Carolyn Bozin

CPSC_326

5/10/23

Final Project: MyPL Classes

Description:

My project entails creating classes in myPL with private and public visibility as well as class member variables and methods. Inheritance is not included in my extension. Classes are a useful extension because they provide more functionality than structs (they allow methods to be attached to class objects).

Example:

Simple Car class that stores information about a car, as well as getter methods for car information.

```
class Car {

private:

public:
    string brand
    string model
    int year

    string get_brand(Car c) {
        return c.brand
    }

    string get_model(Car c) {
        return c.model
    }

    int get_year(Car c) {
        return c.year
    }
}
```

```
void main() {
    Car car = new Car
    car.brand = "Subaru"
    car.model = "Legacy"
    car.year = 2012
    int y = get_year(car)
    string b = get_brand(car)
    string m = get_model(car)
    print("Car brand: ")
    print(b)
    print("\n")
    print("Car model: " )
    print(m)
    print("\n")
    print("Car year: ")
    print(y)
    print("\n")
```

High Level Description of modifications:

LEXER

- Added CLASS, PRIVATE, PUBLIC, and SEMICOLON tokens
- Added ClassDef function to emit tokens of a class definition

PARSER

- Added ClassDef object as an ASTNode, and modified rvalues to check if the var refers to a method

    - ClassDef objects store class name as well as 4 vectors that store public & private members and methods
- Added class_def(), class_stmts(), class_method() functions to parse class definitions and add to the ClassDef object
- Added a check for class objects, members and methods in rvalues, as well as a check for class objects and members in lvalues

SEMANTIC CHECKER

- New semantic checker map that maps class names to class definitions
- Added two helper functions to get methods and members from a class
- Added a check for class objects, members and methods in rvalues, as well as a check for class objects and members in lvalues

VM & CODE GEN

- Created ALLOCC instruction to allocate class, as well as instructions to get, add, and set members and methods
- Created a class heap
- Modified code gen to include visitor function for ClassDefs. Also added a map of class object id to a map of class name and class definition
- Modified code gen to be able to add, get, and set members and methods of a class object, as well as to instantiate a class object

What WAS completed:

- Classes can be created alongside structs and functions
- Classes include both private and public visibility
- Classes can have both members and methods

What WAS NOT completed:

- Did not manage to get private members/methods to be used inside the class
- There are issues with adding struct types and other class types to a class
- Did not figure out how to create arrays of classes
- Method calls do not properly work, so instead of ‘obj.f()’ it works with ‘f(obj)’

What I would do next:

Given more time, I would try to fix all of the problems in my code that I mentioned above. Classes are much more useful if you can mix them with structs and other classes. I think it would also be useful to have protected visibility, but that would entail figuring out inheritance, which could be tricky.

Testing methods:

For testing, I mainly relied on using unit tests. All the tests were related to my class extension, but I separated them by ‘type’, for example lexer tests, code gen tests, etc. For instance, for lexer tests I checked that CLASS tokens were being properly emitted. I also used the command line for testing, especially with pretty printing and semantic checking. When I had segmentation faults I used GDB to

help track down the cause. Lastly, I created various example myPL files to run my code on. The Car class at the top is one of these examples.

Instructions for running code:

To make running the program more easy, I created a makefile which created the executables I needed to run. First I used cmake, and then make to compile the program. Then, I ran files on the command line. For instance, to checked the unit tests I would use ‘./class_tests’, and to test my example files I used ‘./mypl examples/…..mypl’.

Demo Video Link:

https://youtu.be/KrkrSTP4l9k