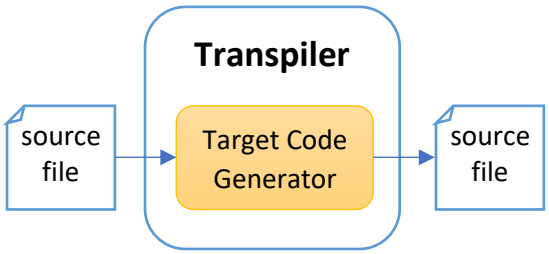
Lab-01 Create minimal source-to-source compilers

In this lab, we are going to implement a set of minimal **source-to-source compilers**, also known as **transpilers** or **transcompilers**. These transpilers will serve as bases for implementing a full retargetable compiler. In this lab, the structure of the compiler is very simple because it contains only one component, the **target code generator**.



The transpiler accepts the input from the input stream and generates the codes in a target high-level language.

1 Objectives:

- Lay down the ground work for implementing a full retargetable compiler.
- Construct a compiler incrementally in a back-to-front approach.

2 Learning outcomes:

By completing this lab, learners should be able to

- create 5 target code generators.
- create 5 minimal compilers.
- recognize the similarities and differences of the target codes in 5 target codes.

3 Task: Print a single integer

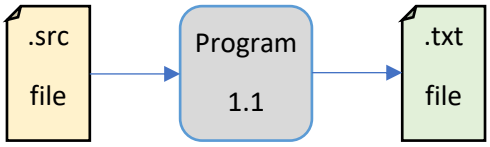
Create 5 transpilers for generating target codes in 5 different target languages (regular text file, Python code, C code, C++ code, and C# code, respectively). The input into each transpiler is a source file (text file with the extension .src) containing only one positive integer. The output from the transpiler is the source code in a target language.

For this lab, we assume that the source file is always correct. There are no needs for error detection. We will handle errors in the following labs.

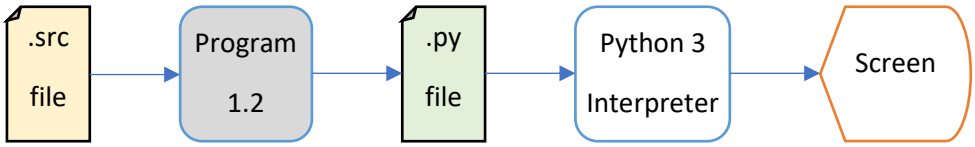
Note that: If the generated source code (except the text copy) is compiled and executed, it should print out the numbers contained in the source file.

The specifications for the target codes are as follows:

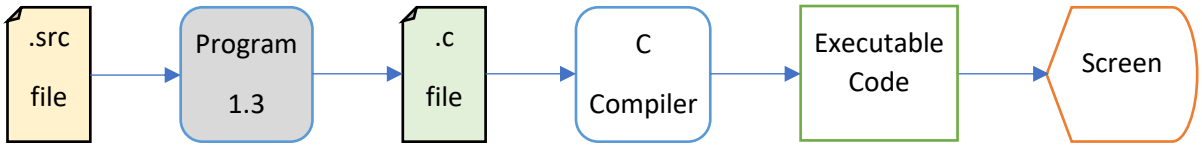
1. A text file with extension .txt containing exactly the number from the source file. In other words, just copy the source file.



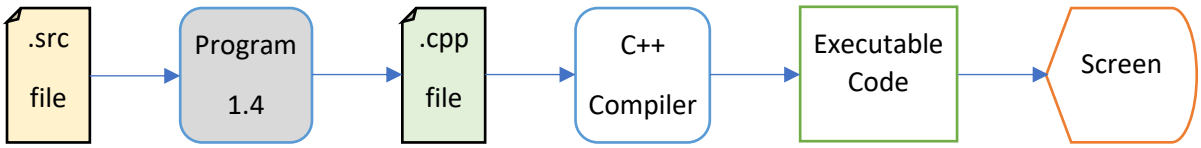
2. A Python 3 source code with extension .py containing the statement to output the number from the source file. The target code should be executable by a Python 3 interpreter and print the out the number from the source file (.src) to the screen.



3. A C source code with extension .c containing the program to output the number from the source file. Learners should be able to compile the target code using a C compiler to generate the executable program. The executable program should print the number from the source file (.src) to the screen.



4. A C++ source code with extension .cpp containing the program to output the number from the source file. Learners should be able to compile the target code using a C++ compiler to generate the executable program. The executable program should print the number from the source file (.src) to the screen.



5. A C# source code with extension .cs containing the program to output the number from the source file. Learners should be able to compile the target code using a C# compiler to generate the executable program. The executable program should print the number from the source file (.src) to the screen.