MatriCs

the ultimate matrix manipulation language

What if you had the power to create matrices of any dimension?

Time to go beyond dimensions...
MatriCs is a specialized language for matrix manipulation.

- **Strongly typed language**
- **C-like syntax**
- **Special operators for vectors and matrices**
- **Compiles into LLVM**
Welcome to the world of Matrices

Let’s learn some Matrices
Primitives: Integer, Boolean, Float, String, Void

Special Data Type: n dimensional Vectors

Comments:
- // - for single line comments
- /* */ - for block comments

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

Control Flow: if, else if, else, while, for, return

Conditionals: ==, !=, <, <=, >, >=

Logical Operators: !, &&, ||

Standard Library: Matrix Addition, Matrix Subtraction, Print Matrices, Transpose, Identity
MatriCs Properties

```c
int main() {
    // This is a comment
    /* This is */
    /* another */
    /* comment */
    int a[2,2,2,2];
    int i;
    int j;
    int k;
    int l;

    a = [[[[[1,2], [3,3]],
          [[7,2], [9,1]]],
          [[[1,2], [3,3]],
          [[7,2], [9,1]]]];

    for (i = 0; i < 2; i++) {
        for (j = 0; j < 2; j++) {
            for (k = 0; k < 2; k++) {
                for (l = 0; l < 2; l++) {
                    print_int(a[i,j,k,l]);
                }
            }
        }
    }
    return 0;
}
```
MatriCs Properties Continued

```c
int main() {
    int[4] b;
    b = [1, 2, 3, 4];
    if (b[1] == 3) {
        printf(true);
    } else {
        printf(false);
    }
}
```

```c
int main() {
    int[2,2,2,2] a;
    int i;
    int j;
    int k;
    int l;
    a = [[[1,2], [3,3]],
         [[7,2], [9,1]]],
         [[[1,2], [3,3]],
         [[7,2], [9,1]]];
    for (i = 0; i < 2; i = i+1) {
        for (j = 0; j < 2; j = j+1) {
            for (k = 0; k < 2; k = k+1) {
                for (l = 0; l < 2; l = l+1) {
                    printf_int(a[i,j,k,l]);
                }
            }
        }
    }
    return 0;
}
```

if/else

for

while
Some Other Very Interesting Features That We Want To Share!!

★ Automatically cast the results of binary operations into a float when we have one integer and one float
★ We can generate matrices of any dimension - even 11 dimensional matrices!!
Behind the Scenes

Compiling Matrices
System Architecture

hello.neo → preprocessor.ml → scanner.ml → parser..mly

hello.ll → codegen.ml → semant.ml

ast.ml → sast.ml
Testing in the Works

Ensuring That MatricCs Always Runs
Build all of the files to ensure that everything works.

Ideal for running simple test cases or with single files - displays the .ll file immediately after successful compilation.

Testing script to test all of the test cases at once.
	est.neo

test.diff

testall.log

test.out
Meet the MatriCs People

“Yes we took the red pill to stay in Wonderland and see how deep the rabbit-hole goes”
Functional programming combines the flexibility and power of abstract mathematics with the intuitive clarity of abstract mathematics.
Lessons Learned

- Start as late as possible to learn about efficiency
- You definitely have to push your limits conceptually in terms of recursion. Downside is that when you try to brag about building a programming language no-one seems to know what that means....
- The LLVM documentation (the actual ones) is a black hole, you can spend your whole life trying to find the meaning of GEP...
- Simple things that you take for granted are often hard to implement
Show Time!!

Time to see MatriCs in action