الرقم الجامعي: AD0039

الاسم: فراس سمير رمضان سليم

Compiler VS Interpreted

Compiler:

In computing, a compiler is a computer program that translates computer code written in one programming language (the source language) into another language (the target language). The name "compiler" is primarily used for programs that translate source code from a high-level programming language to a low-level programming language (e.g. assembly language, object code, or machine code) to create an executable program.

There are many different types of compilers which produce output in different useful forms. A cross-compiler produces code for a different CPU or operating system than the one on which the cross-compiler itself runs. A bootstrap compiler is often a temporary compiler, used for compiling a more permanent or better optimised compiler for a language.

Related software include decompilers, programs that translate from low-level languages to higher level ones; programs that translate between high-level languages, usually called source-to-source compilers or transpilers; language rewriters, usually programs that translate the form of expressions without a change of language; and compiler-compilers, compilers that produce compilers (or parts of them), often in a generic and reusable way so as to be able to produce many differing compilers.

A compiler is likely to perform some or all of the following operations, often called phases: preprocessing, lexical analysis, parsing, semantic analysis (syntax-directed translation), conversion of input programs to an intermediate representation, code optimization and machine specific code generation. Compilers generally implement these phases as modular components, promoting efficient design and correctness of transformations of source input to target output. Program faults caused by incorrect compiler behavior can be very difficult to track down and work around; therefore, compiler implementers invest significant effort to ensure compiler correctness.

Compilers are not the only language processor used to transform source programs. An interpreter is computer software that transforms and then executes the indicated operations. The translation process influences the design of computer languages, which leads to a preference of compilation or interpretation. In theory, a programming language can have both a compiler and an interpreter. In practice, programming languages tend to be associated with just one (a compiler or an interpreter).

Interpreted:

In computer science, an interpreter is a computer program that directly executes instructions written in a programming or scripting language, without requiring them previously to have been compiled into a machine language program. An interpreter generally uses one of the following strategies for program execution:

Parse the source code and perform its behavior directly;

Translate source code into some efficient intermediate representation or object code and immediately execute that;

Explicitly execute stored precompiled bytecode made by a compiler and matched with the interpreter Virtual Machine.

Early versions of Lisp programming language and minicomputer and microcomputer BASIC dialects would be examples of the first type. Perl, Raku, Python, MATLAB, and Ruby are examples of the second, while UCSD Pascal is an example of the third type. Source programs are compiled ahead of time and stored as machine independent code, which is then linked at run-time and executed by an interpreter and/or compiler (for JIT systems). Some systems, such as Smalltalk and contemporary versions of BASIC and Java may also combine two and three. Interpreters of various types have also been constructed for many languages traditionally associated with compilation, such as Algol, Fortran, Cobol, C and C++.

الاسم: فراس سمير رمضان سليم

الرقم الجامعي: AD0039

Compiler VS Interpreted

……

While interpretation and compilation are the two main means by which programming languages are implemented, they are not mutually exclusive, as most interpreting systems also perform some translation work, just like compilers. The terms "interpreted language" or "compiled language" signify that the canonical implementation of that language is an interpreter or a compiler, respectively. A high-level language is ideally an abstraction independent of particular implementations.

Compiler VS Interpreted:

The difference	Compiler:	Interpreter:
Execution Process:	**It compiles the code into machine language or intermediate language before execution.**	**Compiles and executes the code line by line without creating a separate executable file.**
Speed of Execution:	**It usually works faster because all the code is compiled before execution.**	**It may be relatively slower because compile occurs during execution.**
Portability:	**Usually the software is compatible with specific platforms and requires recompiling.**	**The software is usually more portable between platforms without recompiling.**
Debugging:	**It can be more challenging because the code is not compiled directly.**	**Debugging can be easier because the interpreter gives real-time feedback.**
Memory Usage:	**Memory consumption may be lower because there is no need to keep the code in memory.**	**Memory consumption may be higher because the code and interpreter need to be in memory.**
Software security:	**It can be more secure since the original code is not present during execution.**	**It can be less secure due to the original code being present during execution.**
Examples:	**C, C++, Java (bytecode is compiled by the Java compiler)**	**Python, JavaScript, Ruby**