Chapter 1

Introduction

1.1 Interpreter and compiler

There are two approaches to implement a programing language: compilers and interpreters.

Interpreter is an online approach, i.e. the work done by the interpreter is part of running the program. The program we write and the data on which we wish to run the program are inputted into the interpreter, after which the output is produced by the interpreter.

Compiler is an offline approach, i.e. whatever the compiler does is the preprocessing of the program, and it does not take part in the actual execution of the program on the data. The program is translated into an executable by the compiler, and the data is passed to the executable, which then outputs the result.

1.2 History

In 1954, IBM developed the 704 machine. The customers found that the softwares cost more than the hardware, though the hardware already costs a lot. This inspired a lot of people to try to improve the productiveness of programing, among whom was John Backus. He developed "Speedcoding", which from today's the point of view is an interpreter. Speedcoding made it much faster to develop programs, but the programs developed with it ran much slower and also occupied too much memory. Backus continued to develop the FORTRAN project, which is an abbreviation for FORmula TRANslation. With FORTRAN I, he took a compiler approach: formulae written by programmers were translated into a form that could be understood by the machine. FORTRAN I was a successful project not only in the sense that it was soon adopted by most developers back in the 1950s, but also in the sense that its outline is still preserved by modern compilers. A compiler contains 5 phases:

Lexical analysis Syntactic.

Parsing Syntactic.

Semantic analysis Types, scopes, etc.

Optimization

Code generation Translation into another language.