Final Report

Project Manager: Xindi Xu (xx2391)
Language Guru: Qiwen Luo (ql2427)
System Architect: Huaxuan Gao (hg2579)
Tester: Yixin Pan (yp2601)
Project Advisor: Maxwell Levatich (ml4553)

1. Introduction

Marble is a programming language that incorporates matrix manipulation functionalities natively so that the compiled code can solve linear algebra problems efficiently. With Marble, developers can define matrices using Matlab–like `[]` literal syntax, i.e. `M = [0,0,0;0,0,0]` or `M = zeros(3,2)`. Marble includes a bare minimum number of matrix manipulation functions in the language to speed up compiling. Since language is flexible, developers can write functions for other matrix operations that are not provided.

2. Language Manual

2.1 Data Types

In order to accomplish certain operations and functions efficiently, we create the following primitive types as building blocks (each of which contains simple values of a kind).

2.1.1 Primitive types (call by value)

- `int`: Integer under a range of `-2^30 to 2^30 -1`
- `float`: OCaml float type (IEEE 754 with a 53-bit mantissa and exponents from -1022 to 1023)
- `bool`: `true/false`
- `null`: type of variables after declaration and before assignment, type of defined functions, type of variables assigned to functions without return statements

2.1.2 Built-in types (call by reference, non-primitive types)

- `matrix`:
  - accessor: `a[1,0]`
  - dimension: `rows(a) cols(a)`
  - initialization: `matrix A = [1.0,2.0;3.0,4.0]; matrix B = zeros(3,2);`
  - operators:
    - addition, substraction (`+, -`): `[1.0;2.0] + [1.0;2.0]`
    - scalar multiplication (`*`): `3 * [1.0;2.0]` or `[1.0;2.0] * 5.0`
    - matrix multiplication (`*`): `[1.0,2.0] * [1.0;2.0]`
  - Note that all entries in the matrix need to be `float`.

Example:

Java 复制代码

```
// initialize a matrix
matrix A = [1.0,2.0;3.0,4.0];

// access an element in the matrix
A[1,0]; // returns 3.0

// getting the matrix dimension, number of rows, number of cols
rows(A); // returns 2
cols(A); // returns 2
```

2.2 Comments

2.2.1 Single–line comment

The content after the symbol `//` within a line is recognized as a comment in our language and our interpreter will skip the content during the execution.

Example:

```java
// This is a comment
```

2.2.2 Multi–line comment

Any content after `/*` and before `*/` is recognized as a comment in our language and our interpreter will skip the content during the execution.

Example:

```java
/*
This is also a comment
*/
```

2.3 Program

When developers write code in Marble, the file that contains the code is a Marble program. A program consists of a collection of function declarations, variable declarations, and one and only one `main()`.

- Function declarations and variable declarations are optional and can be in any order before the `main()` function
- One `main()` function is required for every program

Example:

```java
matrix numbers;
// error: statements are not allowed in the global scope
int i = 0;
// ok
int j;
int function get(matrix m, int row, int col){
    return m[row,col];
}
int function main(){
    matrix numbers = [1.0,2.0,3.0,4.0;5.0,6.0,7.0,8.0];
    return get(numbers, 0, 1); // 2
}
```

2.4 Functions

2.4.1 Define a Function

A function is a collection of input parameters and statements. A function declaration creates one function and binds the corresponding identifier to it.

- A function must have a name; adding parenthesis `()` to the end of its name will invoke the function
- A function must have a return type. It can be any data type.
- Input parameters are optional and multiple input parameters are separated by commas `,`. Each input parameter must have a type and a name
- Inside the curly braces `{}` is a collection of 0 or more statements
- A function can have 0 or more `return` statements and these return statements must have the match the declared return type
- Within the same scope, functions must have different names

Example:

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

2.4.2 Function Return Statements

- Once any `return` statement is executed, the function will terminate
- The value returned by the function will be available in the context where the function is invoked

Example:

```java
int function get(matrix m, int row, int col){
    return m[row,col];
}
int function set(matrix m, int row, int col, int val){
    m[row,col] = val;
    return 0;
}
matrix m = [1,2;1,2];
float a = get(m,0,0); // a = 1.0
bool b = set(m,1,1,2); // b = null
```

2.4.3 `main` Function

`main` function is a special type of function. In particular, its name must be "main" and it lacks input parameters.

- One program must have one `main` function.
- Return statements are required in the `main` function and they will terminate the program. The value returned from the `main` function is useless
- Code will start executing from `main`.

Example:

```java
int function main(){
    // ...
    return 0;
}
```

2.4.4 Built-in Functions

Function name	Parameters	Notes
`print`	`int i`	Print out an integer
`printb`	`bool b`	Print out a boolean as 0 or 1
`printf`	`float f`	Print out a float
`printmf`	`matrix m`	Print out a matrix
`rows`	`matrix m`	Return number of rows of the matrix
`cols`	`matrix m`	Return number of cols of the matrix
`zeros`	`int r`, `int c`	Return a `r` rows by `c` cols matrix, filled with `0.0`

2.5 Variables

2.5.1 Define a Variable

A variable has a type, a name, and an optional value. A variable declaration creates one variable, binds corresponding identifiers to it, and gives it a type and an initial value.

- One variable can only have one type in its lifetime. There's no way to change its type. A runtime error will be thrown if the variable and the value it is assigned to have mismatched types. See the "2.1 Data Types" section for more details.
- A variable declared without assigning an initial value will have a `null` value. Variables can be reassigned later in the program
- Variables must be declared before assigning a value to it or before using it
- Within the same scope, variables must have different names
- Variables in the global scope cannot

```java
// declare a variable i with type i and assign 0 to it
int i = 0;
// declare a variable j with type int without assigning initial value
// j should be null
int j;

// assign a new value 1 to i
i = 1;
// assign a new value 1.5 to j
// Run-time error: type mismatch
j = 1.5;
```

2.6 Loops

2.6.1 While–loop

The format for while–loop is `while(expr){stmts}`. The expression is the condition part of the loop. The expression is of type boolean and the type check will be done during runtime. The loop–body is a statement list.

Example:

```java
int i = 0;
while(i < 10){
    i = i + 1;
}
```

2.6.2 For–loop

The format for for–loop is `for(assigstmt;expr;assigstmt){stmts}`. The assignment statement, excluding the matrix assignment, is the init part, such as `int i = 0`, `i = 0`, `i += 1` or `i -= 1`. The expression with type boolean is the condition part and the type check will be done during runtime. The part after the condition part is also an assignment statement, which excludes the matrix assignment and will be executed after each iteration. The loop–body is a statement list.

Example:

```java
int n = 1;
for(int i = 0; i < 10; i = i + 1){
    n = n * 10;
}
```

2.7 If–else

The format is `if(expr){stmts}else{stmts}`.
The if–branch is required and can only have one. The else–branch is optional and can have zero or one else. `elif` can be fulfilled by adding another if–else in the if–branch/else–branch.

Example:

```java
if (a<=10) {
    if (a<=20){
        // this is identical to "else if"
    }
}
else {
    // ...
}
```

2.8 Operators

2.8.1 Assignment Operator

The equal sign `=` is used to indicate storing values in variables with the format `type ID = expr;` or `ID = expr;`. Type checking will be done during the runtime.
We also support `+=, -=` and the syntax is `expr += expr;` or `expr -= expr;`. This shortcut is only available for int and float.
If a variable is assigned a value before the declaration, the error will be caught during the compilation.

Example:
`int x = 1;`
`x += 2;`

2.8.2 Arithmetic Operator

The following standard arithmetic operators are provided (only applies to int/float):

- addition `+`
- subtraction and sign negation `-`
- multiplication `*`
- division `/`
- modular `%`

Java 复制代码

```java
int a = 1;
int b = 3;
int c = a + b;
print(c); //out 4
c = b - a;
print(c); //out 2
c = b * c;
print(c); //out 6
c = c / b;
print(c); //out 2
c = c % b;
print(c); //out 2
```

2.8.3 Comparison Operators

The following comparison operators are provided:

- greater than `>`
- less than `<`
- greater than or equal to `>=`
- less than or equal to `<=`
- equal to `==`
- not equal `!=`

All comparison operators will be performed on the values of the operands, not the reference addresses.

Example:

```java
printb(4.0<2.0);
printb(4.0>2.0);
printb(3.0>=3.0);
printb(5.0<=3.0);
printb(3.0==3.0);
printb(3.0!=4.0);

/* The above code will produce
0
1
1
0
1
1
*/
```

2.8.4 Logical Operators

The following logical operators are provided:

- negate `!`
- and `&&`
- or `||`

Only boolean expressions are allowed. Any other expressions will cause runtime errors.

Example:

```
bool isPositive = 1 < 0;
bool isNegaitve = !isPositive;
printb(isPositive && isNegaitve); // 0
printb(isPositive || isNegaitve); // 1
```

2.9 Keywords & Separators

The following keywords are reserved. If used as a variable name, the compiler will throw an error indicating that the keyword cannot be used.

Keywords	Format	Remarks

`if, else`	`if(expr)`
	`{stmts}`
	`else`
	`{stmts}`
		Reserved for conditional statements

`for, while`	`for(assignstmt;expr;assigstmt){stmts}` `while(expr){stmts}`	Reserved for flow control
`main`	`int function main(){stmts}`	main function is used to indicate the starting point to execute the program
`function`	`function foo(){}`	Reserved for functions
`return`	`return expr;`	Reserved for function return statements
`null`		Evaluates to `false` when used as a boolean
`int`, `float`, `bool`, `matrix`	`type ID;`	Built-in datatypes
`( ) [ ] { } , ;`		separators

2.10 Memory

We use call-by-value for both primitive and non-primitive types.

Example:

```java
int function swap(int a, int b){
    int c = a;
    a = b;
    b = c;
    return 0;
}

int function main(){
    int a = 1;
    int b = 2;
    swap(a,b); // a = 1, b = 2
    return 0;
}
```

2.11 Scope

We choose to use static scoping in our language since we want to facilitate modular coding. In this scoping, a variable always refers to its top–level environment.

Example:

```java
int function meth(){
    int a = 0;
    return 0;
}
int function main(){
    // Invalid since "a" is not declared in meth2's scope
    // Error: "undeclared identifier a"
    int b = a + 2;
    return 0;
}
```

```java
int a;
int function meth(){
    return a;
}
int function meth2(){
    int a = 20;
    return meth();
}
int function main(){
    a = 10;
    return meth2(); // 10
}

```

3. Langauge Tutorial

3.1 Initial Setup

To use Marble, please install OCaml and OCaml LLVM binding library, Clang and llvm on local machine first. Then follow the instructions below.

```bash
cd <project folder>
opam init
opam install llvm.<version number>
eval ‘opam config env
```

Note: Do not forget to link lli and llc to OCaml.
For example:

```bash
ln -s /usr/bin/llc-6.0 /usr/bin/llc
```

To run our tests, please run the command `make test`.

3.2 Get Started

3.2.1 Program

Each program has and only has one function called main, which is the entry point of this program. Besides the main function, each program can also have optional global variables and functions.

3.2.2 Function

All functions, including main, are defined as follows.

```
<return type> function <function name> (<param1 type> <param1 name>, ....)
{

  <a list of statements>
  <return statement>

}
```

3.2.3 Variable

Global variables are defined as `<var type> <var name>;`.
Local variables are defined as `<var type> <var name>;` or `<var type> <var name> = <init value>;`

3.2.4 Statements and Expressions

Statements always end with a semi-colon(`;`). Most statements and expressions allowed in mainstream languages are also allowed in Marble. As for a full list of statements and expressions, please refer to our language manual.

3.2.5 Matrix

Matrix is the most important part that distinguishes Marble from other languages. Follow the instructions to learn how to declare and manipulate matrices in Marble!

```
matrix a = [1.1,2.2;3.3,4.4]; // declare a matrix
matrix zero = zeros(3,2); // declare a 3 rows by 2 cols matrix, filled with 0.0

int r = rows(a); // find out number of rows in the matrix
int c = cols(a); // find out number of columns in the matrix

float b = a[1,1]; // access the matrix

printmf(a); // print out the matrix

matrix b = [5.5, 6.6; 7.7, 8.8];
float d = 3.1;
matrix c = a + b; // matrix + matrix
matrix d = a - b; // matrix - matrix
matrix f = d * a; // matrix and scalar multiplication
matrix g = b * a; // matrix multiplication
```

Java 复制代码

3.2.6 Comment

Single-line comments always start with `//`.

Multiple-line comments always start with `/*` and end with `*/`.

3.3 Sample Code

Java 复制代码

```
int function get(matrix mat, int r, int c){
    return mat[r][c];
}

int function set(matrix mat, int r, int c, float val){
    mat[r][c] = val;
    return 0;
}

matrix function main(){
    matrix m1 = [1.1,2.1,3.1;4.1,5.1,6.1];
    matrix m2 = [1.1,2.1,3.1;4.1,5.1,6.1];

    set(m1, 0, 0, 2.1);
    print(get(m1, 0,0)); // 2.1

    matrix res = m1 * m2;
    return res;
}
```

4. Project Plan

4.1 Planning Process

We first set our main goals on a set of features we wanted to implement. Then, we categorized features into basic language features, matrix-related features, and nice-to-have features. We set up milestone deadlines based on these features, the structure of compilers, and suggestions from meetings with our project advisor, Maxwell. We created short-term goals as we work through each milestone. The milestones we set and our full project log are outlined in the following sections 8.1.

We've encountered situations when one member's task was blocked by another member's. We found that constant communication and being flexible could help mitigate the situation. We also did pair programming when implementing the first Hello World milestone and found it extremely useful.

4.2 Specification Process

The initial specification of Marble was based on the set of language features and the main building blocks required to build an end-to-end compiler. Marble is inspired by Java and Python.

The first specification was ambitious and we later cut down these features: dynamic function return type, `elseif`, and operator ref equal for checking variable memory address equal. We first finalized lexical and syntax specifications and implemented the lexer and the parser. Then, we drafted our Language Reference Manual with code samples. In both stages, each member was responsible for a subset of features and a holistic review was conducted with the team and our project advisor before moving forward to the next stage. During the development process, when revisions to the Language Reference Manual and/or parsers are needed, we would weigh choices together as a team.

4.3 Development Process

As suggested by Professor Edwards, we first implemented the lexer and parser based on our context–free grammar, then the AST, SAST, and the semantic checker, then the code generator. We referred to examples from class and past projects to implement our basic Hello World features, including global variables, global functions, the `main` function, and basic integer additions and subtractions. Once our basic compiler ran end–to–end, we developed more features in parallel, such as function type checking, build–in printing functions, function scoping, matrix operations. We also strive for providing intuitive error messages for syntax and runtime errors.

4.4 Testing Process

As we implement features, we wrote multiple success and error unit tests for each feature to ensure code quality. We added functions to visualize syntax errors, the Marble code, and intermediate data structures to debug unexpected test results. We've also built test automation to increase work efficiency.

4.5 Team Responsibilities

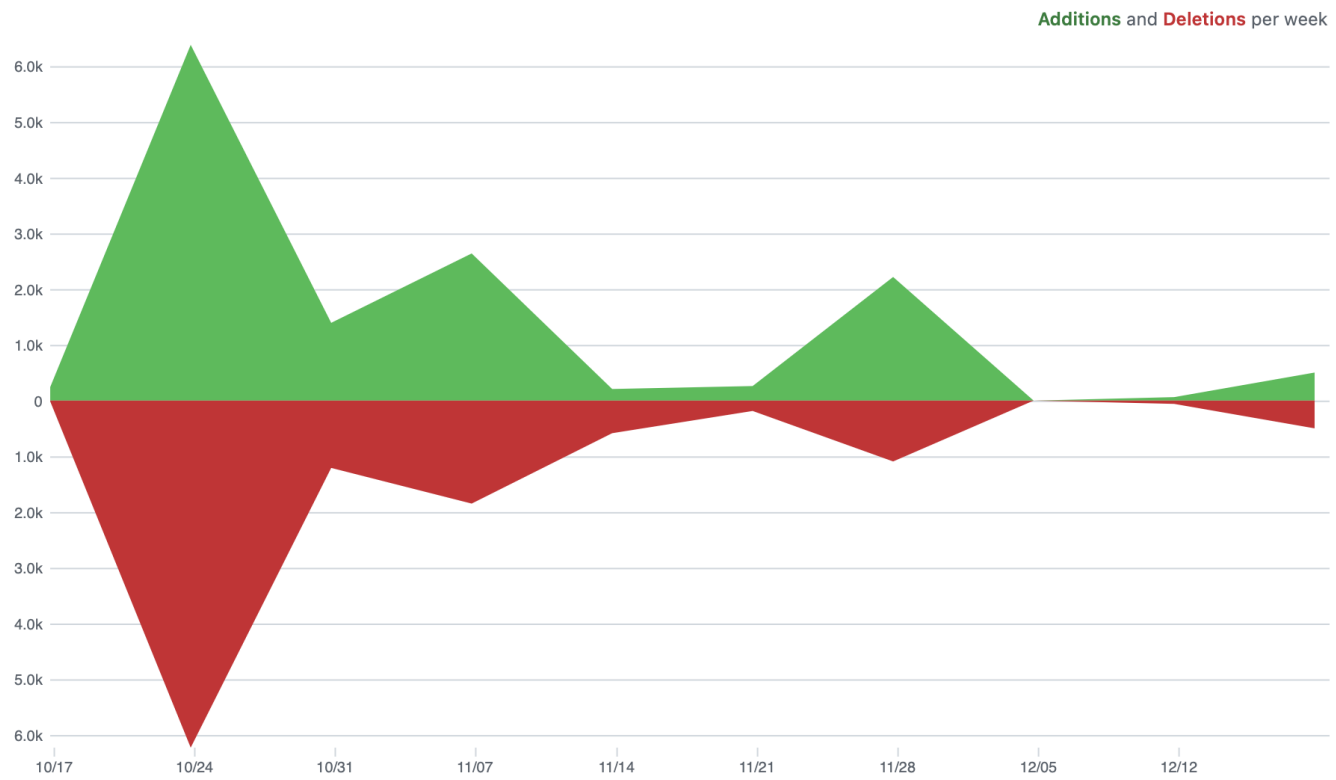
As the project progressed, the roles of each team member became more fluid than the initially assigned roles of *Project Manager, System Architect, Language Guru, and Tester*. In order to streamline the development, each team member is heavily involved with at least two major features of the Marble language, from coding to documentation, as shown below.

Huaxuan Gao: Infrastructure and matrix–related operations
Qiwen Luo: Statements and expressions
Yixin Pan: Statements and expressions
Xindi Xu: Variables, functions, and matrix initialization

4.6 Project Timeline

Date	Milestone
Oct. 28	Context–free grammar completed
Oct. 28	Scanner, parser completed
Nov. 12	Semant, AST, SAST, Codegen completed
Nov. 14	Hello World milestone completed
Nov. 30	Matrix–related features completed
Dec. 4	Function–related features completed
Dec. 4	Statements– and expressions–related features completed
Dec 20	Addressed errors and warnings, sample code completed
Dec 22	Final clean up: remove unused code, formatting

Code Frequency



Project Log

See Appendix 8.1

4.8 Software Development Environment

Technologies used:

- GitHub–Hosted Git Repo: version control
- Ocaml (version 4.x): the programming language for implementing the Marble compiler
  - Ocamlyacc and Ocamllex: compiling scanner and parser frontend
- C (version 13.x): building helper methods for matrix operations in Marble
- LLVM (version 11.x): infrastructure for the Marble compiler
- Shell scripts and Makefile: automation for testing

4.9 Programming Style Guide

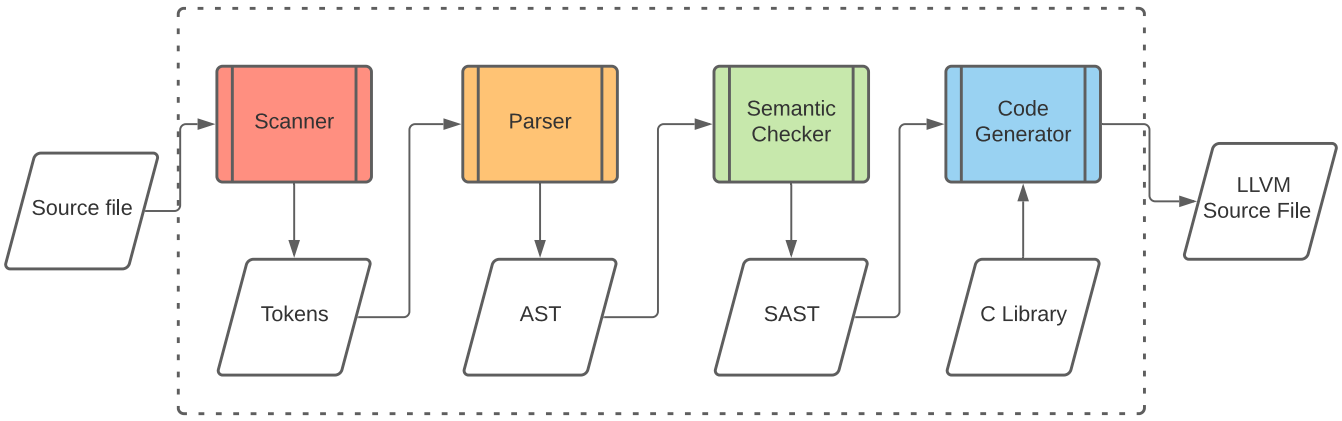
We generally followed the following guidelines while programming the Marble compiler:

- Formatting: use `ocamlformat` to format Ocaml code with the "conventional" Ocaml style guide
- Indentation: use only spaces to ensure consistency over different machines
- Naming: use snake_case as opposed to camelCase for variable and function names
- File Name: files names for passing tests start with `test-` and files names for failing tests start with `fail-`

5. Architecture Design

5.1 Architecture Diagram

This is the overall system architecture. The major components are the scanner, parser, semantic checker, code generator. Each step will generate outputs that can be processed by the following step. The details of each component will be discussed in sections below. From the users' point of view, the system runs end–to–end, takes the input source file, and generates an LLVM IR that can be executed by the LLVM interpreter.



5.2 Scanner

The Marble source files will first go through the scanner. The scanner program of Marble uses `ocamllex` lexical analyzer. It will detect pre–defined keywords and regular expression patterns. It will also discard white spaces, newlines, and comments. After scanning through the source file, the scanner will generate a list of tokens, which can be used by the parser. The scanner is implemented in the `scanner.mll`.

5.3 Parser

The parser of Marble uses `ocamlyacc` to generate the Abstract Syntax Tree (AST) from input tokens. The terminal symbols are declared using `%token`. It also defines the entry point of the grammar with `%start`. Associativity and precedences are also defined in the parser. More importantly, the parser defines a set of rules, that will match the tokens with semantic actions. Semantic actions are arbitrary OCaml expressions, that are evaluated to produce the semantic attribute attached to the defined nonterminal. The parser consists of `parser.mly`, `ast.ml`.

5.4 Semantic Checker

The AST produced by the parser is not annotated with types, and it needs to be semantically checked. The semantic checker will detect duplicated use of variable or function names, and invalid usage of the null type variable declaration. It will also reject invalid types for a given operation. For example, in Marble, logical operators must be performed on two boolean type variables, and arithmetic operators must be performed on variables of the same type. It also makes sure that the value is assigned to the variables with the correct type. The generated Sematically–checked AST (SAST) will be passed to the code generator. The semantic checker is implemented in `semant.ml` and `sast.ml`.

5.5 Code Generator

The code generator takes the AST and generates LLVM intermediate representation (IR). It maps the Marble data types to the corresponding LLVM types and maps Marble operators to LLVM operators. For example, the Marble matrixes are represented by a float pointer pointing to a consecutive memory address, and we access a matrix element by calculating the offset based on row index and column index. In Marble, we have two scopes, the global scope, and the local scope, for each scope, the variable names are stored in a hash table. Finally, we link the LLVM IR with our c library, which includes our implementation of the matrix operations, to generate an executable.

6. Test Plan

6.1 Source Language Programs

6.1.1 Source

Program 1:

This is a simple program where we define two variables a and b with matrix type. Then we conduct matrix addition, subtraction, and multiplication among them. The +, –, * shown in the code below is the syntactical sugar with their actual functionality is calling the built–in function matrix additions, matrix subtractions, and matrix multiplications defined in our library. With the printing function, we can check if the result is consistent with the expected output.

```java
int function main() {
    // mat +/- mat
    matrix a = [1.1, 2.2; 3.3, 4.4];
    matrix b = [5.5, 6.6; 7.7, 8.8];
    matrix c = a + b;
    printmf(c);
    matrix d = a - b;
    printmf(d);
    // int * mat mat * int
    int d = 3;
    matrix e = d * a;
    printmf(e);
    matrix f = a * d;
    printmf(f);
    // float * mat mat * float
    float g = 1.1;
    matrix h = g * a;
    printmf(h);
    matrix i = b * g;
    printmf(i);
    // mat * mat
    matrix j = a * b;
    printmf(j);
    matrix k = b * a;
    printmf(k);
    return 0;
}
```

Program 2:

This is a simple program where we would like to access a matrix element and update its value. Our language support that user can access a matrix element with index (row, col) by mat[row, col] and update a matrix element with a simple assign statement. The `printmf` function would print out the whole matrix where we can check if the matrix element has been updated.

```java
int function main() {
    matrix a = [1.1,2.2;3.3,4.4];
    a[0,1] = 5.5;
    printmf(a);
    return 0;
}
```

6.1.2 Target

Target for program 1:

Java 复制代码

```
; ModuleID = 'Marble'
source_filename = "Marble"

@fmt = private unnamed_addr constant [4 x i8] c"%d\0A\00", align 1
@fmt.1 = private unnamed_addr constant [4 x i8] c"%g\0A\00", align 1

declare i32 @printf(i8*, ...)

declare i32 @printmf(double*)

declare i32* @addm(i32*, i32*)

declare double* @addmf(double*, double*)

declare i32* @subm(i32*, i32*)

declare double* @submf(double*, double*)

declare i32* @scalarm(double, i32*)

declare double* @scalarmf(double, double*)

declare i32* @multiplication(i32*, i32*)

declare double* @multiplicationf(double*, double*)

define i32 @main() {
entry:
  %matrix = alloca [6 x double], align 8
  %ptr = getelementptr inbounds [6 x double], [6 x double]* %matrix, i32 0, i32 0
  store double 2.000000e+00, double* %ptr, align 8
  %ptr1 = getelementptr inbounds [6 x double], [6 x double]* %matrix, i32 0, i32 1
  store double 2.000000e+00, double* %ptr1, align 8
  %ptr2 = getelementptr inbounds [6 x double], [6 x double]* %matrix, i32 0, i32 2
  store double 1.100000e+00, double* %ptr2, align 8
  %ptr3 = getelementptr inbounds [6 x double], [6 x double]* %matrix, i32 0, i32 3
  store double 2.200000e+00, double* %ptr3, align 8
  %ptr4 = getelementptr inbounds [6 x double], [6 x double]* %matrix, i32 0, i32 4
  store double 3.300000e+00, double* %ptr4, align 8
  %ptr5 = getelementptr inbounds [6 x double], [6 x double]* %matrix, i32 0, i32 5
  store double 4.400000e+00, double* %ptr5, align 8
```

```
  %a = getelementptr inbounds [6 x double], [6 x double]* %matrix, i32 0, i32 0
  %a7 = alloca double*, align 8
  store double* %a, double** %a7, align 8
  %matrix8 = alloca [6 x double], align 8
  %ptr9 = getelementptr inbounds [6 x double], [6 x double]* %matrix8, i32 0, i32 0
  store double 2.000000e+00, double* %ptr9, align 8
  %ptr10 = getelementptr inbounds [6 x double], [6 x double]* %matrix8, i32 0, i32 1
  store double 2.000000e+00, double* %ptr10, align 8
  %ptr11 = getelementptr inbounds [6 x double], [6 x double]* %matrix8, i32 0, i32 2
  store double 5.500000e+00, double* %ptr11, align 8
  %ptr12 = getelementptr inbounds [6 x double], [6 x double]* %matrix8, i32 0, i32 3
  store double 6.600000e+00, double* %ptr12, align 8
  %ptr13 = getelementptr inbounds [6 x double], [6 x double]* %matrix8, i32 0, i32 4
  store double 7.700000e+00, double* %ptr13, align 8
  %ptr14 = getelementptr inbounds [6 x double], [6 x double]* %matrix8, i32 0, i32 5
  store double 8.800000e+00, double* %ptr14, align 8
  %b = getelementptr inbounds [6 x double], [6 x double]* %matrix8, i32 0, i32 0
  %b16 = alloca double*, align 8
  store double* %b, double** %b16, align 8
  %a17 = load double*, double** %a7, align 8
  %b18 = load double*, double** %b16, align 8
  %c = call double* @addmf(double* %a17, double* %b18)
  %c19 = alloca double*, align 8
  store double* %c, double** %c19, align 8
  %c20 = load double*, double** %c19, align 8
  %printmf = call i32 @printmf(double* %c20)
  %a21 = load double*, double** %a7, align 8
  %b22 = load double*, double** %b16, align 8
  %d = call double* @submf(double* %a21, double* %b22)
  %d23 = alloca double*, align 8
  store double* %d, double** %d23, align 8
  %d24 = load double*, double** %d23, align 8
  %printmf25 = call i32 @printmf(double* %d24)
  %d26 = alloca i32, align 4
  store i32 3, i32* %d26, align 4
  %d27 = load i32, i32* %d26, align 4
  %a28 = load double*, double** %a7, align 8
  %scalar = sitofp i32 %d27 to double
  %e = call double* @scalarmf(double %scalar, double* %a28)
  %e29 = alloca double*, align 8
  store double* %e, double** %e29, align 8
  %e30 = load double*, double** %e29, align 8
```

```
  %printmf31 = call i32 @printmf(double* %e30)
  %a32 = load double*, double** %a7, align 8
  %d33 = load i32, i32* %d26, align 4
  %scalar34 = sitofp i32 %d33 to double
  %f = call double* @scalarmf(double %scalar34, double* %a32)
  %f35 = alloca double*, align 8
  store double* %f, double** %f35, align 8
  %f36 = load double*, double** %f35, align 8
  %printmf37 = call i32 @printmf(double* %f36)
  %g = alloca double, align 8
  store double 1.100000e+00, double* %g, align 8
  %g38 = load double, double* %g, align 8
  %a39 = load double*, double** %a7, align 8
  %h = call double* @scalarmf(double %g38, double* %a39)
  %h40 = alloca double*, align 8
  store double* %h, double** %h40, align 8
  %h41 = load double*, double** %h40, align 8
  %printmf42 = call i32 @printmf(double* %h41)
  %b43 = load double*, double** %b16, align 8
  %g44 = load double, double* %g, align 8
  %i = call double* @scalarmf(double %g44, double* %b43)
  %i45 = alloca double*, align 8
  store double* %i, double** %i45, align 8
  %i46 = load double*, double** %i45, align 8
  %printmf47 = call i32 @printmf(double* %i46)
  %a48 = load double*, double** %a7, align 8
  %b49 = load double*, double** %b16, align 8
  %j = call double* @multiplicationf(double* %a48, double* %b49)
  %j50 = alloca double*, align 8
  store double* %j, double** %j50, align 8
  %j51 = load double*, double** %j50, align 8
  %printmf52 = call i32 @printmf(double* %j51)
  %b53 = load double*, double** %b16, align 8
  %a54 = load double*, double** %a7, align 8
  %k = call double* @multiplicationf(double* %b53, double* %a54)
  %k55 = alloca double*, align 8
  store double* %k, double** %k55, align 8
  %k56 = load double*, double** %k55, align 8
  %printmf57 = call i32 @printmf(double* %k56)
  ret i32 0
}
```

Target for program 2:

Java 复制代码

```
; ModuleID = 'Marble'
source_filename = "Marble"

@fmt = private unnamed_addr constant [4 x i8] c"%d\0A\00", align 1
@fmt.1 = private unnamed_addr constant [4 x i8] c"%g\0A\00", align 1

declare i32 @printf(i8*, ...)

declare i32 @printmf(double*)

declare i32* @addm(i32*, i32*)

declare double* @addmf(double*, double*)

declare i32* @subm(i32*, i32*)

declare double* @submf(double*, double*)

declare i32* @scalarm(double, i32*)

declare double* @scalarmf(double, double*)

declare i32* @multiplication(i32*, i32*)

declare double* @multiplicationf(double*, double*)

define i32 @main() {
entry:
  %matrix = alloca [6 x double], align 8
  %ptr = getelementptr inbounds [6 x double], [6 x double]* %matrix, i32 0, i32 0
  store double 2.000000e+00, double* %ptr, align 8
  %ptr1 = getelementptr inbounds [6 x double], [6 x double]* %matrix, i32 0, i32 1
  store double 2.000000e+00, double* %ptr1, align 8
  %ptr2 = getelementptr inbounds [6 x double], [6 x double]* %matrix, i32 0, i32 2
  store double 1.100000e+00, double* %ptr2, align 8
  %ptr3 = getelementptr inbounds [6 x double], [6 x double]* %matrix, i32 0, i32 3
  store double 2.200000e+00, double* %ptr3, align 8
  %ptr4 = getelementptr inbounds [6 x double], [6 x double]* %matrix, i32 0, i32 4
  store double 3.300000e+00, double* %ptr4, align 8
  %ptr5 = getelementptr inbounds [6 x double], [6 x double]* %matrix, i32 0, i32 5
  store double 4.400000e+00, double* %ptr5, align 8
```

```
42      %a = getelementptr inbounds [6 x double], [6 x double]* %matrix, i32 0,
    i32 0
43      %a7 = alloca double*, align 8
44      store double* %a, double** %a7, align 8
45      %a8 = load double*, double** %a7, align 8
46      %ptr9 = getelementptr inbounds double, double* %a8, i32 1
47      %cols = load double, double* %ptr9, align 8
48      %colsint = fptosi double %cols to i32
49      %row = mul i32 0, %colsint
50      %row_col = add i32 %row, 1
51      %idx = add i32 2, %row_col
52      %ptr10 = getelementptr inbounds double, double* %a8, i32 %idx
53      store double 5.500000e+00, double* %ptr10, align 8
54      %a11 = load double*, double** %a7, align 8
55      %printmf = call i32 @printmf(double* %a11)
56      ret i32 0
57    }
```

6.2 Test Suite

We start writing all test cases for the "Hello World" iteration where we test primitive type variable declaration, assignments, and related basic arithmetic operations. Then we expanded the test into 1) all types of variables including matrix and float, 2) all types of Binory Op including mod, comparison, and negation associated with all available data type, 3) all statements including if else, while, 4) all function–related issue including its formals, return, and scope, 5) lastly all matrix–related operations such as matrix multiplication, matrix element update, and dimension check.

6.2.1 Passing Tests (27)

Java 复制代码

```
// test-boolOp.mb
int function main(){
    printb(true&&true);
    printb(true&&false);

    printb(true||false);
    printb(true||true);
    printb(false||true);
    printb(false||false);

    printb(!true);
    printb(!false);
    return 0;
}

// test-boolOp.out
1
0
1
1
1
0
0
1
********************************************************************************
//test-comments.mb
/*
Each program must include one `main` function.
*/
int function main(){
    // print 111
    print(111);
    return 0;
}

//test-comments.out
111

********************************************************************************
//test-floatBinop.mb
int function main(){
    printf(1.1+2.2);
    printf(1.1*2.2);
    printf(1.1-2.2);
    printf(1.1/2.2);

    printf(4.1%3.0);

```

```
    printb(4.0<2.0);
    printb(4.0>2.0);
    printb(3.0>=3.0);
    printb(5.0<=3.0);
    printb(3.0==3.0);
    printb(3.0!=4.0);

    return 0;
}

//test-floatBinop.out
3.3
2.42
-1.1
0.5
1.1
0
1
1
0
1
1
**********************************************************************
//test-floatUnop.mb
int function main(){
    printf(-2.1);

    return 0;
}

//test-floatUnop.out
-2.1
**********************************************************************
//test-funcFormal.mb
int function custom_print(int a){
    print(1);
    return 1;
}

int function main(){
    custom_print(1);
    return 0;
}

//test-funcFormal.out
1
**********************************************************************
//test-funcReturn.mb
int function get1(){
    return 1;
```

```
}

float function getPi(){
    return 3.14;
}

bool function getTrue(){
    return true;
}

matrix function getMatrix(){
    matrix a = [1.1, 2.2; 3.3, 4.4];
    return a;
}

int function main(){
    print(get1());
    printf(getPi());
    printb(getTrue());
    printmf(getMatrix());
    return 0;
}

//test-funcReturn.out
1
3.14
1
1.100000 2.200000
3.300000 4.400000
********************************************************************************
//test-intBinop.mb
int function main(){
    print(1+2);
    print(1*2);
    print(2-1);
    print(4/2);

    print(4%3);

    printb(4<2);
    printb(4>2);
    printb(3>=3);
    printb(5<=3);
    printb(3==3);
    printb(3!=4);

    return 0;
}

//test-intBinop.out
```

```
3
2
1
2
1
0
1
1
0
1
1
**********
//test-intUnop.mb
int function main(){
    print(-2);

    return 0;
}
//test-intUnop.out
-2
**********

//test-matrix_access.mb
int function main() {
    matrix a = [1.1,2.2;3.3,4.4];
    float b = a[1,1];
    printf(b);
    return 0;
}

//test-matrix_access.out
4.4
**********

//test-matrix_dim.mb
int function main() {
    matrix a = [1.1,2.2;3.3,4.4; 5.5,6.6];
    int r = rows(a);
    print(r);
    int c = cols(a);
    print(c);
}
//test-matrix_dim.out
3
2
**********

//test-matrix_ops.mb
int function main() {
    // mat +/- mat
```

```
196         matrix a = [1.1, 2.2; 3.3, 4.4];
197         matrix b = [5.5, 6.6; 7.7, 8.8];
198         matrix c = a + b;
199         printmf(c);
200         matrix d = a - b;
201         printmf(d);
202         // int * mat mat * int
203         int d = 3;
204         matrix e = d * a;
205         printmf(e);
206         matrix f = a * d;
207         printmf(f);
208         // float * mat mat * float
209         float g = 1.1;
210         matrix h = g * a;
211         printmf(h);
212         matrix i = b * g;
213         printmf(i);
214         // mat * mat
215         matrix j = a * b;
216         printmf(j);
217         matrix k = b * a;
218         printmf(k);
219         return 0;
220     }
221     //test-matrix_ops.out
222     6.600000 8.800000
223     11.000000 13.200000
224     -4.400000 -4.400000
225     -4.400000 -4.400000
226     3.300000 6.600000
227     9.900000 13.200000
228     3.300000 6.600000
229     9.900000 13.200000
230     1.210000 2.420000
231     3.630000 4.840000
232     6.050000 7.260000
233     8.470000 9.680000
234     22.990000 26.620000
235     52.030000 60.500000
236     27.830000 41.140000
237     37.510000 55.660000
238     ********************************************************************************

239     //test-matrix_update.mb
240     int function main() {
241         matrix a = [1.1,2.2;3.3,4.4];
242         a[0,1] = 5.5;
243         printmf(a);
244         return 0;
```

```
}
//test-matrix_update.out
1.100000 5.500000
3.300000 4.400000
**********

//test-print.mb
int function main(){
    print(111);
    return 0;
}
//test-print.out
111
**********

//test-printb.mb
int function main(){
    bool a = false;
    printb(a);
    printb(true);
    return 0;
}
//test-printb.out
0
1
**********

//test-printf.mb
int function main() {
    float a = 1.1;
    printf(a);
    return 0;
}
//test-printf.out
1.1
**********

//test-printm.mb
int function main(){
    matrix a = [1.0, 2.0; 3.0, 4.0];
    printmf(a);
    matrix b;
    b = [1.0, 2.0; 3.0, 4.0; 5.0, 6.0];
    printmf(b);
    return 0;
}
//test-printm.out
1.000000 2.000000
3.000000 4.000000
1.000000 2.000000
```

```
3.000000 4.000000
5.000000 6.000000
**************************************************************************

//test-scope.mb
int a;

int function foo(){
    int a = 1;
    print(a);
    return 0;
}

int function main(){
    int a = 2;
    foo();
    print(a);
    return 0;
}
//test-scope.out
1
2
**************************************************************************

//test-stmtAssign.mb
int x;

int function main(){
    x = 1;
    x = x + 1;
    print(x);

    x += 2;
    print(x);

    x -= 2;
    print(x);
    return 0;
}
//test-stmtAssign.out
2
4
2
**************************************************************************

//test-stmtFor.mb
int function main(){
    int i = 0;
    for(int x = 0;x < 10;x=x+1){
        i = i + 1;
```

```
    }
    print(i);
    return 0;
}
//test-stmtFor.out
10
**************************************************************************

//test-stmtIfElse.mb
int function main() {
    int a = 1;
    if(a > 0){
        a = -1;
    }
    else{
        a = 1;
    }
    print(a);

    if(a > 0){
        a = -1;
    }
    else{
        a = 1;
    }
    print(a);


    if(a >= 1){
        a = 2;
    }
    print(a);

    return 0;
}
//test-stmtIfElse.out
-1
1
2
**************************************************************************

//test-stmtVDeAssign.mb
int function main(){
    int x = 1;
    print(x);
    return 0;
}
//test-stmtVDeAssign.out
1
```

```
385  ********************************************************************************

386  //test-stmtVDeclare.mb
387  int function main(){
388      int x;
389      x = 2;
390      print(x);
391      return 0;
392  }
393  //test-stmtVDeclare.out
394  2
395  ********************************************************************************

396  //test-stmtWhile.mb
397  int function main(){
398      int i = 1;
399      int x = 1;
400      while(x < 10){
401          x = x + i;
402      }
403      print(x);
404      return 0;
405  }
406  //test-stmtWhile.out
407  10
408
409  ********************************************************************************

410  //test-determinant.mb
411  float function determinant(matrix a){
412      int dim = rows(a);
413      float r_scalar = 1.0;
414
415      // Row ops on matrix a to get in upper triangle form
416      for(int dia = 0; dia < dim; dia=dia+1){
417          for(int r = dia + 1; r < dim; r=r+1){
418              r_scalar = a[r,dia] / a[dia,dia];
419              for(int c = 0; c < dim; c=c+1){
420                  a[r,c] = a[r,c] - r_scalar * a[dia,c];
421              }
422          }
423      }
424
425      // Once matrix a is in upper triangle form
426      // Multiply entries on the diagonal
427      float product = 1.0;
428      for(int i = 0; i < dim; i = i+1){
429          product = product * a[i,i];
430      }
431      return product;
```

```
}


int function main(){
    matrix a = [1.0, 2.0; 3.0, 4.0];
    float det_a = determinant(a);
    printf(det_a);  // -2

    matrix b = [2.0, 0.0, 2.0; 1.0, 1.0, 2.0; 2.0, 1.0, 8.0];
    float det_b = determinant(b);
    printf(det_b);  // 10

    return 0;
}

//test-determinant.out
-2
10
**********

// test-matrix_zeros.mb
int function main() {
    matrix a = zeros(3, 2);
    printmf(a);
    return 0;
}
// test-matrix_zeros.out
0.000000 0.000000
0.000000 0.000000
0.000000 0.000000
**********
//test-fibonacci.mb
int function fib(int n){
    if(n == 1){
        return 1;
    }
    if(n == 2){
        return 2;
    }
    return fib(n-1) + fib(n-2);
}
```

6.2.2 Failing Tests (19)

Java 复制代码

```
//fail-expr_binopType1.mb
int function main(){
    int x = 1;
    float y = 2.0;
    x = x + y;
    print(x);
    return 0;
}
//fail-expr_binopType1.err
Fatal error: exception Failure("illegal binary operator int + float in x + y")
********************************************************************************

//fail-expr_binopType2.mb
int function main(){
    int x = 1;
    float y = 2.0;
    print(x<y);
    return 0;
}
//fail-expr_binopType2.err
Fatal error: exception Failure("illegal binary operator int < float in x < y")
********************************************************************************

//fail-expr_binopType3.mb
int function main(){
    float x = 1.0;
    float y = 2.0;
    print(x&&y);
    return 0;
}
//fail-expr_binopType3.err
Fatal error: exception Failure("illegal binary operator float && float in x && y")
********************************************************************************

//fail-expr_unary1.mb
int function main(){
    int x = 1;
    x = !x;
    print(x);
    return 0;
}
//fail-expr_unary1.err
Fatal error: exception Failure("illegal unary operator !int in !x")
********************************************************************************
```

```
//fail-expr_unary2.mb
int function main(){
    bool x = true;
    x = -x;
    printb(x);
    return 0;
}
//fail-expr_unary2.err
Fatal error: exception Failure("illegal unary operator -Bool in -x")
*********************************************************************************

//fail-funcFormal.mb
int function custom_print(int a){
    print(1);
    return 1;
}

int function main(){
    custom_print();
    return 0;
}
//fail-funcFormal.err
Fatal error: exception Failure("expecting 1 arguments in custom_print();")
*********************************************************************************

//fail-funcReturn.mb
int function main(){
    return true;
}
//fail-funcReturn.err
Fatal error: exception Failure("return gives Bool expected int in true")
*********************************************************************************

//fail-funcUndefined.mb
int function main(){
    custom_print();
    return 0;
}
//fail-funcUndefined.err
Fatal error: exception Failure("unrecognized function custom_print")
*********************************************************************************

// fail-matrix_dim.mb
int function main(){
    matrix a = [1.0, 2.0; 3.0, 4.0, 5.0];
    printmf(a);
    return 0;
}
//fail-matrix_dim.err
```

```
Fatal error: exception Failure("all rows of matrices must have the same number of elemens")
**********************************************************************

//fail-matrix_type.mb
int function main(){
    matrix a = [1.0,2.0;3,4];
    printmf(a);
    return 0;
}
//fail-matrix_type.err
Fatal error: exception Failure("Types in matrix do not match.")
**********************************************************************

//fail-stmt_assign.mb
int x;
int function main(){
    x = 1.1;
    print(x);
    return 0;
}
//fail-stmt_assign.err
Fatal error: exception Failure("Illegal assignment!")
**********************************************************************

//fail-stmt_for.mb
int function main(){
    int i = 0;
    for(int x = 0;x + 10;x=x+1){
        i = i + 1;
    }
    print(i);
    return 0;
}
//fail-stmt_for.err
Fatal error: exception Failure("Expect to have a Bool type here.")
**********************************************************************

//fail-stmt_ifelse.mb
int function main() {
    int a = 1;
    if(a + 0){
        a = -1;
    }
    else{
        a = 1;
    }
    print(a);
    return 0;
}
```

```
//fail-stmt_ifelse.err
Fatal error: exception Failure("Expect to have a Bool type here.")
**********************************************************************

//fail-stmt_ifelse2.mb
int function main() {
    int a = 1;
    if(a + 0){
        a = -1;
    }
    print(a);
    return 0;
}
//fail-stmt_ifelse2.err
Fatal error: exception Failure("Expect to have a Bool type here.")
**********************************************************************

//fail-stmt_v_de_assign.mb
int function main(){
    int x = 1.1;
    print(x);
    return 0;
}
//fail-stmt_v_de_assign.err
Fatal error: exception Failure("Type not correct")
**********************************************************************
//fail-stmt_while.mb
int function main(){
    int i = 1;
    int x = 1;
    while(x + 10){
        x = x + i;
    }
    print(x);
    return 0;
}
//fail-stmt_while.err
Fatal error: exception Failure("Expect to have a Bool type here.")
**********************************************************************
//fail-string.mb
int function main(){
    print("Hello World");
}
//fail-string.err
Fatal error: exception Failure("illegal character "")
**********************************************************************
//fail-scope.mb
int function meth(){
    int a = 0;
    return 0;
```

```
}
int function main(){
    // Invalid since "a" is not declared in meth2's scope
    int b = a + 2;
    return 0;
}
//fail-scope.err
Fatal error: exception Failure("undeclared identifier a")
**********************************************************************
//fail-matrix_zeros.mb
int function main() {
    matrix a = zeros(3, 0);
```

6.3 Test Automation

To increase the efficiency of running all the test suites, we used the test script `testall.sh` which provides a series of instructions for the system to run all the tests (include both passing and failing) with a single command. The output of a passing test would be compared with the expected we designed in the corresponding .out file (the name is consistent for .mb and .out file) while the output of a failing test would be compared to the expected error message we designed in the .err file as well. In both cases, if the content matches, the test is passed; otherwise, it would report failures and show differences in the log file.

6.4 Roles and Responsibilities

The division of test cases is consistent with the division of functionality where Huaxuan writes all test cases regarding the matrix-related operations (operations, dimension, and matrix update) and Infrasture (printing function), Qiwen writes all test cases regarding Statements (if else, while), Yixin writes all test cases regarding Expression (Binop, unary), and Xindi writes all test cases regarding Functions (formals, return, scoping). Huaxuan set up the testing script such as `testall.sh` where all group members can use single command for testing all test cases.

7. Lessons Learned

7.1 Huaxuan Gao

The most important thing that I would like to mention here is the cooperation between team members. The Marble team is very enthusiastic about what we are doing. The collaboration is great among us. For example, when we started to think about what kind of features we should have, Yixin and Qiwen brings in their Java experience while Xindi and I were more familiar with Python, we have some interesting debates on the pros and cons of different languages. These thinkings leads to the idea of Marble. Our team is very hard working as well, I remember a time

when Qiwen submits a pull request at 4 am to fix a bug, and we held 2 meetings every week regularly.

My advice for future teams would be, keep trying until you see the turning point. Because the most exciting moment for me during the whole semester is when I put everything together and print out an integer 0, which means our system finally runs end to end. After that, features are implemented at the incredible speed.

7.2 Qiwen Luo

My biggest takeaway was that it is extremely necessary to hold a regular meeting every week or every other week when working as a team. During our regular meetings, we brainstormed on Marble's features, shared the current progress of each member, solved problems and determined tasks to be completed before the next meeting. Among them, specifying what to be completed before next meeting is the most important. It not only makes each meeting more efficient but also breaks a big project into some small tasks, which makes us feel that the task is simpler and makes the progress monitoring to be easier.

As for advice for future teams, try to determine the context–free grammar for the language as soon as possible. Instead of only discussing the language broadly, the grammar can give us a clearer picture of the language and it makes the feasibility check easier.

7.3 Yixin Pan

Besides the technical aspect I learned over the project such as Ocaml programming and Context–Free Grammar design, what I learned most is about the significance of project management and the team collaboration. To begin with, when doing the topic brainstorm, throwing out as many ideas as possible; those ideas would facilitate the discussion where you can quickly discover the core features you would like to include in the language. Then, during the language design, it would be easier for you to choose the specific features (e.g. static scoping v.s. dynamic scoping) based on the core features. In such circumstances, during the implementation, it would be more efficient to get main feature working while setting aside some nice–to–have features into later iterations which are harder to complete than the expectation. More importantly, setting up a regular implementation cycle is the key to the success. Everyone in the team should stick to the plan they proposed so that the actual implementation is always ahead of schedule rather than behind it.

7.4 Xindi Xu

Identifying a set of features to build is definitely the hardest part but the most important part of any project. Our team often ran into situations where we need to give up fancier features due to time constraints. In such situations, discussing alternative designs with the team and the project advisor early in the process allows us to stay on track. Code reviewing and pair programming also allowed everyone to get on the same page on the language design and implementation of different pieces of the codebase. Having a consistent meeting schedule with the team and the project adviser helped keep us on schedule. As for the more technical part, I benefit a lot from learning Ocaml and compiler design by reading the codebases of MicroC and examples from past projects.

My advice for future teams: start designing the compiler with a bare minimum set of features and gradually expand later. Be conservative when adding new features and try to have everyone's agreement before proceeding.

8. Appendix

8.1 Project Log

Plain Text | 复制代码

```
commit f42d482c37f6a4768f8b779fa3bcf2a98ba8a2cd
Merge: 89469db 5f96e1a
Author: HuaxuanGAO <33063662+HuaxuanGAO@users.noreply.github.com>
Date:   Wed Dec 22 21:11:52 2021 -0500

    Merge pull request #21 from PLT-Marble/fix/remove-unused-code

    remove unused code

commit 5f96e1aafbacfdc9732faae29548c009a7206e31
Author: Xindi Xu <xindixu0@gmail.com>
Date:   Wed Dec 22 21:07:03 2021 -0500

    remove unused code

commit 7b06f28eaf45e0f365da2fe5aee1673d4db06cf1
Author: Xindi Xu <xindixu0@gmail.com>
Date:   Wed Dec 22 20:43:38 2021 -0500

    remove unused code

commit 89469db1ba77d762ef617eddd62ace6d34057fa7
Merge: bb96f8b 724cbdd
Author: TL-QL <40536493+TL-QL@users.noreply.github.com>
Date:   Wed Dec 22 20:40:49 2021 -0500

    Merge pull request #20 from PLT-Marble/feat/malloc-for-matrix

    fix

commit 724cbdd052137448fbc29c96c164e468909309e6
Author: Xindi Xu <xindixu0@gmail.com>
Date:   Wed Dec 22 20:38:33 2021 -0500

    fix

commit bb96f8b4b0bde2715635d2e3a4c57b817051781f
Merge: c933782 8868c57
Author: HuaxuanGAO <33063662+HuaxuanGAO@users.noreply.github.com>
Date:   Wed Dec 22 19:18:21 2021 -0500

    Merge pull request #19 from PLT-Marble/fix/syntax-fix

    fix

commit 8868c57fe43920aaa24f354f03983d7cfdf9d14e
Author: Xindi Xu <xindixu0@gmail.com>
Date:   Wed Dec 22 19:11:44 2021 -0500
```

```
49
50      fix
51
52  commit c93378204d85af35aac47f466a024ae880c2c658
53  Merge: 3f2594f bff40f7
54  Author: Xindi Xu <xindixu0@gmail.com>
55  Date:   Wed Dec 22 12:19:53 2021 -0500
56
57      Merge pull request #18 from PLT-Marble/feat/matrix-generator-zeros
58
59      feat: matrix generator: zeros()
60
61  commit bff40f7bb09da45811db1775751184c6a6f880d9
62  Author: Xindi Xu <xindixu0@gmail.com>
63  Date:   Wed Dec 22 12:07:11 2021 -0500
64
65      rows and cols can't be 0
66
67  commit 3be72c223dd76766d5a16887f0735776fdbfe8e4
68  Author: Xindi Xu <xindixu0@gmail.com>
69  Date:   Wed Dec 22 11:24:18 2021 -0500
70
71      remove more unused code
72
73  commit b66158ccbfc7e0123af135b7ea77be73b0143060
74  Author: Xindi Xu <xindixu0@gmail.com>
75  Date:   Wed Dec 22 11:21:53 2021 -0500
76
77      remove useless code
78
79  commit 4bc00a815b1a63391e80cf449366886fcf24e1f5
80  Author: Xindi Xu <xindixu0@gmail.com>
81  Date:   Wed Dec 22 11:17:36 2021 -0500
82
83      add zeros()
84
85  commit 3f2594f6ef27436fa884b9d1e7efb3714b3c52c5
86  Merge: a020e32 8cc4e28
87  Author: Xindi Xu <xindixu0@gmail.com>
88  Date:   Wed Dec 22 09:41:48 2021 -0500
89
90      Merge pull request #17 from PLT-Marble/wrapup
91
92      fixed for loop with multi stmts in loop-body
93
94  commit 8cc4e2837a8814a3b5a4908618306abb4ef3413e
95  Author: TooLazy-QL <qiwenluo98@gmail.com>
96  Date:   Wed Dec 22 04:06:01 2021 -0500
97
98      fixed for loop with multi stmts in loop-body
```

```
 99
100   commit a020e328323a67a4c4cf1fecd71da5cda4fb6578
101   Merge: d5c3770 a1c86c2
102   Author: Xindi Xu <xindixu0@gmail.com>
103   Date:   Tue Dec 21 22:39:02 2021 -0500
104
105       Merge pull request #15 from PLT-Marble/recursive-fibonacci
106
107       fibonacci test passed
108
109   commit d5c37706047bdd132fa4e61747fa52df7143e55a
110   Merge: b9c9c18 47e86fd
111   Author: Xindi Xu <xindixu0@gmail.com>
112   Date:   Tue Dec 21 22:28:06 2021 -0500
113
114       Merge pull request #16 from PLT-Marble/feat/demo-code
115
116       feat/add demo code-determinant
117
118   commit 47e86fd147e7bc0a1788279161b1fe691872e1f3
119   Author: Xindi Xu <xindixu0@gmail.com>
120   Date:   Tue Dec 21 22:15:17 2021 -0500
121
122       update variable name
123
124   commit 609a0e885827cd78daa3c6aa53ec3123b564f394
125   Author: Xindi Xu <xindixu0@gmail.com>
126   Date:   Tue Dec 21 22:09:55 2021 -0500
127
128       add demo code
129
130   commit a1c86c2e68032e771af6f26dcccd3c37a9b35dad
131   Author: Huaxuan Gao <gogoghx@163.com>
132   Date:   Mon Dec 20 15:13:47 2021 -0500
133
134       fibonacci test passed
135
136   commit b9c9c180f98d97948f797a7bd207a89f11ffa701
137   Merge: 7006f8d c51058f
138   Author: TL-QL <40536493+TL-QL@users.noreply.github.com>
139   Date:   Mon Dec 20 00:59:15 2021 -0500
140
141       Merge pull request #14 from PLT-Marble/wrapup
142
143       Resolved all warnings and parser conflicts
144
145   commit c51058f94edc17ec73687a4f97e9fa88bbd98eeb
146   Author: TooLazy-QL <qiwenluo98@gmail.com>
147   Date:   Mon Dec 20 00:47:18 2021 -0500
148
```

```
    Resolved all warnings and parser conflicts

commit 7006f8d768ecbb65b8c6d1ffb4a0d4cb0cfe1062
Merge: 2893ee0 cc703cc
Author: HuaxuanGAO <33063662+HuaxuanGAO@users.noreply.github.com>
Date:   Sat Dec 18 15:35:14 2021 -0500

    Merge pull request #13 from PLT-Marble/command-line

    Command line

commit cc703ccf4cfa50c3ecc461fda5076c3ee56260a4
Merge: eb9908f 2893ee0
Author: Huaxuan Gao <gogoghx@163.com>
Date:   Sat Dec 18 15:29:00 2021 -0500

    Merge branch 'main' into command-line

commit 2893ee0e8888d0ac04ba8ed942cd20591406697a
Merge: b34a55c 6b17a89
Author: HuaxuanGAO <33063662+HuaxuanGAO@users.noreply.github.com>
Date:   Sat Dec 18 15:28:34 2021 -0500

    Merge pull request #12 from PLT-Marble/clean-code

    remove redundant code and test case

commit eb9908f141af4914e102bcb8945bf67cc1e823b3
Author: Huaxuan Gao <gogoghx@163.com>
Date:   Sat Dec 18 15:27:31 2021 -0500

    command line tool marble interpreter

commit 6b17a891bfaa5c71d5f58a0d8f848c1b635dec05
Author: Huaxuan Gao <gogoghx@163.com>
Date:   Sat Dec 18 10:27:52 2021 -0500

    remove redundant code and test case

commit b34a55cc7a574e943e05fff17bfa116d361c084e
Merge: d5da2d9 3ca6390
Author: Xindi Xu <xindixu0@gmail.com>
Date:   Thu Dec 16 20:48:23 2021 -0500

    Merge pull request #11 from PLT-Marble/test/more-tests

    Tests: function failure tests, comment

commit 3ca63900b199416c600ecafb79e7584a8af1cb94
Author: Xindi Xu <xindixu0@gmail.com>
```

```
Date:   Thu Dec 16 20:36:39 2021 -0500

    failed tests

commit a7bb021d036694ad64b3e9695737bfa4288e9e31
Author: Xindi Xu <xindixu0@gmail.com>
Date:   Thu Dec 16 20:29:32 2021 -0500

    add more tests

commit d5da2d9607c63c6c0a58900e3265097dca7b188b
Merge: acbca4c 54dd6b3
Author: Xindi Xu <xindixu0@gmail.com>
Date:   Sat Dec 4 15:26:46 2021 -0500

    Merge pull request #10 from PLT-Marble/stmtNexpr

    stmt and expr done

commit 54dd6b380f747450b3b0c841846c5ff83f80dded
Author: Xindi Xu <xindixu0@gmail.com>
Date:   Sat Dec 4 15:26:10 2021 -0500

    update tests

commit 59896807874ffd7b1db3aca971b09d7c8313185f
Merge: 2885489 acbca4c
Author: Xindi Xu <xindixu0@gmail.com>
Date:   Sat Dec 4 15:19:39 2021 -0500

    Merge branch 'main' into stmtNexpr

commit acbca4ca0c850de1a2bde637ef06a94d813c41d4
Merge: 7631fb2 cde43e9
Author: HuaxuanGAO <33063662+HuaxuanGAO@users.noreply.github.com>
Date:   Sat Dec 4 15:07:02 2021 -0500

    Merge pull request #5 from PLT-Marble/feat/explicit-function-return-
type

    Feat: function return type, formal, and main

commit cde43e9e958412f6ba5bafb8260b9eab6b46dcfa
Merge: d2f6b21 67f53dc
Author: Xindi Xu <xindixu0@gmail.com>
Date:   Thu Dec 2 20:45:35 2021 -0500

    Merge pull request #9 from PLT-Marble/feat/treat-main-function-as-
normal-functions

```

```
    feat: treat main function as normal functions

commit 67f53dcc7e353d88c505ead3960f6c24380ffc5c
Merge: 258aed8 0ea75c1
Author: Huaxuan Gao <gogoghx@163.com>
Date:   Thu Dec 2 20:38:05 2021 -0500

    Merge branch 'feat/treat-main-function-as-normal-functions' of ssh://github.com/PLT-Marble/Marble into feat/treat-main-function-as-normal-functions

commit 258aed81830ddc95d12ad829ea261edefcd44a49
Author: Huaxuan Gao <gogoghx@163.com>
Date:   Thu Dec 2 20:37:52 2021 -0500

    comment type cast in test case

commit 0ea75c1d44dd7a5437af79d7bddef0058329469c
Author: Xindi Xu <xindixu0@gmail.com>
Date:   Thu Dec 2 20:35:34 2021 -0500

    fix: reset marble.ml

commit 2aea72085b539b1e912ae1475d86b48853f1c067
Author: Xindi Xu <xindixu0@gmail.com>
Date:   Thu Dec 2 20:24:52 2021 -0500

    fix: typo

commit 15ad1608c67158eff919d2479fe7a62057e65900
Author: Xindi Xu <xindixu0@gmail.com>
Date:   Thu Dec 2 20:22:33 2021 -0500

    fix: typo

commit 48ec5fcf0c2bbc421dbb20261b701dfdcca28f88
Author: Xindi Xu <xindixu0@gmail.com>
Date:   Thu Dec 2 20:15:46 2021 -0500

    feat: treat main function as normal functions

commit d2f6b21edc26b9ab8cc20be7f4ea487ffa45ef94
Author: Xindi Xu <xindixu0@gmail.com>
Date:   Thu Dec 2 01:27:43 2021 -0500

    test: return values

commit 8f31518cf4983f1d95cc805869b85f02564341ce
Author: Xindi Xu <xindixu0@gmail.com>
Date:   Thu Dec 2 00:13:44 2021 -0500
```

```
295
296         clean up: function_decls
297
298     commit 336f2202823acbb468fddce2d7edbb4ba9d1eb8f
299     Author: Xindi Xu <xindixu0@gmail.com>
300     Date:   Wed Dec 1 23:48:24 2021 -0500
301
302         fix: add return types for predefined functions
303
304     commit 28ccce4e8ab8eaa906153b28f9491a466e79869f
305     Author: Xindi Xu <xindixu0@gmail.com>
306     Date:   Wed Dec 1 18:21:11 2021 -0500
307
308         revert .gitignore
309
310     commit 288548998990164a732e2cf15dbc652dd0bc35b6
311     Author: TooLazy-QL <qiwenluo98@gmail.com>
312     Date:   Wed Dec 1 00:04:49 2021 -0500
313
314         stmt and expr done
315
316     commit f6040da7e76c703d9a77d24dd4573cf57bfccce9
317     Author: Xindi Xu <xindixu0@gmail.com>
318     Date:   Wed Dec 1 00:04:25 2021 -0500
319
320         fix after merge
321
322     commit 843205c5f4e88251de300ed0ad5c75bdff47ee00
323     Author: Xindi Xu <xindixu0@gmail.com>
324     Date:   Tue Nov 30 23:56:00 2021 -0500
325
326         update after merging
327
328     commit d4a8d1ad1ca521fe4b73957238990b286d75b71f
329     Author: Xindi Xu <xindixu0@gmail.com>
330     Date:   Tue Nov 30 19:56:57 2021 -0500
331
332         update gitignore
333
334     commit 9188212918e74c0c91610178952eea894500bb5d
335     Merge: 89e5786 7631fb2
336     Author: Xindi Xu <xindixu0@gmail.com>
337     Date:   Tue Nov 30 19:56:22 2021 -0500
338
339         Merge branch 'main' into feat/explicit-function-return-type
340
341     commit 89e5786c828b6d79faa8c7f76c08362eff119096
342     Merge: 3775515 c47bce0
343     Author: Xindi Xu <xindixu0@gmail.com>
344     Date:   Tue Nov 30 19:50:37 2021 -0500
```

```
345
346         Merge pull request #7 from PLT-Marble/feat/func-formal-final
347
348         Feat/func formal final
349
350     commit c47bce0d0729ed77eb707b1b9ad230f442c54fbe
351     Author: Xindi Xu <xindixu0@gmail.com>
352     Date:   Tue Nov 30 19:24:26 2021 -0500
353
354         feat: handle formal for functions
355
356     commit 377551500885091abcb614b1673017c2fa87b7ed
357     Author: Xindi Xu <xindixu0@gmail.com>
358     Date:   Tue Nov 30 18:21:25 2021 -0500
359
360         update test script
361
362     commit 7631fb24ff5580a80d03458ed948ea685f01b18c
363     Merge: f46342c 15349eb
364     Author: HuaxuanGAO <33063662+HuaxuanGAO@users.noreply.github.com>
365     Date:   Tue Nov 30 18:10:52 2021 -0500
366
367         Merge pull request #4 from PLT-Marble/matrix
368
369         Matrix
370
371     commit f46342cbda36b92c6fa13c159f473229619eb90e
372     Merge: a7cf14a 69e312e
373     Author: HuaxuanGAO <33063662+HuaxuanGAO@users.noreply.github.com>
374     Date:   Tue Nov 30 18:10:44 2021 -0500
375
376         Merge pull request #3 from PLT-Marble/float-and-bool-types
377
378         Float and bool types
379
380     commit c91154df6d9aa5dff28b3060f3ecd585c29d5cde
381     Author: Xindi Xu <xindixu0@gmail.com>
382     Date:   Tue Nov 30 01:29:48 2021 -0500
383
384         function return and formals
385
386     commit 0231b410acff532a53894a905fadf3bb8e14d5cb
387     Author: Xindi Xu <xindixu0@gmail.com>
388     Date:   Tue Nov 30 01:28:01 2021 -0500
389
390         add and check functions
391
392     commit eefba159b94c4819e1a67ecfe443cf3bcd9baf17
393     Author: Xindi Xu <xindixu0@gmail.com>
394     Date:   Tue Nov 30 01:27:01 2021 -0500
```

```
395
396         format
397
398     commit fa91c7bb5e46d81a9478529d6021953734344242
399     Author: Xindi Xu <xindixu0@gmail.com>
400     Date:   Mon Nov 29 23:12:24 2021 -0500
401
402         function with explicit return type
403
404     commit 15349ebdf3d200990a788839d6e9cce3c06ba7e6
405     Author: Huaxuan Gao <gogoghx@163.com>
406     Date:   Mon Nov 29 16:00:20 2021 -0500
407
408         rows cols for mat
409
410     commit 13209a22d45cb4ec45c678487f292a1d07930bc7
411     Author: Huaxuan Gao <gogoghx@163.com>
412     Date:   Mon Nov 29 15:44:20 2021 -0500
413
414         mat bin op done
415
416     commit 05e64b69a3926ef94778557b436e333948604f50
417     Author: Huaxuan Gao <gogoghx@163.com>
418     Date:   Mon Nov 29 15:07:45 2021 -0500
419
420         mat getter setter
421
422     commit 8f382633467d64f7e38f22b7e43c3cd7a6e8ed06
423     Author: Huaxuan Gao <gogoghx@163.com>
424     Date:   Mon Nov 29 12:57:58 2021 -0500
425
426         print matrix
427
428     commit 73e5cd5002f5f7e0ffcde8e725ba129ce8035c83
429     Merge: 5bf5714 69e312e
430     Author: Huaxuan Gao <gogoghx@163.com>
431     Date:   Mon Nov 29 11:33:44 2021 -0500
432
433         Merge branch 'float-and-bool-types' into matrix
434
435     commit 5bf57142305b86c061ee5231407a91c8c25c2067
436     Author: Huaxuan Gao <gogoghx@163.com>
437     Date:   Mon Nov 29 11:23:37 2021 -0500
438
439         test forbin push to main
440
441     commit 69e312ee0325f01d114d8f9aebbd76f048cebffb
442     Merge: 4c14bf5 a7cf14a
443     Author: AlbertG <gogoghx@163.com>
444     Date:   Mon Nov 29 11:14:28 2021 -0500
```

```
445
446       Merge branch 'main' into float-and-bool-types
447
448   commit 4c14bf5400854c2bfca138f4a5b7a48d48a4691d
449   Author: Huaxuan Gao <gogoghx@163.com>
450   Date:   Mon Nov 29 11:01:41 2021 -0500
451
452       printb
453
454   commit 384990fc400bd60172b6f8866c51e06c764ff835
455   Author: Huaxuan Gao <gogoghx@163.com>
456   Date:   Mon Nov 29 10:03:14 2021 -0500
457
458       float op done
459
460   commit f1da3ac675c36aa9e3ac25fff1711c7eb20c8347
461   Author: Huaxuan Gao <gogoghx@163.com>
462   Date:   Sun Nov 28 17:42:08 2021 -0500
463
464       add float type
465
466   commit 2d16f98cd755c85d4e88bb120e251353319fc09a
467   Author: Xindi Xu <xindixu0@gmail.com>
468   Date:   Fri Nov 26 22:41:41 2021 -0500
469
470       format
471
472   commit a7cf14ae3bb3da15704893cb6d3354d7f169ddc8
473   Author: TooLazy-QL <qiwenluo98@gmail.com>
474   Date:   Fri Nov 26 00:29:07 2021 -0500
475
476       Added while, for, if-else(no elif) - semant&codegen
477
478   commit 1b06bf8f45ac386879d348a1de6433cdf4ad401c
479   Author: TooLazy-QL <qiwenluo98@gmail.com>
480   Date:   Fri Nov 26 00:28:37 2021 -0500
481
482       Added while, for, if-else(no elif)
483
484   commit 42f1ea6fe3354f7c17e1e7fe739b93931d221cc6
485   Author: Huaxuan Gao <gogoghx@163.com>
486   Date:   Mon Nov 22 10:56:07 2021 -0500
487
488       add Make file
489
490   commit 86101c96dc1535c465e184ad2fda61d1cfb2ad97
491   Merge: a9571fb 8154ccc
492   Author: Xindi Xu <xindixu0@gmail.com>
493   Date:   Sat Nov 20 16:01:10 2021 -0500
494
```

```
    Merge pull request #2 from PLT-Marble/helloworld

    Helloworld

commit 8154ccca8b9e7cfb455fd894278c913c68cbd805
Author: Huaxuan Gao <gogoghx@163.com>
Date:   Sun Nov 14 15:10:47 2021 -0500

    update readme with instructions to run

commit 4774d996d7e14150f00b2bf403c6769b4a9df4ab
Author: Huaxuan Gao <gogoghx@163.com>
Date:   Sun Nov 14 10:36:29 2021 -0500

    update makefile

commit 9d5f0ff663f9f782743738b9f3b3c5f75feff4ab
Author: TooLazy-QL <qiwenluo98@gmail.com>
Date:   Sun Nov 14 02:57:22 2021 -0500

    Completed helloworld - stmt&expr in sement&codegen with tests

commit f231d732568c93f48874fb542b8b6a8ee65f79b8
Author: Huaxuan Gao <gogoghx@163.com>
Date:   Sat Nov 13 17:42:09 2021 -0500

    added test script

commit f51f427612c92e96b05e88c1c53b11c05bb24119
Author: Huaxuan Gao <gogoghx@163.com>
Date:   Sat Nov 13 17:12:48 2021 -0500

    print(1) success

commit a9571fb25284754c4b0ca6f3720dbb03c6b66847
Author: Xindi Xu <xindixu0@gmail.com>
Date:   Sat Nov 13 00:41:27 2021 -0500

    handle vars and funcs in codegen

commit 489d2ea86769280c7864d46f11a3fe77abf6f321
Author: TooLazy-QL <qiwenluo98@gmail.com>
Date:   Fri Nov 12 23:16:01 2021 -0500

    Added _tags

commit 04be1cdb8aaba4849cbe4d95d5d0b8ada01007fe
Author: TooLazy-QL <qiwenluo98@gmail.com>
Date:   Fri Nov 12 23:14:53 2021 -0500

```

```
    Comment out functionalities not included in this iter

commit 5914695363640a7b4392f8c4687cf2c23f1e9141
Author: TooLazy-QL <qiwenluo98@gmail.com>
Date:   Fri Nov 12 21:25:04 2021 -0500

    Updated semant=stmt

commit 4de66c32018c94b0dc5113c193e9fe3983c4fa50
Author: Xindi Xu <xindixu0@gmail.com>
Date:   Fri Nov 12 17:29:23 2021 -0500

    update after merging

commit 719eb020d36a5510c21bd823de5a6e093d264daf
Author: Huaxuan Gao <gogoghx@163.com>
Date:   Fri Nov 12 17:23:12 2021 -0500

    marble.ml

commit 89a04bfa67790e3eb9c195a7837c2c8dc05794c0
Merge: ca01390 c03d547
Author: Xindi Xu <xindixu0@gmail.com>
Date:   Fri Nov 12 17:19:02 2021 -0500

    Merge pull request #1 from PLT-Marble/pretty-printing

    Pretty printing

commit c03d547e0080425e6394d1abba12e92dde6fafed
Merge: 5b39dac ca01390
Author: Xindi Xu <xindixu0@gmail.com>
Date:   Fri Nov 12 17:17:04 2021 -0500

    Merge branch 'main' into pretty-printing

commit 5b39dac0cf77ba62d8d41c77a0aed9dca370dff3
Merge: e7246b4 b522a17
Author: Huaxuan Gao <gogoghx@163.com>
Date:   Fri Nov 12 16:55:14 2021 -0500

    merge

commit ca0139015642347807a416a2b3b93ba6d7687872
Author: Yixin Pan <ypan37@jhu.edu>
Date:   Fri Nov 12 16:52:42 2021 -0500

    expr in codegen

commit e7246b42afce23555e3bb2a7aed74fe6707742f5
```

```
Author: Huaxuan Gao <gogoghx@163.com>
Date:   Fri Nov 12 16:50:29 2021 -0500

    semant

commit 856c7b7987d7c1d2c499f859fcff689650bf6e25
Author: Yixin Pan <ypan37@jhu.edu>
Date:   Fri Nov 12 16:32:22 2021 -0500

    update semant for expr

commit 8925cc6207b52357d935c40889abdd0d30ba6714
Author: TooLazy-QL <qiwenluo98@gmail.com>
Date:   Thu Nov 11 23:59:36 2021 -0500

    Added codegen.ml-stmt part

commit 0fa61fbcf95c08cde1c0127aac724d18fb8c1efe
Author: TooLazy-QL <qiwenluo98@gmail.com>
Date:   Thu Nov 11 22:50:07 2021 -0500

    Updated semant.ml-stmt part

commit d154b7f20383466f3d54e2def07530d56dfb2ede
Author: TooLazy-QL <qiwenluo98@gmail.com>
Date:   Thu Nov 11 21:36:26 2021 -0500

    Added semant.ml-stmt part

commit 4f9388660c7488e4593fd81ae7328db79004811b
Author: TooLazy-QL <qiwenluo98@gmail.com>
Date:   Thu Nov 11 21:35:51 2021 -0500

    Updated VDeclare in stmt

commit 82f0cdb69a2d53101512707ababd8be30f8c74b5
Author: Huaxuan Gao <gogoghx@163.com>
Date:   Thu Nov 11 15:08:49 2021 -0500

    microc semant

commit fb19b768b406ca43ed41105ee62a6b5a9e648b46
Author: Huaxuan Gao <gogoghx@163.com>
Date:   Thu Nov 11 14:05:49 2021 -0500

    half semant

commit b522a17951cdacace1aa3efbac50f6c93f46ff08
Author: TooLazy-QL <qiwenluo98@gmail.com>
Date:   Wed Nov 10 20:41:22 2021 -0500
```

```
645
646         Created semant.ml and codegen.ml
647
648     commit cbdbdfec2396ec79109ce9cdb6063b50cff49da0
649     Merge: f9eeaf9 f3bb6fe
650     Author: Xindi Xu <xindixu0@gmail.com>
651     Date:   Tue Nov 9 14:33:39 2021 -0500
652
653         Merge branch 'main' of https://github.com/PLT-Marble/Marble
654
655     commit f9eeaf99db5b8a62873472a1548e0415d251a3f3
656     Author: Xindi Xu <xindixu0@gmail.com>
657     Date:   Tue Nov 9 14:33:37 2021 -0500
658
659         meeting notes with TA
660
661     commit f3bb6fe3f01af1be1b75dfe7ad7a5163f1a81459
662     Author: Yixin Pan <ypan37@jhu.edu>
663     Date:   Mon Nov 8 23:56:21 2021 -0500
664
665         update on sast for expr
666
667     commit fd4b357ae3150ef22257d7f1609500c95aef666f
668     Author: TooLazy-QL <qiwenluo98@gmail.com>
669     Date:   Mon Nov 8 21:31:15 2021 -0500
670
671         Updated stmt&elifstmts&assignstmt for ast and sast
672
673     commit 9405274dd48e121bb6a3f4c17eb745922d9b1823
674     Author: Yixin Pan <ypan37@jhu.edu>
675     Date:   Mon Nov 8 20:52:25 2021 -0500
676
677         quick fix for MLit
678
679     commit b99d0f916489d60ba3e0a89e0f9d066b9267525d
680     Author: Yixin Pan <ypan37@jhu.edu>
681     Date:   Mon Nov 8 16:52:19 2021 -0500
682
683         update on expr for ast and sast
684
685     commit 6c5eaf9c5b1f0bba55e4828bff165b85c0552282
686     Author: Yixin Pan <ypan37@jhu.edu>
687     Date:   Mon Nov 8 16:43:53 2021 -0500
688
689         quick update on parser
690
691     commit 5d6241fc0c74600d5ecb493c6217fe446ad0f269
692     Author: Huaxuan Gao <gogoghx@163.com>
693     Date:   Sat Nov 6 20:29:57 2021 -0400
694
```

```
    print matrix

commit 16bbfbcc76b4f6fcf90fc4274344d1d627376e1a
Author: Huaxuan Gao <gogoghx@163.com>
Date:   Sat Nov 6 17:56:31 2021 -0400

    add prefix to print

commit 6d200eff2d72d9e45c56c547ec88ddf973b50744
Author: Huaxuan Gao <gogoghx@163.com>
Date:   Sat Nov 6 17:11:55 2021 -0400

    printable

commit 0dd98dfd30dbee16ffa017ec54835307c2c33ef3
Author: Xindi Xu <xindixu0@gmail.com>
Date:   Sat Nov 6 16:01:35 2021 -0400

    update ast and add sast

commit 0ef760cedc54fc92bbf341f74b146a5dffd46e47
Author: TooLazy-QL <qiwenluo98@gmail.com>
Date:   Fri Nov 5 15:22:08 2021 -0400

    Changed the order of stmt append to match with stmts in fdecl

commit 6073268b7094a605f54b86a85a6152075e891d2e
Author: TooLazy-QL <qiwenluo98@gmail.com>
Date:   Fri Nov 5 11:45:49 2021 -0400

    clean file

commit 83dc1e2921beb64a28a487934d775e181c606be7
Author: TooLazy-QL <qiwenluo98@gmail.com>
Date:   Fri Nov 5 11:43:56 2021 -0400

    Fixed NOT,AND,OR precedence and associativity

commit 1a5f059c8e4483ebe7d9ce0b06d016f7049a5e3c
Author: Xindi Xu <xindixu0@gmail.com>
Date:   Fri Nov 5 11:06:23 2021 -0400

    TA comments Nov. 5

commit a47cdd277d997e6ce2faeba9bde04f7d9291ec0d
Author: Yixin Pan <ypan37@jhu.edu>
Date:   Sat Oct 30 20:56:18 2021 -0400

    clean parser file output

```

```
commit be855c1a7cc0f93bf55afc97ba9e9f6d4d5021a7
Author: Yixin Pan <ypan37@jhu.edu>
Date:   Sat Oct 30 20:54:50 2021 -0400

    fix logic operator

commit 64b259d129ee1934f6e711cac22a433e86dd573c
Merge: 36fa031 b445df9
Author: Huaxuan Gao <huaxuan@A-G.localdomain>
Date:   Sat Oct 30 20:40:51 2021 -0400

    Merge branch 'main' of ssh://github.com/PLT-Marble/Marble into main

commit 36fa031859c3437b478dc1baf26316408e439a73
Author: Huaxuan Gao <huaxuan@A-G.localdomain>
Date:   Sat Oct 30 20:39:06 2021 -0400

    matrix op keywords

commit b445df9b3417c4dc8e50bc78c2f3fc9af51136c9
Author: Xindi Xu <xindixu0@gmail.com>
Date:   Thu Oct 28 22:59:43 2021 -0400

    fix: mod

commit 4c7cf223e284b402318794eb477723e1a8f56d24
Author: Xindi Xu <xindixu0@gmail.com>
Date:   Thu Oct 28 22:30:44 2021 -0400

    fix: update program, decls rules

commit 83f996ac0d9217beb7f6ab663a7244be4ed73163
Merge: ab6236d 55fca9c
Author: Xindi Xu <xindixu0@gmail.com>
Date:   Thu Oct 28 22:03:20 2021 -0400

    Merge branch 'main' of https://github.com/PLT-Marble/Marble

commit ab6236d9842691fabe27bfe8fa484b4b97cb43fe
Author: Xindi Xu <xindixu0@gmail.com>
Date:   Thu Oct 28 21:16:22 2021 -0400

    fixme: add TA review comments

commit 55fca9c80a6d5cc4a14f03f59402106af4ab1834
Merge: 34668a7 4bc6f4e
Author: TooLazy-QL <qiwenluo98@gmail.com>
Date:   Thu Oct 28 17:01:21 2021 -0400

    Merge branch 'main' of https://github.com/PLT-Marble/Marble into main
```

```
795
796   commit 34668a7e39bbcfa594a6a190748233efb6845c5d
797   Author: TooLazy-QL <qiwenluo98@gmail.com>
798   Date:   Thu Oct 28 17:00:52 2021 -0400
799
800       Added assignment
801
802   commit 4bc6f4e22ee20331663b03ea4b1b1c706c2d033c
803   Author: Huaxuan Gao <huaxuan@A-G.localdomain>
804   Date:   Thu Oct 28 11:45:47 2021 -0400
805
806       add TRUE/FALSE to keywords, remove BOOL
807
808   commit a910c799c754771cf42c2e9a4f0eb66924ff6216
809   Author: Huaxuan Gao <huaxuan@A-G.localdomain>
810   Date:   Thu Oct 28 11:27:36 2021 -0400
811
812       fix shift/reduce conflict
813
814   commit 7f1d49c4b4ee9e67368029b98eb2ebd539c25769
815   Author: Huaxuan Gao <huaxuan@A-G.localdomain>
816   Date:   Thu Oct 28 11:17:07 2021 -0400
817
818       fix symbol not found and some typo
819
820   commit 4b4de680eb38469d6772314d61c72523e156431f
821   Author: Yixin Pan <ypan37@jhu.edu>
822   Date:   Wed Oct 27 23:31:52 2021 -0400
823
824       quick fix for continue and break
825
826   commit b871f4a803e052d98e01f14ca8402df56d059cbf
827   Author: Yixin Pan <ypan37@jhu.edu>
828   Date:   Wed Oct 27 23:30:13 2021 -0400
829
830       parser for expr
831
832   commit a87667e512b0ecbdeeae449b411843ab6db0b525
833   Merge: b4a0733 fef682c
834   Author: TooLazy-QL <qiwenluo98@gmail.com>
835   Date:   Wed Oct 27 19:47:05 2021 -0400
836
837       Merge branch 'main' of https://github.com/PLT-Marble/Marble into main
838
839   commit b4a0733f01c702318b638d3491d31eaee09285e4
840   Author: TooLazy-QL <qiwenluo98@gmail.com>
841   Date:   Wed Oct 27 19:46:41 2021 -0400
842
843       Added stmts, stmt and elifstmts
844
```

```
commit fef682ce9bbedce910f14e07e579113fca4e70dc
Author: Xindi Xu <xindixu0@gmail.com>
Date:   Wed Oct 27 19:31:58 2021 -0400

    fix: use epsion instead of opt for formals

commit 7a73b1d76cc36f7a4b4f7f9be76cb5efbee76774
Author: Xindi Xu <xindixu0@gmail.com>
Date:   Wed Oct 27 19:28:10 2021 -0400

    feat: add CFG (program to type)

commit c5bbdcd24753494e1db2ab8c6228ee198999b16a
Author: Xindi Xu <xindixu0@gmail.com>
Date:   Wed Oct 27 19:27:41 2021 -0400

    fix: add formatter

commit 578df2460c80da6553f69ba72c7d62bfec9e4562
Merge: a395a7a d14b4c4
Author: Huaxuan Gao <huaxuan@A-G.localdomain>
Date:   Tue Oct 26 19:21:43 2021 -0400

    Merge branch 'main' of ssh://github.com/PLT-Marble/Marble into main

commit a395a7aee2eeb23bb359e19ab84f80a807702ccc
Author: Huaxuan Gao <huaxuan@A-G.localdomain>
Date:   Tue Oct 26 19:21:41 2021 -0400

    MLIT

commit d14b4c4a3599dff20f0799d7a262a28e41cd37e1
Author: TooLazy-QL <qiwenluo98@gmail.com>
Date:   Tue Oct 26 14:58:21 2021 -0400

    Moved . to operator

commit 034149badc2841b72e5a8b5f0231b12696d8d71e
Merge: 9aa36e5 7b06435
Author: TooLazy-QL <qiwenluo98@gmail.com>
Date:   Tue Oct 26 14:57:10 2021 -0400

    revert

commit 9aa36e5b2e55e309f7921aa6492b6e2d3a3c316a
Author: TooLazy-QL <qiwenluo98@gmail.com>
Date:   Tue Oct 26 14:55:08 2021 -0400

    Moved . to operator

```

```
commit 7b06435b702e9b3a12e56b22337ebc09e32bde9e
Author: Huaxuan Gao <huaxuan@A-G.localdomain>
Date:   Tue Oct 26 13:19:57 2021 -0400

    parser initial setup

commit 60807419fe02004b049f64e5e8f0f096c7a546d7
Author: TooLazy-QL <qiwenluo98@gmail.com>
Date:   Tue Oct 26 12:16:41 2021 -0400

    Added constructor to keywords

commit ccdbeabb56299c6f8486c6f028710b3ec512c635
Author: TooLazy-QL <qiwenluo98@gmail.com>
Date:   Mon Oct 25 23:38:39 2021 -0400

    Added matrix literal to tokems

commit fbe2b19688650ffa34bf9be771cb22c57c4946b2
Author: TooLazy-QL <qiwenluo98@gmail.com>
Date:   Mon Oct 25 21:05:17 2021 -0400

    Deleted matrix from keywords and added class to keywords

commit dc78ff00f8674d95fcf32f91cc6dcc70876d153c
Author: TooLazy-QL <qiwenluo98@gmail.com>
Date:   Mon Oct 25 20:22:06 2021 -0400

    Add . as a seperator

commit 7842ad8afc20d7ac3cb5206cfecc592c45917ad8
Author: TooLazy-QL <qiwenluo98@gmail.com>
Date:   Mon Oct 25 19:42:24 2021 -0400

    Added true/false to tokens

commit 6f690c091c2394becb4bbab6cea09843ad512b0e
Author: TooLazy-QL <qiwenluo98@gmail.com>
Date:   Mon Oct 25 19:33:28 2021 -0400

    Added keywords for types

commit a97b17d02226a10368f1c8c53a0234cce7b11e90
Author: TooLazy-QL <qiwenluo98@gmail.com>
Date:   Mon Oct 25 19:09:12 2021 -0400

    Added += and -= as tokens

commit d15b2eef0f8fe499e213a2c711f737e8e2e49a44
Author: TooLazy-QL <qiwenluo98@gmail.com>
```

```
Date:   Mon Oct 25 18:59:22 2021 -0400

    Changed float format to digits.digits

commit f325a3973b28a9fec4a4c514a75c75d0552b2df7
Author: TooLazy-QL <qiwenluo98@gmail.com>
Date:   Mon Oct 25 18:53:38 2021 -0400

    Deleted keyword try, catch, throw and export

commit a18cbe0785bb192e16da50127ab737754e7a7572
Author: Huaxuan Gao <huaxuan@A-G.localdomain>
Date:   Sat Oct 23 14:37:15 2021 -0400

    test scanner

commit cdfd2d658fa58f173826ef45e74ab5675e202e77
Author: Huaxuan Gao <huaxuan@A-G.localdomain>
Date:   Sat Oct 23 10:23:23 2021 -0400

    init scanner

commit 1caafd34ff5c14c8be91f20634d8b35c0748f042
Author: HuaxuanGAO <33063662+HuaxuanGAO@users.noreply.github.com>
Date:   Sat Oct 23 07:47:26 2021 -0400

    Initial commit
```

8.2 Code Listing

8.2.1 scanner.mll

OCaml | 复制代码

```
{ open Parser
(* Reference: https://ocaml.org/manual/lexyacc.html *)
(* 2.7 keywords *)
let keyword_table = Hashtbl.create 53
  let _ =
    List.iter (fun (kwd, tok) -> Hashtbl.add keyword_table kwd tok)
      [
        "return", RETURN;
        "function", FUNCTION;
        "int", INT;
        "float", FLOAT;
        "bool", BOOL;
        "matrix", MATRIX;
        "null", NULL;
        "while", WHILE;
        "for", FOR;
        "if", IF;
        "else", ELSE;
      ]
}

let digit = ['0' - '9']
let digits = digit+
let float = digits '.' digits
let quote = ['\'' '\"']

rule tokenize = parse
  [' ' '\t' '\r' '\n'] { tokenize lexbuf }
(* 2.2 comments *)
| "//" { comment lexbuf }
| "/*" { comments lexbuf }
(* 2.1 types *)
| digits as lit { ILIT(int_of_string lit) }
| float as lit { FLIT(float_of_string lit) }
| "true" { BLIT(true) }
| "false" { BLIT(false) }
(* 2.6 operators *)
| '='        { ASSIGN }
| "+="       { PLUSASSIGN }
| "-="       { MINUSASSIGN }
| '+'        { PLUS }
| '-'        { MINUS }
| '*'        { TIMES }
| '/'        { DIVIDE }
| '%'        { MOD }
| "=="       { EQ }
| "!="       { NEQ }
| '<'        { LT }
```

```
49   | "<="       { LEQ }
50   | ">"        { GT }
51   | ">="       { GEQ }
52   | "!"        { NOT }
53   | "&&"       { AND }
54   | "||"       { OR }
55   (* 2.7 seperators *)
56   | '('        { LPAREN }
57   | ')'        { RPAREN }
58   | '['        { LBRACK }
59   | ']'        { RBRACK }
60   | '{'        { LBRACE }
61   | '}'        { RBRACE }
62   | ','        { COMMA }
63   | ';'        { SEMI }
64   (* 2.7 keywords *)
65   | ['A'-'Z' 'a'-'z'] ['A'-'Z' 'a'-'z' '0'-'9' '_'] * as id
66     { (*print_endline "scanner find id: ";
67       print_endline id; *)
68       try
69         Hashtbl.find keyword_table id
70       with Not_found ->
71         ID id }
72   (*  *)
73   | eof { EOF }
74   | _ as char { raise (Failure("illegal character " ^ Char.escaped char)) }
75
76   and comments = parse
77     "*/" { tokenize lexbuf }
78     | _    { comments lexbuf }
79   and comment = parse
80     '\n' { tokenize lexbuf }
81     | _ { comment lexbuf }
82
```

8.2.2 parser.mly

OCaml | 复制代码

```
%{
open Ast
open Lexing
let parse_error s =
      begin
        try
          let start_pos = Parsing.symbol_start_pos ()
          and end_pos = Parsing.symbol_end_pos () in
          Printf.printf "File \"%s\", line %d, characters %d-%d: \n"
            start_pos.pos_fname
            start_pos.pos_lnum
            (start_pos.pos_cnum - start_pos.pos_bol)
            (end_pos.pos_cnum - start_pos.pos_bol)
        with Invalid_argument(_) -> ()
      end;
      Printf.printf "Syntax error: %s\n" s;
      raise Parsing.Parse_error
%}

%token PLUS MINUS TIMES DIVIDE MOD
%token EQ NEQ LT LEQ GT GEQ
%token NOT AND OR
%token ASSIGN PLUSASSIGN MINUSASSIGN
%token LPAREN RPAREN LBRACK RBRACK LBRACE RBRACE COMMA SEMI
%token IF ELSE
%token WHILE FOR
%token RETURN FUNCTION
%token NULL
%token INT FLOAT BOOL MATRIX

%token <int> ILIT
%token <float> FLIT
%token <bool> BLIT
%token <string> ID
%token EOF

%left IF ELSE
%right ASSIGN
%left OR
%left AND
%left EQ NEQ
%left LT GT LEQ GEQ
%left PLUS MINUS
%left TIMES DIVIDE MOD
%left LBRACK RBRACK
%right NOT

%start program
```

```
%type <Ast.program> program

%%

program:
  decls EOF { $1 }

decls:
   /* nothing */ { ([], [])               }
 | decls vdecl { (($2 :: fst $1), snd $1) }
 | decls fdecl { (fst $1, ($2 :: snd $1)) }


vdecl: dtype ID SEMI { $1, $2 }

fdecl: dtype FUNCTION ID LPAREN formals RPAREN LBRACE stmts RBRACE {
    {
        return = $1;
        fname = $3;
        formals = List.rev $5;
        stmts = List.rev $8;
    }
}

formals:
/* nothing */ { [] }
| dtype ID { [($1, $2)] }
| formals COMMA dtype ID { ($3, $4) :: $1 }

dtype:
| INT { Int }
| FLOAT { Float }
| BOOL { Bool }
| MATRIX { Matrix }

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

stmt:
  expr SEMI { Expr ($1) }
| RETURN expr SEMI { Return ($2) }
| dtype ID SEMI { VDeclare($1, $2) }
| assignstmt SEMI { AssignStmt($1) }
| WHILE LPAREN expr RPAREN LBRACE stmts RBRACE  {While($3, List.rev $6)}
| FOR LPAREN assignstmt SEMI expr SEMI assignstmt RPAREN LBRACE stmts
RBRACE {For($3, $5, $7, List.rev $10)}
| IF LPAREN expr RPAREN LBRACE stmts RBRACE {If($3, List.rev $6)}
| IF LPAREN expr RPAREN LBRACE stmts RBRACE ELSE LBRACE stmts RBRACE
{IfElse($3, List.rev $6, List.rev $10)}
```

```
 97
 98  assignstmt:
 99    dtype ID ASSIGN expr { VDeAssign($1, $2, $4) }
100  | ID PLUSASSIGN expr { Assign($1, Binop(Id($1), Add, $3)) }
101  | ID MINUSASSIGN expr { Assign($1, Binop(Id($1), Sub, $3)) }
102  | ID ASSIGN expr { Assign($1, $3) }
103  | expr LBRACK expr COMMA expr RBRACK ASSIGN expr { MAssign($1, $3, $5, $8)
     }
104
105  expr:
106   ILIT  { ILit($1) }
107  | FLIT { FLit($1) }
108  | BLIT { BLit($1) }
109  | matrix { MLit($1) }
110  | expr LBRACK expr COMMA expr RBRACK { Access($1, $3, $5) }
111  | ID   { Id($1) }
112  | expr PLUS expr   { Binop($1, Add, $3) }
113  | expr MINUS expr  { Binop($1, Sub, $3) }
114  | expr TIMES expr  { Binop($1, Mul, $3) }
115  | expr DIVIDE expr { Binop($1, Div, $3) }
116  | ID LPAREN inputs RPAREN { Func($1, $3) }
117  | MINUS expr %prec NOT   { Unary(Neg, $2) }
118  | NOT expr         { Unary(Not, $2) }
119  | expr AND expr    { Binop($1, And, $3) }
120  | expr OR expr     { Binop($1, Or, $3) }
121  | expr MOD expr { Binop($1, Mod, $3) }
122  | expr EQ expr  { Binop($1, Eq, $3) }
123  | expr NEQ expr { Binop($1, Neq, $3) }
124  | expr LT expr  { Binop($1, Less, $3) }
125  | expr LEQ expr { Binop($1, Leq, $3) }
126  | expr GT expr  { Binop($1, Greater, $3) }
127  | expr GEQ expr { Binop($1, Geq, $3) }
128
129  inputs:
130   /* nothing */  { [] }
131  | expr           { [$1] }
132  | expr COMMA inputs    { $1 :: $3 }
133
134  matrix:
135    LBRACK matrix_row_list RBRACK { $2 }
136
137  matrix_row_list:
138  |  matrix_row { [$1] }
139  | matrix_row SEMI matrix_row_list { $1 :: $3 }
140
141  matrix_row:
142  | expr { [$1] }
143  | expr COMMA matrix_row { $1 :: $3 }
```

8.2.3 ast.ml

OCaml | 复制代码

```
type operator =
  | Add
  | Sub
  | Mul
  | Div
  | Mod
  | And
  | Or
  | Eq
  | Neq
  | Less
  | Leq
  | Greater
  | Geq

type uop = Neg | Not

type expr =
  | Binop of expr * operator * expr
  | ILit of int
  | FLit of float
  | BLit of bool
  | MLit of expr list list
  | Id of string
  | Func of string * expr list
  | Access of expr * expr * expr
  | Unary of uop * expr

type dtype = Int | Float | Bool | Matrix | Null

(*type elifstmt = Elif of expr * stmt list*)

type assignstmt =
  | VDeAssign of dtype * string * expr
  | Assign of string * expr
  | MAssign of expr * expr * expr * expr

type stmt =
  | Expr of expr
  | Return of expr
  | VDeclare of dtype * string
  | AssignStmt of assignstmt
  | If of expr * stmt list
  | IfElse of expr * stmt list * stmt list
  | For of assignstmt * expr * assignstmt * stmt list
  | While of expr * stmt list

type bind = dtype * string
```

```
49
50  type fdecl = {
51    return : dtype;
52    fname : string;
53    formals : bind list;
54    stmts : stmt list;
55  }
56
57  type program = bind list * fdecl list
58
59  (* Pretty-printing functions from microc *)
60  let string_of_typ = function
61    | Int -> "int"
62    | Null -> "null"
63    | Float -> "float"
64    | Bool -> "Bool"
65    | Matrix -> "matrix"
66
67  let string_of_op = function
68    | Add -> "+"
69    | Sub -> "-"
70    | Mul -> "*"
71    | Div -> "/"
72    | Mod -> "%"
73    | Eq -> "=="
74    | Neq -> "!="
75    | Less -> "<"
76    | Leq -> "<="
77    | Greater -> ">"
78    | Geq -> ">="
79    | And -> "&&"
80    | Or -> "||"
81
82  let string_of_uop = function Neg -> "-" | Not -> "!"
83
84  let rec string_of_expr = function
85    | ILit l -> string_of_int l
86    | FLit l -> string_of_float l
87    | BLit l -> string_of_bool l
88    | MLit l ->
89        let string_of_row l = String.concat "" (List.map string_of_expr l)
    in
90        String.concat "" (List.map string_of_row l)
91    | Id s -> s
92    | Binop (e1, o, e2) ->
93        string_of_expr e1 ^ " " ^ string_of_op o ^ " " ^ string_of_expr e2
94    | Unary (o, e) -> string_of_uop o ^ string_of_expr e
95    | Func (id, inputs) ->
96        id ^ "(" ^ String.concat ", " (List.map string_of_expr inputs) ^
    ");\n"
```

```
97      | Access (e1, e2, e3) ->
98          string_of_expr e1 ^ " " ^ string_of_expr e2 ^ " " ^ string_of_expr
    e3
99
100   let string_of_assignstmt = function
101     | VDeAssign (t, id, expr) ->
102         "VDeAssign: " ^ string_of_typ t ^ id ^ string_of_expr expr ^ ";\n"
103     | Assign (v, e) -> "Assign: " ^ v ^ " = " ^ string_of_expr e ^ ";\n"
104     | MAssign (id, r, c, v) ->
105         "MAssign: " ^ string_of_expr id ^ "[" ^ string_of_expr r ^ ", "
106         ^ string_of_expr c ^ "] = " ^ string_of_expr v ^ ";\n"
107
108   let rec string_of_stmt = function
109     | Expr expr -> string_of_expr expr ^ ";\n"
110     | Return expr -> "return: " ^ string_of_expr expr ^ ";\n"
111     | AssignStmt assignstmt -> string_of_assignstmt assignstmt
112     | VDeclare (t, id) -> "VDeclare: " ^ string_of_typ t ^ " " ^ id ^ ";\n"
113     | While (e, ss) ->
114         "While(" ^ string_of_expr e ^ ") { "
115         ^ String.concat "\n" (List.map string_of_stmt ss)
116         ^ " }\n"
117     | For (as1, e, as2, ss) ->
118         "For( " ^ string_of_assignstmt as1 ^ "; " ^ string_of_expr e ^ "; "
119         ^ string_of_assignstmt as2 ^ ") { "
120         ^ String.concat "\n" (List.map string_of_stmt ss)
121         ^ " }\n"
122     | If (e, ss) ->
123         "If( " ^ string_of_expr e ^ ") {"
124         ^ String.concat "\n" (List.map string_of_stmt ss)
125         ^ " }\n"
126     | IfElse (e, ss1, ss2) ->
127         "If( " ^ string_of_expr e ^ ") {"
128         ^ String.concat "\n" (List.map string_of_stmt ss1)
129         ^ " } Else{ "
130         ^ String.concat "\n" (List.map string_of_stmt ss2)
131         ^ " }\n"
132
133   let string_of_vdecl (t, id) = "vdecl: " ^ string_of_typ t ^ " " ^ id ^
    ";\n"
134
135   let string_of_fdecl fdecl =
136     "fdecl: " ^ fdecl.fname ^ "("
137     ^ String.concat ", " (List.map snd fdecl.formals)
138     ^ ")\n{\n"
139     ^ String.concat "" (List.map string_of_stmt fdecl.stmts)
140     ^ "}\n"
141
142   let string_of_program (vars, funcs) =
143     String.concat "" (List.map string_of_vdecl vars)
144     ^ "\n"
```

```
145     ^ String.concat "\n" (List.map string_of_fdecl funcs)
146
```

8.2.4 sast.ml

OCaml | 复制代码

```ocaml
open Ast

type sexpr = dtype * sx

and sx =
  | SBinop of sexpr * operator * sexpr
  | SId of string
  | SILit of int
  | SFLit of float
  | SBLit of bool
  | SMLit of sexpr list list
  | SFunc of string * sexpr list
  | SAccess of sexpr * sexpr * sexpr
  | SUnary of uop * sexpr

type sassignstmt =
  | SVDeAssign of dtype * string * sexpr
  | SAssign of string * sexpr
  | SMAssign of sexpr * sexpr * sexpr * sexpr

type sstmt =
  | SExpr of sexpr
  | SReturn of sexpr
  | SVDeclare of dtype * string
  | SAssignStmt of sassignstmt
  | SIf of sexpr * sstmt list
  | SIfElse of sexpr * sstmt list * sstmt list
  | SFor of sassignstmt * sexpr * sassignstmt * sstmt list
  | SWhile of sexpr * sstmt list

type sbind = dtype * string

type sfdecl = {
  sreturn : dtype;
  sfname : string;
  sformals : sbind list;
  sstmts : sstmt list;
}

type sprogram = sbind list * sfdecl list

(* Pretty-printing functions from microc *)

let rec string_of_sexpr (t, e) =
  "(" ^ string_of_typ t ^ " : "
  ^ (match e with
    | SILit l -> string_of_int l
    | SFLit l -> string_of_float l
```

```
49      | SBLit l -> string_of_bool l
50      | SMLit l ->
51          let string_of_row l = String.concat "" (List.map string_of_sexpr
    l) in
52          String.concat "" (List.map string_of_row l)
53      | SId s -> s
54      | SBinop (e1, o, e2) ->
55          string_of_sexpr e1 ^ " " ^ string_of_op o ^ " " ^ string_of_sexpr
    e2
56      | SUnary (o, e) -> string_of_uop o ^ string_of_sexpr e
57      | SFunc (id, inputs) ->
58          id ^ "(" ^ String.concat ", " (List.map string_of_sexpr inputs) ^
    ";\n"
59      | SAccess (e1, e2, e3) ->
60          string_of_sexpr e1 ^ " " ^ string_of_sexpr e2 ^ " " ^
    string_of_sexpr e3)
61    ^ ")"
62
63  let string_of_sassignstmt = function
64    | SAssign (v, e) -> "Assign: " ^ v ^ " = " ^ string_of_sexpr e ^ ";\n"
65    | SVDeAssign (t, id, sexpr) ->
66        "VDeAssign: " ^ string_of_typ t ^ id ^ string_of_sexpr sexpr ^ ";\n"
67    | SMAssign (id, r, c, v) ->
68        "MAssign: " ^ string_of_sexpr id ^ "[" ^ string_of_sexpr r ^ ", "
69        ^ string_of_sexpr c ^ "] = " ^ string_of_sexpr v ^ ";\n"
70
71  let rec string_of_sstmt = function
72    | SExpr sexpr -> string_of_sexpr sexpr ^ ";\n"
73    | SReturn sexpr -> "return: " ^ string_of_sexpr sexpr ^ ";\n"
74    | SAssignStmt sassignstmt -> string_of_sassignstmt sassignstmt
75    | SVDeclare (t, id) -> "VDeclare: " ^ string_of_typ t ^ " " ^ id ^ ";\n"
76    | SWhile (e, ss) ->
77        "While(" ^ string_of_sexpr e ^ ") { "
78        ^ String.concat "\n" (List.map string_of_sstmt ss)
79        ^ " }\n"
80    | SFor (as1, e, as2, ss) ->
81        "For( " ^ string_of_sassignstmt as1 ^ "; " ^ string_of_sexpr e ^ ";
    "
82        ^ string_of_sassignstmt as2 ^ ") { "
83        ^ String.concat "\n" (List.map string_of_sstmt ss)
84        ^ " }\n"
85    | SIf (e, ss) ->
86        "If( " ^ string_of_sexpr e ^ ") {"
87        ^ String.concat "\n" (List.map string_of_sstmt ss)
88        ^ " }\n"
89    | SIfElse (e, ss1, ss2) ->
90        "If( " ^ string_of_sexpr e ^ ") {"
91        ^ String.concat "\n" (List.map string_of_sstmt ss1)
92        ^ " } Else{ "
93        ^ String.concat "\n" (List.map string_of_sstmt ss2)
```

```
94         ^ " }\n"
95
96   let string_of_svdecl (t, id) = "vdecl: " ^ string_of_typ t ^ " " ^ id ^
     ";\n"
97
98   let string_of_sfdecl fdecl =
99     "fdecl: " ^ fdecl.sfname ^ "("
100    ^ String.concat ", " (List.map snd fdecl.sformals)
101    ^ ")\n{\n"
102    ^ String.concat "" (List.map string_of_sstmt fdecl.sstmts)
103    ^ "}\n"
104
105  let string_of_sprogram (svars, sfuncs) =
106    String.concat "" (List.map string_of_svdecl svars)
107    ^ "\n"
108    ^ String.concat "\n" (List.map string_of_sfdecl sfuncs)
109
```

8.2.5 semant.ml

OCaml | 复制代码

```ocaml
open Ast
open Sast
module StringMap = Map.Make (String)

let check (globals, functions) =
  let check_binds kind binds =
    List.iter
      (function
        | Null, b -> raise (Failure ("illegal null " ^ 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

  (**** Check global variables ****)
  check_binds "global" globals;

  (**** Check functions ****)
  let built_in_decls =
    let add_bind map (name, ftypes, ret) =
      StringMap.add name
        {
          return = ret;
          fname = name;
          (* Handles an list of formals types*)
          formals = List.map (fun t -> (t, "PLACEHOLDER")) ftypes;
          stmts = [];
        }
        map
    in
    List.fold_left add_bind StringMap.empty
      [
        ("print", [ Int ], Null);
        ("printb", [ Bool ], Null);
        ("printf", [ Float ], Null);
        ("printmf", [ Matrix ], Null);
        ("rows", [ Matrix ], Int);
        ("cols", [ Matrix ], Int);
        ("zeros", [ Int; Int ], Matrix);
      ]
  in

```

```
49      let add_func map fd =
50        let built_in_err = "function " ^ fd.fname ^ " may not be defined"
51        and dup_err = "duplicate function " ^ fd.fname
52        and make_err er = raise (Failure er)
53        and n = fd.fname (* Name of the function *) in
54        match fd with
55        (* No duplicate functions or redefinitions of built-ins *)
56        | _ when StringMap.mem n built_in_decls -> make_err built_in_err
57        | _ when StringMap.mem n map -> make_err dup_err
58        | _ -> StringMap.add n fd map
59      in
60
61      (* Collect all function names into one symbol table *)
62      let function_decls = List.fold_left add_func built_in_decls functions in
63
64      (* Return a function from our symbol table *)
65      let find_func s =
66        try StringMap.find s function_decls
67        with Not_found -> raise (Failure ("unrecognized function " ^ s))
68      in
69
70      let _ = find_func "main" in
71
72      let check_function func =
73        check_binds "formal" func.formals;
74
75        let check_assign lvaluet rvaluet err =
76          if lvaluet = rvaluet then lvaluet else raise (Failure err)
77        in
78
79        let symbols =
80          List.fold_left
81            (fun m (ty, name) -> StringMap.add name ty m)
82            StringMap.empty (globals @ func.formals)
83        in
84
85        let type_of_identifier s env =
86          try StringMap.find s env
87          with Not_found -> raise (Failure ("undeclared identifier " ^ s))
88        in
89
90        let rec check_expr e env =
91          match e with
92          | Id n -> (type_of_identifier n env, SId n)
93          | ILit l -> (Int, SILit l)
94          | FLit l -> (Float, SFLit l)
95          | BLit l -> (Bool, SBLit l)
96          | MLit l ->
97              let find_inner_type l =
98                match l with
```

```
 99              | hd :: _ ->
100                  let t, _ = check_expr hd env in
101                  t
102              | _ -> Null
103            in
104            let find_type mat =
105              match mat with (* tl *)
106              | hd :: _ -> find_inner_type hd | _ -> Null
107            in
108            let my_type = find_type l in
109            let rec matrix_expr l =
110              match l with
111              | hd :: tl ->
112                  let ty, e = check_expr hd env in
113                  if ty != my_type then
114                    raise (Failure "Types in matrix do not match.");
115                  (ty, e) :: matrix_expr tl
116              | _ -> []
117            in
118            (Matrix, SMLit (List.map matrix_expr l))
119        | Binop (e1, op, e2) ->
120            let t1, e1' = check_expr e1 env and t2, e2' = check_expr e2 env
    in
121            (* All binary operators require operands of the same type *)
122            let same = t1 = t2 in
123            (* Determine expression type based on operator and operand types
    *)
124            let ty =
125              match op with
126              | (Add | Sub | Mul | Div | Mod) when same && t1 = Int -> Int
127              | (Add | Sub | Mul | Div | Mod) when same && t1 = Float ->
    Float
128              | (Add | Sub) when same && t1 = Matrix -> Matrix
129              | (Eq | Neq) when same -> Bool
130              | (Less | Leq | Greater | Geq) when same && (t1 = Int || t1 =
    Float)
131                ->
132                  Bool
133              | (And | Or) when same && t1 = Bool -> Bool
134              | Mul when same && t1 = Matrix -> Matrix
135              | Mul when t1 = Matrix && (t2 = Float || t2 = Int) -> Matrix
136              | Mul when t2 = Matrix && (t1 = Float || t1 = Int) -> Matrix
137              | _ ->
138                  raise
139                    (Failure
140                       ("illegal binary operator " ^ string_of_typ t1 ^ " "
141                      ^ string_of_op op ^ " " ^ string_of_typ t2 ^ " in "
142                      ^ string_of_expr e))
143            in
144            (ty, SBinop ((t1, e1'), op, (t2, e2')))
```

```
145         | Unary (op, e1) as e ->
146             let tp, c = check_expr e1 env in
147             let ty =
148               match op with
149               | Neg when tp = Int || tp = Float -> tp
150               | Not when tp = Bool -> Bool
151               | _ ->
152                   raise
153                     (Failure
154                        ("illegal unary operator " ^ string_of_uop op
155                       ^ string_of_typ tp ^ " in " ^ string_of_expr e))
156             in
157             (ty, SUnary (op, (tp, c)))
158         | Func (id, inputs) as func ->
159             let fd = find_func id in
160             let param_length = List.length fd.formals in
161
162             if List.length inputs != param_length then
163               raise
164                 (Failure
165                    ("expecting " ^ string_of_int param_length ^ " arguments
    in "
166                     ^ string_of_expr func))
167             else
168               let check_call (ft, _) e =
169                 let et, e' = check_expr e env in
170                 let err =
171                   "illegal argument found " ^ string_of_typ et ^ " expected
    "
172                   ^ string_of_typ ft ^ " in " ^ string_of_expr e
173                 in
174                 (check_assign ft et err, e')
175               in
176
177               let args' = List.map2 check_call fd.formals inputs in
178               (fd.return, SFunc (id, args'))
179         | Access (m, r, c) ->
180             let r_t, _ = check_expr r env in
181             let c_t, _ = check_expr c env in
182             if r_t != Int || c_t != Int then
183               raise (Failure "index must be of type int");
184             let _, _ = check_expr m env in
185             (Float, SAccess (check_expr m env, check_expr r env, check_expr
    c env))
186       in
187
188       (* Check stmts - Return a semantically-checked statement i.e.
    containing sexprs *)
189       let rec check_stmt env stmt =
190         match stmt with
```

```
191         | Expr e -> (SExpr (check_expr e env), env)
192         | Return e ->
193             let t, e' = check_expr e env in
194             if t = func.return then (SReturn (t, e'), env)
195             else
196               raise
197                 (Failure
198                    ("return gives " ^ string_of_typ t ^ " expected "
199                   ^ string_of_typ func.return ^ " in " ^ string_of_expr e))
200         | VDeclare (t, s) -> (SVDeclare (t, s), StringMap.add s t env)
201         | AssignStmt astmt -> (
202             match astmt with
203             | VDeAssign (t, s, e) ->
204                 let t', _ = check_expr e env in
205                 let decl_type = check_assign t t' "Type not correct" in
206                 ( SAssignStmt (SVDeAssign (t, s, check_expr e env)),
207                   StringMap.add s decl_type env )
208             | Assign (s, e) ->
209                 let left_typ = type_of_identifier s env
210                 and right_typ, e' = check_expr e env in
211                 let error = "Illegal assignment!" in
212                 ignore (check_assign left_typ right_typ error);
213                 (SAssignStmt (SAssign (s, (right_typ, e'))), env)
214             | MAssign (m, r, c, e) ->
215                 let r_t, _ = check_expr r env in
216                 let c_t, _ = check_expr c env in
217                 let e_t, _ = check_expr e env in
218                 if r_t != Int || c_t != Int then
219                   raise (Failure "index must be of type int");
220                 if e_t != Float then raise (Failure "value must be of type
    float");
221                 ( SAssignStmt
222                     (SMAssign
223                        ( check_expr m env,
224                          check_expr r env,
225                          check_expr c env,
226                          check_expr e env )),
227                   env ))
228         | While (e, stmts) ->
229             let typ, _ = check_expr e env in
230             if typ != Bool then raise (Failure "Expect to have a Bool type
    here.");
231             (SWhile (check_expr e env, check_stmts env stmts), env)
232         | For (astmt, e2, astmt2, stmts) ->
233             let sastmt, env2 = check_assignstmt env astmt in
234             let sastmt2, _ = check_assignstmt env2 astmt2 in
235             let typ, _ = check_expr e2 env2 in
236             if typ != Bool then raise (Failure "Expect to have a Bool type
    here.");
```

```
237             ( SFor (sastmt, check_expr e2 env2, sastmt2, check_stmts env2
    stmts),
238               env )
239         | If (e, stmts) ->
240             let typ, _ = check_expr e env in
241             if typ != Bool then raise (Failure "Expect to have a Bool type
    here.");
242             (SIf (check_expr e env, check_stmts env stmts), env)
243         | IfElse (e, stmts1, stmts2) ->
244             let typ, _ = check_expr e env in
245             if typ != Bool then raise (Failure "Expect to have a Bool type
    here.");
246             ( SIfElse
247                 (check_expr e env, check_stmts env stmts1, check_stmts env
    stmts2),
248               env )
249       and check_assignstmt env astmt =
250         match astmt with
251         | VDeAssign (t, s, e) ->
252             let t', _ = check_expr e env in
253             let decl_type = check_assign t t' "Type not correct" in
254             (SVDeAssign (t, s, check_expr e env), StringMap.add s decl_type
    env)
255         | Assign (s, e) ->
256             let left_typ = type_of_identifier s env
257             and right_typ, e' = check_expr e env in
258             let error = "Illegal assignment!" in
259             ignore (check_assign left_typ right_typ error);
260             (SAssign (s, (right_typ, e')), env)
261         | _ -> raise (Failure "Illegal assignment in for-loop header!")
262       and check_stmts env stmts =
263         match stmts with
264         | [ (Return _ as s) ] ->
265             let st, _ = check_stmt env s in
266             [ st ]
267         | Return _ :: _ -> raise (Failure "Unreachable statments after
    return")
268         | s :: ss ->
269             let st, env2 = check_stmt env s in
270             st :: check_stmts env2 ss
271         | [] -> []
272       in
273
274       {
275         sreturn = func.return;
276         sfname = func.fname;
277         sformals = func.formals;
278         sstmts = check_stmts symbols func.stmts;
279       }
280     in
```

```
281     (globals, List.map check_function functions)
```

8.2.6 codegen.ml

OCaml | 复制代码

```ocaml
(* Code generation: translate takes a semantically checked AST and
   produces LLVM IR

   LLVM tutorial: Make sure to read the OCaml version of the tutorial

   http://llvm.org/docs/tutorial/index.html

   Detailed documentation on the OCaml LLVM library:

   http://llvm.moe/
   http://llvm.moe/ocaml/
*)

module L = Llvm
module A = Ast
open Sast
module StringMap = Map.Make (String)

(* translate : Sast.program -> Llvm.module *)
let translate (globals, functions) =
  let context = L.global_context () in

  (* Create the LLVM compilation module into which
     we will generate code *)
  let the_module = L.create_module context "Marble" in

  (* Get types from the context *)
  let i32_t = L.i32_type context
  and i8_t = L.i8_type context
  and i1_t = L.i1_type context
  and float_t = L.double_type context
  and void_t = L.void_type context in

  (* Return the LLVM type for a Marble type *)
  let ltype_of_typ = function
    | A.Int -> i32_t
    | A.Float -> float_t
    | A.Bool -> i1_t
    | A.Matrix -> L.pointer_type float_t
    | A.Null -> void_t
  in

  (* Create a map of global variables after creating each *)
  let global_vars : L.llvalue StringMap.t =
    let global_var m (t, n) =
      let init =
        match t with
        | A.Int -> L.const_int (ltype_of_typ t) 0
```

```
          | _ -> L.const_int (ltype_of_typ t) 0
        in
        StringMap.add n (L.define_global n init the_module) m
      in
      List.fold_left global_var StringMap.empty globals
    in

    (* built-in function *)
    let printf_t : L.lltype =
      L.var_arg_function_type i32_t [| L.pointer_type i8_t |]
    in
    let printf_func : L.llvalue =
      L.declare_function "printf" printf_t the_module
    in

    let printmf_t : L.lltype =
      L.function_type i32_t [| L.pointer_type float_t |]
    in
    let printmf_func : L.llvalue =
      L.declare_function "printmf" printmf_t the_module
    in

    (* matrix addition*)
    let addmf_t : L.lltype =
      L.function_type (L.pointer_type float_t)
        [| L.pointer_type float_t; L.pointer_type float_t |]
    in
    let addmf_func : L.llvalue = L.declare_function "addmf" addmf_t
the_module in

    (* subtraction *)
    let submf_t : L.lltype =
      L.function_type (L.pointer_type float_t)
        [| L.pointer_type float_t; L.pointer_type float_t |]
    in
    let submf_func : L.llvalue = L.declare_function "submf" submf_t
the_module in

    (* scalar multiplication *)
    let scalarmf_t : L.lltype =
      L.function_type (L.pointer_type float_t)
        [| float_t; L.pointer_type float_t |]
    in
    let scalarmf_func : L.llvalue =
      L.declare_function "scalarmf" scalarmf_t the_module
    in

    (* matrix multiplication *)
    let multiplicationmf_t : L.lltype =
      L.function_type (L.pointer_type float_t)
```

```
97          [| L.pointer_type float_t; L.pointer_type float_t |]
98      in
99      let multiplicationmf_func : L.llvalue =
100       L.declare_function "multiplicationmf" multiplicationmf_t the_module
101     in
102
103     (* functions to easily get number of rows/columns of a matrix *)
104     let get_matrix_rows matrix builder =
105       (* matrix has already gone through expr *)
106       let typ = L.string_of_lltype (L.type_of matrix) in
107       let ret =
108         match typ with
109         | "double*" ->
110             let rows = L.build_load matrix "rows" builder in
111             L.build_fptosi rows i32_t "rowsint" builder
112         | _ -> L.build_load matrix "rows" builder
113       in
114       ret
115     in
116     let get_matrix_cols matrix builder =
117       let ptr =
118         L.build_in_bounds_gep matrix [| L.const_int i32_t 1 |] "ptr" builder
119       in
120       let typ = L.string_of_lltype (L.type_of ptr) in
121       let ret =
122         match typ with
123         | "double*" ->
124             let cols = L.build_load ptr "cols" builder in
125             L.build_fptosi cols i32_t "colsint" builder
126         | _ -> L.build_load ptr "cols" builder
127       in
128       ret
129     in
130     (* Define each function (arguments and return type) so we can
131        call it even before we've created its body *)
132     let function_decls : (L.llvalue * sfdecl) StringMap.t =
133       let function_decl m fdecl =
134         let name = fdecl.sfname in
135         let formal_types =
136           Array.of_list (List.map (fun (t, _) -> ltype_of_typ t)
    fdecl.sformals)
137         in
138         let ftype = L.function_type (ltype_of_typ fdecl.sreturn)
    formal_types in
139         StringMap.add name (L.define_function name ftype the_module, fdecl)
    m
140       in
141       List.fold_left function_decl StringMap.empty functions
142     in
143
```

```
144     (* Fill in the body of the given function *)
145     let build_function_body fdecl =
146       let the_function, _ = StringMap.find fdecl.sfname function_decls in
147       let builder = L.builder_at_end context (L.entry_block the_function) in
148
149       let int_format_str = L.build_global_stringptr "%d\n" "fmt" builder
150       and float_format_str = L.build_global_stringptr "%g\n" "fmt" builder
    in
151
152       let local_vars = Hashtbl.create 20 in
153       let add_formal (t, n) p =
154         L.set_value_name n p;
155         let local = L.build_alloca (ltype_of_typ t) n builder in
156         ignore (L.build_store p local builder);
157         Hashtbl.add local_vars n local
158       in
159
160       (* add formals *)
161       List.iter2 add_formal fdecl.sformals (Array.to_list (L.params
    the_function));
162
163       (* Return the value for a variable or formal argument.
164          Check local names first, then global names *)
165       let lookup n =
166         try Hashtbl.find local_vars n
167         with Not_found -> (
168           try StringMap.find n global_vars
169           with Not_found ->
170             raise (Failure ("Runtime: undeclared identifier " ^ n)))
171       in
172
173       (* Construct code for an expression; return its value *)
174       let rec expr builder ((_, e) : sexpr) =
175         match e with
176         | SILit i -> L.const_int i32_t i
177         | SFLit f -> L.const_float float_t f
178         | SBLit b -> L.const_int i1_t (if b then 1 else 0)
179         | SMLit l ->
180             let find_inner_type l =
181               match l with
182               | hd :: _ ->
183                   let t, _ = hd in
184                   t
185               | _ -> A.Int
186             in
187             let find_type mat =
188               match mat with hd :: _ -> find_inner_type hd | _ -> A.Int
189             in
190             let my_type = find_type l in
191             let make_matrix =
```

```
192               match my_type with
193               | A.Float ->
194                   let count a = List.fold_left (fun x _ -> x + 1) 0 a in
195                   let rows = float_of_int (count l) in
196                   let cols = float_of_int (count (List.hd l)) in
197                   let rec valid_dims m =
198                     match m with
199                     | hd :: tl ->
200                         if count hd == count (List.hd l) then valid_dims tl
201                         else false
202                     | _ -> true
203                   in
204                   if not (valid_dims l) then
205                     raise
206                       (Failure
207                          "all rows of matrices must have the same number of
    \
208                           elemens")
209                   else
210                     (* allocate space 2 + rows * cols*)
211                     let matrix =
212                       L.build_malloc
213                         (L.array_type float_t
214                            (2 + (int_of_float rows * int_of_float cols)))
215                         "matrix" builder
216                     in
217
218                     let eval_row row =
219                       List.fold_left
220                         (fun eval_row x -> eval_row @ [ expr builder x ])
221                         [] row
222                     in
223                     let unfolded =
224                       List.fold_left (fun unfld row -> unfld @ eval_row row)
    [] l
225                     in
226                     let unfolded =
227                       [ L.const_float float_t rows; L.const_float float_t
    cols ]
228                       @ unfolded
229                     in
230                     let rec store idx lst =
231                       match lst with
232                       | hd :: tl ->
233                           let ptr =
234                             L.build_in_bounds_gep matrix
235                               [| L.const_int i32_t 0; L.const_int i32_t idx
    |]
236                               "ptr" builder
237                           in
```

```
238                         ignore (L.build_store hd ptr builder);
239                         store (idx + 1) tl
240                     | _ -> ()
241                   in
242                   store 0 unfolded;
243                   L.build_in_bounds_gep matrix
244                     [| L.const_int i32_t 0; L.const_int i32_t 0 |]
245                     "matrix" builder
246             | _ -> raise (Failure "invalid matrix type")
247           in
248           make_matrix
249       | SId s -> L.build_load (lookup s) s builder
250       | SBinop (((A.Matrix, _) as m1), op, m2) ->
251           let m1' = expr builder m1 and m2' = expr builder m2 in
252           let ret =
253             match op with
254             | A.Add -> L.build_call addmf_func [| m1'; m2' |] "addmf"
    builder
255             | A.Sub -> L.build_call submf_func [| m1'; m2' |] "submf"
    builder
256             | A.Mul ->
257                 let t', _ = m2 in
258                 let ret_val' =
259                   match t' with
260                   | A.Int ->
261                       let scalar =
262                         L.build_sitofp m2' float_t "scalar" builder
263                       in
264                       L.build_call scalarmf_func [| scalar; m1' |]
    "scalarmf"
265                         builder
266                   | A.Float ->
267                       L.build_call scalarmf_func [| m2'; m1' |] "scalarmf"
268                         builder
269                   | _ ->
270                       L.build_call multiplicationmf_func [| m1'; m2' |]
    "matmf"
271                         builder
272                 in
273                 ret_val'
274             | _ -> raise (Failure "internal error: semant should have
    rejected")
275           in
276           ret
277       | SBinop ((_ as m1), (_ as op), ((A.Matrix, _) as m2)) ->
278           let m1' = expr builder m1 and m2' = expr builder m2 in
279           let ret =
280             match op with
281             | A.Mul ->
282                 let t, _ = m1 in
```

```
283                 let ret_val =
284                   match t with
285                   | A.Int ->
286                       let scalar =
287                         L.build_sitofp m1' float_t "scalar" builder
288                       in
289                       L.build_call scalarmf_func [| scalar; m2' |]
    "scalarmf"
290                         builder
291                   | A.Float ->
292                       L.build_call scalarmf_func [| m1'; m2' |] "scalarm"
293                         builder
294                   | _ -> raise (Failure "should be caught elsewhere")
295                 in
296                 ret_val
297             | _ -> raise (Failure "internal error: semant should have
    rejected")
298           in
299           ret
300       | SBinop (((A.Float, _) as e1), op, e2) ->
301           let e1' = expr builder e1 and e2' = expr builder e2 in
302           (match op with
303           | A.Add -> L.build_fadd
304           | A.Sub -> L.build_fsub
305           | A.Mul -> L.build_fmul
306           | A.Div -> L.build_fdiv
307           | A.Mod -> L.build_frem
308           | A.Eq -> L.build_fcmp L.Fcmp.Oeq
309           | A.Neq -> L.build_fcmp L.Fcmp.One
310           | A.Less -> L.build_fcmp L.Fcmp.Olt
311           | A.Leq -> L.build_fcmp L.Fcmp.Ole
312           | A.Greater -> L.build_fcmp L.Fcmp.Ogt
313           | A.Geq -> L.build_fcmp L.Fcmp.Oge
314           | A.And | A.Or ->
315               raise
316                 (Failure
317                    "internal error: semant should have rejected and/or on
    float"))
318             e1' e2' "tmp" builder
319       | SBinop (e1, op, e2) ->
320           let e1' = expr builder e1 and e2' = expr builder e2 in
321           (match op with
322           | A.Add -> L.build_add
323           | A.Sub -> L.build_sub
324           | A.Mul -> L.build_mul
325           | A.Div -> L.build_sdiv
326           | A.Mod -> L.build_srem
327           | A.Eq -> L.build_icmp L.Icmp.Eq
328           | A.Neq -> L.build_icmp L.Icmp.Ne
329           | A.Less -> L.build_icmp L.Icmp.Slt
```

```
330             | A.Leq -> L.build_icmp L.Icmp.Sle
331             | A.Greater -> L.build_icmp L.Icmp.Sgt
332             | A.Geq -> L.build_icmp L.Icmp.Sge
333             | A.And -> L.build_and
334             | A.Or -> L.build_or)
335               e1' e2' "tmp" builder
336         | SUnary (op, ((t, _) as e)) ->
337             (* Unary and Negate *)
338             let e' = expr builder e in
339             (match op with
340             | A.Neg when t = A.Float -> L.build_fneg
341             | A.Neg -> L.build_neg
342             | A.Not -> L.build_not)
343               e' "tmp" builder
344         | SAccess (((_, _) as m), r, c) ->
345             (* get desired pointer location *)
346             let matrix = expr builder m
347             and row_idx = expr builder r
348             and col_idx = expr builder c in
349             let cols = get_matrix_cols matrix builder in
350             (* row = row_idx * cols *)
351             let row = L.build_mul row_idx cols "row" builder in
352             (* row_col = (row_idx * cols) + col_idx *)
353             let row_col = L.build_add row col_idx "row_col" builder
354             and offset = L.const_int i32_t 2 in
355             (* idx = 2 + (row_idx * cols) + col_idx *)
356             let idx = L.build_add offset row_col "idx" builder in
357             let ptr = L.build_in_bounds_gep matrix [| idx |] "ptr" builder
    in
358             L.build_load ptr "element" builder
359         (* Function call *)
360         | SFunc ("print", [ e ]) | SFunc ("printb", [ e ]) ->
361             L.build_call printf_func
362               [| int_format_str; expr builder e |]
363               "printf" builder
364         | SFunc ("printf", [ e ]) ->
365             L.build_call printf_func
366               [| float_format_str; expr builder e |]
367               "printf" builder
368         | SFunc ("printmf", [ e ]) ->
369             L.build_call printmf_func [| expr builder e |] "printmf" builder
370         | SFunc ("rows", [ e ]) ->
371             let matrix = expr builder e in
372             get_matrix_rows matrix builder
373         | SFunc ("cols", [ e ]) ->
374             let matrix = expr builder e in
375             get_matrix_cols matrix builder
376         | SFunc ("zeros", [ (_, rows); (_, cols) ]) -> (
377             match (rows, cols) with
378             | SILit r, SILit c ->
```

```
379                 if r == 0 || c == 0 then
380                   raise (Failure "rows and cols can't be zero")
381                 else
382                   let matrix =
383                     L.build_malloc
384                       (L.array_type float_t (2 + (r * c)))
385                       "matrix" builder
386                   in
387
388                   let eval_row row =
389                     List.fold_left
390                       (fun eval_row _ -> eval_row @ [ L.const_float float_t
    0.0 ])
391                       [] row
392                   in
393                   let unfolded =
394                     List.fold_left (fun unfld row -> unfld @ eval_row row)
    [] []
395                   in
396                   let unfolded =
397                     [
398                       L.const_float float_t (float_of_int r);
399                       L.const_float float_t (float_of_int c);
400                     ]
401                     @ unfolded
402                   in
403
404                   let rec store idx lst =
405                     match lst with
406                     | hd :: tl ->
407                         let ptr =
408                           L.build_in_bounds_gep matrix
409                             [| L.const_int i32_t 0; L.const_int i32_t idx |]
410                             "ptr" builder
411                         in
412                         ignore (L.build_store hd ptr builder);
413                         store (idx + 1) tl
414                     | _ -> ()
415                   in
416                   store 0 unfolded;
417                   L.build_in_bounds_gep matrix
418                     [| L.const_int i32_t 0; L.const_int i32_t 0 |]
419                     "matrix" builder
420             | _ -> raise (Failure "Rows and cols of a matrix must be
    intergers."))
421         | SFunc (f, args) ->
422             let fdef, fdecl = StringMap.find f function_decls in
423             let llargs = List.rev (List.map (expr builder) (List.rev args))
    in
424             let result =
```

```
425               match fdecl.sreturn with A.Null -> "null" | _ -> f ^ "_result"
426             in
427             L.build_call fdef (Array.of_list llargs) result builder
428       in
429
430       (* LLVM insists each basic block end with exactly one "terminator"
431          instruction that transfers control.  This function runs "instr
      builder"
432          if the current block does not already have a terminator.  Used,
433          e.g., to handle the "fall off the end of the function" case. *)
434       let add_terminal builder instr =
435         match L.block_terminator (L.insertion_block builder) with
436         | Some _ -> ()
437         | None -> ignore (instr builder)
438       in
439       let rec stmt builder = function
440         | SExpr e ->
441             ignore (expr builder e);
442             builder
443         | SReturn e ->
444             ignore
445               (match fdecl.sreturn with
446               (* Special "return nothing" instr *)
447               | A.Null -> L.build_ret_void builder
448               | A.Matrix ->
449                   let e' = expr builder e in
450                   L.build_ret
451                     (L.build_bitcast e' (ltype_of_typ A.Matrix) "tmp"
      builder)
452                     builder
453               | _ -> L.build_ret (expr builder e) builder);
454             builder
455         | SVDeclare (t, s) ->
456             let local_var = L.build_alloca (ltype_of_typ t) s builder in
457             Hashtbl.add local_vars s local_var;
458             builder
459         | SAssignStmt sastmt -> (
460             match sastmt with
461             | SVDeAssign (t, s, se) ->
462                 let e' = expr builder se in
463                 L.set_value_name s e';
464                 let local_var = L.build_alloca (ltype_of_typ t) s builder in
465                 ignore (L.build_store e' local_var builder);
466                 Hashtbl.add local_vars s local_var;
467                 builder
468             | SAssign (s, se) ->
469                 let e' = expr builder se in
470                 ignore (L.build_store e' (lookup s) builder);
471                 builder
472             | SMAssign (m, r, c, e) ->
```

```
473                 (* get desired pointer location *)
474                 let matrix = expr builder m
475                 and row_idx = expr builder r
476                 and col_idx = expr builder c in
477                 let cols = get_matrix_cols matrix builder in
478                 (* row = row_idx * cols *)
479                 let row = L.build_mul row_idx cols "row" builder in
480                 (* row_col = (row_idx * cols) + col_idx *)
481                 let row_col = L.build_add row col_idx "row_col" builder
482                 and offset = L.const_int i32_t 2 in
483                 (* idx = 2 + (row_idx * cols) + col_idx *)
484                 let idx = L.build_add offset row_col "idx" builder in
485                 let ptr = L.build_in_bounds_gep matrix [| idx |] "ptr"
    builder in
486                 (* update value at that location *)
487                 let e' = expr builder e in
488                 let m_typ = L.string_of_lltype (L.type_of matrix)
489                 and e_typ = L.string_of_lltype (L.type_of e') in
490                 let e_fixed =
491                   match (m_typ, e_typ) with
492                   | "double*", "i32" ->
493                       L.build_uitofp e' float_t "float_e" builder
494                   | _ -> e'
495                 in
496                 ignore (L.build_store e_fixed ptr builder);
497                 builder)
498         | SWhile (se, sstmts) ->
499             let se_bb = L.append_block context "while" the_function in
500             ignore (L.build_br se_bb builder);
501             let body_bb = L.append_block context "while_body" the_function
    in
502             add_terminal
503               (List.fold_left stmt (L.builder_at_end context body_bb)
    sstmts)
504               (L.build_br se_bb);
505             let pred_builder = L.builder_at_end context se_bb in
506             let bool_val = expr pred_builder se in
507             let merge_bb = L.append_block context "merge" the_function in
508             ignore (L.build_cond_br bool_val body_bb merge_bb pred_builder);
509             L.builder_at_end context merge_bb
510         | SFor (sastmt, se2, sastmt2, sstmts) ->
511             List.fold_left stmt builder
512               [
513                 SAssignStmt sastmt; SWhile (se2, sstmts @ [ SAssignStmt
    sastmt2 ]);
514               ]
515         | SIf (se, sstmts) ->
516             let bool_val = expr builder se in
517             let merge_bb = L.append_block context "merge" the_function in
518             let b_br_merge = L.build_br merge_bb in
```

```
519             let then_bb = L.append_block context "then" the_function in
520             add_terminal
521               (List.fold_left stmt (L.builder_at_end context then_bb)
    sstmts)
522               b_br_merge;
523             let else_bb = L.append_block context "else" the_function in
524             add_terminal
525               (List.fold_left stmt (L.builder_at_end context else_bb) [])
526               b_br_merge;
527             ignore (L.build_cond_br bool_val then_bb else_bb builder);
528             L.builder_at_end context merge_bb
529         | SIfElse (se, sstmts1, sstmts2) ->
530             let bool_val = expr builder se in
531             let merge_bb = L.append_block context "merge" the_function in
532             let b_br_merge = L.build_br merge_bb in
533             let then_bb = L.append_block context "then" the_function in
534             add_terminal
535               (List.fold_left stmt (L.builder_at_end context then_bb)
    sstmts1)
536               b_br_merge;
537             let else_bb = L.append_block context "else" the_function in
538             add_terminal
539               (List.fold_left stmt (L.builder_at_end context else_bb)
    sstmts2)
540               b_br_merge;
541             ignore (L.build_cond_br bool_val then_bb else_bb builder);
542             L.builder_at_end context merge_bb
543       in
544       (* Build the code for each statement in the function *)
545       let builder = List.fold_left stmt builder fdecl.sstmts in
546
547       (* Add a return if the last block falls off the end *)
548       add_terminal builder
549         (match fdecl.sreturn with
550         | A.Null -> L.build_ret_void
551         | A.Float -> L.build_ret (L.const_float float_t 0.0)
552         | t -> L.build_ret (L.const_int (ltype_of_typ t) 0))
553     in
554
555     List.iter build_function_body functions;
556
557     the_module
```

8.2.7 marble.ml

OCaml 复制代码

```
(* Top-level of the MicroC compiler: scan & parse the input,
   check the resulting AST and generate an SAST from it, generate LLVM IR,
   and dump the module *)

type action = Ast | Sast | LLVM_IR | Compile

let () =
  let action = ref Compile in
  let set_action a () = action := a in
  let speclist =
    [
      ("-a", Arg.Unit (set_action Ast), "Print the AST");
      ("-s", Arg.Unit (set_action Sast), "Print the SAST");
      ("-l", Arg.Unit (set_action LLVM_IR), "Print the generated LLVM
IR");
      ( "-c",
        Arg.Unit (set_action Compile),
        "Check and print the generated LLVM IR (default)" );
    ]
  in
  let usage_msg = "usage: ./microc.native [-a|-s|-l|-c] [file.mc]" in
  let channel = ref stdin in
  Arg.parse speclist (fun filename -> channel := open_in filename)
usage_msg;

  let lexbuf = Lexing.from_channel !channel in
  let ast = Parser.program Scanner.tokenize lexbuf in
  match !action with
  | Ast -> print_string (Ast.string_of_program ast)
  | _ -> (
      let sast = Semant.check ast in
      match !action with
      | Ast -> ()
      | Sast -> print_string (Sast.string_of_sprogram sast)
      | LLVM_IR ->
          print_string (Llvm.string_of_llmodule (Codegen.translate sast))
      | Compile ->
          let m = Codegen.translate sast in
          Llvm_analysis.assert_valid_module m;
          print_string (Llvm.string_of_llmodule m))

```

8.2.8 matrix_helper.c

C 复制代码

```c
/*
 *  matrix_helper.c
 *  reference: https://github.com/emilydringel/MATRIX_MANIA/blob/main/c_functions/matrix_functions.c
 */

#include <stdlib.h>
#include <stdio.h>
#include <string.h>

void printmf(double* element_one) /* prints the matrix<int> */
{
    int rows = (int) element_one[0];
    int cols = (int) element_one[1];
    int size = rows * cols;
    char matrix[size*100]; /* maybe should be better about size?? */
    strcpy(matrix, "");
    for(int i = 0; i < rows; i++){
        for(int j = 0; j < cols; j++){
            int location = 2 + (i*cols) + j;
            char buffer[1000];
            sprintf(buffer, "%f", element_one[location]);
            strcat(matrix, buffer);
            if(j!=cols-1){
                strcat(matrix, " ");
            }
        }
        if(i!=rows-1){
            strcat(matrix, "\n");
        }
    }
    printf("%s\n", matrix);
}

double* addmf(double* m1, double* m2)
{
    if(m1[0]!=m2[0] || m1[1]!=m2[1]){
        printf("RUNTIME ERROR: matrices being added do not have the same dimensions.\n");
        exit(1);
    }
    int rows = (int) m1[0];
    int cols = (int) m1[1];
    int size = 2 + rows * cols;
    double *empty = malloc(size * sizeof(double));
    empty[0] = rows;
    empty[1] = cols;
```

```
46      for (int i = 2; i < size; i++) {
47          empty[i] = m1[i] + m2[i];
48      }
49      return empty;
50  }
51
52  double* submf(double* m1, double* m2)
53  {
54      if(m1[0]!=m2[0] || m1[1]!=m2[1]){
55          printf("RUNTIME ERROR: matrices being subtracted do not have the
    same dimensions.\n");
56          exit(1);
57      }
58      int rows = (int) m1[0];
59      int cols = (int) m1[1];
60      int size = 2 + rows * cols;
61      double *empty = malloc(size * sizeof(double));
62      empty[0] = rows;
63      empty[1] = cols;
64      for (int i = 2; i < size; i++) {
65          empty[i] = m1[i] - m2[i];
66      }
67      return empty;
68  }
69
70  double* scalarmf(double x, double* m){
71      int rows = (int) m[0];
72      int cols = (int) m[1];
73      int size = 2 + rows * cols;
74      double *empty = malloc(size * sizeof(double));
75      for (int i = 0; i < size; i++) {
76          empty[i] = m[i];
77          if(i>=2){
78              empty[i] *= x;
79          }
80      }
81      return empty;
82  }
83
84  double* multiplicationmf(double* m1, double* m2){
85      if(m1[1]!=m2[0]){
86          printf("RUNTIME ERROR: matrices being multiplied do not have
    complementary dimensions.\n");
87          exit(1);
88      }
89      int rows_one = (int) m1[0];
90      int rows_two = (int) m2[0];
91      int cols_one = (int) m1[1];
92      int cols_two = (int) m2[1];
93      double *empty = malloc(rows_one * cols_two * sizeof(int));
```

```
94        empty[0] = (double) rows_one;
95        empty[1] = (double) cols_two;
96        for(int row=0; row<rows_one; row++){
97            for(int col=0; col<cols_two; col++){
98                empty[2+(cols_two*row)+col] = 0;
99                for(int val=0; val<cols_one; val++){
100                   double x = m1[2+cols_one*row+val]*m2[2+cols_two*val+col];
101                   empty[2+cols_two*row+col] += x;
102               }
103           }
104       }
105       return empty;
106   }
107
```