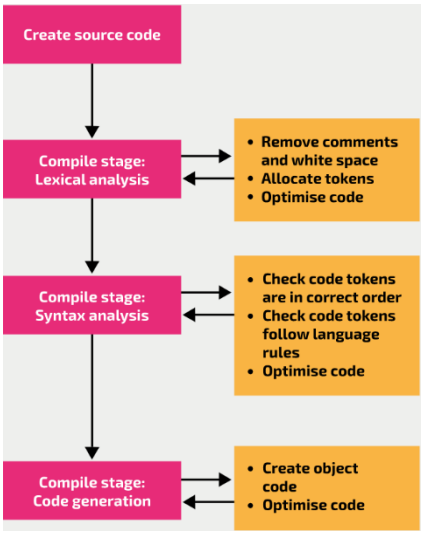
Investigation about programing – Fundamental algorithms

By César Sobrino Pech

Stage of program compilation



Why is important know about programing flow when we are studying Data engineer?

The answer is easy, we must know how all our works, or how we can write and develop tools or how to make our work easy to become in a computer software which set of step-by-step instructions that directs the computer to do the tasks you want it to do and produce the results you want.

The compilation is that computers do when we give to them human language and the compiler makes our orders in machine language. There are 6 stages of compiling a program

- lexical analysis
- symbol table construction
- syntax analysis
- code generation
- optimization

Lexical analysis

is the first stage of the compilation process, where the source code created by the programmer is tokenized for translation into executable code. The tokens can be converted on instructions than u can say at algorithm-logic procreator what it should do with the instructions and variables.

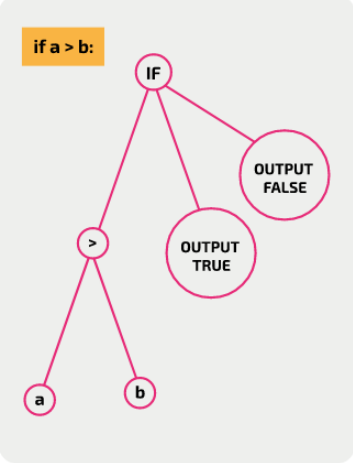
The process can be represented diagrammatically as follows (the individual items are referred to as **lexemes**):

Lexeme	Token	Pattern
user_name	identifier	Letter followed by digits or letters
=	operator	=
"gwen"	literal	Any string between a pair of single or double quotes

Symbol table construction are all lexemes and codes than the syntaxis analexed can be represent ate Symbol Table is an important data structure created and maintained by the compiler in order to keep track of semantics of variables i.e. it stores information about the scope and binding information about names, information about instances of various entities such as variable and function names, classes, objects, etc.

syntax analysis

Syntax analysis is the compilation stage immediately following lexical analysis. Once tokens have been assigned to the code elements, the compiler checks that the tokens are in the correct order and follow the rules of the language. For example, in Python the command `print(user_name)` is syntactically correct, in that it follows the rules for a print statement: print in lowercase immediately followed by a bracket followed by an identifier and closed by a righthand bracket. In abstraction, this is no different from natural languages such as English; the sentence "he Moved wearily" fails on 3 syntax points: the sentence does not start with a capital letter; a capital letter is used incorrectly in the second word; there is no full stop to end the sentence. The syntax rules for programming languages are finite but must be followed.



Syntax rules differ between languages:

- `if a > b:` is syntactically correct in Python
- `if a > b` is not syntactically correct in Python - the rule for using a colon is broken
- `if (a == b)` is syntactically correct in Java
- `if (a = b)` is not syntactically incorrect in Java - a single equals sign is not a conditional operator

code generator

This stage follows the stages of lexical and syntax analysis. A separate program is created that is distinct from the original source code. The code generated is the object code, which is the binary equivalent of the source code. This is the executable version of the code, before linked libraries are included.

Code generation is a major distinguishing feature between compilation and interpretation; interpreters do not produce a separate executable file.

Code optimization

Code optimization is carried out throughout the compilation process and as part of the code generation stage. The optimizer may identify redundant or repeated code and remove or rearrange the code as necessary. Examples are removing procedures that are never called or moving an assignment statement that had incorrectly been placed inside a loop, potentially causing it to be inefficiently executed multiple times.

Levels of programing

A programming language defines a set of instructions that are compiled together to perform a specific task by the CPU (Central Processing Unit). The programming language mainly refers to high-level languages such as C, C++, Pascal, Ada, COBOL, etc.

Each programming language contains a unique set of keywords and syntax, which are used to create a set of instructions. Thousands of programming languages have been developed till now, but each language has its specific purpose. These languages vary in the level of abstraction they provide from the hardware. Some programming languages provide less or no abstraction while some provide higher abstraction. Based on the levels of abstraction, they can be classified into two categories:

- Low-level language
- High-level language

Type of Language	Example Language	Description	Example Instructions
High-level Language	Python, Visual Basic, Java, C++	Independent of hardware (portable). Translated using either a compiler or interpreter. One statement translates into many machine code instructions.	payRate = 7.38 Hours = 37.5 Salary = payRate * Hours
Low-level Language	Assembly Language	Translated using an assembler. One statement translates into one machine code instruction.	LDA181 ADD93 STO185
	Machine Code	Executable binary code produced either by a compiler, interpreter or assembler.	10101000110101010100100101010101

Low-level language

The low-level language is a programming language that provides no abstraction from the hardware, and it is represented in 0 or 1 forms, which are the machine instructions. The languages that come under this category are the Machine level language and Assembly language.

High-Level Language

The high-level language is a programming language that allows a programmer to write the programs which are independent of a particular type of computer. The high-level languages are considered as high-level because they are closer to human languages than machine-level languages.

When writing a program in a high-level language, then the whole attention needs to be paid to the logic of the problem.

A compiler is required to translate a high-level language into a low-level language.

https://bournetocode.com/projects/GCSE_Computing_Fundamentals/pages/3-2-9-class_prog_langs.html

https://www.bbc.co.uk/bitesize/guides/zmthsrd/revision/3