CompA: Complex Analyzer

Xiping Liu(xl2639)
Jianshuo Qiu(jq2253)
Tianwu Wang(tw2576)
Yingshuang Zheng(yz3083)
Zhanpeng Su(zs2329)

Introduction

CompA is a strongly typed language supports complex number and matrix computations. The frequent use of matrix and complex number in engineering related problems inspire us to built this language that is good at complex arithmetic and matrix manipulation.

Our Goal

Design a imperative language specialized in complex arithmetic and matrix manipulation

Simplicity of treating a complex number as a tuple

Matrix data type for matrix functions and manipulation

Complex data type for complex arithmetic

A sufficient rich library of complex functions and functions of matrix manipulation

Overview

C-like syntax

Main Data Type: matrix, complex

Rich complex-oriented standard library

Compiles to the LLVM

Basics

Primitive Types:

int, float, bool, string

Data Types:

complex, matrix

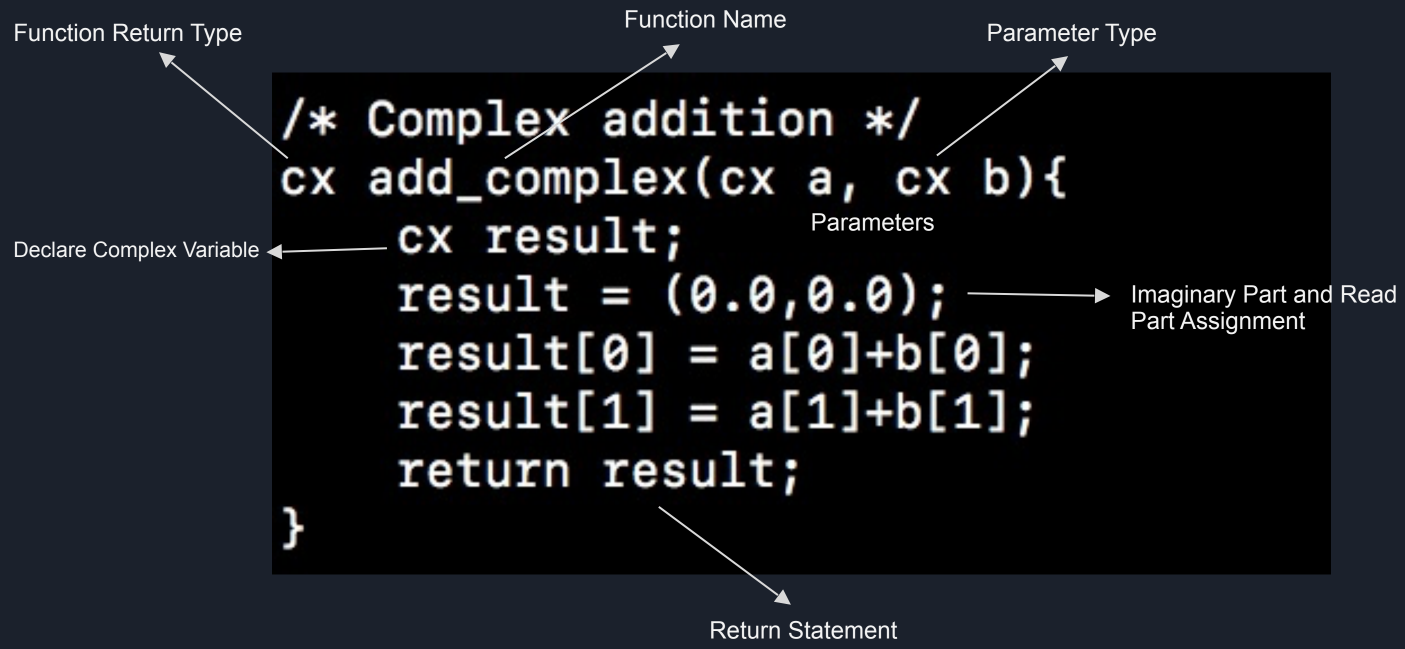
Declaration and Initialization:

```
cx a;

a= (1.2, 2.4);

int[2][3] m;

populate_2D_int(%%m, 1, 2, 3);
```

Function Declaration:

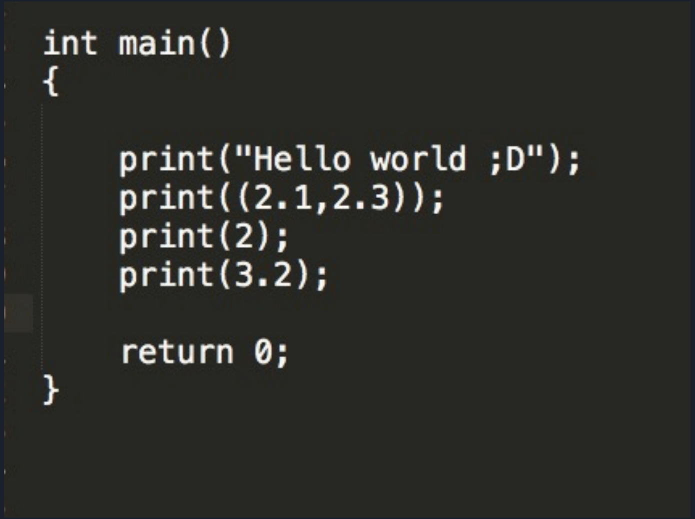
```
void populate_2D_int(int[ ][ ]  x,  int a, int
h, int w) {
    int i;
    for (i = 0; i < (h*w); i = i + 1) {
        #x = a;
        x = ++x;
    }
}
```

Simple Tutorial

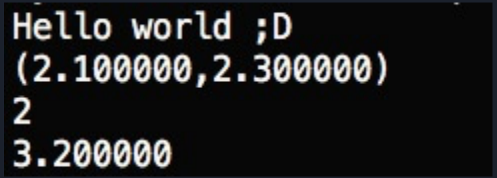


```
/* Complex addition */
cx add_complex(cx a, cx b){
    cx result;
    result = (0.0,0.0);
    result[0] = a[0]+b[0];
    result[1] = a[1]+b[1];
    return result;
}
```

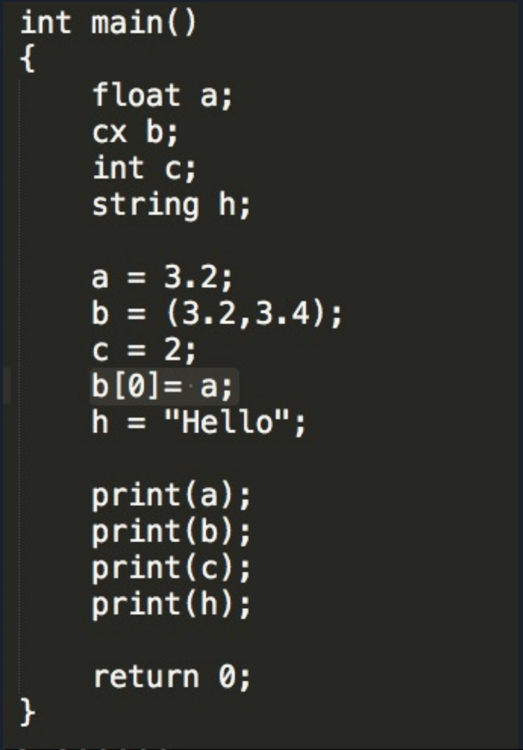
Generic Type Printing



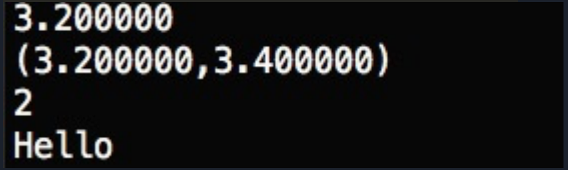
```
int main()
{

    print("Hello world ;D");
    print((2.1,2.3));
    print(2);
    print(3.2);

    return 0;
}
```



```
Hello world ;D
(2.100000,2.300000)
2
3.200000
```

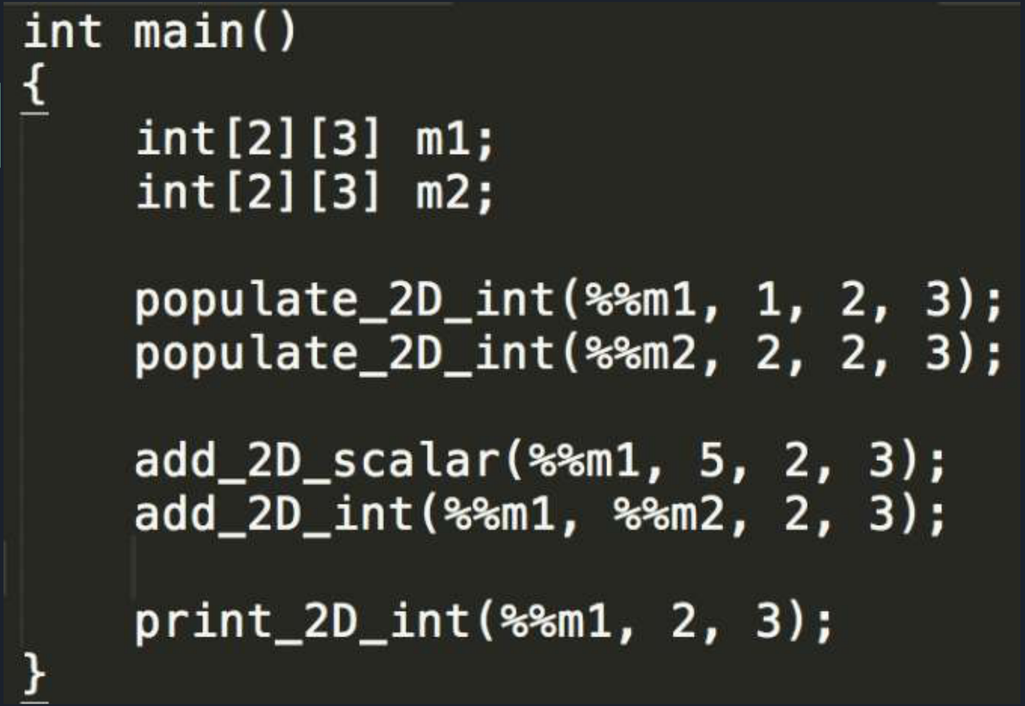


```
int main()
{
    float a;
    cx b;
    int c;
    string h;

    a = 3.2;
    b = (3.2,3.4);
    c = 2;
    b[0]= a;
    h = "Hello";

    print(a);
    print(b);
    print(c);
    print(h);

    return 0;
}
```



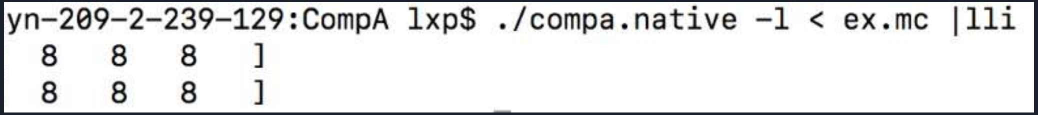
```
3.200000
(3.200000,3.400000)
2
Hello
```

Matrix Addition & Printing



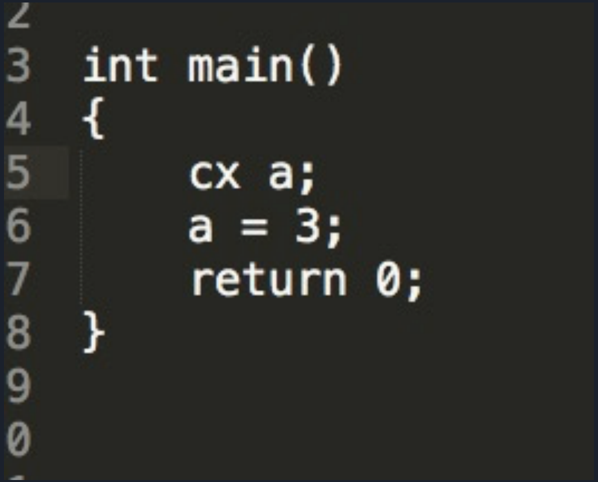
```
int main()
{
    int[2][3] m1;
    int[2][3] m2;

    populate_2D_int(%%m1, 1, 2, 3);
    populate_2D_int(%%m2, 2, 2, 3);

    add_2D_scalar(%%m1, 5, 2, 3);
    add_2D_int(%%m1, %%m2, 2, 3);

    print_2D_int(%%m1, 2, 3);
}
```

Output



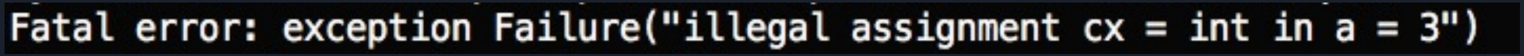
```
yn-209-2-239-129:CompA lxp$ ./compa.native -l < ex.mc |lli
  8   8   8   ]
  8   8   8   ]
```

Semantic Check



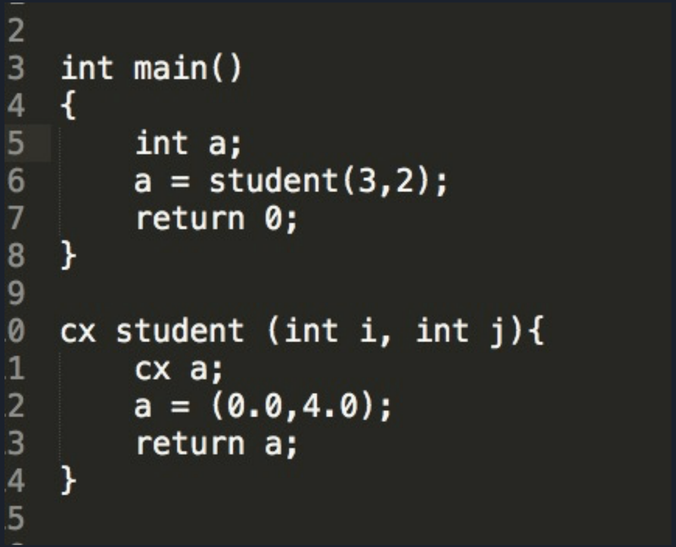
```
int main()
{
    cx a;
    a = 3;
    return 0;
}
```

Error Message



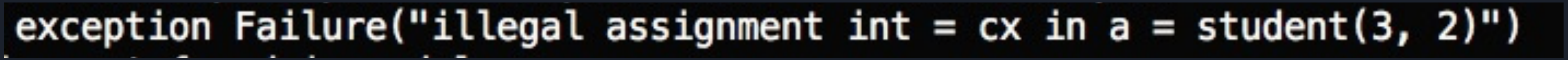
```
Fatal error: exception Failure("illegal assignment cx = int in a = 3")
```

Semantic Check

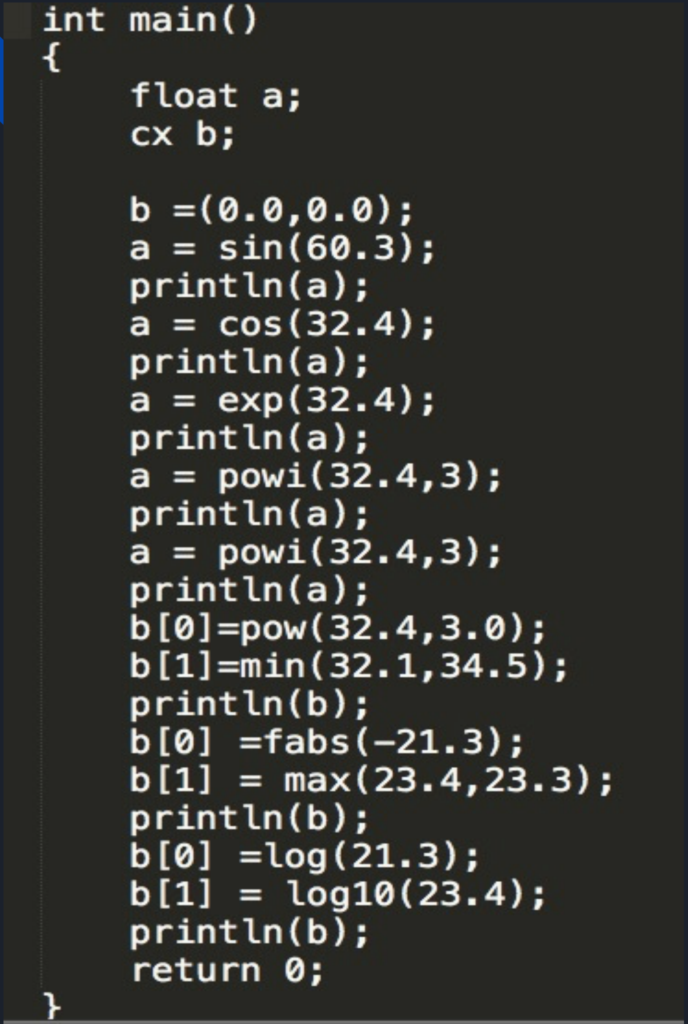


```
int main()
{
    int a;
    a = student(3,2);
    return 0;
}

cx student (int i, int j){
    cx a;
    a = (0.0,4.0);
    return a;
}
```

Error Message



```
exception Failure("illegal assignment int = cx in a = student(3, 2)")
```

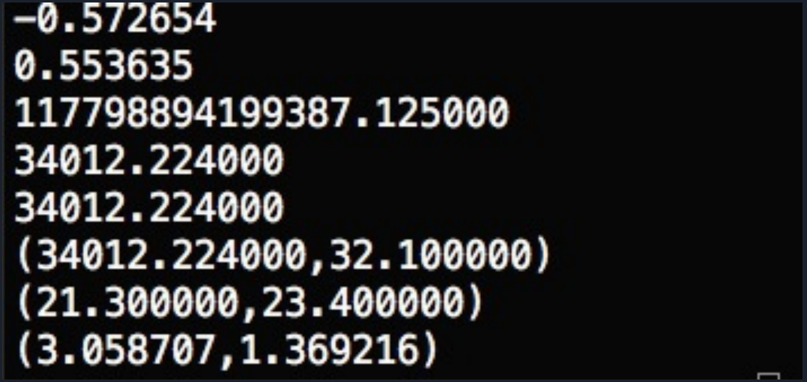


```
int main()
{
    float a;
    cx b;

    b =(0.0,0.0);
    a = sin(60.3);
    println(a);
    a = cos(32.4);
    println(a);
    a = exp(32.4);
    println(a);
    a = powi(32.4,3);
    println(a);
    a = powi(32.4,3);
    println(a);
    b[0]=pow(32.4,3.0);
    b[1]=min(32.1,34.5);
    println(b);
    b[0] =fabs(-21.3);
    b[1] = max(23.4,23.3);
    println(b);
    b[0] =log(21.3);
    b[1] = log10(23.4);
    println(b);
    return 0;
}
```

Built-In Functions

sqrt, sin, cos,
powi, pow, exp,
log, log10,
fabs, min, max

Output



```
-0.572654
0.553635
117798894199387.125000
34012.224000
34012.224000
(34012.224000,32.100000)
(21.300000,23.400000)
(3.058707,1.369216)
```

Demo

Q & A

Thank you!