Santiago Calvillo
CPSC 326
Organization of programming languages
5/11/2023

Final Report

This report will cover the extension I designed for the mypl program implementation. During the semester we implemented the mypl language going through the pipeline of processes it needs to go through. For the extension we were asked to develop, I decided to elaborate a transpiler that will convert the mypl language into C# code and will be able to execute it. What this means is that when a mypl program is typed, there is a –csharp flag that will perform a series of steps. The steps that will execute are the following:

1. Save the name of the mypl file to name the csharp files.
2. Create the mypl lexer with what has been written and the parser with the programs from HW2 and 3 and build the AST.
3. Use the AST in a newly built CSharpPrintVisitor to create a file of the mypl program in csharp syntax using the same logic as the pretty printer developed in class but puts the output in a file instead of output in terminal.
4. Create a new directory to store all the new project information as csharp programs require more than the file to be able to run. Creates a folder in the C#TestOuputs folder.
5. Move to that directory and remove folder if is the same name.
6. Using dotnet it will create a new console that creates the files able to run the program in chsarp and replaces the default program, Program.cs with the .cs file generated in the pretty printer.
7. Run the program.

Santiago Calvillo
CPSC 326
Organization of programming languages
5/11/2023

This is the basic pipeline at a high level that the program follows to execute the csharp program. As to the modifications that had to be made, there were two parts that had to be changed. One is the mypl.cpp that is where the csharp flag was added. This is were all the file creation and were dotnet is called to run the program, and it was made so that automatically when doing ./mypl –csharp “filename” it will execute the program in csharp without having to input another command. The other part I had to do was to create a file that will make a pretty printer for csharp. To do this I used the abstract class visitor to create a CSharpPrintVisitor that will print the csharp code modifying the parts to adapt the syntax. The biggest differences between csharp and mypl I would say are the built in functions. Input output are written as Console.Write and Console.Read respectively. For the length it is .length instead of length() and this happens the same for the tostring. Other than some minor changes the only big thing is that you must have a main class created and all the functions must be static so I adjusted that.

I will say I pretty much completed everything I wanted to do in my project as to make the transpiler as it successfully takes a mypl program and converts it. The only part I would’ve addressed if I would’ve had more time would be to adapt the scope. Different to mypl csharp has some restrictions in the scope of variable declarations. Specifically, I found out that you cannot name functions and names the same way and you cannot use a variable that has been used inside a for loop. If I would’ve had more time, I would’ve probably addressed that problem. The other issue is probably the time it takes too long to build the csharp folder. This happens because it has to build a new folder and run dotnet commands and I will say it was the most challenging part of the project. Finding the proper extensions and going through the

documentation was the most heavy assignment and it was quite challenging, probably the reason this project is enlisted as a quite challenging one as it took most of my time to figure out how to run the program.

Finally for my testing I decided it was not worth it to build unit tests as what has to be tested is that the code runs as it should run with csharp producing the same output as the mypl. This is why I created several mypl programs and put instructions to run them in mypl and csharp and put the output in text files to then compare the files with cmp checking the output is the same. I developed 7 programs that tested most of the functions mypl offers like loops, structs and operations. In addition one thing I noted when running the programs is that even though the csharp programs take around 10 seconds to build as it builds the dependencies they are incredibly faster compared to the mypl. For example a Fibonacci program will take less time to run in csharp compared to mypl even though mypl runs instantly. In mypl it took 12 seconds compared to csharp's 11 seconds, where 10 of those are to build the folders so really impressive the difference of speed although it might only be seen in recursive long programs that take several computation time. In conclusion a very interesting project to make in which I was able to dive more into computation and processing time for an interpretation language like mypl and compare it with a compilation program, much faster.

Finally to run the program and tests these are the instructions:

To run program "filename" in csharp: **./mypl –csharp "filename"**

To run bash file for compare output of tests: **./test_runner.sh** (chmod +x test_runner.sh if it is not compiled)