Transpiler For C To Python

CS 252: Advanced Programming Languages

Parth Jayantilal Jain
San José State University

December 11, 2017

Abstract

There are many languages available to write a program for a computer. Many languages are domain specific. It is very difficult for a person to know all the languages. If a person knows the concept and knows the application then the programming language should not be a barrier. With this motivation i decided to implement a transpiler to convert a program given in C language to a program in Python.

Contents

1 Introduction

As there are so many programming languages available and this shouldn't be a barrier for implementing the ideas, I have implemented a transpiler in haskell to convert a C code to a python code.

1.1 What is Transpiler?

A transpiler or source-to-source compiler, transcompiler is a type of compiler that takes the source code of a program written in one programming language as its input and produces the equivalent source code in another programming language.[2]

1.2 Transpiler VS Compiler

A transpiler translates between programming languages that operate at approximately the same level of abstraction, while a traditional compiler translates from a higher level programming language to a lower level programming language. For example, a transpiler may perform a translation of a program from Python to C, whereas a compiler may perform a translation of a program from java to assembly language code.[2]

1.3 Languages used

A transpiler involves three languages, language for input program, language for output program, and the language in which the transpiler is used. In my project, the input program is in C language, the output program is in Python and the transpiler is written in haskell.

2 Features Supported

- Main function
- return 0
- include statements
- if loop
- if loop with boolean operators (and, or, not)
- while loop
- while loop with boolean operators (and, or, not)
- variable assignment (int, float, char)
- arithmetic operations
- support for relational operations
- print string
- print variables
- increment
- decrement
- array initialization
- open a file

3 Implementation

The transpiler reads an input file and using parsec library from haskell parses the input file. Then using the concept of AST, i am doing pattern matching for the input program and then the appropriate python code is written to the output file.Now as we know there are no brackets used to describe the scope of a block in python. Python uses spacing or indentation for declaring the scope.Therefore, i wrote a python program to give indentation to the output code.
I have included features for if loop and while loop, which can have one or two conditions, not more than that and these conditions are separated by either and operator, or operator or one condition consist of not operator. Both the loops can consist of multiple expressions inside the loop. Variables can be assigned an int value, a float value, a char, another variable, an arithmetic operation. The arithmetic operations supported are plus, minus, multiplication and division. It also has a feature of assigning an array with values, but all the values needs to be assigned at once. I have also added feature to open a file and handle main function. The program can consist of only one function that is the main function. I have also added features for handling relational operators, but only one operator can be used at a time. It also consist of features to print string and print variables. If you want to print a string and variable together, then the variable needs to be at the end of string. I have also added feature of increment and decrement operator.[1]

An example of C code:

```c
#include <stdio.h>

int main()
{
    int x = 1;
    int y = 1;
    char a = 'a';
    int array [3] = {1,2,3};
    float b = 1.2;
    FILE *file;
    file = fopen("tt.txt", "r");
    if (x==1)
    {
        printf("First if loop");
    }
    if ((x == 1) && (y==1))
    {
        printf("second if loop %i",x);
    }
    if ((x == 1) && (y==1))
    {
        printf("second if loop %i",x);
    }
    if ((x == 2) || (y == 1))
    {
        printf("%c",a);
    }
    int counter = 1;
    while (x<=5)
    {
        printf("%i",x);
        x++;
    }
    y = y-1;
    if (!(x==1))
    {
        printf("%i",x);
    }

    return 0;
}
```

To run the transpiler, paste the file "project.hs" in the same folder as the input program. Open terminal and type the command runhaskell project.hs <inputfilename> > <outputfilename>. The outputfile should be a .txt file. Now paste the file indentation.py in the same folder as project.hs and now run the file indentation.py. The program will now ask for input filename, give .txt file generated by project.hs, then the program will ask for the output file, give the name of the output .py file. Now run the output python file.

The final output file generated is:

```
def main():
    x = 1
    y = 1
    a = 'a'
    array = [1,2,3]
    b = 1.2
    file = open("tt.txt", 'r')
    if x == 1:
        print "First if loop"
    if ((x == 1) and (y == 1)):
        print "second if loop  " + str(x)
    if ((x == 1) and (y == 1)):
        print "second if loop  " + str(x)
    if ((x == 2) or (y == 1)):
        print " " + str(a)
    counter = 1
    while x <= 5:
        print " " + str(x)
        x = x + 1
    y = y-1
    if not(x == 1):
        print " " + str(x)
if __name__ == '__main__': main()

```

One of the challenges faced during adding feature for initializing variable with floating point value, was that if you parse the numbers after decimal using "many digit" or "many1 digit" then it is not able to identify zero. I solved this problem by parsing it in string format and separating the digits to the left and to right by using sepBy.

4 Scope and Future Work

In future, more features can be added like multiple arithmetic operations can be used in a single expression, loops can support more then two conditions. values of array can be accessed, and a particular value can be assigned to the array. Features to read, write and modify files can be added. The current version of transpiler supports some features and not the entire language, so more features can be added in future.

5 Conclusion

This project gave me an opportunity to learn how to design a transpiler, it also helped to understand the structure of the language design and how a compiler works. It also helped explore haskell language and its application.

References

[1] Thomas Austin. Parsec. `http://www.cs.sjsu.edu/~austin/cs252-fall17/slides/CS252-Day11-Parsec.pdf`, 2017. [Online; accessed 20-Nov-2017].

[2] Wikipedia. Source-to-source compiler. `https://en.wikipedia.org/wiki/Source-to-source_compiler`, 2017. [Online; accessed 03-Dec-2017].