Ben Puryear
5/11/23
CPSC 326-01, OPL
Final Project Report

For my final project I was tasked to implement Default Values and Keyword Arguments. This involved heavy modifications of CallExpr in all MyPL source code files. ast_parser.cpp, ast.h, code_generator.cpp, semantic_checker.cpp, simple_parser.cpp, vm_frame.h, vm.h were all modified in order to implement Default Values and Keyword Arguments.

Description of the extension with examples.

Default Values

```
# Default_Values.mypl


int default_value_example(int x, int y = 5){
    return (x + y)
}
default_value_example(3, 7)
# -> 10


default_value_example(5)
# -> 10
```

In this example we can see that the variable y is set to their default value if and only if there is no parameter passed in for that value. Because of relying on how many arguments were passed in to determine what variables are set to their default value, all parameters with default values must be at the end of the function declaration.

Keyword Arguments

```
# Keyword_Argument.mypl
void keyword_argument_example(x, y, z){
    print(x)
    print(y)
    print(z)
}
keyword_argument_example(10, 20, 30)
# -> 102030


keyword_argument_example(z=10, y=20, x=30)
```

```
# -> 302010


keyword_argument_example(10, z=20, y=30)
# -> 103020
```

In this example you can see that with the use of Keyword Arguments, one can pass in the arguments in an order different from the function declaration. To do this the user needs to specify the specific variable, then use an equal sign and set it to its specific value.

High level descriptions of the parts of the pipeline modified.

- ast_parser.cpp
  - Modified the params function to support default values.
- ast.h
  - Added an optional expression called defaultExpr to VarDef.
  - Added a vector of expressions called keyword_args to CallExpr.
  - Added a Boolean called keyword_values to CallExpr. This will be true if there are any keyword_args in the CallExpr.
- code_generator.cpp
  - When looping through the params of a function, if the param has a default_value, push back the value of default_value to the curr_frame's vector "default_values".
  - Added extensive support for keyword_arguments and default values.
    - A large majority of the added code is here, handling how to order the instructions to push the certain values.
- semantic_checker.cpp
  - Added checking for the correct parameter names for keyword arguments.
  - Similar implementation of code_generator.cpp for checking for correct grammar by checking the parameters for keyword_arguments.
- simple_parser.cpp
  - Added checking for ASSIGN tokens when checking for params (implementing keyword_arguments) and type-checking the values.
- vm_frame.h
  - Included ast.h.
  - Added a vector of Expr named default_values.
  - Added a vector of strings named parameter_names.
- vm.h
  - Moved frame_info to be public.

What was completed?

Everything.

How was this tested?

I created 2 different test files, as well as 20 specialized tests. These tests, 10 for each, covered the ast_parser, simple_parser, code_generator, and VM changes.

As well as creating these 20 tests, I also copied in the 497 tests from the previous homework assignments to double check that any of my changes broke previous MyPL functions.

How to build and run this extensions

After cloning the repo, run `cmake .` followed by `make`.

To run all tests, including the old tests, use the command `make -f AllTests.mk`.

To run the new tests, use the following commands:

```
./default_value_tests
./keyword_argument_tests
```

Demo Video

https://www.youtube.com/watch?v=z0pYvjoqXmw