b__element)
  %b__array11 = load i32*, i32** %b
  %b__element_ptr12 = getelementptr inbounds i32, i32* %b__array11,
   i32 4
  %b__element13 = load i32, i32* %b__element_ptr12
  %b__array14 = load i32*, i32** %b
  %b__element_ptr15 = getelementptr inbounds i32, i32* %b__array14,
   i32 0
  store i32 %b__element13, i32* %b__element_ptr15
  %b__array16 = load i32*, i32** %b
  %b__element_ptr17 = getelementptr inbounds i32, i32* %b__array16,
   i32 0
  %b__element18 = load i32, i32* %b__element_ptr17
  call void @int_println(i32 %b__element18)
  %b__array19 = load i32*, i32** %b
  %b__element_ptr20 = getelementptr inbounds i32, i32* %b__array19,
   i32 0
  store i32 7, i32* %b__element_ptr20
  %b__array21 = load i32*, i32** %b
  %b__element_ptr22 = getelementptr inbounds i32, i32* %b__array21,
   i32 0
  %b__element23 = load i32, i32* %b__element_ptr22
  call void @int_println(i32 %b__element23)
  call void @int_println(i32 3)
  ret i32 0
}

declare noalias i8* @malloc(i32)
```

Aapeli: Classes

Zhonglin: Sorting Algorithm

test-sorting_arr.kazm

```
int main() {
        array int[6] my_arr = [5,3,0,2,1,4];
        int len = my_arr.length;
        int i = 0;
        while(i < len) {
                int j = 0;
```

```
                while(j < len - i - 1) {
                        if (my_arr[j] > my_arr[j + 1]) {
                                int temp = my_arr[j];
                                my_arr[j] = my_arr[j + 1];
                                my_arr[j + 1] = temp;
                        }
                        j = j + 1;
                }
                i = i + 1;
        }
        int n = 0;
        while(n < len) {
                int_print(my_arr[n]);
                n = n + 1;
        }
}
```

test-sorting_arr.ll

```
; ModuleID = 'kazm'
source_filename = "kazm"

declare void @print(i8*)

declare void @println(i8*)

declare void @int_print(i32)

declare void @int_println(i32)

declare void @double_print(double)

declare void @double_println(double)

declare void @char_println(i8)

declare i32 @next_int()

define i32 @main() {
entry:
  %malloccall = tail call i8* @malloc(i32 mul (i32 ptrtoint (i1**
    getelementptr (i1*, i1** null, i32 1) to i32), i32 6))
  %array_literal = bitcast i8* %malloccall to i32**
  %array_ptr = bitcast i32** %array_literal to i32*
  %array_element = getelementptr inbounds i32, i32* %array_ptr, i32
    0
```

```
  store i32 5, i32* %array_element
  %array_element1 = getelementptr inbounds i32, i32* %array_ptr, i32
    1
  store i32 3, i32* %array_element1
  %array_element2 = getelementptr inbounds i32, i32* %array_ptr, i32
    2
  store i32 0, i32* %array_element2
  %array_element3 = getelementptr inbounds i32, i32* %array_ptr, i32
    3
  store i32 2, i32* %array_element3
  %array_element4 = getelementptr inbounds i32, i32* %array_ptr, i32
    4
  store i32 1, i32* %array_element4
  %array_element5 = getelementptr inbounds i32, i32* %array_ptr, i32
    5
  store i32 4, i32* %array_element5
  %my_arr = alloca i32*
  store i32* %array_ptr, i32** %my_arr
  %len = alloca i32
  store i32 6, i32* %len
  %i = alloca i32
  store i32 0, i32* %i
  br label %start

start:                                            ; preds = %entry,
   %end
  %id41 = load i32, i32* %i
  %id42 = load i32, i32* %len
  %binop_res43 = icmp slt i32 %id41, %id42
  br i1 %binop_res43, label %loop, label %end40

loop:                                             ; preds = %start
  %j = alloca i32
  store i32 0, i32* %j
  br label %start6

start6:                                           ; preds = %loop, %
   join
  %id32 = load i32, i32* %j
  %id33 = load i32, i32* %len
  %id34 = load i32, i32* %i
  %binop_res35 = sub i32 %id33, %id34
  %binop_res36 = sub i32 %binop_res35, 1
  %binop_res37 = icmp slt i32 %id32, %binop_res36
  br i1 %binop_res37, label %loop7, label %end
```

```
loop7:                                            ; preds = %start6
  %id = load i32, i32* %j
  %my_arr__array = load i32*, i32** %my_arr
  %my_arr__element_ptr = getelementptr inbounds i32, i32* %
   my_arr__array, i32 %id
  %my_arr__element = load i32, i32* %my_arr__element_ptr
  %id8 = load i32, i32* %j
  %binop_res = add i32 %id8, 1
  %my_arr__array9 = load i32*, i32** %my_arr
  %my_arr__element_ptr10 = getelementptr inbounds i32, i32* %
   my_arr__array9, i32 %binop_res
  %my_arr__element11 = load i32, i32* %my_arr__element_ptr10
  %binop_res12 = icmp sgt i32 %my_arr__element, %my_arr__element11
  br i1 %binop_res12, label %take, label %dont_take

take:                                             ; preds = %loop7
  %id13 = load i32, i32* %j
  %my_arr__array14 = load i32*, i32** %my_arr
  %my_arr__element_ptr15 = getelementptr inbounds i32, i32* %
   my_arr__array14, i32 %id13
  %my_arr__element16 = load i32, i32* %my_arr__element_ptr15
  %temp = alloca i32
  store i32 %my_arr__element16, i32* %temp
  %id17 = load i32, i32* %j
  %binop_res18 = add i32 %id17, 1
  %my_arr__array19 = load i32*, i32** %my_arr
  %my_arr__element_ptr20 = getelementptr inbounds i32, i32* %
   my_arr__array19, i32 %binop_res18
  %my_arr__element21 = load i32, i32* %my_arr__element_ptr20
  %id22 = load i32, i32* %j
  %my_arr__array23 = load i32*, i32** %my_arr
  %my_arr__element_ptr24 = getelementptr inbounds i32, i32* %
   my_arr__array23, i32 %id22
  store i32 %my_arr__element21, i32* %my_arr__element_ptr24
  %id25 = load i32, i32* %temp
  %id26 = load i32, i32* %j
  %binop_res27 = add i32 %id26, 1
  %my_arr__array28 = load i32*, i32** %my_arr
  %my_arr__element_ptr29 = getelementptr inbounds i32, i32* %
   my_arr__array28, i32 %binop_res27
  store i32 %id25, i32* %my_arr__element_ptr29
  br label %join

dont_take:                                        ; preds = %loop7
  br label %join
```

```
join:                                             ; preds = %
    dont_take, %take
  %id30 = load i32, i32* %j
  %binop_res31 = add i32 %id30, 1
  store i32 %binop_res31, i32* %j
  br label %start6

end:                                              ; preds = %start6
  %id38 = load i32, i32* %i
  %binop_res39 = add i32 %id38, 1
  store i32 %binop_res39, i32* %i
  br label %start

end40:                                            ; preds = %start
  %n = alloca i32
  store i32 0, i32* %n
  br label %start44

start44:                                          ; preds = %end40,
    %loop45
  %id53 = load i32, i32* %n
  %id54 = load i32, i32* %len
  %binop_res55 = icmp slt i32 %id53, %id54
  br i1 %binop_res55, label %loop45, label %end52

loop45:                                           ; preds = %start44
  %id46 = load i32, i32* %n
  %my_arr__array47 = load i32*, i32** %my_arr
  %my_arr__element_ptr48 = getelementptr inbounds i32, i32* %
    my_arr__array47, i32 %id46
  %my_arr__element49 = load i32, i32* %my_arr__element_ptr48
  call void @int_print(i32 %my_arr__element49)
  %id50 = load i32, i32* %n
  %binop_res51 = add i32 %id50, 1
  store i32 %binop_res51, i32* %n
  br label %start44

end52:                                            ; preds = %start44
  ret i32 0
}

declare noalias i8* @malloc(i32)
```

Molly: Recursive Fibonacci

test-fib.kazm

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

int main() {
  int f = fib(10);

  int_println(f);

}
```

test-fib.ll

```
; ModuleID = 'kazm'
source_filename = "kazm"

declare void @print(i8*)

declare void @println(i8*)

declare void @int_print(i32)

declare void @int_println(i32)

declare void @double_print(double)

declare void @double_println(double)

declare void @char_println(i8)

declare i32 @next_int()

define i32 @fib(i32 %n) {
entry:
  %n_local = alloca i32
  store i32 %n, i32* %n_local
  %id = load i32, i32* %n_local
  %binop_res = icmp sle i32 %id, 1
  br i1 %binop_res, label %take, label %dont_take
```

```
take:                                             ; preds = %entry
  %id2 = load i32, i32* %n_local
  ret i32 %id2

dont_take:                                        ; preds = %entry
  br label %join

join:                                             ; preds = %
    dont_take
  %id3 = load i32, i32* %n_local
  %binop_res4 = sub i32 %id3, 1
  %0 = call i32 @fib(i32 %binop_res4)
  %id5 = load i32, i32* %n_local
  %binop_res6 = sub i32 %id5, 2
  %1 = call i32 @fib(i32 %binop_res6)
  %binop_res7 = add i32 %0, %1
  ret i32 %binop_res7
}

define i32 @main() {
entry:
  %0 = call i32 @fib(i32 10)
  %f = alloca i32
  store i32 %0, i32* %f
  %id = load i32, i32* %f
  call void @int_println(i32 %id)
  ret i32 0
}
```

Appendix: Listings

Code listing

Kazm

Contents of `kazm.ml`:

```
let _ =
  let lexbuf = Lexing.from_channel stdin in
  let ast = Parser.program Scanner.tokenize lexbuf in
  let sast = Checker.check ast in
  (* print_endline (Sast.string_of_sprogram sast) *)
  let m = Codegen.gen sast in
  print_endline (Llvm.string_of_llmodule m)
```

Scanner

Contents of `scanner.ml`:

```
{ open Parser }

let newline = '\n' | '\r' | "\r\n"
let digit = ['0'-'9']
let exp = 'e'['-''+']?['0'-'9']+
let double = (
    ((digit)+'.'(digit)* (exp)?) |
    ((digit)* '.'(digit)+(exp)?) |
    ((digit)+exp))
let singquote = '\''
let print_char = [' '-'~']


rule tokenize = parse
  [' ' '\t' '\r' '\n'] { tokenize lexbuf }
(* single line comment starts with // and carries to end of line *)
| "//" [^'\n']* { tokenize lexbuf }
(* multi line comment starts with /*)
| "/*" { multicomment lexbuf }
| '(' { PAREN_L }
| ')' { PAREN_R }
| '{' { BRACE_L }
| '}' { BRACE_R }
| "[]" { SQB_PAIR }
| '[' { SQB_L }
| ']' { SQB_R }
| '.' { DOT }
| ';' { SEMI }
| ',' { COMMA }
| '%' { MOD }
```

```
| '=' { ASSIGN  }
| '+' { PLUS }
| '-' { MINUS }
| '*' { TIMES }
| '/' { DIVIDE }
| "+=" { PLUSEQ }
| "-=" { MINUSEQ }
| "*+" { TIMESEQ }
| "/=" { DIVIDEQ }
| "&&" { AND }
| "||" { OR }
| '!' { NOT } (* "!" in C-Net *)
| "==" { EQ }
| "!=" { NEQ }
| '<' { LT }
| "<=" { LEQ }
| '>' { GT }
| ">=" { GEQ }
| "~" { TWIDDLE }
| "class" { CLASS }
| "void" { VOID }
| "array"  { ARRAY }
| "length" { LENGTH }
| "bool" { BOOL }
| "char" { CHAR }
| "int" { INT }
| "double" { DOUBLE }
| "string" {STRING}
| "if" { IF }
| "else" { ELSE }
| "for" { FOR }
| "while" { WHILE }
| "return" { RETURN }
| "true" { TRUE }
| "false" { FALSE }
(* | "\"" ([^'\"']+ as str) "\"" { STRING_LITERAL(str) } *)
| ['0'-'9']+ as int { INT_LITERAL(int_of_string int) }
| double as doublelit {DOUBLE_LITERAL(float_of_string doublelit)}
| ['a'-'z''_']['A'-'Z''a'-'z''_''0'-'9']* as str { IDENTIFIER(str) }
| ['A'-'Z']['a'-'z''A'-'Z''_''0'-'9']*  as classLit {
   CLASS_IDENTIFIER(classLit) }
| '"'   { STRING_LITERAL(parse_string (Buffer.create 100) lexbuf) }
| singquote print_char singquote as char   {CHAR_LITERAL(char.[1])}
| eof { EOF }
| _ as char     { raise (Failure("Undefined character: " ^ Char.
   escaped char)) }
```

```
and parse_string buffer = parse
  '"'                 { Buffer.contents buffer }
| newline             { Buffer.add_string buffer (Lexing.lexeme
   lexbuf); parse_string buffer lexbuf }
| [^ '"'  '\n' '\r']+ { Buffer.add_string buffer (Lexing.lexeme
   lexbuf); parse_string buffer lexbuf }
| eof                 { raise (Failure("Non-terminated double quotes
   ")) }

and multicomment = parse
"*/" { tokenize lexbuf }
| '\n' { Lexing.new_line lexbuf; multicomment lexbuf }
| eof { raise (Failure("reached end of file with an unclosed
   multiline comment"))}
| _ { multicomment lexbuf }
```

Parser

Contents of `parser.mly`:

```
/* Ocamlyacc parser for Kazm */
%{
open Ast
%}

%token PAREN_L PAREN_R BRACE_L BRACE_R SQB_L SQB_R SQB_PAIR /* ( ) {
    } [ ] */
%token DOT SEMI COMMA MOD ASSIGN TWIDDLE /* . ; , * % = ~ */
%token PLUS MINUS TIMES DIVIDE  /* + - * / */
%token PLUSEQ MINUSEQ TIMESEQ DIVIDEQ /* + - * / += -= *= /= */
%token AND OR NOT  /* && || ! */
%token EQ NEQ LT LEQ GT GEQ /* == != < <= > >= */
%token VOID BOOL CHAR INT DOUBLE STRING
%token IF ELSE FOR WHILE
%token RETURN
%token CLASS
%token ARRAY LENGTH
%token TRUE FALSE

%token<string> IDENTIFIER CLASS_IDENTIFIER
%token<string> CLASS_NAME
%token<string> STRING_LITERAL
%token<float> DOUBLE_LITERAL
%token<char> CHAR_LITERAL
%token<int> INT_LITERAL
```

```
/* %token<bool> BOOL_LITERAL */
%token EOF

%nonassoc NOELSE
%nonassoc ELSE
%nonassoc PAREN_L PAREN_R BRACE_L BRACE_R SQB_L SQB_R
%left SEMICO
%left IF
%right ASSIGN PLUSEQ MINUSEQ TIMESEQ DIVIDEQ
%left OR
%left AND
%left EQ NEQ
%left LT GT GEQ LEQ
%left PLUS MINUS
%left TIMES DIVIDE MOD
%right NOT

%left DOT

%start program
%type <Ast.program> program
%%

program:
  decls EOF { $1 }

decls:
    /* nothing */ { ([], []) }
  | decls fdecl {
    let (f, s) = $1 in
    (f @ [$2], s)
    }
  | decls cdecl {
    let (f, s) = $1 in
    (f, s @ [$2])
    }

fdecl:
   typ IDENTIFIER PAREN_L formals_opt PAREN_R BRACE_L stmts BRACE_R
     { { typ = $1;
         fname = $2;
         formals = List.rev $4;
         body = $7 } }

cdecl:
    CLASS CLASS_IDENTIFIER BRACE_L class_body BRACE_R SEMI {
```

```
      let (vars, mthds, constructors, destructors) = $4 in
      { cname = $2; cvars = vars; cmethods = mthds; cconstructors =
   constructors; cdestructors = destructors; }
    }

class_body:
    { ([], [], [], []) }
  | class_body var_decl {
   let (f, s, t, h) = $1 in
   (f @ [$2], s, t, h)
  }
  | class_body mdecl {
   let (f, s, t, h) = $1 in
   (f, s @ [$2], t, h)
  }
  | class_body constructor_decl {
   let (f, s, t, h) = $1 in
   (f, s, t @ [$2], h)
  }
  | class_body destructor_decl {
   let (f, s, t, h) = $1 in
   (f, s, t, h @ [$2])
  }

mdecl:
   typ IDENTIFIER PAREN_L formals_opt PAREN_R BRACE_L stmts BRACE_R
     { { typ = $1;
         fname = $2;
         formals = List.rev $4;
         body = $7 } }

constructor_decl:
   CLASS_IDENTIFIER PAREN_L formals_opt PAREN_R BRACE_L stmts
   BRACE_R
     { { typ = Void;
         fname = $1;
         formals = List.rev $3;
         body = $6 } }

destructor_decl:
   TWIDDLE CLASS_IDENTIFIER PAREN_L PAREN_R BRACE_L stmts BRACE_R
     { { typ = Void;
         fname = $2;
         formals = [];
         body = $6 } }
```

```
formals_opt:
    /* nothing */ { [] }
  | formal_list   { $1 }

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

typ:
    VOID { Void }
  | BOOL { Bool }
  | CHAR { Char }
  | INT { Int }
  | DOUBLE { Double }
  | STRING { String }
  | CLASS_IDENTIFIER { ClassT($1) }
  | ARRAY typ SQB_L INT_LITERAL SQB_R {ArrT($2, $4)}

var_decl:
    typ IDENTIFIER SEMI { ($1, $2) }

stmts:
    { [] }
  | stmts stmt { $1 @ [$2] }

stmt:
    expr SEMI { Expr $1 }
  | scope { $1 }
  | return_stmt SEMI { $1 }
  | if_stmt { $1 }
  | while_stmt { $1 }
  | for_stmt { $1 }
  | var_decl_stmt SEMI { $1 }

scope:
    BRACE_L block_stmt BRACE_R { StmtScope($2) }

block_stmt:
    stmts { Block($1) }

return_stmt:
    RETURN expr { Return $2 }
  | RETURN { EmptyReturn }

if_stmt:
    IF PAREN_L expr PAREN_R scope ELSE scope { If($3, $5, $7) }
```

```
  | IF PAREN_L expr PAREN_R scope { If($3, $5, Block([])) }

while_stmt:
    WHILE PAREN_L expr PAREN_R scope { While($3, $5) }

for_stmt:
    FOR PAREN_L expr SEMI expr SEMI expr PAREN_R scope { For($3, $5,
    $7, $9) }

var_decl_stmt:
    typ IDENTIFIER { Initialize(($1, $2), None) }
  | typ IDENTIFIER ASSIGN expr { Initialize(($1, $2), Some $4) }

expr:
    INT_LITERAL        { Literal($1) }
  | STRING_LITERAL     { StringLit($1) }
  | DOUBLE_LITERAL     { Dliteral(string_of_float $1)}
  | CHAR_LITERAL       { CharLit(Char.escaped $1)}
  | TRUE               { BoolLit(true) }
  | FALSE              { BoolLit(false) }
  | expr PLUS   expr   { Binop($1, Add,   $3)   }
  | expr MINUS  expr   { Binop($1, Sub,   $3)   }
  | expr TIMES  expr   { Binop($1, Mult,  $3)   }
  | expr DIVIDE expr   { Binop($1, Div,   $3)   }
  | expr MOD expr      { Binop($1, Mod,   $3)   }
  | expr EQ     expr   { Binop($1, Equal, $3)   }
  | expr NEQ    expr   { Binop($1, Neq,   $3)   }
  | expr LT     expr   { Binop($1, Less,  $3)   }
  | expr LEQ    expr   { Binop($1, Leq,   $3)   }
  | expr GT     expr   { Binop($1, Greater, $3) }
  | expr GEQ    expr   { Binop($1, Geq,   $3)   }
  | expr AND    expr   { Binop($1, And,   $3)   }
  | expr OR     expr   { Binop($1, Or,    $3)   }
  | NOT expr           { Unop(Not, $2) }
  | MINUS expr %prec NOT { Unop(Neg, $2)      }
  | PAREN_L expr PAREN_R { $2 }
  | fq_identifier ASSIGN expr { Assign($1, $3) }
  | fq_identifier PAREN_L expr_list PAREN_R { Call($1, $3) }
  | fq_identifier      { Id($1) }
  | SQB_L expr_list SQB_R { ArrayLit($2) }
  | IDENTIFIER SQB_L expr SQB_R { ArrayAccess($1, $3) }
  | IDENTIFIER SQB_L expr SQB_R ASSIGN expr { ArrayAssign($1, $3, $6
   ) }
  | IDENTIFIER DOT LENGTH { ArrayLength($1) }

fq_identifier:
```

```
    IDENTIFIER { [$1] }
  | IDENTIFIER DOT IDENTIFIER { $1::$3::[] }

expr_list:
    { [] }
  | expr_list COMMA expr { $1 @ [$3] }
  | expr { [$1] }
```

AST

Contents of `ast.ml`:

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

type uop = Neg | Not

type class_t = string
type typ = Int | Bool | Double | Void | String | Char | ClassT of
   class_t | ArrT of typ * int
type bind = typ * string

type ref = string list

type expr =
    Literal of int
  | Dliteral of string
  | BoolLit of bool
  | StringLit of string
  | CharLit of string
  | Id of ref
  | Binop of expr * op * expr
  | Unop of uop * expr
  | Assign of ref * expr
  | Call of ref * expr list
  | Noexpr
  | ArrayLit of expr list
  | ArrayAccess of string * expr
  | ArrayAssign of string * expr * expr
  | ArrayLength of string

type stmt =
    Block of stmt list
  | StmtScope of stmt
  | Expr of expr
```

```
  | Return of expr
  | If of expr * stmt * stmt
  | For of expr * expr * expr * stmt
  | While of expr * stmt
  | EmptyReturn
  | Initialize of bind * expr option

type func_decl = {
    typ : typ;
    fname : string;
    formals : bind list;
    body : stmt list;
}

type class_decl = {
    cname : class_t;
    cvars : bind list;
    cmethods : func_decl list;
    cconstructors : func_decl list;
    (* We'll check there is one and only one of these later *)
    cdestructors : func_decl list;
}

type program = func_decl list * class_decl list

(* TODO: maybe fix later *)
let string_of_op = function
    Add -> "+"
  | Sub -> "-"
  | Mult -> "*"
  | Div -> "/"
  | Equal -> "=="
  | Neq -> "!="
  | Less -> "<"
  | Leq -> "<="
  | Greater -> ">"
  | Geq -> ">="
  | And -> "&&"
  | Or -> "||"
  | Mod -> "%"


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

```
let string_of_typ = function
    Int -> "int"
  | Bool -> "bool"
  | Double -> "double"
  | Void -> "void"
  | String -> "string"
  | Char -> "char"
  | ClassT(name) -> "class " ^ name
  | ArrT(t, l) -> "array"

let rec string_of_expr = function
    Literal(l) -> string_of_int l
  | Dliteral(l) -> l
  | BoolLit(true) -> "true"
  | BoolLit(false) -> "false"
  | StringLit(s) -> "\"" ^ s ^ "\""
  | CharLit(c) -> "\'" ^ c ^ "\'"
  | Id(s) -> String.concat ", " s
  | Binop(e1, o, e2) ->
      string_of_expr e1 ^ " " ^ string_of_op o ^ " " ^
   string_of_expr e2
  | Unop(o, e) -> string_of_uop o ^ string_of_expr e
  | Assign(v, e) -> String.concat " " v ^ " = " ^ string_of_expr e
  | Call(f, el) ->
      String.concat " " f ^ "(" ^ String.concat ", " (List.map
   string_of_expr el) ^ ")"
  | Noexpr -> ""
  | ArrayLit(values) -> "values"
  | ArrayAccess(name, value) -> name
  | ArrayAssign(name, index, value) -> string_of_expr index
  | ArrayLength(name) -> "length"

let rec string_of_stmt = function
    Block(stmts) ->
      "{\n" ^ String.concat "" (List.map string_of_stmt stmts) ^ "}\
   n"
  | StmtScope(stmt) ->
    string_of_stmt stmt
  | Expr(expr) -> string_of_expr expr ^ ";\n";
  | Return(expr) -> "return " ^ string_of_expr expr ^ ";\n";
  | EmptyReturn -> "return;\n";
  | If(e, s, Block([])) -> "if (" ^ string_of_expr e ^ ")\n" ^
   string_of_stmt s
  | If(e, s1, s2) ->  "if (" ^ string_of_expr e ^ ")\n" ^
      string_of_stmt s1 ^ "else\n" ^ string_of_stmt s2
  | For(e1, e2, e3, s) ->
```

```
      "for (" ^ string_of_expr e1  ^ " ; " ^ string_of_expr e2 ^ " ;
    " ^
      string_of_expr e3  ^ ") " ^ string_of_stmt s
  | While(e, s) -> "while (" ^ string_of_expr e ^ ") " ^
   string_of_stmt s


let string_of_vdecl (t, id) = string_of_typ t ^ " " ^ id ^ ";\n"

let string_of_fdecl fdecl =
  string_of_typ fdecl.typ ^ " " ^
  fdecl.fname ^ "(" ^ String.concat ", " (List.map snd fdecl.formals
   ) ^
  ")\n{\n" ^
  String.concat "" (List.map string_of_stmt fdecl.body) ^
  "}\n"

let string_of_cdecl cdecl =
  "class" ^ " " ^ cdecl.cname ^ " " ^
  cdecl.cname ^ "\n{\n" ^ String.concat ", " (List.map
   string_of_vdecl cdecl.cvars) ^
  ")\n{\n" ^
  String.concat "" (List.map string_of_fdecl cdecl.cmethods) ^
  "}\n"

let string_of_program (vars, funcs, classes) =
  String.concat "" (List.map string_of_vdecl vars) ^ "\n" ^
  String.concat "\n" (List.map string_of_fdecl funcs) ^ "\n" ^
  String.concat "\n" (List.map string_of_cdecl classes)
```

SAST

Contents of `sast.ml`:

```
(* Semantically-checked AST and functions for printing it *)

open Ast

type sref = (typ * string) list

type sexpr = typ * sx
and sx =
    SLiteral of int
  | SDliteral of string
  | SBoolLit of bool
  | SCharLit of string
```

```
  | SStringLit of string
  | SId of ref
  | SBinop of sexpr * op * sexpr
  | SUnop of uop * sexpr
  | SAssign of ref * sexpr
  | SCall of ref * sexpr list
  | SNoexpr
  | SArrayLit of typ * sx list
  | SArrayAccess of string * sexpr
  | SArrayAssign of string * sexpr * sexpr
  | SArrayLength of string

type sstmt =
    SBlock of sstmt list
  | SExpr of sexpr
  | SReturn of sexpr
  | SIf of sexpr * sstmt * sstmt
  | SFor of sexpr * sexpr * sexpr * sstmt
  | SWhile of sexpr * sstmt
  | SEmptyReturn
  | SInitialize of bind * sexpr option

type sfunc_decl = {
    styp : typ;
    sfname : string;
    sformals : bind list;
    sbody : sstmt list;
}

type sclass_decl = {
    scname : class_t;
    scvars : bind list;
    scmethods : sfunc_decl list;
    scconstructors : sfunc_decl list;
    scdestructor : sfunc_decl option;
}

type sprogram = sfunc_decl list * sclass_decl list

(* Pretty-printing functions *)
(* TODO: maybe fix later *)
let rec string_of_sexpr (t, e) =
  "(" ^ string_of_typ t ^ " : " ^ (match e with
    SLiteral(l) -> string_of_int l
  | SBoolLit(true) -> "true"
  | SBoolLit(false) -> "false"
```

```
  | SCharLit c -> "\'" ^ c ^ "\'"
  | SStringLit s -> "\"" ^ s ^ "\""
  | SDliteral(l) -> l
  (* | SId(s) -> String.concat ", " s *)
  | SBinop(e1, o, e2) ->
      string_of_sexpr e1 ^ " " ^ string_of_op o ^ " " ^
    string_of_sexpr e2
  | SUnop(o, e) -> string_of_uop o ^ string_of_sexpr e
  (* | SAssign(v, e) -> v ^ " = " ^ string_of_sexpr e *)
  (* | SCall(f, el) ->
      f ^ "(" ^ String.concat ", " (List.map string_of_sexpr el) ^
    ")" *)
  | SNoexpr -> ""
                 ) ^ ")"

let rec string_of_sstmt = function
    SBlock(stmts) ->
      "{\n" ^ String.concat "" (List.map string_of_sstmt stmts) ^ "
    }\n"
  | SExpr(expr) -> string_of_sexpr expr ^ ";\n";
  | SReturn(expr) -> "return " ^ string_of_sexpr expr ^ ";\n";
  | SIf(e, s, SBlock([])) ->
      "if (" ^ string_of_sexpr e ^ ")\n" ^ string_of_sstmt s
  | SIf(e, s1, s2) ->  "if (" ^ string_of_sexpr e ^ ")\n" ^
      string_of_sstmt s1 ^ "else\n" ^ string_of_sstmt s2
  | SFor(e1, e2, e3, s) ->
      "for (" ^ string_of_sexpr e1  ^ " ; " ^ string_of_sexpr e2 ^ "
    ; " ^
      string_of_sexpr e3  ^ ") " ^ string_of_sstmt s
  | SWhile(e, s) -> "while (" ^ string_of_sexpr e ^ ") " ^
    string_of_sstmt s
  | SEmptyReturn -> "return;"

let string_of_sfdecl fdecl =
  string_of_typ fdecl.styp ^ " " ^
  fdecl.sfname ^ "(" ^ String.concat ", " (List.map snd fdecl.
    sformals) ^
  ")\n{\n" ^
  String.concat " " (List.map string_of_sstmt fdecl.sbody) ^
  "}\n"

let string_of_sprogram (vars, funcs) =
  String.concat "" (List.map string_of_vdecl vars) ^ "\n" ^
  String.concat "\n" (List.map string_of_sfdecl funcs)
```

Semantic analyzer

Contents of `checker.ml`:

```
open Ast
open Sast

module StringMap = Map.Make(String)

(* Semantic checking of the AST. Returns an SAST if successful,
   throws an exception if something is wrong.
   Check each function and each class *)

let check (functions, classes) =

  (* Verify a list of bindings has no void types or duplicate names
   *)
  let check_binds (kind : string) (binds : bind list) =
    List.iter (function
    (Void, b) -> raise (Failure ("illegal void " ^ kind ^ " " ^ b))
      | _ -> ()) binds;
    let rec dups = function
        [] -> ()
      |    ((_,n1) :: (_,n2) :: _) when n1 = n2 ->
      raise (Failure ("duplicate " ^ kind ^ " " ^ n1))
      | _ :: t -> dups t
    in dups (List.sort (fun (_,a) (_,b) -> compare a b) binds)
  in


  (* Collect function declarations for built-in functions: no bodies
    *)
  let built_in_decls =
    let add_bind map (name, ty) = StringMap.add name {
      typ = Void;
      fname = name;
      formals = [(ty, "x")];
      body = [] } map
    in let smap =  StringMap.add "next_int" {
      typ = Int;
      fname = "next_int";
      formals = [];
      body = [] } StringMap.empty
    in List.fold_left add_bind smap [ ("print", String);
                                ("println", String);
                                ("int_print", Int);
                                ("int_println", Int);
```

```
                              ("double_print", Double);
                              ("double_println", Double);
                              ("char_println", Char)]
in

(* Add function name to symbol table *)
let add_func map fd =
  let built_in_err = "function " ^ fd.fname ^ " may not be defined
 "
  and dup_err = "duplicate function " ^ fd.fname
  and make_err er = raise (Failure er)
  and n = fd.fname
  in match fd with (* No duplicate functions or redefinitions of
 built-ins *)
       _ when StringMap.mem n built_in_decls -> make_err
 built_in_err
     | _ when StringMap.mem n map -> make_err dup_err
     | _ ->  StringMap.add n fd map
in

(* Collect all function names into one symbol table *)
let function_decls = List.fold_left add_func built_in_decls
 functions
in

(* Add class name to class symbol table with cvars *)
let add_class map cls =
  let dup_error = "duplicate class " ^ cls.cname
  and make_err er = raise (Failure er)
  and n = cls.cname
  and vars = List.fold_left (fun m (ty, name) -> StringMap.add
 name ty m) StringMap.empty cls.cvars
  in match cls with
      _ when StringMap.mem n map -> make_err dup_error
    | _ -> StringMap.add n vars map
in

(* Collect all class info into class symbol table *)
let class_decls = List.fold_left add_class StringMap.empty classes
  in

(* Collect all class_method info into class_method symbol table *)
let class_methods_decls = List.fold_left
        (fun map cls -> StringMap.add cls.cname
            (List.fold_left add_func StringMap.empty cls.cmethods)
  map) StringMap.empty classes
```

```
in

(* Return a function from our function symbol table *)
let find_func s =
  try StringMap.find s function_decls
  with Not_found -> raise (Failure ("unrecognized function " ^ s))
in

let check_main func_typ =
  match func_typ with
    Int -> true
  | _ ->  raise (Failure("main function type is expected to be int
  instead of " ^ string_of_typ func_typ))
in

let main_func = find_func "main" in
let main_func_typ = main_func.typ in
let _ = check_main main_func_typ in

(* Check functions *)
let check_function vars func =
  (* Make sure no formals are void or duplicates *)
  check_binds "formal" func.formals;

  (* Raise an exception if the given rvalue type cannot be
 assigned to
     the given lvalue type *)
  let check_assign lvaluet rvaluet err =
     if lvaluet = rvaluet then lvaluet else raise (Failure err)
  in

  (* Build local symbol table of variables for this function *)
  let symbols = List.fold_left (fun m (ty, name) -> StringMap.add
 name ty m)
                 StringMap.empty (func.formals @ vars)
  in

  (* Return a variable type from our local symbol table *)
  let type_of_identifier s scope =
    try StringMap.find s scope
    with Not_found -> try StringMap.find s symbols
              with Not_found -> raise (Failure ("undeclared
 identifier " ^ s))
  in

  (* Return a ref (e.g. car.power) type *)
```

```
 let trace_type cls_instance cls_var scope =
   match type_of_identifier cls_instance scope with
     ClassT(clsname) -> (if StringMap.mem clsname class_decls =
false
                         then raise (Failure ("no class type " ^
clsname))
                         else let vars_map = StringMap.find
clsname class_decls in
                              try StringMap.find cls_var
vars_map
                              with Not_found -> raise (Failure (
"undeclared variable " ^ cls_var)))
     | _ -> raise (Failure (cls_instance ^ " must be a class
instance"))
 in

 (* Return a method function *)
 let trace_method clsname methodname =
     if StringMap.mem clsname class_decls = false
     then raise (Failure ("no class type " ^ clsname))
     else let methods_map = StringMap.find clsname
class_methods_decls in
           try StringMap.find methodname methods_map
           with Not_found -> raise (Failure ("undeclared method "
 ^ methodname ^ " in class " ^ clsname))
 in

 (* Return a semantically-checked expression, i.e., with a type
*)
 let rec expr scope e =
   match e with
     Literal  l -> (Int, SLiteral l)
   | Dliteral l -> (Double, SDliteral l)
   | BoolLit l  -> (Bool, SBoolLit l)
   | CharLit c -> (Char, SCharLit c)
   | StringLit s -> (String, SStringLit s)
   | Noexpr     -> (Void, SNoexpr)
   | Id (s :: []) -> (type_of_identifier s scope, SId([s]))
   | Id (s :: var :: []) ->  (trace_type s var scope, SId(s :: [
var]))
   | Id (_) -> raise(Failure ("Usage: cls_instance.cls_var (e.g.
car.power)"))
   | Assign(ref, e) as ex ->
       let lt = match ref with
         var :: [] -> type_of_identifier var scope
       | s :: var :: [] -> trace_type s var scope
```

```
      | _ -> raise (Failure ("Usage: cls_instance.cls_var (e.g.
car.power)"))
      and (rt, e') = expr scope e in
      let err = "illegal assignment " ^ string_of_typ lt ^ " = "
 ^
                string_of_typ rt ^ " in " (* ^ string_of_expr
ex *)
      in (check_assign lt rt err, SAssign(ref, (rt, e')))
  | Unop(op, e) as ex ->
      let (t, e') = expr scope e in
      let ty = match op with
        Neg when t = Int || t = Double -> t
      | Not when t = Bool -> Bool
      | _ -> raise (Failure ("illegal unary operator " ^
                             string_of_uop op ^ string_of_typ t
^
                             " in " ^ string_of_expr ex))
      in (ty, SUnop(op, (t, e')))
  | Binop(e1, op, e2) as e ->
      let (t1, e1') = expr scope e1
      and (t2, e2') = expr scope e2 in
      (* All binary operators require operands of the same type
*)
      let same = t1 = t2 in
      let ty = match op with
        Add | Sub | Mult when same && t1 = Int -> Int
      | Div | Mod when same && t1 = Int   -> if e2' = SLiteral
(0)
        then raise(Failure("Div by 0: " ^ string_of_expr e))
else Int
      | Add | Sub | Mult when same && t1 = Double -> Double
      | Div when same && t1 = Double -> if e2' = SDliteral("0.")
        then raise(Failure("Div by 0.0: " ^ string_of_expr e))
else Double
      | Equal | Neq            when same               -> Bool
      | Less | Leq | Greater | Geq
                when same && (t1 = Int || t1 = Double) -> Bool
      | And | Or when same && t1 = Bool -> Bool
      | _ -> raise (
      Failure ("illegal binary operator " ^
                  string_of_typ t1 ^ " " ^ string_of_op op ^ "
" ^
                  string_of_typ t2 ^ " in " ^ string_of_expr e)
)
      in (ty, SBinop((t1, e1'), op, (t2, e2')))
  | Call(ref, args) as call ->
```

```
      let fd = match ref with
        fname :: [] -> find_func fname
      | s :: methodname :: [] -> (match type_of_identifier s
scope with
                                    ClassT(clsname) ->
trace_method clsname methodname
                                  | _ -> raise (Failure (s ^ "
must be a class instance")))
      | _ -> raise (Failure ("Usage: cls_instance.cls_var (e.g.
car.power)"))
      in
        let param_length = List.length fd.formals in
        if List.length args != param_length then
        raise (Failure ("expecting " ^ string_of_int
param_length ^
                        " arguments in " ^ string_of_expr call))
        else let check_call (ft, _) e =
          let (et, e') = expr scope e in
          let err = "illegal argument found " ^ string_of_typ et
 ^
            " expected " ^ string_of_typ ft ^ " in " ^
string_of_expr e
          in (check_assign ft et err, e')
        in
        let args' = List.map2 check_call fd.formals args
        in (fd.typ, SCall(ref, args'))
  | ArrayLit(values) ->
    let array_body = List.map (expr scope) values in
    (* Type of first element in array *)
    let array_t, _ = List.hd array_body in
    let type_check el =
      let el_t = fst el in
      if el_t != array_t then raise (Failure ("Types in array
literal must all match")) else ()
    in
    (* Check all the types match *)
    ignore (List.map type_check array_body);
    (ArrT(array_t, List.length values), SArrayLit(array_t, List.
map snd array_body))
  | ArrayAccess(v, e) -> (* array name and array index *)
    (* check if type of e is an int *)
    let (typ', sx') = expr scope e in
      if typ' != Int
      then raise(Failure("Wrong type of array index in array
access"))
      else
```

```
        let v_ty = type_of_identifier v scope in
        let e_ty = match v_ty with
          ArrT(t, l) ->  match e with
                           Literal l1 -> if l1 >= l then raise(
Failure("Array (" ^ v ^") index (" ^
                                                 string_of_expr e ^ ")
out of bounds (" ^ string_of_int l ^")")) else t
                         |_ -> t
        | _ -> raise(Failure("Wrong type of variable in array
access"))
        in (e_ty, SArrayAccess(v, (typ', sx')))
  | ArrayAssign(v, e1, e2) -> (* array name array index value to
 be assigned *)
    (* check if type of e1 is int *)
    let (typ', sx') = expr scope e1 in
      if typ' != Int
      then raise(Failure("Wrong type of array index in array
access"))
      else (* check if type of v is array *)
        let v_ty = type_of_identifier v scope in
        let e_ty = match v_ty with
            ArrT(t, l) -> match e1 with
                           Literal l1 -> if l1 >= l then raise(
Failure("Array (" ^ v ^") index (" ^
                                                 string_of_expr e ^ ")
out of bounds (" ^ string_of_int l ^")")) else t
                         |_ -> t
          | _ -> raise(Failure("Wrong type of variable in array
assign"))
        in
        let (typ'', sx'') = expr scope e2 in
        (e_ty, SArrayAssign(v, (typ', sx'), (typ'', sx'')))
  | ArrayLength(name) -> (*return the length of array *)
    let v_ty = type_of_identifier name scope in
    let e_ty = match v_ty with
        ArrT(t, l) -> Int
      | _ -> raise(Failure("Must call .length on an array. " ^
name ^ " is not an array"))
    in
    (e_ty, SArrayLength(name))
 in

 let check_bool_expr e scope =
   let (t', e') = expr scope e
   and err = "expected Boolean expression in " ^ string_of_expr e
   in if t' != Bool then raise (Failure err) else (t', e')
```

```
 in

 (* Return a semantically-checked statement i.e. containing
sexprs *)
 let rec check_stmt stmt scope =
   match stmt with
     Expr e -> SExpr (expr scope e)
   | Initialize (_, _) -> raise (Failure ("Initialize stmts are
inside Block."))
   | If(p, b1, b2) -> SIf(check_bool_expr p scope, check_stmt b1
scope, check_stmt b2 scope)
   | For(e1, e2, e3, st) -> check_stmt (Block([Expr(e1); While(e2
, Block([st; Expr(e3)]))])) scope
   | While(p, s) -> SWhile(check_bool_expr p scope, check_stmt s
scope)
   | EmptyReturn -> SEmptyReturn
   | Return e -> let (t, e') = expr scope e in
     if t = func.typ then SReturn (t, e')
     else raise (
   Failure ("return gives " ^ string_of_typ t ^ " expected " ^
         string_of_typ func.typ ^ " in " ^ string_of_expr e))

   | StmtScope(block) ->
     check_stmt block scope
     (* A block is correct if each statement is correct and
nothing
         follows any Return statement.  Nested blocks are
flattened. *)
   | Block sl ->
       let rec check_stmt_list stmts scope =
         match stmts with
           [Return _ as s] -> [check_stmt s scope]
         | Initialize ((ClassT c, name), None) :: ss ->
             if StringMap.mem c class_decls = false then raise (
Failure (c ^" class " ^ "is undefined"))
             else SInitialize((ClassT c, name), None) ::
check_stmt_list ss (StringMap.add name (ClassT c) scope)
         | Initialize (bd, None) :: ss ->
             let (typ, name) = bd in
             if StringMap.mem name scope = true
             then raise (Failure ("cannot initialize " ^ name ^ "
 twice"))
             else SInitialize(bd, None) :: check_stmt_list ss (
StringMap.add name typ scope)
         | Initialize ((ClassT c, name), Some e) :: ss ->
             let se = expr scope e in
```

```
            if StringMap.mem c class_decls = false then raise (
Failure (c ^" class " ^ "is undefined"))
            else SInitialize((ClassT c, name), None) ::
check_stmt_list ss (StringMap.add name (ClassT c) scope)
        | Initialize (bd, Some e) :: ss ->
            let (typ, name) = bd in
            let se = expr scope e in
            let (typ', sx') = se in
            let _ = match typ with
                ArrT(t, l) -> let e' = (match e with
                    ArrayLit(ex) -> ex
                  | _ -> raise(Failure(name ^" needs to be
initialized to an array literal")))
                in
                (* Check if array is declared and init with
wrong length & wrong type *)
                if (List.length e') != l then raise(Failure("
Array (" ^ name ^ ") " ^
                  "declared with length (" ^ string_of_int l ^")
 but init with length (" ^ string_of_int (List.length e') ^")" ))
 else t
              | _ -> typ
            in
            if typ <> typ' then raise(Failure("initialize:
variable and value to be assigned of different types"))
            else
              (if StringMap.mem name scope = true
                        then raise (Failure ("cannot initialize
" ^ name ^ " twice"))
                        else SInitialize(bd, Some se) ::
check_stmt_list ss (StringMap.add name typ scope))
        | Return _ :: _   -> raise (Failure "nothing may follow
a return")
        | Block sl :: ss  -> check_stmt_list (sl @ ss) scope
        | s :: ss         -> check_stmt s scope ::
check_stmt_list ss scope
        | []              -> []
      in SBlock(check_stmt_list sl scope)

 in (* body of check_function *)
 { styp = func.typ;
   sfname = func.fname;
   sformals = func.formals;
   sbody = match check_stmt (Block func.body) StringMap.empty
with
 SBlock(sl) -> sl
```

```
      | _ -> raise (Failure ("internal error: block didn't become a
   block?"))
    }
  in

  let check_class cls =
    let ctrs = cls.cconstructors in
    let dtrs = cls.cdestructors in
    let check_cd_name func =
      if func.fname <> cls.cname then raise (Failure ("Class
   constructor or destructor does not have same name as class,
   expected " ^ cls.cname ^ " (class) == " ^ func.fname)) else ()
    in
    ignore (List.map check_cd_name ctrs);
    ignore (List.map check_cd_name dtrs);
    let check_dtrs = function
        [] -> None
      | hd::[] -> Some (check_function ((ClassT(cls.cname), "me")
   ::[]) hd)
      | _ -> raise (Failure ("Can only have one or zero destructors"
   ))
    in
    { scname = cls.cname;
      scvars = cls.cvars;
      scmethods = List.map (check_function ((ClassT(cls.cname), "me"
   )::[])) cls.cmethods;
      scconstructors = List.map (check_function ((ClassT(cls.cname),
    "me")::[])) ctrs;
      scdestructor = check_dtrs dtrs;
    }
  in
  (List.map (check_function []) functions, List.map check_class
   classes)
```

Code generation

Contents of `codegen.ml`:

```
module A = Ast
module L = Llvm
open Sast

module SMap = Map.Make(String)

(* A variable scope, contains variables and a ref to the parent
   scope *)
```

```
type vscope = Scope of (vscope option) * (L.llvalue * A.typ * bool)
    SMap.t
(* A codegen context: builder and variable scope *)
type ctx_t = Ctx of L.llbuilder * vscope

let gen (sfunction_decls, sclass_decls) =
  (* Set up module & context *)
  let context = L.global_context () in
  let m = L.create_module context "kazm" in

  let mangle_method_name cname mname = cname ^ "__" ^ mname in

  (* Set up types in the context *)
  let i1_t = L.i1_type context in
  let i8_t = L.i8_type context in
  let i32_t = L.i32_type context in
  let i64_t = L.i64_type context in
  let double_t = L.double_type context in
  let string_t = L.pointer_type (L.i8_type context) in
  let char_t = i8_t in
  let void_t = L.void_type context in
  let char_ptr_t = L.pointer_type char_t in
  let void_ptr_t = L.pointer_type i8_t in

  let gen_cls_t map cls =
    let name = cls.scname in
    let cls_t = L.named_struct_type context name in
    SMap.add name cls_t map
  in

  let cls_ts = List.fold_left gen_cls_t SMap.empty sclass_decls in

  (* Map our AST type to LLVM type *)
  let rec typ_to_t = function
      A.Void -> void_t
    | A.Bool -> i1_t
    | A.Int -> i32_t
    | A.String -> string_t
    | A.Double -> double_t
    | A.ClassT(name) ->
      let cls_t = SMap.find name cls_ts in
      L.pointer_type cls_t
    | A.Char -> i8_t
    | A.ArrT(ty, _) -> L.pointer_type (typ_to_t ty)
  in
```

```
let get_default = function
  | A.Bool -> L.const_int i1_t 0
  | A.Int -> L.const_int i32_t 0
  | A.Double -> L.const_float double_t (float_of_string "0.0")
  | A.Char -> L.const_int i8_t 109
in

(* Codegen function definitions *)
let get_func_sig func =
  let ret_t = typ_to_t func.styp in
  let arg_types = List.map (fun sformal -> typ_to_t (fst sformal))
  func.sformals in
  (ret_t, arg_types)
in

let codegen_class_decl map cls =
  let name = cls.scname in
  let member_ts = List.map (fun v -> typ_to_t (fst v)) cls.scvars
 in
  let cls_t = SMap.find name cls_ts in
  let add_method map mthd =
    let mname = mthd.sfname in
    let mangled_name = mangle_method_name name mname in
    let (ret_t, arg_ts) = get_func_sig mthd in
    let me_t = L.pointer_type cls_t in
    let func_t = L.function_type ret_t (Array.of_list (me_t::
 arg_ts)) in
    let func_def = L.define_function mangled_name func_t m in
    SMap.add mname (mthd, mangled_name, func_def) map
  in
  let dtrs = match cls.scdestructor with
    None -> []
  | Some d -> [d]
  in
  let mthds = List.fold_left add_method SMap.empty cls.scmethods
 in
  let ctrs = List.fold_left add_method SMap.empty cls.
 scconstructors in
  let dtr = List.fold_left add_method SMap.empty dtrs in
  ignore (L.struct_set_body cls_t (Array.of_list member_ts) false)
 ;
  SMap.add name (cls, cls_t, mthds, ctrs, dtr) map
in

let all_classes = List.fold_left codegen_class_decl SMap.empty
 sclass_decls in
```

```
let codegen_func_decl name ret_t arg_ts =
  let func_t = L.function_type ret_t (Array.of_list arg_ts) in
    L.declare_function name func_t m
in

let add_func_decl map name ret_t arg_ts =
  SMap.add name (codegen_func_decl name ret_t arg_ts) map
in

let codegen_func_def name ret_t arg_ts =
  let func_t = L.function_type ret_t (Array.of_list arg_ts) in
    L.define_function name func_t m
in

let add_func_def map name ret_t arg_ts =
  SMap.add name (codegen_func_def name ret_t arg_ts) map
in

(* Given a builder and our type, build a dummy return (e.g. if
 there's a missing return) *)
let build_default_return typ ctx =
  let Ctx(builder, sp) = ctx in
  match typ with
    A.Void -> ignore (L.build_ret_void builder); ctx
  | A.Bool -> ignore (L.build_ret (L.const_int i1_t 0) builder);
 ctx
  | A.Int -> ignore (L.build_ret (L.const_int i32_t 0) builder);
 ctx
  | A.Double -> ignore (L.build_ret (L.const_float double_t 0.)
 builder); ctx
  | A.Char -> ignore (L.build_ret (L.const_int i8_t 109) builder);
   ctx
in

let all_funcs = SMap.empty in
(* Builtins... *)
let all_funcs = add_func_decl all_funcs "print" void_t [char_ptr_t
 ] in
let all_funcs = add_func_decl all_funcs "println" void_t [
 char_ptr_t] in
let all_funcs = add_func_decl all_funcs "int_print" void_t [i32_t]
  in
let all_funcs = add_func_decl all_funcs "int_println" void_t [
 i32_t] in
let all_funcs = add_func_decl all_funcs "double_print" void_t [
```

```
 double_t] in
let all_funcs = add_func_decl all_funcs "double_println" void_t [
 double_t] in
let all_funcs = add_func_decl all_funcs "char_println" void_t [
 i8_t] in
let all_funcs = add_func_decl all_funcs "next_int" i32_t [] in

(* Codegen function definitions *)
let codegen_func_sig all_funcs func =
  let (ret_t, arg_ts) = get_func_sig func in
  add_func_def all_funcs func.sfname ret_t arg_ts
in

let all_funcs = List.fold_left codegen_func_sig all_funcs
 sfunction_decls in

let new_scope ctx =
  let Ctx(_, parent_scope) = ctx in
  Scope(Some parent_scope, SMap.empty)
in

let rec debug_scope_inspect name scope =
  (* Prints out all vars in all scopes *)
  match scope with
    Scope(None, map) -> _debug_print_all_vars_in_map "top" map
  | Scope(Some parent, map) -> _debug_print_all_vars_in_map name
 map; debug_scope_inspect (name ^ "_up") parent
and _debug_print_all_vars_in_map name map =
  SMap.iter (fun name _ -> ignore (print_endline ("Found var " ^
 name ^ " in " ^ name)))
in

let rec find_var scope name =
  match scope with
    Scope(None, map) -> SMap.find name map
  | Scope(Some parent, map) -> if SMap.mem name map then SMap.find
   name map else find_var parent name
in

let add_var scope name var vtyp own =
  let Scope(p, map) = scope in
  let map' = SMap.add name (var, vtyp, own) map in
  Scope(p, map')
in

let build_class_alloc cname name builder =
```

```
  (* Class info *)
  let (cls, cls_t, mthds, ctrs, dtr) = SMap.find cname all_classes
  in
  (* A pointer to the right struct *)
  let ptr_var = L.build_alloca (L.pointer_type cls_t) name builder
  in
  (* Malloc the memory *)
  let mallocd = L.build_malloc cls_t ("_malloc_" ^ name) builder
 in
  ignore (L.build_store mallocd ptr_var builder);
  let me = L.build_load ptr_var "me" builder in
  (* Let's call the constructor, if any *)
  (match SMap.find_opt cname ctrs with
    None -> ()
  | Some (mthd, mangled_name, fn) -> ignore (L.build_call fn (
 Array.of_list ([me])) "" builder));
  ptr_var
in

let destroy_var name var vtyp builder =
  match vtyp with
  (* TODO: *)
    (* For classes, call the destructor and free the memory *)
    | A.ClassT(cname) ->
      (* Class info *)
      let (cls, cls_t, mthds, ctrs, dtr) = SMap.find cname
 all_classes in
      let me = L.build_load var "me" builder in
      (* Call destructor if it exists *)
      (match SMap.find_opt cname dtr with
        None -> ()
      | Some (mthd, mangled_name, fn) -> ignore (L.build_call fn (
 Array.of_list ([me])) "" builder));
      ignore (L.build_free me builder)
    (* For arrays, call the desctructors on each and free the
 memory *)
    | A.ArrT(ty, _) ->
      ()
    (* For primitive types, we don't have to do anything *)
    | _ -> ()
in

let find_fq_var builder scope = function
  (* Unqualified access *)
    ref::[] -> let (var, _, _) = find_var scope ref in var
  (* Qualified access *)
```

```
  | hd::tl::[] ->
    (* Get info about the variable *)
    let (cval, ClassT(cname), own) = find_var scope hd in
    (* Get info about the class *)
    let (cls, cls_t, mthds, ctrs, dtr) = SMap.find cname
 all_classes in
    (* Members names with indexes *)
    let mems = List.mapi (fun ix v -> (ix, snd v)) cls.scvars in
    (* Filter out the members that have the same name as the
 sought after member (there should only be 1) *)
    let (mem_pos_in_class, _) = List.hd (List.filter (fun (ix, v)
 -> (tl = v)) mems) in
    (* Load address of the struct *)
    let load = L.build_load cval ("_struct_" ^ hd) builder in
    L.build_struct_gep load mem_pos_in_class (tl ^ "_ptr") builder
  | _ -> raise (Failure("find_fq_var: cannot be other patterns"))
in

let build_scope_exit ctx =
  (* Builds destructors, etc *)
  let Ctx(builder, sp) = ctx in
  let Scope(_, vars) = sp in
  let dest_var name (var, vtyp, own) =
    if own then destroy_var name var vtyp builder else ()
  in
  ignore (SMap.iter dest_var vars)
in

(* Codegen for an expression *)
let rec codegen_expr ctx ((typ, e) : sexpr) =
  let Ctx(builder, sp) = ctx in
  match e with
  (* Function call *)
    SCall(ref, exprs) ->
      let (ctx', args) = Future.fold_left_map codegen_expr ctx
 exprs in
      let ex = match ref with
      | fname :: [] ->
        L.build_call (SMap.find fname all_funcs) (Array.of_list
 args) "" builder
      | cvar :: methodname :: [] ->
        let (var, A.ClassT(cname), own) = find_var sp cvar in
        let (cls, cls_t, mthds, ctrs, dtr) = SMap.find cname
 all_classes in
        let (mthd, mangled_name, fn) = SMap.find methodname mthds
 in
```

```
      let me = L.build_load var "me" builder in
      L.build_call fn (Array.of_list (me::args)) "" builder
  | _ -> raise (Failure("codegen_expr SCall:cannot be other
patterns"))
    in
 (ctx', ex)
 (* New bool literal *)
 | SBoolLit(value) -> (ctx, L.const_int i1_t (if value then 1
else 0))
 (* New 32-bit integer literal *)
 | SLiteral(value) -> (ctx, L.const_int i32_t value)
 | SDliteral(value) -> (ctx, L.const_float double_t (
float_of_string value))
 (* New string literal (just make a new global string) *)
 | SStringLit(value) -> (ctx, L.build_global_stringptr value "
globalstring" builder)
 | SCharLit(value) -> (ctx, L.const_int i8_t (Char.code (String.
get value 0)))
 | SUnop(op, ((t, _) as e)) ->
    let (ctx1, e') = codegen_expr ctx e in
    let lbuild = match op with
        A.Neg when t = A.Double -> L.build_fneg
      | A.Neg                   -> L.build_neg
      | A.Not                   -> L.build_not
    in
    let Ctx(builder, sp) = ctx1 in
    let new_expr = lbuild e' "unop_res" builder in
    (ctx1, new_expr)
 | SBinop(e1, op, e2) ->
   (* Lookup right thing to build in llvm *)
   let lbuild = match op with
       A.Add when typ = Int -> L.build_add
     | A.Add when typ = Double -> L.build_fadd
     | A.Sub when typ = Int -> L.build_sub
     | A.Sub when typ = Double -> L.build_fsub
     | A.Mult -> L.build_mul
     (* TODO: Simply tested but more tests may be needed *)
     | A.Div when typ = Int -> L.build_sdiv
     (* | A.Div when typ = Double ->L.build_fdiv *)
     | A.Div when typ = Double ->  let (Double, SDliteral l) = e2
 in
                                   raise(Failure(l))
     | A.Mod -> L.build_srem
     | A.And -> L.build_and
     | A.Or -> L.build_or
     | A.Equal -> L.build_icmp L.Icmp.Eq
```

```
      | A.Neq -> L.build_icmp L.Icmp.Ne
      | A.Less -> L.build_icmp L.Icmp.Slt
      | A.Leq -> L.build_icmp L.Icmp.Sle
      | A.Greater -> L.build_icmp L.Icmp.Sgt
      | A.Geq -> L.build_icmp L.Icmp.Sge
    in
    let (ctx1, first) = codegen_expr ctx e1 in
    let (ctx2, second) = (codegen_expr ctx1 e2) in
    let Ctx(builder, sp) = ctx2 in
    let new_expr = lbuild first second "binop_res" builder in
    (ctx2, new_expr)
  | SId(fqn) ->
    let var = find_fq_var builder sp fqn in
    (ctx, L.build_load var "id" builder)
  (* Assign expression e to a new bind(type, name) *)
  | SAssign(fqn, value) ->
    let var = find_fq_var builder sp fqn in
    let (ctx', e') = codegen_expr ctx value in
    ignore (L.build_store e' var builder);
    (ctx, e')
  | SArrayLit(arr_t, exs) ->
    (* arr: sexpr list = typ * sx list *)
    let size = List.length exs in
    let ty = typ_to_t (A.ArrT(arr_t, size)) in
    (* allocate memory for array *)
    let arr_alloca = L.build_array_malloc ty (L.const_int i32_t
size) "array_literal" builder in
    (* bitcast -- pointer-to-int *)
    let arr_ptr = L.build_pointercast arr_alloca ty "array_ptr"
builder in
    (* store an element in slot `ix` and pass off context to the
next one *)
    let store_el (ctx, ix) el =
      let (ctx', gex) = codegen_expr ctx (arr_t, el) in
      let element_ptr = L.build_in_bounds_gep arr_ptr [|  (L.
const_int i32_t ix) |] "array_element" builder in
      ignore (L.build_store gex element_ptr builder);
      (ctx', ix + 1)
    in
    let (ctx', _) = List.fold_left store_el (ctx, 0) exs in
    (ctx', arr_ptr)
  | SArrayAccess(name, pos_ex) ->
    let (ctx', element) = build_array_element_ptr ctx name pos_ex
in
    (ctx', L.build_load element (name ^ "__element") builder)
  | SArrayAssign(name, pos_ex, assign_ex) ->
```

```
    let (ctx', assign) = codegen_expr ctx assign_ex in
    let (ctx'', element) = build_array_element_ptr ctx' name
 pos_ex in
    (ctx'', L.build_store assign element builder)
  | SArrayLength(name) ->
    let var = find_var sp name in
    let (ll, vtyp, own) = var in (* llvalue and Ast typ *)
    let ArrT(t, l) = vtyp in (* Ast typ and length of array *)
    (ctx, L.const_int i32_t l)
and build_array_element_ptr ctx name pos_ex =
  let Ctx(builder, sp) = ctx in
  let (ctx', pos) = codegen_expr ctx pos_ex in
  let array_ptr = find_fq_var builder sp [name] in
  (* Load the array address into a register *)
  let array = L.build_load array_ptr (name ^ "__array") builder in
  (ctx', L.build_in_bounds_gep array [| pos |] (name ^ "
 __element_ptr") builder)
in

(* Add terminator to end of a basic block *)
let add_terminator ctx build_terminator =
  let Ctx(builder, _) = ctx in
  (* llvm.moe: block_terminator returns the terminator of the BB,
    insertion_block returns the current block we're inserting into
  with builder *)
  match L.block_terminator (L.insertion_block builder) with
    Some _ -> ()
  | None -> ignore (build_terminator ctx)
in

let build_alloc vtyp name builder =
  match vtyp with
    | A.ClassT(cname) ->
      build_class_alloc cname name builder
    | _ ->
      L.build_alloca (typ_to_t vtyp) name builder
in

(* Codegen for function body *)
let raw_gen_func body typ name formals fn =
  (* Codegen for a statement *)
  (* Takes ctx and statement and returns a ctx *)
  let rec codegen_stmt ctx stmt =
    let Ctx(builder, sp) = ctx in
    match stmt with
    (* For expressions we just codegen the expression *)
```

```
      SExpr(e) ->
        let (ctx', ex) = codegen_expr ctx e in
        ctx'
    | SEmptyReturn -> build_default_return typ ctx
    | SReturn(expr) ->
      let (ctx', gexpr) = codegen_expr ctx expr in
      ignore (L.build_ret gexpr builder); ctx'
    (* If-statements *)
    | SIf(cond, true_stmts, false_stmts) ->
      (* Codegen the condition evaluation *)
      let (ctx', gcond) = codegen_expr ctx cond in

      (* Add the block we go to if we take this branch (cond is
true) *)
      let true_blk = L.append_block context "take" fn in
      (* Add the block we go to if we don't take this branch (cond
 is false) *)
      let false_blk = L.append_block context "dont_take" fn in

      (* Generate the block that we come back to after both
branches *)
      let join_blk = L.append_block context "join" fn in
      let join_builder = L.builder_at_end context join_blk in

      (* This function takes a builder and builds a `br`
instruction, which
         (br)anches back to the start of the join_blk block made
above *)
      let build_join ctx =
        let Ctx(bldr, _) = ctx in
        L.build_br join_blk bldr
      in

      (* True branch building *)
      let true_ctx = Ctx(L.builder_at_end context true_blk,
new_scope ctx') in
      (* Build this branch's statements into this block *)
      let true_ctx' = codegen_stmt true_ctx true_stmts in
      ignore (build_scope_exit true_ctx');
      add_terminator true_ctx' build_join;

      (* False branch building *)
      let false_ctx = Ctx(L.builder_at_end context false_blk,
new_scope ctx') in
      let false_ctx' = codegen_stmt false_ctx false_stmts in
      ignore (build_scope_exit false_ctx');
```

```
      add_terminator false_ctx' build_join;

      (* Build the actual conditional branch *)
      ignore (L.build_cond_br gcond true_blk false_blk builder);
      (* Finally return the new builder at end of merge *)
      let Ctx(_, sp) = ctx' in
      Ctx(join_builder, sp)
    (* While-statements *)
    | SWhile(cond, stmt) ->
      (* Start at the start block and its builder *)
      let start_blk = L.append_block context "start" fn in
      let start_builder = L.builder_at_end context start_blk in

      (* Loop block that we keep repeating *)
      let loop_blk = L.append_block context "loop" fn in
      let loop_builder = L.builder_at_end context loop_blk in
      (* Loop body (an iteration) *)
      let while_ctx = Ctx(loop_builder, new_scope ctx) in
      let while_ctx' = codegen_stmt while_ctx stmt in
      ignore (build_scope_exit while_ctx');
      (* Back to start after a loop iteration *)
      let Ctx(loop_builder', _) = while_ctx' in
      ignore (L.build_br start_blk loop_builder');

      (* Generate the end block where we end up after the while
cond becomes false *)
      let end_blk = L.append_block context "end" fn in
      let end_builder = L.builder_at_end context end_blk in

      (* Codegen the condition evaluation *)

      let start_ctx = Ctx(start_builder, sp) in
      let (_, gcond) = codegen_expr start_ctx cond in
      (* Build the branch instr *)
      ignore (L.build_cond_br gcond loop_blk end_blk start_builder
);

      (* Branch to start *)
      ignore (L.build_br start_blk builder);

      (* Continue building after the end of the loop *)
      let Ctx(_, sp) = ctx in
      Ctx(end_builder, sp)
    (* For a block of statements, just fold *)
    | SBlock(stmts) ->
      let ctx' = Ctx(builder, new_scope ctx) in
```

```
      let ctx'' = List.fold_left codegen_stmt ctx' stmts in
      ignore (build_scope_exit ctx'');
      let Ctx(_, sp') = ctx' in
      let Ctx(builder'', _) = ctx'' in
      Ctx(builder'', sp')
    | SInitialize((vtyp, name), expr) ->
        (match vtyp with
          A.ClassT(cname) ->
            if expr != None then raise (Failure ("Can't assign
init class")) else
            Ctx(builder, add_var sp name (build_class_alloc cname
name builder) vtyp true)
          | _ ->
            let (ctx', value) =
              (match expr with
              | Some e ->
                codegen_expr ctx e
              | None ->
                match vtyp with
                  | A.ArrT(t, l) ->
                    let arr_lit = (match t with
                          A.Int -> SArrayLit(A.Int, List.init l (
fun x -> SLiteral(0)))
                        | A.Bool -> SArrayLit(A.Bool, List.init l
(fun x -> SBoolLit(false)))
                        | A.Double -> SArrayLit(A.Double, List.
init l (fun x -> SDliteral("0.0")))
                      ) in
                    codegen_expr ctx (A.ArrT(t, l), arr_lit)
                  | A.String ->
                    (ctx, L.build_global_stringptr "" "
default_string" builder)
                  | _ -> (ctx, get_default vtyp)
              )
            in
            let var = L.build_alloca (typ_to_t vtyp) name builder
in
            let Ctx(_, sp') = ctx' in
            ignore (L.build_store value var builder);
            Ctx(builder, add_var sp' name var vtyp true)
    )
 in

 let fn_builder = L.builder_at_end context (L.entry_block fn) in

 let add_param map (ptyp, name) param =
```

```
    L.set_value_name name param;
    let local_copy = L.build_alloca (typ_to_t ptyp) name
 fn_builder in
    ignore (L.set_value_name (name ^ "_local") local_copy);
    ignore (L.build_store param local_copy fn_builder);
    SMap.add name (local_copy, ptyp, false) map
  in
  let params = List.fold_left2 add_param SMap.empty formals (Array
 .to_list (L.params fn)) in
  let fn_scope = Scope(None, params) in
  let initialize_var scope (vtyp, name, own) =
    add_var scope name (build_alloc vtyp name fn_builder) vtyp own
  in
  let fn_scope' = List.fold_left initialize_var fn_scope [] in
  let fn_ctx = Ctx(fn_builder, fn_scope') in
  (* Build all statements *)
  let ctx' = codegen_stmt fn_ctx (SBlock body) in
  ignore (build_scope_exit ctx');
  ignore (add_terminator ctx' (build_default_return typ))
in

let gen_func func =
  let body = func.sbody in
  let typ = func.styp in
  let name = func.sfname in
  let formals = func.sformals in
  let fn = SMap.find name all_funcs in
  raw_gen_func body typ name formals fn
in

let gen_methods cname (cls, cls_t, mthds, ctrs, dtr) =
  let gen_method name (mthd, mangled_name, fn) =
    let body = mthd.sbody in
    let typ = mthd.styp in
    let me_formal = (A.ClassT(cname), "me") in
    let formals = me_formal :: mthd.sformals in
    raw_gen_func body typ mangled_name formals fn
  in
  ignore (SMap.iter gen_method mthds);
  ignore (SMap.iter gen_method ctrs);
  ignore (SMap.iter gen_method dtr);
in

ignore (List.map gen_func sfunction_decls);
ignore (SMap.iter gen_methods all_classes);
```

```
  m
```

C Builtins

Contents of `builtins.c`:

```
#include <stdio.h>
#include <malloc.h>

void println(char* str) {
  printf("%s\n", str);
}

void print(char* str) {
  printf("%s", str);
}

void int_print(int val) {
  printf("%d", val);
}

void int_println(int val) {
  printf("%d\n", val);
}

void char_println(char val){
  printf("%c\n", val);
}

void double_print(double val) {
  printf("%f", val);
}

void double_println(double val) {
  printf("%f\n", val);
}

// Useful before variables work to test loops, etc
int _the_next_int = 0;
int next_int() {
  return _the_next_int++;
}
```

Backported OCaml functions

Contents of `future.ml`:

```
(* Copied from ocaml source since this is only available in >4.11 *)
(* https://github.com/ocaml/ocaml/blob/
   cce52acc7c7903e92078e9fe40745e11a1b944f0/stdlib/list.ml#L272-L278
    *)
let fold_left_map f accu l =
  let rec aux accu l_accu = function
    | [] -> accu, List.rev l_accu
    | x :: l ->
        let accu, x = f accu x in
        aux accu (x :: l_accu) l in
  aux accu [] l

(* https://github.com/ocaml/ocaml/blob/
   cce52acc7c7903e92078e9fe40745e11a1b944f0/stdlib/list.ml#L247-L252
    *)
let filteri p l =
  let rec aux i acc = function
  | [] -> List.rev acc
  | x::l -> aux (i + 1) (if p i x then x::acc else acc) l
  in
  aux 0 [] l
```

Compile helper

Contents of `compile.py`:

```
#!/bin/python3

from pathlib import Path
import shutil
import subprocess
import argparse

parser = argparse.ArgumentParser(description="Kazm compiler")
parser.add_argument("file", help="the .kazm file to compile")
parser.add_argument("--run", dest="run", action="store_true", help="
   Run the compiled file")
parser.set_defaults(run=False, select="")
args = parser.parse_args()

# clean build dir
tmp_dir = Path(".") / "_kazm_build"
shutil.rmtree(tmp_dir, ignore_errors=True)
tmp_dir.mkdir()

def pipe_through(args, input_):
```

```
    out = subprocess.run(args, input=input_, capture_output=True)
    return out.returncode, out.stdout, out.stderr

def run(args):
    out = subprocess.run(args, capture_output=True)
    return out.returncode, out.stdout.decode("utf8"), out.stderr.
   decode("utf8")

print(f"Building kazm...")
print(f" Compiling kazm.native")
code, stdout, stderr = run(["opam", "config", "exec", "--", "
   ocamlbuild", "-use-ocamlfind", "kazm.native"])
if code != 0:
    print("Failed to compile kazm.native")
    print(stderr)
    print(stdout)
    exit(1)
print(f" Compiling builtins")
builtins_o = f"{tmp_dir}/builtins.o"
code, _, _ = run(["cc", "-c", "builtins.c", "-o", builtins_o])
assert code == 0, "Failed to compile builtins"

filename = args.file

assert filename.endswith(".kazm"), "File must end with .kazm"
name = filename[:-len(".kazm")]

print(f"Compiling '{name}'")
# LLVM IR file
ll_file = f"{tmp_dir}/{name}.ll"
# assembly file
s_file = f"{tmp_dir}/{name}.s"
# executable
ex_file = f"./{name}"
print(f" Compiling '{name}' through kazm...")
with open(filename, "rb") as f:
    code, data, err = pipe_through(["./kazm.native"], f.read())
    # failed to compile
if code != 0:
    exit("Failed to compile with kazm:\n\n" + err.decode("utf8"))
with open(ll_file, "wb") as o:
    o.write(data)
print(f" Compiling '{name}' through LLVM...")
code, _, stderr = run(["llc", "--relocation-model=pic", ll_file, f"-
   o={s_file}"])
# failed to compile llvm
```

```
if code != 0:
    exit("Failed to compile with llvm:\n\n" + stderr)
print(f" Linking '{name}' through cc...")
code, _, stderr = run(["cc", "-o", ex_file, builtins_o, s_file])
if code != 0:
    exit("Failed to link:\n\n" + stderr)
print(f"OK.")
print(f"Compiled code written to '{ex_file}'")

if args.run:
    print(f" Running '{name}'...")
    code, stdout, stderr = run([ex_file])
    if code == 0:
        print()
        print(stdout)
    else:
        exit("Failed to run:\n\n" + stderr)
```

Git log

```
commit 8362a38d5a348016a3e9b54bfc5b26211e627000
Author: Aapeli <git@aapelivuorinen.com>
Date:   Wed Dec 29 19:33:26 2021 +0200

    Clean up

commit eaf13d1b011cda00cac82344cd94a1928a8fbd03
Author: Aapeli <git@aapelivuorinen.com>
Date:   Wed Dec 29 19:32:49 2021 +0200

    Update readme

commit 54a24d4884012429521359028162341 1e817abe9
Merge: 9a74426 0ca6d3e
Author: Aapeli <git@aapelivuorinen.com>
Date:   Wed Dec 29 19:27:32 2021 +0200

    Merge branch 'develop' of github.com:aapeliv/plt into develop

commit 9a744260f8b989f32b4c13db9aad736cadc8835b
Author: Aapeli <git@aapelivuorinen.com>
Date:   Wed Dec 29 19:27:26 2021 +0200

    Add bit from report
```

```
commit 0ca6d3e77184e33bf7d52d195147c34c20007579
Merge: 862b06e 583b792
Author: mrm2234 <60762357+mrm2234@users.noreply.github.com>
Date:   Wed Dec 29 11:45:57 2021 -0500

    Merge branch 'develop' of https://github.com/aapeliv/plt into
   develop

commit 862b06e3d49ee3abc5373b376efcfce2162fe332
Author: mrm2234 <60762357+mrm2234@users.noreply.github.com>
Date:   Wed Dec 29 11:45:48 2021 -0500

    string + string

commit 583b792cb327254bedd5e6c70571a67ab443823c
Author: mrm2234 <60762357+mrm2234@users.noreply.github.com>
Date:   Wed Dec 29 11:40:08 2021 -0500

    Update tests.md

commit 8192dae09095fe2f934a5abe1bc6b8c333d19c38
Author: mrm2234 <60762357+mrm2234@users.noreply.github.com>
Date:   Wed Dec 29 11:28:57 2021 -0500

    Update Makefile

commit 5db2d949baa12e84b51804f5c3ad06e39ba1948c
Author: mrm2234 <60762357+mrm2234@users.noreply.github.com>
Date:   Wed Dec 29 11:27:16 2021 -0500

    got rid of break

commit b712802ad942f86c3e7da8bcd1f2e9f490ca09b1
Author: Katie Kim <jk4534@columbia.edu>
Date:   Wed Dec 29 11:06:33 2021 -0500

    comment for array test

commit 4f5c2551db9716dc4c42c8464e4a21670b55c8cb
Merge: badc15e 98a55e9
Author: Katie Kim <jk4534@columbia.edu>
Date:   Wed Dec 29 10:25:20 2021 -0500

    Merge branch 'develop' of https://github.com/aapeliv/plt into
   develop
```

```
commit badc15ea147c1d389a797949fd4489c371ed7673
Author: Katie Kim <jk4534@columbia.edu>
Date:   Wed Dec 29 10:25:05 2021 -0500

    added array test

commit 98a55e9feda72bda3adf7de7fecbae1a773f7dfc
Author: mrm2234 <60762357+mrm2234@users.noreply.github.com>
Date:   Wed Dec 29 09:13:27 2021 -0500

    remove one test file in tests.md

commit 62b17c9ca211e21bbcda8d21d074c03fb18d70fd
Merge: 7f00bde 549c924
Author: mrm2234 <60762357+mrm2234@users.noreply.github.com>
Date:   Wed Dec 29 09:12:44 2021 -0500

    Merge branch 'develop' of https://github.com/aapeliv/plt into
   develop

commit 7f00bde70f78f2fb333526ef93a6751913a14c1e
Author: mrm2234 <60762357+mrm2234@users.noreply.github.com>
Date:   Wed Dec 29 09:12:38 2021 -0500

    nested if

commit 549c924e96c97c13987e6f9e5beaba7175311056
Author: Katie Kim <jk4534@columbia.edu>
Date:   Wed Dec 29 08:34:07 2021 -0500

    cleanup

commit 5913fedd1e8c7fc58def752f993925833b1dfd79
Author: mrm2234 <60762357+mrm2234@users.noreply.github.com>
Date:   Wed Dec 29 01:47:31 2021 -0500

    Makefile

commit 649b9492d9e8cd9c43509a053174127f72ab20bf
Author: mrm2234 <60762357+mrm2234@users.noreply.github.com>
Date:   Wed Dec 29 01:21:53 2021 -0500

    tests and test.md

commit bd7ff12e88a346beb06b091d4bf80ea49de524cb
Author: mrm2234 <60762357+mrm2234@users.noreply.github.com>
```

```
Date:   Wed Dec 29 00:30:57 2021 -0500

    new class files

commit 30a4c0541016cea809e11fc09d0c3546a4e541b2
Author: mrm2234 <60762357+mrm2234@users.noreply.github.com>
Date:   Wed Dec 29 00:23:13 2021 -0500

    Update tests.md

commit a5dd2246a8656292785253abd53f1b1dd165bbba
Author: mrm2234 <60762357+mrm2234@users.noreply.github.com>
Date:   Wed Dec 29 00:22:10 2021 -0500

    Update tests.md

commit 02ed6a1a7a251fbb8d3303d38c6e2314c4440262
Author: mrm2234 <60762357+mrm2234@users.noreply.github.com>
Date:   Wed Dec 29 00:15:23 2021 -0500

    Update tests.md

commit b0caeaa5458355beb045190e5815a3e9493dd697
Author: mrm2234 <60762357+mrm2234@users.noreply.github.com>
Date:   Wed Dec 29 00:02:56 2021 -0500

    Update tests.md

commit b5f5e67954c01d7d2c7e1fc9293038e723178df9
Author: mrm2234 <60762357+mrm2234@users.noreply.github.com>
Date:   Tue Dec 28 23:56:10 2021 -0500

    Update tests.md

commit 11a11acec744b738b070ef649ffe6676e0b09060
Author: mrm2234 <60762357+mrm2234@users.noreply.github.com>
Date:   Tue Dec 28 23:49:09 2021 -0500

    Update tests.md

commit a50b4109ef27772c2bfd97ba6a8f8b933f3ed71c
Author: mrm2234 <60762357+mrm2234@users.noreply.github.com>
Date:   Tue Dec 28 23:47:32 2021 -0500

    finished tests.md
```

```
commit 3248a9c82b318c37b4e41405c3aebba6f10aca13
Merge: 9965d4f 9aa4b49
Author: mrm2234 <60762357+mrm2234@users.noreply.github.com>
Date:   Tue Dec 28 21:26:34 2021 -0500

    Merge branch 'develop' of https://github.com/aapeliv/plt into
   develop

commit 9965d4fd9396089d3cf373b3e9263f6f1d057b19
Author: mrm2234 <60762357+mrm2234@users.noreply.github.com>
Date:   Tue Dec 28 21:26:32 2021 -0500

    tests.md

commit 9aa4b49e4cacbcb13b89c343c45465fe595e3c04
Merge: b9942f3 65cebd3
Author: yangzhonglin <zhonglinyang98@gmail.com>
Date:   Tue Dec 28 21:22:13 2021 -0500

    Merge branch 'develop' of https://github.com/aapeliv/plt into
   develop

commit b9942f3ec9634da8f37a8b1a4486d3405b76339a
Merge: 55fd973 cd7f8e2
Author: yangzhonglin <zhonglinyang98@gmail.com>
Date:   Tue Dec 28 21:19:54 2021 -0500

    Merge branch 'develop' of https://github.com/aapeliv/plt into
   develop

commit 65cebd38c06a96b21a9e106c949fbcd9f642cb3d
Merge: d781223 cd7f8e2
Author: mrm2234 <60762357+mrm2234@users.noreply.github.com>
Date:   Tue Dec 28 21:19:49 2021 -0500

    void main

commit 55fd973e9e7e3146dcdafbd34ed682c6a9343953
Author: yangzhonglin <zhonglinyang98@gmail.com>
Date:   Tue Dec 28 21:19:06 2021 -0500

    Test stmts after return

commit d781223318cbaf53d2bce40197aad85add23ad1e
Author: mrm2234 <60762357+mrm2234@users.noreply.github.com>
Date:   Tue Dec 28 21:18:34 2021 -0500
```

```
    tests.md

commit a92f4d5657ab169fa427f492b8c9c55aca6d4092
Author: mrm2234 <60762357+mrm2234@users.noreply.github.com>
Date:   Tue Dec 28 21:16:56 2021 -0500

    made fib fib not exp

commit cd7f8e26262899238eb1fdc7aee639599276eed1
Author: Katie Kim <jk4534@columbia.edu>
Date:   Tue Dec 28 21:15:25 2021 -0500

    update tests.md

commit 3a54f569e9dfe50d7e8a3ce522c453bdd6a537d5
Author: yangzhonglin <zhonglinyang98@gmail.com>
Date:   Tue Dec 28 21:08:28 2021 -0500

    Check main function type

commit c0072702f138db0c231a084dbe27a2b5509430d8
Author: mrm2234 <60762357+mrm2234@users.noreply.github.com>
Date:   Tue Dec 28 21:01:42 2021 -0500

    test cases

commit cf4f9584dde6dbc316bc0a218959d8260975f32c
Merge: ed16984 2bd7c89
Author: mrm2234 <60762357+mrm2234@users.noreply.github.com>
Date:   Tue Dec 28 20:51:28 2021 -0500

    Merge branch 'develop' of https://github.com/aapeliv/plt into
   develop

commit ed1698480bd9f63cfbb3de7c581d252c9f5ea659
Author: mrm2234 <60762357+mrm2234@users.noreply.github.com>
Date:   Tue Dec 28 20:51:22 2021 -0500

    default val of char

commit 2bd7c89acb92963e6fa97abcf61d4fee8c4c9307
Author: Katie Kim <jk4534@columbia.edu>
Date:   Tue Dec 28 20:45:53 2021 -0500

    test refactoring and tally
```

```
commit 21836afdfbb0398c0f9d1aea60399a8c962218aa
Author: yangzhonglin <zhonglinyang98@gmail.com>
Date:   Tue Dec 28 20:35:52 2021 -0500

    Get rid of global variables

commit 5a8a4b4870ed65f0cd07ec76e8ce88eaa7ee6d51
Merge: 360b698 898fa01
Author: Aapeli <git@aapelivuorinen.com>
Date:   Wed Dec 29 02:26:57 2021 +0200

    Merge branch 'constructors-and-destructors' into develop

commit 898fa016de33bb37191c3ff9b6fc89042a432500
Author: Aapeli <git@aapelivuorinen.com>
Date:   Wed Dec 29 02:26:35 2021 +0200

    Use constructor in a bit of everything

commit fe3df42e22781ddd8271164b4149001ac2fc0256
Author: Aapeli <git@aapelivuorinen.com>
Date:   Wed Dec 29 02:22:32 2021 +0200

    Fix up block scope bug

commit 4889b40cb6fabab0746564e1539fe54fe6f184c1
Author: Aapeli <git@aapelivuorinen.com>
Date:   Wed Dec 29 02:09:31 2021 +0200

    Get constructors and destructors working if they are defined

commit 360b698f4947f2ef6c4600cfd7e968b5a794ddaa
Author: yangzhonglin <zhonglinyang98@gmail.com>
Date:   Tue Dec 28 17:40:08 2021 -0500

    change locals to scope in checker.ml

commit 6ba0b74a29f36ae7cc939102b838cf0f6e33bd55
Author: Aapeli <git@aapelivuorinen.com>
Date:   Wed Dec 29 00:21:00 2021 +0200

    Add class method as a method param test

commit 51bde1e549ecb4c22163800ea9575c030f307595
Author: Aapeli <git@aapelivuorinen.com>
```

```
Date:   Wed Dec 29 00:13:34 2021 +0200

    Add sample method stuff from presentation

commit 97a125deaf84ac34b07a89a62a914d5abf05c5c3
Author: Aapeli <git@aapelivuorinen.com>
Date:   Tue Dec 28 23:41:06 2021 +0200

    Add some more tests and fix _tags

commit 63c15398a8f8fddd96d31b0b5168582b29e3b9a6
Author: Katie Kim <jk4534@columbia.edu>
Date:   Tue Dec 28 16:00:35 2021 -0500

    cleanup

commit ba766333c956641a1acff628eea8fec880f33b98
Merge: 042c653 a499fd5
Author: Katie Kim <jk4534@columbia.edu>
Date:   Tue Dec 28 15:54:01 2021 -0500

    Merge branch 'develop' of https://github.com/aapeliv/plt into
   develop

commit 042c653b6bb0008c132dd97f468b35fb3973fe63
Author: Katie Kim <jk4534@columbia.edu>
Date:   Tue Dec 28 15:53:55 2021 -0500

    fix tests

commit a499fd55d68187c4d059e456ac99bc9b05a2407c
Merge: a1d61bc 2b270fd
Author: mrm2234 <60762357+mrm2234@users.noreply.github.com>
Date:   Tue Dec 28 15:49:59 2021 -0500

    Merge branch 'develop' of https://github.com/aapeliv/plt into
   develop

commit a1d61bc3b51481f451d08b01536875b77981b376
Author: mrm2234 <60762357+mrm2234@users.noreply.github.com>
Date:   Tue Dec 28 15:49:50 2021 -0500

    ocamlyacc -v parser.mly runs clean

commit 2b270fd4eb0e98a7482058529792e0228e7a7e6a
Merge: 6536cf4 94f36d8
```

```
Author: Katie Kim <jk4534@columbia.edu>
Date:   Tue Dec 28 15:49:33 2021 -0500

    Merge branch 'develop' of https://github.com/aapeliv/plt into
   develop

commit 6536cf48604f71a70253b1d74f05bdc7e7ddacf1
Author: Katie Kim <jk4534@columbia.edu>
Date:   Tue Dec 28 15:49:25 2021 -0500

    enable disabled tests

commit da0f79cf5d1aeb39f204409beafec93b950ebe09
Author: mrm2234 <60762357+mrm2234@users.noreply.github.com>
Date:   Tue Dec 28 15:44:19 2021 -0500

    removed run_one.py and updated test.py to have accuracy rate for
    loggy.txt

commit 94f36d8c278d8dbe140f93645ffc73bcf8603c88
Author: Aapeli <git@aapelivuorinen.com>
Date:   Tue Dec 28 22:31:21 2021 +0200

    Add hello world for report

commit b1b0400699bd3124abc55d2e6ce87d7dd1fc01f0
Merge: 91e9c00 52c99fc
Author: Aapeli <git@aapelivuorinen.com>
Date:   Tue Dec 28 22:29:23 2021 +0200

    Merge branch 'develop' of github.com:aapeliv/plt into develop

commit 52c99fca53ed3a8502c255ff26700aa9701108e1
Author: yangzhonglin <zhonglinyang98@gmail.com>
Date:   Tue Dec 28 15:28:43 2021 -0500

    using my_arr.length

commit 91e9c000233a0b552505d4f1ddfbae2e82547a6d
Author: Aapeli <git@aapelivuorinen.com>
Date:   Tue Dec 28 22:28:08 2021 +0200

    Add compile helper

commit 9a5a12a2cfff7cf93f6a8d062744b12b0466b071
Author: mrm2234 <60762357+mrm2234@users.noreply.github.com>
```

```
Date:   Tue Dec 28 15:22:56 2021 -0500

    revert to old msg

commit 9af53f92bfa9af354e1877bb0ae59620aa314c0e
Author: mrm2234 <60762357+mrm2234@users.noreply.github.com>
Date:   Tue Dec 28 15:16:46 2021 -0500

    Better error msg for wrong type bind + zhonglin's checker change
    for array len

commit a05f0518bb6803d194ec6777e4f68fa83313844b
Merge: 8f8f3a9 712afb9
Author: Aapeli <git@aapelivuorinen.com>
Date:   Tue Dec 28 22:02:47 2021 +0200

    Merge branch 'develop' of github.com:aapeliv/plt into develop

commit 8f8f3a9c0daadcd9731f2b6fa7796869aa38fd3e
Author: Aapeli <git@aapelivuorinen.com>
Date:   Tue Dec 28 22:01:53 2021 +0200

    Clean up initialization codegen

commit 712afb9bfb0b857e0f1e1e08ab462404e559719d
Merge: aef0e17 d1875fe
Author: mrm2234 <60762357+mrm2234@users.noreply.github.com>
Date:   Tue Dec 28 14:53:29 2021 -0500

    Merge branch 'develop' of https://github.com/aapeliv/plt into
   develop

commit aef0e17ee43ffafabe5faa41918de533b68e469c
Author: mrm2234 <60762357+mrm2234@users.noreply.github.com>
Date:   Tue Dec 28 14:53:28 2021 -0500

    char

commit d1875fe775c3d5b69162068b592b6fa8321405a7
Merge: ab7efe4 3926864
Author: Aapeli <git@aapelivuorinen.com>
Date:   Tue Dec 28 21:45:48 2021 +0200

    Merge branch 'develop' of github.com:aapeliv/plt into develop

commit ab7efe4069b7bde2ec218ef06282eb9d3c97cb42
```

```
Author: Aapeli <git@aapelivuorinen.com>
Date:   Tue Dec 28 21:45:43 2021 +0200

    Clean up codegen

commit 392686440d6264f67be076ae9ef89ab2496da138
Author: yangzhonglin <zhonglinyang98@gmail.com>
Date:   Tue Dec 28 14:34:37 2021 -0500

    deleted one array fail test

commit 031f68e90c79698a55f9ce7a8e424d537f49aff3
Merge: 9e7aa8c faada6d
Author: yangzhonglin <zhonglinyang98@gmail.com>
Date:   Tue Dec 28 14:28:55 2021 -0500

    modified index out of bounds checking

commit 9e7aa8c57911626513180af5648f565fa076fa04
Author: yangzhonglin <zhonglinyang98@gmail.com>
Date:   Tue Dec 28 14:27:46 2021 -0500

    modified index out of bounds checking

commit faada6d7eca20d66326fd35ff6694775055ab1a5
Merge: d63b383 9b99c2d
Author: mrm2234 <60762357+mrm2234@users.noreply.github.com>
Date:   Tue Dec 28 14:04:58 2021 -0500

    Merge branch 'develop' of https://github.com/aapeliv/plt into
   develop

commit d63b3833d51b31f3c5548865b8e47d78d2b9612d
Author: mrm2234 <60762357+mrm2234@users.noreply.github.com>
Date:   Tue Dec 28 14:04:55 2021 -0500

    Start of char

commit 9b99c2df1a28124257880a2dbb9feb9a0d68301a
Author: Aapeli <git@aapelivuorinen.com>
Date:   Tue Dec 28 21:04:05 2021 +0200

    Fix for loop

commit 1e8882428221fb3712752572a1787854680f0821
Merge: 66464d1 61c8c69
```

```
Author: Aapeli <git@aapelivuorinen.com>
Date:   Tue Dec 28 20:56:35 2021 +0200

    Merge branch 'develop' of github.com:aapeliv/plt into develop

commit 66464d178961075b95f5c883edf7e19dc1c10b97
Author: Aapeli <git@aapelivuorinen.com>
Date:   Tue Dec 28 20:56:29 2021 +0200

    Implement for loop

commit 61c8c69f097255fa30385387eb68cf77501eb419
Author: yangzhonglin <zhonglinyang98@gmail.com>
Date:   Tue Dec 28 13:55:54 2021 -0500

    test sorting arr

commit 97b2612c00813fc812665b08d398e9541620471e
Merge: 36147bc c297894
Author: Aapeli <git@aapelivuorinen.com>
Date:   Tue Dec 28 20:51:40 2021 +0200

    Merge branch 'develop' of github.com:aapeliv/plt into develop

commit 36147bcac8bee8db5e72fd3aa2f7766039c43087
Author: Aapeli <git@aapelivuorinen.com>
Date:   Tue Dec 28 20:51:04 2021 +0200

    Fix while loop building

commit c297894dbe9fa91f2fdde4bbddec76439e474537
Merge: b239880 7d1e5bc
Author: yangzhonglin <zhonglinyang98@gmail.com>
Date:   Tue Dec 28 13:32:59 2021 -0500

    Merge branch 'develop' of https://github.com/aapeliv/plt into
   develop

commit b239880816690d36b5cfb2fe25ab814918c20dab
Author: yangzhonglin <zhonglinyang98@gmail.com>
Date:   Tue Dec 28 13:29:39 2021 -0500

    test nested while loop

commit e3ffa2b5489c3da0a81dbb5c92e7d71a0579a20e
Merge: 2a803c5 7d1e5bc
```

```
Author: mrm2234 <60762357+mrm2234@users.noreply.github.com>
Date:   Tue Dec 28 13:10:44 2021 -0500

    Merge branch 'develop' of https://github.com/aapeliv/plt into
   develop

commit 2a803c50d899938027ae6d5cd21a8a7f93c2eda9
Author: mrm2234 <60762357+mrm2234@users.noreply.github.com>
Date:   Tue Dec 28 13:10:42 2021 -0500

    new arr test cases

commit 7d1e5bcfa902a9f37c5fdfbbbad1113545c94700
Author: Aapeli <git@aapelivuorinen.com>
Date:   Tue Dec 28 19:44:16 2021 +0200

    Formatting

commit 6a3a365fa2454998824f577be0ec1211483fb64f
Author: Aapeli <git@aapelivuorinen.com>
Date:   Tue Dec 28 19:44:10 2021 +0200

    Resurrect tests

commit 8fa08abd6f8545eeb4e22fa5077f6a5432b84b09
Merge: ead7e59 c19ee59
Author: mrm2234 <60762357+mrm2234@users.noreply.github.com>
Date:   Tue Dec 28 12:03:40 2021 -0500

    Merge branch 'develop' of https://github.com/aapeliv/plt into
   develop

commit ead7e59967f034e766ab4e789bf07e1f74c27180
Author: mrm2234 <60762357+mrm2234@users.noreply.github.com>
Date:   Tue Dec 28 12:03:33 2021 -0500

    arr.length

commit c19ee592be9b2503b38d7351dbc8ff181fb17df6
Merge: 58383f4 7757873
Author: Aapeli <git@aapelivuorinen.com>
Date:   Tue Dec 28 18:55:39 2021 +0200

    Merge branch 'develop' of github.com:aapeliv/plt into develop

commit 58383f4fe772914f730971e876f6936ef93f0611
```

```
Author: Aapeli <git@aapelivuorinen.com>
Date:   Tue Dec 28 18:55:34 2021 +0200

    Add more scoping tests

commit 7757873617d6a427af551efd64341cd011ae639f
Merge: c80f7b1 9b54678
Author: Katie Kim <jk4534@columbia.edu>
Date:   Tue Dec 28 11:54:33 2021 -0500

    Merge branch 'develop' of https://github.com/aapeliv/plt into
   develop

commit c80f7b1aa6047a2b42145054a0e723fbc9fd7ed1
Author: Katie Kim <jk4534@columbia.edu>
Date:   Tue Dec 28 11:54:09 2021 -0500

    fix tests

commit 9b54678218429c5ab6c28835d1af4801b80d264f
Merge: 1747e1c bfec589
Author: Aapeli <git@aapelivuorinen.com>
Date:   Tue Dec 28 18:48:20 2021 +0200

    Merge branch 'develop' of github.com:aapeliv/plt into develop

commit 1747e1c04c6725f5a9a166810d54864641cccfd2
Author: Aapeli <git@aapelivuorinen.com>
Date:   Tue Dec 28 18:48:18 2021 +0200

    Add disabled encapsulation test

commit bfec58950fbc27efe6eafd2a61eebcff08b01f37
Author: mrm2234 <60762357+mrm2234@users.noreply.github.com>
Date:   Tue Dec 28 11:40:58 2021 -0500

    reworked test cases

commit afaa49810a528145f27a3c9a9d254b4ca954f3fd
Merge: 82abdf7 a9c8047
Author: mrm2234 <60762357+mrm2234@users.noreply.github.com>
Date:   Tue Dec 28 11:35:01 2021 -0500

    Merge branch 'develop' of https://github.com/aapeliv/plt into
   develop
```

```
commit 82abdf790dc42535e2664e0dc09fc1352b6309c5
Author: mrm2234 <60762357+mrm2234@users.noreply.github.com>
Date:   Tue Dec 28 11:34:50 2021 -0500

    new tests

commit a9c8047bade54f3109e7aa01310f07ea29cbd371
Merge: 1ecef09 f283553
Author: Aapeli <git@aapelivuorinen.com>
Date:   Tue Dec 28 18:34:20 2021 +0200

    Merge branch 'develop' of github.com:aapeliv/plt into develop

commit 1ecef09737e36b56897321002e8907df99555604
Author: Aapeli <git@aapelivuorinen.com>
Date:   Tue Dec 28 18:34:15 2021 +0200

    Fix up class member access

commit f2835539338634b08909ae88b672f74369ea9584
Author: Katie Kim <jk4534@columbia.edu>
Date:   Tue Dec 28 11:33:55 2021 -0500

    array index out of bounds

commit 8c9cd5c0798bd3e0b2f36538a6b1fcadae24e6d2
Merge: b9b2bae 212336c
Author: mrm2234 <60762357+mrm2234@users.noreply.github.com>
Date:   Tue Dec 28 11:24:28 2021 -0500

    Merge branch 'develop' of https://github.com/aapeliv/plt into
   develop

commit b9b2baebc46629291fb873e03f3f07f72d1616a2
Merge: 28fe41b 820e262
Author: mrm2234 <60762357+mrm2234@users.noreply.github.com>
Date:   Tue Dec 28 11:24:26 2021 -0500

    wrong length arr test case

commit 212336cd48e53633211464031d8ba476407abb8a
Merge: c33b64d 820e262
Author: Aapeli <git@aapelivuorinen.com>
Date:   Tue Dec 28 18:22:54 2021 +0200

    Merge branch 'develop' of github.com:aapeliv/plt into develop
```

```
commit c33b64d8f535f37e25b894028cd7711771da81cc
Author: Aapeli <git@aapelivuorinen.com>
Date:   Tue Dec 28 18:22:47 2021 +0200

    Fix up type_to_t

commit 820e262aff071ad99aa9a8429599be0fe2ddbd62
Merge: 617ddb5 d650f17
Author: Katie Kim <jk4534@columbia.edu>
Date:   Tue Dec 28 11:19:55 2021 -0500

    Merge branch 'develop' of https://github.com/aapeliv/plt into
   develop

commit 617ddb52874b32a9f9b2573364eeeaca96362cbc
Author: Katie Kim <jk4534@columbia.edu>
Date:   Tue Dec 28 11:19:50 2021 -0500

    move arraylength to develop

commit d650f1768e2919bde8941291ed3c369941725465
Author: Aapeli <git@aapelivuorinen.com>
Date:   Tue Dec 28 18:15:35 2021 +0200

    Add me to class methods

commit c2a044299fa30b72571950635f4472b40c64d0d3
Merge: 442ec8f 47c182e
Author: Aapeli <git@aapelivuorinen.com>
Date:   Tue Dec 28 18:12:29 2021 +0200

    Merge branch 'develop' of github.com:aapeliv/plt into develop

commit 442ec8ff54e1736f980bf352297d969498f1d381
Author: Aapeli <git@aapelivuorinen.com>
Date:   Tue Dec 28 18:12:26 2021 +0200

    Make simple methods test right thing

commit 28fe41b133b2b4edd78368818da73037a32b7025
Merge: 76d9aee 47c182e
Author: mrm2234 <60762357+mrm2234@users.noreply.github.com>
Date:   Tue Dec 28 11:12:11 2021 -0500

    array indexing
```

```
commit a7597d23b997849346c3ab25d60ff5b54a86d97c
Author: Aapeli <git@aapelivuorinen.com>
Date:   Tue Dec 28 18:06:12 2021 +0200

    Fix up most of class methods and me

commit 76d9aee0ce29a0febd7e9ee2eafe7835a570754b
Merge: b2a466e 36b9c96
Author: mrm2234 <60762357+mrm2234@users.noreply.github.com>
Date:   Tue Dec 28 11:00:09 2021 -0500

    ony one tests folder'

commit e88551dc4d3422b935c76501fc63dfb1062a2a11
Author: Aapeli <git@aapelivuorinen.com>
Date:   Tue Dec 28 17:58:31 2021 +0200

    Work on class methods

commit 47c182e910ace28b61e8012569036fa5288efb9c
Author: ktjkim <37836749+ktjkim@users.noreply.github.com>
Date:   Tue Dec 28 10:40:33 2021 -0500

    Update test-arr_init_decl.kazm

commit 34fa45da46f6408f466357b4f9659e60018037d6
Merge: 6de3bae e0d1451
Author: Katie Kim <jk4534@columbia.edu>
Date:   Tue Dec 28 10:39:01 2021 -0500

    fix tests

commit 6de3baec3c9898053d0ef2be94c860c664d5233e
Author: Katie Kim <jk4534@columbia.edu>
Date:   Tue Dec 28 10:38:20 2021 -0500

    fix tests

commit e0d1451cc92a99a3db2bdfca2a7affa8e146f1b9
Author: Katie Kim <jk4534@columbia.edu>
Date:   Tue Dec 28 10:31:40 2021 -0500

    fix tests

commit 388ffb78bd3810b973028c127543bc83a179222f
```

```
Author: Katie Kim <jk4534@columbia.edu>
Date:   Tue Dec 28 10:17:55 2021 -0500

    type checking for declaration with initialization

commit e6d0422c746d317353a80de5749b2f030972d6f3
Author: Katie Kim <jk4534@columbia.edu>
Date:   Tue Dec 28 10:05:48 2021 -0500

    declaration with initialization for int double bool string

commit ed877380b198f89b56c9c3b7503c4889bbb8c5b7
Author: Aapeli <git@aapelivuorinen.com>
Date:   Tue Dec 28 16:39:43 2021 +0200

    Start on class method name mangling and me pointer

commit 36b9c96ac7440085938f6d87d52e20e193a99bdd
Author: Aapeli <git@aapelivuorinen.com>
Date:   Tue Dec 28 14:02:22 2021 +0200

    Add scope in parser

commit 6217093c29a3d75dc5230975165cbc1c99c57273
Author: Aapeli <git@aapelivuorinen.com>
Date:   Tue Dec 28 13:48:38 2021 +0200

    Unreverse stmts and remove braces from block

commit f3c9a427c20c5cfe2ed09387ca43399cb3d090ca
Author: Aapeli <git@aapelivuorinen.com>
Date:   Tue Dec 28 13:45:46 2021 +0200

    Add scope destruction

commit e2b0eedcf366286b14ef257fa08cbdb0ee382ace
Author: Aapeli <git@aapelivuorinen.com>
Date:   Tue Dec 28 13:26:51 2021 +0200

    Add option to run only selected tests

commit 099ef52f83ef6c520f6a01c36b060ddae0b602a0
Author: Aapeli <git@aapelivuorinen.com>
Date:   Tue Dec 28 13:03:33 2021 +0200

    Move arrays onto the heap and use inbounds gep
```

```
commit 4a52dd5cd11f17698c6da4838df1d7c8975883cd
Author: Aapeli <git@aapelivuorinen.com>
Date:   Tue Dec 28 12:32:49 2021 +0200

    Use LLVM builtin in malloc builder

commit 197bbb29d4728a80b759f83eff58ef8136034e87
Author: Aapeli <git@aapelivuorinen.com>
Date:   Tue Dec 28 12:28:17 2021 +0200

    Clean up array element computation

commit c43366736a67fd4d9d043a69247f389fe923e047
Author: Aapeli <git@aapelivuorinen.com>
Date:   Tue Dec 28 12:22:05 2021 +0200

    Clean up array assign

commit 7058a95bb15b7a034908ae7a66136a5b69a4611f
Author: Aapeli <git@aapelivuorinen.com>
Date:   Tue Dec 28 12:18:29 2021 +0200

    Clean up array access

commit 2e034810678f376ef161754b0e96a92f4a8ce368
Author: Aapeli <git@aapelivuorinen.com>
Date:   Tue Dec 28 11:52:55 2021 +0200

    Clean up

commit a9626e6941547c341d7584c3978be1db3607cdce
Author: Aapeli <git@aapelivuorinen.com>
Date:   Tue Dec 28 11:42:39 2021 +0200

    Tabs to spaces

commit cd8496510ec8c4a9fa1e4bc47e1d3ccbc4e73c92
Author: Aapeli <git@aapelivuorinen.com>
Date:   Tue Dec 28 11:40:23 2021 +0200

    Clean up

commit c286ea7fb40d1a14be1754d600218d61078a97ef
Author: Aapeli <git@aapelivuorinen.com>
Date:   Tue Dec 28 11:37:31 2021 +0200
```

```
    Fix array literal SAST type and checking

commit 7bc4fe5f26af728465550ca9553e1e6325a89c76
Author: Aapeli <git@aapelivuorinen.com>
Date:   Tue Dec 28 11:04:53 2021 +0200

    Add failing test for array checker bug

commit 876cef0092b06ea700ad6e604f41e2f86ca31f72
Author: Aapeli <git@aapelivuorinen.com>
Date:   Tue Dec 28 10:56:11 2021 +0200

    Clean up

commit 4302dcafe614cf37353f3c2e318b1ff99f02d805
Author: Aapeli <git@aapelivuorinen.com>
Date:   Tue Dec 28 10:47:34 2021 +0200

    Move temptests into tests

commit caa986e2321aebe47a53d7d0d5635b18b177049b
Author: yangzhonglin <zhonglinyang98@gmail.com>
Date:   Tue Dec 28 00:17:39 2021 -0500

    string type added and checker checks undefined class

commit b2a466ea9e7b5e18dfaf39c171167ca12c00b665
Author: mrm2234 <60762357+mrm2234@users.noreply.github.com>
Date:   Mon Dec 27 17:34:43 2021 -0500

    test-class operator=

commit 2513d1c09014b0a3aec4c3adfbcb9768fbae104c
Merge: 4e547db e4314e2
Author: Aapeli <git@aapelivuorinen.com>
Date:   Mon Dec 27 21:15:01 2021 +0200

    Merge branch 'classes2' into develop

commit e4314e2e7330a194c76230b8b6841ffd5ec70bdd
Author: Aapeli <git@aapelivuorinen.com>
Date:   Mon Dec 27 21:14:48 2021 +0200

    Clean up
```

```
commit d1739417468 67f363160dfb05c72bda299a8348c
Author: Aapeli <git@aapelivuorinen.com>
Date:   Mon Dec 27 20:12:07 2021 +0200

    Clean up

commit 4e547dbd99edad08832a48306580d1f239583f11
Author: Aapeli <git@aapelivuorinen.com>
Date:   Mon Dec 27 21:07:21 2021 +0200

    Clean up

commit b7c654c32aa83f5617ab8cc875d44ce7cc3af128
Author: Aapeli <git@aapelivuorinen.com>
Date:   Mon Dec 27 21:06:21 2021 +0200

    Move tests into tests/

commit 9aff8fae90b2cfb76439e62d1435b4bdd6eb2fec
Merge: c46cd3a 98e6fd6
Author: ktjkim <37836749+ktjkim@users.noreply.github.com>
Date:   Mon Dec 27 14:00:49 2021 -0500

    Merge pull request #11 from aapeliv/kt

    Arrays

commit 98e6fd69be1de53698b06168f246fad3a1eb31ce
Author: Katie Kim <jk4534@columbia.edu>
Date:   Mon Dec 27 13:27:46 2021 -0500

    change default value to bool

commit 07822244f022940aaf41b6782baf1d2a082fae64
Author: Katie Kim <jk4534@columbia.edu>
Date:   Mon Dec 27 13:21:02 2021 -0500

    support for array declaration without initialization

commit ee6d53378316edbd2f93dc5d85d4068d7c460f45
Merge: 66080f3 c46cd3a
Author: Katie Kim <jk4534@columbia.edu>
Date:   Mon Dec 27 13:07:27 2021 -0500

    merge develop into kt
```

```
commit c46cd3a96e0ba603c3c70cf7f93bc6cbd5424882
Author: yangzhonglin <zhonglinyang98@gmail.com>
Date:   Mon Dec 27 03:11:06 2021 -0500

    TODO in codegen

commit 66080f3b16e9923da232740605c8e597aaf6e72c
Author: mrm2234 <60762357+mrm2234@users.noreply.github.com>
Date:   Sun Dec 26 23:32:05 2021 -0500

    negative indexing

commit 550872e0d0f830c64f75146f9acef57543864e94
Merge: c704711 7bd136c
Author: mrm2234 <60762357+mrm2234@users.noreply.github.com>
Date:   Sun Dec 26 23:29:52 2021 -0500

    Merge branch 'kt' of https://github.com/aapeliv/plt into kt

commit c7047110cf939c23c053a09a4a24440487efedf0
Author: mrm2234 <60762357+mrm2234@users.noreply.github.com>
Date:   Sun Dec 26 23:29:34 2021 -0500

    Index oob

commit 7bd136cc841059eba64f7cdaf574982b849c841d
Author: Katie Kim <jk4534@columbia.edu>
Date:   Sun Dec 26 23:13:36 2021 -0500

    delete unnecessary files

commit 8ff110c36f4f5db6981680375e2d143da4d29988
Author: Katie Kim <jk4534@columbia.edu>
Date:   Sun Dec 26 22:45:29 2021 -0500

    stuff works

commit fab0cad9a8e032dfe5a3d09ce22490acb8218701
Author: Katie Kim <jk4534@columbia.edu>
Date:   Sun Dec 26 22:28:41 2021 -0500

    checkpoint: arrayaccess seems to work

commit 20b75c71ded94da76e41e5b0bde99b47ed900894
Author: Katie Kim <jk4534@columbia.edu>
Date:   Sun Dec 26 21:44:27 2021 -0500
```

```
    checkpoint: initialize with array e.g. array int[3] = [1, 2, 3];
    runs

commit e5563110ea3137d6f491a80dbc55b5c38b079fe1
Author: Katie Kim <jk4534@columbia.edu>
Date:   Sun Dec 26 21:22:13 2021 -0500

    checkpoint: arraylit seems to work

commit 2a183bcc27835fc84cf6006685277034d64f7dd4
Author: mrm2234 <60762357+mrm2234@users.noreply.github.com>
Date:   Sun Dec 26 15:29:27 2021 -0500

    bool

commit 18e003d6bbcfec3ea4cea842173b91a19d617733
Author: mrm2234 <60762357+mrm2234@users.noreply.github.com>
Date:   Sun Dec 26 14:56:45 2021 -0500

    moving tests

commit 588996432bc14f89c486edad288aec687fd8e74b
Author: mrm2234 <60762357+mrm2234@users.noreply.github.com>
Date:   Sun Dec 26 14:11:51 2021 -0500

    Moved test files around

commit 3427b73ccd8e21a33120cea370d23e351ca26f34
Author: mrm2234 <60762357+mrm2234@users.noreply.github.com>
Date:   Sun Dec 26 13:36:43 2021 -0500

    Fixed div by 0 to work with div and mod of int

commit 0ddd87c0c59402e6885c0d7e6c42ccd988e6df49
Author: mrm2234 <60762357+mrm2234@users.noreply.github.com>
Date:   Sun Dec 26 13:20:49 2021 -0500

    Div by 0 working for int

commit edd22a762fffb533c48176aa485a886959af225c
Author: mrm2234 <60762357+mrm2234@users.noreply.github.com>
Date:   Sun Dec 26 12:30:21 2021 -0500

    %
```

```
commit b7c142373e3e03428762321e1ca54639cfa3a65f
Author: mrm2234 <60762357+mrm2234@users.noreply.github.com>
Date:   Sun Dec 26 12:14:44 2021 -0500

    new tests

commit 1cd594a94d01295e7c5c911bf47f66f58bd6f9ab
Author: Aapeli <git@aapelivuorinen.com>
Date:   Sun Dec 26 18:57:45 2021 +0200

    Add test naming convention

commit 9cd116e2176ad7707b6d5345d8627d7673ab5fdd
Merge: cd21024 c76ab51
Author: Aapeli <git@aapelivuorinen.com>
Date:   Thu Dec 23 18:54:15 2021 +0200

    Merge branch 'class-methods' into develop

commit cd2102456ebb2ebd198b6d83a879c11302eed7df
Author: Aapeli <git@aapelivuorinen.com>
Date:   Thu Dec 23 18:52:24 2021 +0200

    Disable failing tests

commit 6c5bd17c44ad13e3947d4638af5ccd8f32645f24
Author: mrm2234 <60762357+mrm2234@users.noreply.github.com>
Date:   Thu Dec 23 11:35:58 2021 -0500

    Renaming

commit b5a42cf86bdfbb32913ce19e3c652b583ab2abc6
Author: mrm2234 <60762357+mrm2234@users.noreply.github.com>
Date:   Thu Dec 23 11:29:51 2021 -0500

    Renamed tests for test/fail

commit 979e71b7faa709ad5b9fc72cbe3307b85b2afd47
Author: mrm2234 <60762357+mrm2234@users.noreply.github.com>
Date:   Thu Dec 23 11:07:47 2021 -0500

    new tests

commit 282e5c3a33ab7625a83cbba29355e8882e85a15c
Merge: a94a0a8 21f441a
Author: mrm2234 <60762357+mrm2234@users.noreply.github.com>
```

```
Date:   Sun Dec 19 22:29:03 2021 -0500

    Merge branch 'develop' of https://github.com/aapeliv/plt into
   develop

commit a94a0a86190c619d8d24c4e168b690cfbb82f449
Author: mrm2234 <60762357+mrm2234@users.noreply.github.com>
Date:   Sun Dec 19 22:28:44 2021 -0500

    test files

commit 21f441a53a7229c42cb325a90c65ad54976186b0
Author: mrm2234 <60762357+mrm2234@users.noreply.github.com>
Date:   Sun Dec 19 22:15:49 2021 -0500

    Added overleaf link

commit c76ab5177eeb04d34b5c9f25764bbc52a40f0c41
Author: yangzhonglin <zhonglinyang98@gmail.com>
Date:   Sun Dec 19 21:14:34 2021 -0500

    SCall works TODO: SInitialize with expr

commit 4a8cff1c0ed7df24e9572be268c7dab90525bf67
Author: yangzhonglin <zhonglinyang98@gmail.com>
Date:   Sun Dec 19 20:31:29 2021 -0500

    TODO: SCall

commit 86a3b913fc2a01f2e518784e4037e7984996c78e
Author: yangzhonglin <zhonglinyang98@gmail.com>
Date:   Sun Dec 19 19:06:49 2021 -0500

    Initialize without expr fixed

commit 3199e7901a0e8791cda758a84a3897654d39c04f
Author: yangzhonglin <zhonglinyang98@gmail.com>
Date:   Sun Dec 19 18:30:32 2021 -0500

    add check_class

commit 12a1c2ffe23ba569801a665575b6a78c8885a6da
Author: ktjkim <37836749+ktjkim@users.noreply.github.com>
Date:   Sun Dec 19 11:47:18 2021 -0500

    Update readme.md
```

```
commit d35b67fbd2b103c74abfcdda6f34ec1f8285f078
Author: mrm2234 <60762357+mrm2234@users.noreply.github.com>
Date:   Sat Dec 18 00:20:39 2021 -0500

    run_one.py

commit 3e78d74ee8b4dd5a2e8fd36b9ef3955b6df541d8
Author: mrm2234 <60762357+mrm2234@users.noreply.github.com>
Date:   Fri Dec 17 23:40:57 2021 -0500

    Tests

commit 44aec26495a5e6415064aa45625edfc4e316b644
Author: mrm2234 <60762357+mrm2234@users.noreply.github.com>
Date:   Mon Dec 13 18:15:31 2021 -0500

    reassignment

commit 41d42b5709e394f3735af47eb4dc666470d7855a
Author: mrm2234 <60762357+mrm2234@users.noreply.github.com>
Date:   Mon Dec 13 18:14:36 2021 -0500

    changed one test

commit 64af40de2c91427c2ab3cb64c398ec2634312c45
Author: mrm2234 <60762357+mrm2234@users.noreply.github.com>
Date:   Mon Dec 13 17:49:57 2021 -0500

    new tests

commit 5e44243a7e601a0cceff2192dfdde9550e6f02e5
Author: Aapeli <git@aapelivuorinen.com>
Date:   Mon Dec 13 00:25:25 2021 -0500

    WIP: start on class methods

commit d77ce87fe5b3e8aa498fc20fb38fbbd594f45c33
Author: Aapeli <git@aapelivuorinen.com>
Date:   Sun Dec 12 11:40:08 2021 -0500

    Make it possible to pass classes to functions

commit 382159b6bfaa59afcc32d2fb312b7cd161fc92a3
Author: Aapeli <git@aapelivuorinen.com>
Date:   Sun Dec 12 11:25:01 2021 -0500
```

```
    Move classes onto the heap

commit c5035b652f7a8b0d91ab1e9aa470a3041d059940
Merge: 32d6969 bae55c9
Author: Aapeli <git@aapelivuorinen.com>
Date:   Sat Dec 11 19:40:59 2021 -0500

    Merge branch 'classes' into develop

commit bae55c9b08d54f082c37fb12f3329bd210a5c058
Author: Aapeli <git@aapelivuorinen.com>
Date:   Sat Dec 11 19:28:10 2021 -0500

    Implement fully qualified assignment

commit c39a793d1e737cfab5bade34fb1a0ff43381f13b
Merge: 83e572f 8d61f92
Author: Aapeli <git@aapelivuorinen.com>
Date:   Sat Dec 11 18:42:51 2021 -0500

    Merge branch 'classes' of github.com:aapeliv/plt into classes

commit 83e572f8ef87ee2555e3637b426d59fdf4089fb4
Author: Aapeli <git@aapelivuorinen.com>
Date:   Sat Dec 11 18:33:07 2021 -0500

    Get class member access to work

commit 5eff562138c6c3243ce9ca6ec70fc1a2e3a8a41e
Author: Aapeli <git@aapelivuorinen.com>
Date:   Sat Dec 11 18:17:46 2021 -0500

    WIP

commit 8d61f92d5b466cad12c472748ddec40624922cc5
Author: mrm2234 <60762357+mrm2234@users.noreply.github.com>
Date:   Sat Dec 11 18:17:19 2021 -0500

    parser

commit 48c96db8faf2e592e51ca5f7eac1a6b6964323d1
Author: mrm2234 <60762357+mrm2234@users.noreply.github.com>
Date:   Sat Dec 11 18:08:19 2021 -0500

    Changed naming convention
```

```
commit 7bff65637cabcea26f0f999745b37c8cacf920da
Author: mrm2234 <60762357+mrm2234@users.noreply.github.com>
Date:   Sat Dec 11 17:50:25 2021 -0500

    Constructor

commit 2b44be197b059b7a7e49dd9473aa573294f6a9d2
Author: Aapeli <git@aapelivuorinen.com>
Date:   Sat Dec 11 17:07:47 2021 -0500

    Temp fix for parser issue with class keyword

commit 55684096dd9e83323caf37b2778956c1f6f260c9
Author: Aapeli <git@aapelivuorinen.com>
Date:   Sat Dec 11 17:03:06 2021 -0500

    Temp fix tests

commit 0f7c9d579b8c5eaca72081b756bb623155f18633
Author: Aapeli <git@aapelivuorinen.com>
Date:   Sat Dec 11 16:56:13 2021 -0500

    Figure out named structs

commit 27abc37d69453f1d4d844bb91b88ea933b7c0b3f
Author: Aapeli <git@aapelivuorinen.com>
Date:   Sat Dec 11 16:52:05 2021 -0500

    WIP: declare structs

commit f420ab58af3dcce9c63283825d6eadd8cf458077
Author: Aapeli <git@aapelivuorinen.com>
Date:   Sat Dec 11 16:35:24 2021 -0500

    Add option to only run last failed tests to runner

commit 4311c5afab35f63d040c832c141a6f6f7e29af00
Author: Aapeli <git@aapelivuorinen.com>
Date:   Sat Dec 11 16:22:43 2021 -0500

    Start on struct codegen

commit e689e4b97d528e3a0e86c50deda23553a1e51cc4
Author: mrm2234 <60762357+mrm2234@users.noreply.github.com>
Date:   Sat Dec 11 16:07:05 2021 -0500
```

```
    Create team_project.out

commit cec5de2ee5ff90c44dd7f875128600bbe867b5b8
Author: mrm2234 <60762357+mrm2234@users.noreply.github.com>
Date:   Sat Dec 11 16:06:28 2021 -0500

    First Class

commit a5be28e936f068be659ac99ef0835f02263435fd
Author: Aapeli <git@aapelivuorinen.com>
Date:   Sat Dec 11 16:02:47 2021 -0500

    Add structs to parser

commit 32d6969de163acf22cd95221438e96303660741b
Author: Aapeli <git@aapelivuorinen.com>
Date:   Sat Dec 11 12:08:22 2021 -0500

    Add a test

commit 1babf562e20c7f480994f18bf462c011cf8edb7a
Author: Aapeli <git@aapelivuorinen.com>
Date:   Sat Dec 11 11:30:26 2021 -0500

    Add reassignment test

commit 8a74d8a494220d4554c38723413f25a0acf9c862
Author: Aapeli <git@aapelivuorinen.com>
Date:   Sat Dec 11 00:36:34 2021 -0500

    Enable int arg test

commit a5ca9327d527603d324f287982558a865dbeb83e
Author: Aapeli <git@aapelivuorinen.com>
Date:   Sat Dec 11 00:35:00 2021 -0500

    Add variable based tests

commit dd9aed37bd47baaf698c67b0da5a1c8132a01d41
Author: Aapeli <git@aapelivuorinen.com>
Date:   Fri Dec 10 23:53:18 2021 -0500

    Clean up

commit b3764afa93f5fbbf46213fa51f1c4de168aa12ef
```

```
Author: Aapeli <git@aapelivuorinen.com>
Date:   Fri Dec 10 23:50:12 2021 -0500

    Implement function args

commit c66b34eaebd537cbbdabbd5f548b53ae35fd5a75
Merge: b3d07df a44179a
Author: Aapeli <git@aapelivuorinen.com>
Date:   Fri Dec 10 23:09:48 2021 -0500

    Merge branch 'scopes' into develop

commit b3d07df85215078b241d6df90c7cf3f7ad15f845
Merge: 242cf61 f4ec8cf
Author: Aapeli <git@aapelivuorinen.com>
Date:   Fri Dec 10 23:09:38 2021 -0500

    Merge branch 'develop' of github.com:aapeliv/plt into develop

commit a44179a1d27bb853615ed67237435a0e1bf60dca
Author: Aapeli <git@aapelivuorinen.com>
Date:   Fri Dec 10 23:09:25 2021 -0500

    Add comments

commit 5b506ba16df374b8a68715cd21f12ba36940cb54
Author: Aapeli <git@aapelivuorinen.com>
Date:   Fri Dec 10 23:07:26 2021 -0500

    Types like a Pro

commit ed0bb04c942ec2713aa66aa4301217baf818a5ff
Author: Aapeli <git@aapelivuorinen.com>
Date:   Fri Dec 10 22:53:25 2021 -0500

    Clean up

commit 33386a4c994e9882caa2bf5b74b510cfa14e2334
Author: Aapeli <git@aapelivuorinen.com>
Date:   Fri Dec 10 22:49:28 2021 -0500

    Fix while bug

commit 7c4f301d30227c1e517543e5b86ee324189c075e
Author: Aapeli <git@aapelivuorinen.com>
Date:   Fri Dec 10 22:42:53 2021 -0500
```

```
    Try to add ctx

commit 242cf61169708f0aa6f940fe24eca55fcd2f8e6e
Author: Aapeli <git@aapelivuorinen.com>
Date:   Fri Dec 10 21:32:10 2021 -0500

    Start work on scopes

commit d9c0d348b0541e5683194f47e7f1e09213f28e91
Author: Aapeli <git@aapelivuorinen.com>
Date:   Fri Dec 10 21:16:06 2021 -0500

    Add some variable tests

commit f4ec8cf47e3871586978c77fd08c9fe6144109d4
Author: mrm2234 <60762357+mrm2234@users.noreply.github.com>
Date:   Fri Dec 10 21:13:59 2021 -0500

    file #s

commit 20b4cbf986987591072914ad5c06d00b0cc3a164
Merge: 3af2746 aaffec5
Author: mrm2234 <60762357+mrm2234@users.noreply.github.com>
Date:   Fri Dec 10 21:10:03 2021 -0500

    Merge branch 'develop' of https://github.com/aapeliv/plt into
   develop

commit 3af27469b1dc1672848bb94ddd156c3ff457982f
Author: mrm2234 <60762357+mrm2234@users.noreply.github.com>
Date:   Fri Dec 10 21:09:54 2021 -0500

    Makefile

commit aaffec554a1cc0d9e80814bac4bb6c803ab9ba8e
Author: Aapeli <git@aapelivuorinen.com>
Date:   Fri Dec 10 21:08:51 2021 -0500

    Add loggy.txt to .gitignore

commit c2ea491c394cc5d1fde00abaec15f32618d55bd5
Merge: 3ec9284 b870cfe
Author: Aapeli <git@aapelivuorinen.com>
Date:   Fri Dec 10 21:07:54 2021 -0500
```

```
    Merge branch 'develop' of github.com:aapeliv/plt into develop

commit 3ec92841b87efbe14c6a4d792f15ee9334e558a3
Author: Aapeli <git@aapelivuorinen.com>
Date:   Fri Dec 10 21:07:50 2021 -0500

    Implement user defined function arguments

commit b870cfee29d1c58dc095e43212bc5265b82680ff
Author: mrm2234 <60762357+mrm2234@users.noreply.github.com>
Date:   Fri Dec 10 21:07:40 2021 -0500

    test.py loggy.txt

commit 93432d0e3a4dbd0f3792ef6604970cf31bd3da0f
Author: Aapeli <git@aapelivuorinen.com>
Date:   Fri Dec 10 19:39:33 2021 -0500

    Disable failing test

commit 9c97c03905817235f6ccc3db34ff12219a82c4b5
Author: Aapeli <git@aapelivuorinen.com>
Date:   Fri Dec 10 19:37:50 2021 -0500

    Fix github actions

commit 7283877a1f52349df66ef18f7bacb6bf9d9d244e
Author: Aapeli <git@aapelivuorinen.com>
Date:   Fri Dec 10 19:35:56 2021 -0500

    Fix github actions

commit bbb985fcbe0e188c20859fdba6130b76dabe165d
Author: Aapeli <git@aapelivuorinen.com>
Date:   Fri Dec 10 19:30:50 2021 -0500

    Run opam init

commit 8b1ad1cc8dfc81b8ac8fcb78bec1d5e4e49f417e
Author: Aapeli <git@aapelivuorinen.com>
Date:   Fri Dec 10 19:27:13 2021 -0500

    Also print stderr if kazm fails to build

commit a372a949099ff4c3032a1bad5670a8b963fa6ff4
Author: Aapeli <git@aapelivuorinen.com>
```

```
Date:   Fri Dec 10 19:24:54 2021 -0500

    Return 0 only on all tests passing

commit 210dd5731ec13bc4a1fbd1583ba8b03d2d98416e
Author: Aapeli <git@aapelivuorinen.com>
Date:   Fri Dec 10 19:22:50 2021 -0500

    Add github action test

commit 8c018c37cac904e0563d5d6a22fb1289755a8bf5
Author: Aapeli <git@aapelivuorinen.com>
Date:   Fri Dec 10 19:19:18 2021 -0500

    Add while return test

commit 6cbaa3226e9ea39706973396136a887b00e81c14
Author: Aapeli <git@aapelivuorinen.com>
Date:   Fri Dec 10 19:16:16 2021 -0500

    Change behavior of empty return

commit 8a31d331cc29d31816039fab89994ee2063a16cd
Merge: 4694dd5 3b85fe2
Author: Aapeli <git@aapelivuorinen.com>
Date:   Fri Dec 10 19:09:27 2021 -0500

    Merge branch 'develop' of github.com:aapeliv/plt into develop

commit 4694dd53df10b65fb0e68d5b809d53ba6178babf
Author: Aapeli <git@aapelivuorinen.com>
Date:   Fri Dec 10 19:09:08 2021 -0500

    Add return type mismatch tests

commit e04aba4c07b9c22adb6385fcaf79ec6d3c23ce80
Author: Aapeli <git@aapelivuorinen.com>
Date:   Fri Dec 10 19:05:46 2021 -0500

    Clean up

commit 4a799a8e585c5230d0fb5d851e1b3f790a0fa9f7
Author: Aapeli <git@aapelivuorinen.com>
Date:   Fri Dec 10 19:05:31 2021 -0500

    Fix missing terminator in conditionals
```

```
commit 4db0cf83020a5736f49e6ce17af0183dae1da0e1
Author: Aapeli <git@aapelivuorinen.com>
Date:   Fri Dec 10 18:55:44 2021 -0500

    Abstract terminator builder a bit

commit 84c661c09cc86445be75c5826053d317f6b755ee
Author: Aapeli <git@aapelivuorinen.com>
Date:   Fri Dec 10 18:51:28 2021 -0500

    Add missing return at end of function bb

commit 3b85fe26990d4aee06310a1737a4d30cdd27e38f
Author: mrm2234 <60762357+mrm2234@users.noreply.github.com>
Date:   Fri Dec 10 18:33:57 2021 -0500

    Update Makefile

commit 8c43f8c35655a5c67f62d8ff89cf6366c00a3041
Author: mrm2234 <60762357+mrm2234@users.noreply.github.com>
Date:   Fri Dec 10 18:31:45 2021 -0500

    Update Makefile

commit 841ec4dd23506534e8a92ba02c0a8ef866e0ac49
Author: mrm2234 <60762357+mrm2234@users.noreply.github.com>
Date:   Fri Dec 10 18:29:18 2021 -0500

    fixed testall

commit 4382cab6722e641444fd4e44b4334125bb801608
Author: Aapeli <git@aapelivuorinen.com>
Date:   Fri Dec 10 18:18:34 2021 -0500

    Add nested if tests

commit 7f5e5d96c5ed5e2a2fe0ae42aa55b7551f3136c6
Author: Aapeli <git@aapelivuorinen.com>
Date:   Fri Dec 10 18:03:31 2021 -0500

    Add nested ifs test

commit c346534272e0b6bd733542b37050417677673a87
Author: Aapeli <git@aapelivuorinen.com>
Date:   Fri Dec 10 17:59:26 2021 -0500
```

```
    Add duplicate function test

commit b766b3c2dc1b4b0d72f5a0d10aebd4e5b41614b6
Author: Aapeli <git@aapelivuorinen.com>
Date:   Fri Dec 10 17:53:47 2021 -0500

    Add user function sig mismatch test

commit 5bc64c606557270581135112f36e33e66800ab7e
Author: Aapeli <git@aapelivuorinen.com>
Date:   Fri Dec 10 17:49:10 2021 -0500

    Clean up

commit e173cf86bba1bf288ac736d09edbc5ce83339aa8
Author: Aapeli <git@aapelivuorinen.com>
Date:   Fri Dec 10 17:43:16 2021 -0500

    Add type mistmatch test and support for compile error tests

commit 643714a5cafb87aaf41a0aa99ec6f0fa7ad901ce
Author: Aapeli <git@aapelivuorinen.com>
Date:   Fri Dec 10 17:33:59 2021 -0500

    Clean up

commit dab421d7bb305539d44d611e2a7707d658fa0cbe
Author: Aapeli <git@aapelivuorinen.com>
Date:   Fri Dec 10 17:28:27 2021 -0500

    Add test instructions into readme

commit 795648870e1e4971bf67a70d2dd4fa0624907c2f
Author: Aapeli <git@aapelivuorinen.com>
Date:   Fri Dec 10 17:26:25 2021 -0500

    Clean up

commit 6352a8217f371bb0c1e40033db49b8e23064fab3
Author: Aapeli <git@aapelivuorinen.com>
Date:   Fri Dec 10 17:25:21 2021 -0500

    Run tests in deterministic order

commit d3d437834b5062b502251ad91d660700bbe2bcc2
```

```
Author: Aapeli <git@aapelivuorinen.com>
Date:   Fri Dec 10 17:24:47 2021 -0500

    Add void return test

commit c0d06628ca83952d5e2f5b5afa07d09f905934db
Author: Aapeli <git@aapelivuorinen.com>
Date:   Fri Dec 10 17:22:24 2021 -0500

    Clean up testing

commit 70dd822317f3645755ce2e82719d13af38a537b4
Author: Aapeli <git@aapelivuorinen.com>
Date:   Fri Dec 10 17:17:55 2021 -0500

    Add some tests

commit cdbcd5236e9476ad3b8f7582a68613b3479cf76d
Author: Aapeli <git@aapelivuorinen.com>
Date:   Fri Dec 10 15:29:48 2021 -0500

    Add John's dockerfile

commit 7c6d35c2c62341e2e22e9182383d4c2a07a3a1d8
Author: Aapeli <git@aapelivuorinen.com>
Date:   Fri Dec 10 14:59:14 2021 -0500

    Clean up

commit b9c1419bd229da9fde5e21bfdfc0daa8bbe5af0f
Author: Aapeli <git@aapelivuorinen.com>
Date:   Fri Dec 10 14:58:47 2021 -0500

    Add more generated stuff to .gitignore

commit 655e3192a1dc8aae31cc6540dbd135f4bed4c24d
Author: Aapeli <git@aapelivuorinen.com>
Date:   Fri Dec 10 14:57:04 2021 -0500

    Clean up

commit a6ddcbe0c1d45849ca44cae112b32f5c23e841b4
Merge: f9b556e 95e2090
Author: yangzhonglin <zhonglinyang98@gmail.com>
Date:   Thu Dec 9 16:49:42 2021 -0500
```

```
    Merge branch 'develop' of https://github.com/aapeliv/plt into
   develop

    hello_world fixed

commit f9b556ef013d60cc94fe1e9305f54be4e7532acb
Author: yangzhonglin <zhonglinyang98@gmail.com>
Date:   Thu Dec 9 16:49:23 2021 -0500

    hello_world fixed

commit 95e20907775d5036faa983efcea89e479215b885
Author: Zhonglin Yang <55119328+yzl1998@users.noreply.github.com>
Date:   Mon Dec 6 11:07:07 2021 -0500

    Update readme.md

commit b0fd60035acca30fd2c86de574f7333ebc230af5
Author: Zhonglin Yang <55119328+yzl1998@users.noreply.github.com>
Date:   Mon Dec 6 02:00:50 2021 -0500

    Update readme.md

commit 378a45e7ffb8cdebcdc591551e862cdef7698dc1
Author: yangzhonglin <zhonglinyang98@gmail.com>
Date:   Mon Dec 6 01:59:02 2021 -0500

    codegen fixed, sast_test can compile now

commit d6126844fbba3720ee5676ed529e72b2574f6e26
Merge: edc48c1 8cc015f
Author: yangzhonglin <zhonglinyang98@gmail.com>
Date:   Fri Dec 3 15:18:28 2021 -0500

    merge develop_sast and develop

commit edc48c1f900938d5d388ee935908ace2b9247259
Author: yangzhonglin <zhonglinyang98@gmail.com>
Date:   Thu Dec 2 18:05:47 2021 -0500

    sast checker

commit 8cc015fa592640eb626a603bd9a25759a71f23a3
Author: mrm2234 <60762357+mrm2234@users.noreply.github.com>
Date:   Mon Nov 29 17:19:35 2021 -0500
```

```
    Update Makefile

commit 26460a8e09489565fce01acec946b95718229a7c
Author: mrm2234 <60762357+mrm2234@users.noreply.github.com>
Date:   Mon Nov 29 17:15:15 2021 -0500

    Updated testall.sh

commit 908623b416d46f284252eb771543748498672109
Author: mrm2234 <60762357+mrm2234@users.noreply.github.com>
Date:   Mon Nov 29 17:11:19 2021 -0500

    testall.sh

commit 60f359370bacff51ca7a08a3e71abeee3e13858c
Author: mrm2234 <60762357+mrm2234@users.noreply.github.com>
Date:   Mon Nov 29 16:58:13 2021 -0500

    oops

commit 65b93c7f09e3dfbc0bb721a20af55b0cb8b3cf59
Author: mrm2234 <60762357+mrm2234@users.noreply.github.com>
Date:   Mon Nov 29 16:57:22 2021 -0500

    Makefile

commit c0db447b8fa91e881e9f44d7171e136c53e0c844
Merge: 2ef5b44 97b3f94
Author: Aapeli <git@aapelivuorinen.com>
Date:   Mon Nov 29 11:18:39 2021 -0500

    Merge branch 'control-flow' into develop

commit 97b3f94f762b42b5cab327914801bcd5d2b0d0a7
Author: Aapeli <git@aapelivuorinen.com>
Date:   Mon Nov 29 11:18:15 2021 -0500

    Implement while loop

commit ce40dc1fa52e97a3a262a06b1dcf3da992636a47
Author: Aapeli <git@aapelivuorinen.com>
Date:   Mon Nov 29 11:06:16 2021 -0500

    Add next_int builtin for testing loops

commit edf45d03ae574861a1c99bf27d1499375f72e817
```

```
Author: Aapeli <git@aapelivuorinen.com>
Date:   Mon Nov 29 11:02:29 2021 -0500

    Add integer comparisons and start on while

commit c110c41120b1272421aa9238d469029049d38082
Author: Aapeli <git@aapelivuorinen.com>
Date:   Mon Nov 29 10:51:24 2021 -0500

    Add back rest of kazm demo

commit f4bfea45adf9e8106544244c4e47bec539ca1ea5
Author: Aapeli <git@aapelivuorinen.com>
Date:   Mon Nov 29 10:51:12 2021 -0500

    Implement else for if

commit 47f2a6ff2e13684ebcb354f1bfa5b08864a9c9c2
Author: Aapeli <git@aapelivuorinen.com>
Date:   Mon Nov 29 10:34:14 2021 -0500

    Implement simple if with no else

commit 61e1c190d2a69375f917f6fdef1cf2fc8e51fade
Author: Aapeli <git@aapelivuorinen.com>
Date:   Mon Nov 29 10:11:25 2021 -0500

    Make codegen_stmt recursive so I can do non-straight line code

commit c17ba4aba112def5c013c364bf6b7370a901519d
Author: Aapeli <git@aapelivuorinen.com>
Date:   Mon Nov 29 10:00:15 2021 -0500

    Clean up

commit 9c928f4693325a3aef6364a5cec3842d1a4ca1f2
Author: Aapeli <git@aapelivuorinen.com>
Date:   Mon Nov 29 09:55:02 2021 -0500

    Implement return values

commit d7ef49cc0fd7e797cd7342ff49a369ac57cc10e5
Author: Aapeli <git@aapelivuorinen.com>
Date:   Mon Nov 29 09:41:19 2021 -0500

    Add integer arithmetic
```

```
commit d207c5a66ac1605e5a07b32d50ea14c12920e884
Author: Aapeli <git@aapelivuorinen.com>
Date:   Mon Nov 29 09:31:16 2021 -0500

    Add support for doubles

commit e1657cd19dfa2520d27a925e3bed2d7dee52a591
Author: Aapeli <git@aapelivuorinen.com>
Date:   Mon Nov 29 09:19:20 2021 -0500

    Implement other functions and calling functions

commit 1a08b3658a7387e2c7476e4d98e07be9fbac5cdd
Author: Aapeli <git@aapelivuorinen.com>
Date:   Mon Nov 29 09:15:59 2021 -0500

    WIP

commit ad3da7791823e76194bc6ae3b16edf5a9aa3234d
Author: Aapeli <git@aapelivuorinen.com>
Date:   Sun Nov 28 21:25:15 2021 -0500

    More int printing

commit 9c97fb76768dc09aac9b92708c2bd68fdd69ee3c
Author: Aapeli <git@aapelivuorinen.com>
Date:   Sun Nov 28 21:20:25 2021 -0500

    Add print_int

commit 81ae0dd5542a253dc68032c12a868efa07c2f484
Author: Aapeli <git@aapelivuorinen.com>
Date:   Sun Nov 28 21:18:16 2021 -0500

    Store func decls in a map

commit 33046c5f0ca10ad70cedd388e431bed39d06d5ff
Author: Aapeli <git@aapelivuorinen.com>
Date:   Sun Nov 28 20:55:57 2021 -0500

    Start on reference to variable and clean up

commit 11fa3ea8b8d8844fa1633dd149134a8097ad360d
Author: Aapeli <git@aapelivuorinen.com>
Date:   Sun Nov 28 20:46:36 2021 -0500
```

```
    Add string literals, function calls with multiple expr args

commit 9444e060f9f97d18723fd3c4a8c5385aa1d5ebfa
Author: Aapeli <git@aapelivuorinen.com>
Date:   Sun Nov 28 20:34:40 2021 -0500

    Add start of real expressions and assignment

commit 830c7c30db2d2c66b36046f91009aadb5c565bc2
Author: Aapeli <git@aapelivuorinen.com>
Date:   Sun Nov 28 16:43:45 2021 -0500

    Reorg

commit 9b7f47ff913db994e5a7603b67de933443e9aff0
Author: Aapeli <git@aapelivuorinen.com>
Date:   Sun Nov 28 16:18:46 2021 -0500

    WIP

commit 2ef5b440f3892d861fbb16c9a4381dffc1385a82
Author: Aapeli <git@aapelivuorinen.com>
Date:   Sun Nov 28 15:52:41 2021 -0500

    Remove proposal

commit 85f5ad3e958c5b4ed6d6cd4a530e0a97d5f1cbf1
Author: Aapeli <git@aapelivuorinen.com>
Date:   Fri Nov 19 12:15:29 2021 -0500

    Separate stmt and expr

commit d03a5935a867821bc377f993b2c50ade5603c6db
Author: Aapeli <git@aapelivuorinen.com>
Date:   Fri Nov 19 11:34:06 2021 -0500

    Clean up

commit ca28ddbb7cea7262d0147203686e04b3d41be146
Merge: 3b18a5d 200a4cf
Author: Aapeli <git@aapelivuorinen.com>
Date:   Fri Nov 19 11:24:58 2021 -0500

    Merge branch 'hello-world' into develop
```

```
commit 200a4cfdb18e66abccc441ba70a0a6093f0ffd78
Author: Aapeli <git@aapelivuorinen.com>
Date:   Fri Nov 19 11:24:36 2021 -0500

    Add _tags file

commit fa800548dd6c6216546e814ef5d4788aa013d0ba
Author: Aapeli <git@aapelivuorinen.com>
Date:   Fri Nov 12 18:45:37 2021 -0500

    Remove accidentally added binary

commit 52fd5fde4cedced835260e1ddc247c4f2c8555ae
Author: Aapeli <git@aapelivuorinen.com>
Date:   Fri Nov 12 18:41:33 2021 -0500

    Simplify hello world

commit b067820d8654da574d35886ae526f169496cf51d
Author: Aapeli <git@aapelivuorinen.com>
Date:   Fri Nov 12 18:41:22 2021 -0500

    Write code generator and instructions in readme

commit 122d165c132b74503cf78d6d45407ccd3bf20f10
Author: Aapeli <git@aapelivuorinen.com>
Date:   Fri Nov 12 16:15:33 2021 -0500

    Remove user defined funcs

commit c103cf15b5dddbcb68cf575e51cd8d62f62f088f
Author: Aapeli <git@aapelivuorinen.com>
Date:   Fri Nov 12 16:09:43 2021 -0500

    Add semantic checker

commit 8e2538e6231b5b37d50f8cf5e7b6b5af2d24894e
Author: Aapeli <git@aapelivuorinen.com>
Date:   Fri Nov 12 15:45:24 2021 -0500

    Simplify syntax and construct AST

commit 9f3c88df117b63f8f92e64a182365be2b271b511
Author: Aapeli <git@aapelivuorinen.com>
Date:   Fri Nov 12 14:50:40 2021 -0500
```

```
    Start on simple AST which tracks junk

commit 3b18a5d49484f015a6479c97109a458f7d3e8f80
Author: yangzhonglin <zhonglinyang98@gmail.com>
Date:   Sat Oct 30 17:54:43 2021 -0400

    add multi line comment

commit a2996ba18f6f8f5de6ebe49d7d752ac73f9c9023
Author: Aapeli <git@aapelivuorinen.com>
Date:   Sat Oct 30 17:47:26 2021 -0400

    Clean up

commit 541d71c9c51820874388a2db30c929a46d37d2f0
Merge: 2bcaa6b a07eda2
Author: Aapeli <git@aapelivuorinen.com>
Date:   Sat Oct 30 17:47:06 2021 -0400

    Merge branch 'literals' into develop

commit a07eda2c6f22adecf5bea2e23c9a3f6c98b3b14d
Author: yangzhonglin <zhonglinyang98@gmail.com>
Date:   Sat Oct 30 17:45:50 2021 -0400

    Add literals (char, double, string, etc.)

commit 2bcaa6b98b5389d2f6cf6615835c4a4687aa0e3e
Author: Aapeli <git@aapelivuorinen.com>
Date:   Sat Oct 30 17:44:06 2021 -0400

    Add array access

commit 7f1d4365767081d022443ac013d969ba74bd3024
Author: Aapeli <git@aapelivuorinen.com>
Date:   Sat Oct 30 17:34:20 2021 -0400

    Make anything ending with .identifier a full name

commit 3c419106c0e5beab827f7d90cf159ed2063a4b2a
Author: Aapeli <git@aapelivuorinen.com>
Date:   Sat Oct 30 17:25:58 2021 -0400

    Implement constructors

commit aabf7c5c32b1f76993be07968d466bb8cac15f2e
```

```
Author: Aapeli <git@aapelivuorinen.com>
Date:   Sat Oct 30 17:24:31 2021 -0400

    Add String class from LRM

commit 6e717e72de2eff54759c743ef92e9a61f7d68a0b
Author: Aapeli <git@aapelivuorinen.com>
Date:   Sat Oct 30 17:20:11 2021 -0400

    Typo

commit b0afa0e85865ba8aa4da72e2d96f6def3f7a1244
Author: Aapeli <git@aapelivuorinen.com>
Date:   Sat Oct 30 17:19:30 2021 -0400

    Add instructions for running the parser

commit fbad3a35e6b11cb64eafa57a465a27f808f7c9c8
Author: Aapeli <git@aapelivuorinen.com>
Date:   Sat Oct 30 17:18:42 2021 -0400

    Add book stuff to sample

commit 3e32d9af31235ea492bb665191f4152d99144ed8
Author: Aapeli <git@aapelivuorinen.com>
Date:   Sat Oct 30 17:11:17 2021 -0400

    Allow declaring custom dtypes

commit 7e49a0d83b689ce3d3bb091a80c3e13715f11de7
Author: Aapeli <git@aapelivuorinen.com>
Date:   Sat Oct 30 17:06:41 2021 -0400

    Add more to sample

commit 1d899db579ce5fbd521ec5fe54b50e811e11df76
Author: Aapeli <git@aapelivuorinen.com>
Date:   Sat Oct 30 17:00:24 2021 -0400

    Add true/false

commit 13b70ff5459714324652a64461f625da441f8fa6
Author: Aapeli <git@aapelivuorinen.com>
Date:   Sat Oct 30 16:52:00 2021 -0400

    Add classes and accessor to parser
```

```
commit 19fd09861b46d51198e2e88d34c53e9fab5dcd28
Author: Aapeli <git@aapelivuorinen.com>
Date:   Sat Oct 30 16:35:50 2021 -0400

    Get rid of empty

commit a0f7e42cc64719d7252b2b819de7898738541e3a
Author: Aapeli <git@aapelivuorinen.com>
Date:   Sat Oct 30 16:35:06 2021 -0400

    Add break

commit 61758e592d7f7604ae5105130b3b6b3c8ac2245d
Author: Aapeli <git@aapelivuorinen.com>
Date:   Sat Oct 30 16:29:14 2021 -0400

    Make assignments and decl's expressions so they can be used in
   for loop exprs

commit 44a7cef8e4311b67ab878b6b4eb72f5e88f77cbe
Author: Aapeli <git@aapelivuorinen.com>
Date:   Sat Oct 30 16:20:11 2021 -0400

    Add single line comments

commit 0061ce3a27548e352038af012f144dc51de5dd50
Author: Aapeli <git@aapelivuorinen.com>
Date:   Sat Oct 30 16:16:56 2021 -0400

    Get rid of imports

commit 3ec37dc06125fe6c899ce3c9cbf734f8a2757c4e
Author: Aapeli <git@aapelivuorinen.com>
Date:   Sat Oct 30 16:16:34 2021 -0400

    Add for loop

commit 045a575229f47420f8247485af608009eecf760e
Author: Aapeli <git@aapelivuorinen.com>
Date:   Sat Oct 30 15:58:07 2021 -0400

    Clean up

commit 7527362f8cf446dac62289780b2f424253f9a2e3
Merge: b1b8b2a 3ea462f
```

```
Author: Aapeli <git@aapelivuorinen.com>
Date:   Sat Oct 30 15:49:51 2021 -0400

    Merge

commit 3ea462f2d24d69bbd2a138c5ae8d4c5101247811
Author: Aapeli <git@aapelivuorinen.com>
Date:   Sat Oct 30 15:46:57 2021 -0400

    Add string literals that are different from variable/function
   names

commit b1b8b2a6b26d23974051dd078bbec97df86e69d8
Author: Katie Kim <jk4534@columbia.edu>
Date:   Sat Oct 30 15:33:20 2021 -0400

    no yacc errors

commit ab5dc49672ec6562ee0942af05b57408fa96080b
Author: Aapeli <git@aapelivuorinen.com>
Date:   Sat Oct 30 15:28:29 2021 -0400

    Start on while

commit ef08f5aac7cbae9fdf8dc0833cc91408f25f3ac3
Author: Aapeli <git@aapelivuorinen.com>
Date:   Sat Oct 30 15:18:25 2021 -0400

    Use expression in if condition

commit 6328c0c2b915360033cfe9c16ace6aa9e79334bd
Author: Aapeli <git@aapelivuorinen.com>
Date:   Sat Oct 30 15:15:28 2021 -0400

    Work on parser

commit 758cbde5d0bb8152b36c1bf6fb093743a8f1e737
Author: Aapeli <git@aapelivuorinen.com>
Date:   Fri Oct 29 11:39:35 2021 -0400

    Make parser build again

commit 80c60550c554fc68010c2812046dcfedbeccd5a1
Author: Aapeli <git@aapelivuorinen.com>
Date:   Tue Oct 26 13:52:38 2021 -0400
```

```
    Remove generated files

commit f943abaabf1f535a9a03b7dd5c259a5a9c3888d9
Author: mrm2234 <60762357+mrm2234@users.noreply.github.com>
Date:   Fri Oct 29 10:58:18 2021 -0400

    .ml

commit e0d9806c1ecf6ca55b33ac8df43b6aa807c2f19d
Author: mrm2234 <60762357+mrm2234@users.noreply.github.com>
Date:   Fri Oct 29 10:56:22 2021 -0400

    Oct 29

commit fc5f0472305778ae47a7720cae70646e2033cf45
Author: mrm2234 <60762357+mrm2234@users.noreply.github.com>
Date:   Sun Oct 24 22:40:21 2021 -0400

    molly changes

commit 653570d9d0153eb23a660f384ac49d4376275ea2
Author: Aapeli <git@aapelivuorinen.com>
Date:   Sat Oct 23 17:16:25 2021 -0400

    Fix up if stuff

commit 7ac4f0f8433bf7b57727d8262cdf34fad26ef9b7
Author: Aapeli <git@aapelivuorinen.com>
Date:   Sat Oct 23 17:04:17 2021 -0400

    Implement some kind of statement stuff

commit 8e8a0c151c08d77227ed2228ec3316f43f674a19
Author: Aapeli <git@aapelivuorinen.com>
Date:   Sat Oct 23 16:43:16 2021 -0400

    Add function decl syntax to parser

commit 456499b8893cf03e04139ab5558216798f702727
Author: Aapeli <git@aapelivuorinen.com>
Date:   Fri Oct 22 20:15:36 2021 -0400

    Work on parser, wip

commit 89d9fa9c317b63da56989ce7ad50c06d3bc1955a
Author: Aapeli <git@aapelivuorinen.com>
```

```
Date:   Thu Oct 21 16:07:02 2021 -0400

    Make 'string ast'

commit d5b0976f9693d9e990b3dbe4d9a9cd1de112b6f7
Author: Aapeli <git@aapelivuorinen.com>
Date:   Thu Oct 21 16:00:28 2021 -0400

    Start on lexer and parser

commit c29cf26bdf6e2197a53c5e8750c78c9131089e30
Author: Aapeli <git@aapelivuorinen.com>
Date:   Mon Oct 4 15:21:28 2021 -0400

    Start on proposal

commit 6ed81ebebc51ff71e663515fed17b53fda0bac79
Author: Aapeli <git@aapelivuorinen.com>
Date:   Fri Sep 24 16:13:01 2021 -0400

    Initial commit
```