

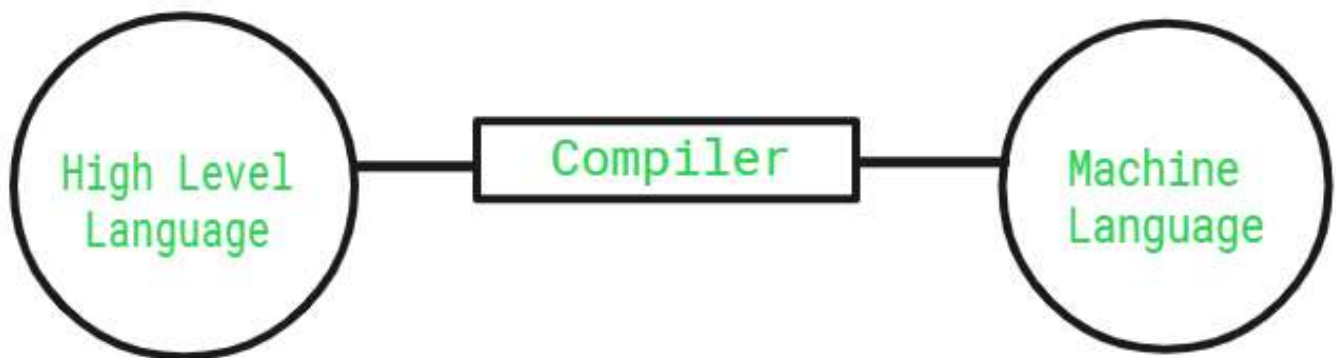


Introduction To Compilers

Trending Now Data Structures Algorithms Topic-wise Practice Python Machine Learning Data Science

A Compiler is a software that typically takes a high level language (Like C++ and Java) code as input and converts the input to a lower level language at once. It lists all the errors if the input code does not follow the rules of its language. This process is much faster than interpreter but it becomes difficult to debug all the errors together in a program.

A compiler is a translating program that translates the instructions of high level language to machine level language. A program which is input to the compiler is called a **Source program**. This program is now converted to a machine level language by a compiler is known as the **Object code**.



How does a compiler work?

The process of compiling code involves several steps. Basically, Code passes through these steps sequentially and if there is any mistake in code then it will be examined through these steps and thus compilation process stops in-between and show compilation error. Otherwise if everything is OK then compiler does not show any error and compile the source code .

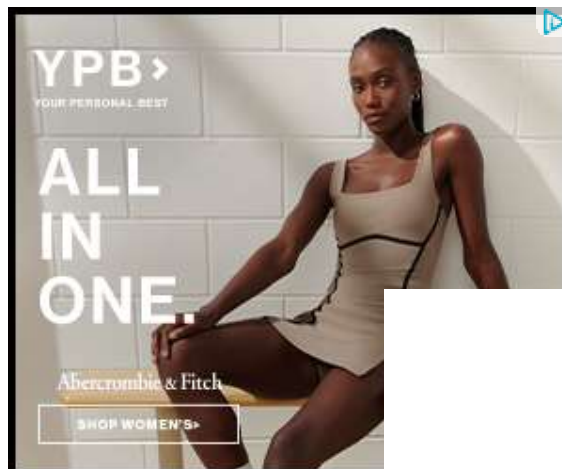
There are the following steps which are involved during compiling:

Lexical Analysis

The source code is first tested by the compiler's lexer, which breaks the source code into tokens, such as keywords, identifiers, operators, and punctuation.

Syntax Analysis

The next step is syntax analysis, where the compiler's parser checks the code for syntax errors and ensures that it follows the rules of the programming language. The compiler generates an Abstract Syntax Tree (AST) that represents the structure of the code.



Semantic Analysis

Once the code is syntactically correct, the compiler performs semantic analysis on parsed code to find the meaning. The compiler check for logical errors, such as type mismatches, undeclared variables, and incorrect usage of operators.

Optimization

The compiler may perform various optimizations to improve the performance of the resulting code. It is an optional phase of compiler. It removes dead code and arranges the sequence of instructions in order to boost the program execution.

Code generations

The last step is code generation, where the compiler translates the AST into machine-readable code. The code generator creates assembly language code, which is then translated into binary code that can be executed by the computer.

Why are compilers important?

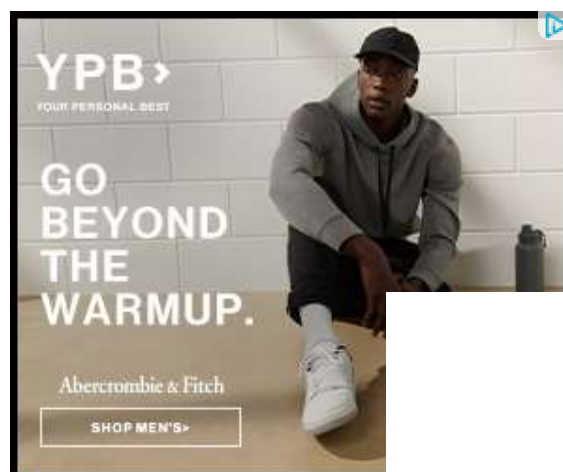
There are the following reasons why compilers are important to us:

1)Enabling concept of High Level Language

They enable developers to write code in high-level programming languages, which are easier to understand and more human-readable than machine code.

2)One time effort(Efficiency)

Compilers convert high-level code into machine code .Now, this machine code is stored in a file having extension(.exe) which is directly executed by computer. So,programmer need only one time effort to compile code into machine code and after that they can use code whenever they want using (.exe) file. This process allows programs to run much faster and more efficiently than if they were interpreted at runtime.



3)Portability

Compilers translates source code which is written in high-level programming language into machine code. And thus, it enables portability feature because this machine code can be run on many different operating systems and hardware architectures because it is platform independent.

4) Security

Compiler can provide programmer security by preventing memory-related errors, such as buffer overflows, by analyzing and optimizing the code. It can also generate warnings or errors if it detects potential memory issues.

There are different Compilers :

- **Cross-Compiler** – The compiled program can run on a computer whose CPU or Operating System is different from the one on which the compiler runs.
- **Bootstrap Compiler** – The compiler written in the language that it intends to compile.
- **Decompiler** – The compiler that translates from a low-level language to a higher level one.
- **Transcompiler** – The compiler that translates high level languages.

A compiler can translate only those source programs which have been written in the language for which the compiler is meant. Each high level programming language requires a separate compiler for the conversion.

For Example, a **FORTRAN** compiler is capable of translating into a **FORTRAN** program. A computer system may have more than one compiler to work for more than one high level languages.

Top most Compilers used according to the Computer Languages –

- **C**– Turbo C, Tiny C Compiler, GCC, Clang, Portable C Compiler
- **C++** -GCC, Clang, Dev C++, Intel C++, Code Block

- **JAVA**– IntelliJ IDEA, Eclipse IDE, NetBeans, BlueJ, JDeveloper
- **Kotlin**– IntelliJ IDEA, Eclipse IDE
- **Python**– CPython, JPython, Wing, Spyder
- **JavaScript**– WebStorm, Atom IDE, Visual Studio Code, Komodo Edit



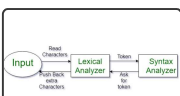

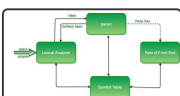
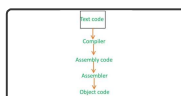
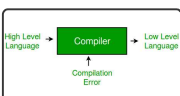
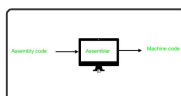

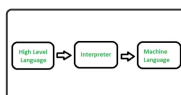
Conclusion:

compilers are critical tools for software development. They enable developers to write code in high-level programming languages, ensure that the code is correct and efficient, and make it possible to develop software for different platforms and architectures. Understanding compilers is essential for any programmer who wants to develop efficient, reliable, and scalable software.

Last Updated : 11 May, 2023

25

Similar Reads

	Single pass, Two pass, and Multi pass Compilers		Compiler Design - Science of Building a Compilers
	Introduction of Lexical Analysis		Introduction to Syntax Analysis in Compiler Design
	Parsing Set 1 (Introduction, Ambiguity and Parsers)		Introduction of Object Code in Compiler Design
	Introduction of Compiler Design		Introduction of Assembler
	Introduction to YACC		Introduction to Interpreters

[Previous](#)

[Next](#)