تفاوت کامپایلر و مفسر

کامپایلر چیست؟

کامپایلر (compiler) نوعی ابزار ترجمه است که ورودی را به شکل زبان سطح بالا می‌گیرد و خروجی را به زبان سطح پایین مانند زبان ماشین یا اسمبلی تولید می‌کند. [1]

در کامپایلر باید قبل از اجرای برنامه، کدها یکبار به زبان ماشین کامپایل شوند تا برنامه به درستی اجرا شود. هر تغییر کوچکی در کدها اعمال شود، باید یکبار کدها از اول کامپایل شوند. بعد از کامپایل شدن کدها، یک فایل خروجی از برنامه (برای مثال exe) برای اجرا ساخته می‌شود.اگر خطایی در کدها وجود داشته باشد، کدها کامپایل نمی‌شوند و خروجی فایل تحویل داده نمی‌شود و با خطای Compile error مواجه می‌شویم.[2]

مفسر چیست؟

مفسر به انگلیسی Interpreter نیز مانند کامپایلر، نرم‌افزاری است که کدهای برنامه نویسی را از زبان سطح بالای قابل فهم برای انسان به زبان قابل فهم برای کامپیوتر یا «زبان ماشین» تبدیل می‌کند.[2]

نقش مفسر چیست؟

نقش مفسر ترجمه کد منبع به زبان مقصد است. مفسرها با پردازش کد خط به خط عمل می‌کنند و هر خط را قبل از رفتن به خط بعدی اجرا می‌کنند. آن‌ها دستورالعمل‌های زبان سطح بالا (High Level Language) را به دستورالعمل‌های زبان ماشین تبدیل می‌کنند که می‌تواند مستقیماً توسط رایانه قابل درک و اجرا باشد.[1]

مزایا و معایب مفسر [1]

مزایای مفسر:

1. اشکال‌زدایی آسان‌تر: برنامه‌هایی که به زبان تفسیر شده نوشته شده‌اند، عموماً اشکال‌زدایی آسان‌تری دارند. از آنجایی که مفسرها کد را خط به خط اجرا می‌کنند، بازخورد فوری ارائه کرده و به توسعه‌دهندگان اجازه می‌دهند تا خطاها را به طور مؤثرتری شناسایی و تصحیح کنند.

2. مدیریت خودکار حافظه: مفسرها اغلب مدیریت حافظه را به طور خودکار انجام داده و خطر خطاهای مربوط به حافظه مانند نشت حافظه یا سرریز شدن بافر را کاهش می‌دهند. این می‌تواند فرآیند توسعه را ساده کرده و ثبات کلی برنامه را بهبود بخشد.
3. انعطاف‌پذیری: زبان‌های تفسیر شده انعطاف‌پذیری بیشتری نسبت به زبان‌های کامپایل ارائه می‌دهند. مفسرها امکان تایپ پویا، تخصیص حافظه پویا و اصلاح کد زمان اجرا را فراهم می‌کنند و آن‌ها را برای نمونه‌سازی سریع، اسکریپت‌نویسی و محیط‌های توسعه تعاملی مناسب می‌سازد.

معایب مفسر:

1. اجرای برنامه محدود: مفسرها فقط می‌توانند برنامه‌هایی را اجرا کنند که به طور خاص برای مفسر مربوطه طراحی شده‌اند. این بدان معناست که کد نوشته شده برای یک مفسر ممکن است مستقیماً توسط مفسر یا کامپایلر دیگر بدون تغییرات قابل‌اجرا نباشد.
2. سرعت اجرای کمتر: کد تفسیر شده در مقایسه با کد کامپایل شده کندتر اجرا می‌شود زیرا در طول زمان اجرا خط به خط ترجمه و اجرا می‌شود. مفسرها از مزیت بهینه‌سازی پیش از زمان انجام شده توسط کامپایلرها برخوردار نیستند که می‌تواند بر عملکرد در وظایف محاسباتی فشرده تأثیر بگذارد.

مزایا و معایب کامپایلر [1]

مزایای کامپایلرها:

1. افزایش سرعت اجرا: کد کامپایل شده در مقایسه با کد تفسیر شده سریع‌تر اجرا می‌شود، زیرا قبل از اجرا مستقیماً به کد ماشین ترجمه خواهد شد و اینجاست که تفاوت کامپایلر و مفسر خودش را نشان می‌دهد.
2. افزایش امنیت برنامه‌ها: کامپایلرها به بهبود امنیت برنامه‌ها کمک می‌کنند. فرآیند کامپایل شامل بهینه‌سازی کد و شناسایی آسیب‌پذیری‌های بالقوه است و در نتیجه خطر نقض امنیت را کاهش می‌دهد.
3. قابلیت‌های اشکال زدایی: کامپایلرها اغلب ابزارهای اشکال زدایی را ارائه می‌دهند که به شناسایی و اصلاح خطاهای کد کمک می‌کند. این ابزارها فرآیند اشکال‌زدایی را ساده کرده و پیدا کردن و رفع مشکلات را برای توسعه‌دهندگان آسان‌تر می‌کنند.

معایب کامپایلرها:

1. تشخیص خطای محدود: در حالی که کامپایلرها می‌توانند خطاهای نحوی و برخی از خطاهای معنایی را شناسایی کنند، ممکن است انواع خطاها یا مسائل منطقی را در کد تشخیص ندهند. برخی از خطاها ممکن است فقط در زمان اجرا آشکار شوند.
2. افزایش زمان کامپایل برای پایگاه‌های کد بزرگ: کامپایل کدهای حجیم می‌تواند زمان‌بر باشد. با افزایش اندازه پایگاه کد، فرآیند کامپایل ممکن است بیشتر طول بکشد و منجر به تأخیرهای احتمالی در چرخه توسعه شود.

زبان‌های برنامه نویسی کامپایلری

C,C++,C#,Java,Go

زبان‌های برنامه نویسی مفسری

Python,Javascript,PHP,Ruby

شباهت کامپایلر و مفسر چیست؟

با تعریفی که از کامپایلر و مفسر در قسمت بالا کردیم، متوجه شدیم که شباهت کامپایلر و مفسر در این است که هر دو یک نرم‌افزاری هستند که کدهای زبان برنامه نویسی سطح بالا را که برای انسان قابل فهم است، به کدهای زبان برنامه نویسی سطح پایین یعنی «زبان ماشین» تبدیل می‌کنند تا برای کامپیوتر قابل فهم باشند.

تفاوت کامپایلر و مفسر چیست؟

در زبان‌های کامپایلری قبل از اجرای برنامه، همه‌ی کدها به یکباره به «زبان ماشین» تبدیل می‌شوند. اما در زبان‌های مفسری موقع اجرای برنامه، کدها به‌صورت خط به خط به زبان ماشین ترجمه می‌شوند.
در زبان‌های کامپایلری بعد از کامپایل شدن یک فایل خروجی (مانند exe) ساخته می‌شود که به زبان ماشین ترجمه شده است و در دفعات بعدی نیازی به کامپایل و ترجمه دوباره کدها نیست. اما در زبان‌های مفسری هیچ فایل خروجی ساخته نمی‌شود و همیشه موقع اجرا، این کدها به‌صورت خط به خط به «زبان ماشین» ترجمه می‌شوند.

اگر خطایی در کد زبان‌های برنامه نویسی کامپایلری رخ بدهد، کامپایلر با خطای Compile error مواجه می‌شود و فایل خروجی ساخته نمی‌شود و نمی‌توان آن را اجرا کرد. اگر در کد زبان‌های برنامه نویسی مفسری خطایی رخ دهد، چون به‌صورت خط به خط این ترجمه انجام می‌شود، برنامه تا زمانی که خطایی در کد وجود نداشته باشد اجرا شده و موقع رسیدن به آن خط از کد که خطا وجود دارد، برنامه متوقف می‌شود.

سرعت اجرای برنامه در زبان‌های کامپایلری به دلیل اینکه خروجی کامپایل شده برای دفعات بعدی نیاز به ترجمه نیست، بالاتر از سرعت اجرای برنامه در زبان‌های مفسری است چون که همیشه موقع اجرا این ترجمه باید انجام شود و مقداری زمان‌بر است.

پیدا کردن باگ (Bug) در زبان‌های برنامه نویسی مفسری آسان‌تر از زبان‌های برنامه نویسی کامپایلری است، چون در زبان‌های مفسری به‌صورت خط به خط ترجمه انجام می‌شود و در هر خط از کد، خطا (Bug) وجود داشته باشد، برنامه متوقف شده و خطا در آن خط قابل مشاهده است اما در زبان‌های کامپایلری چون کل کدها کامپایل می‌شوند، موقع خطا، متوجه نمی‌شویم که در کدام یک از خطها باگ وجود دارد.

زبان مفسری وابسته به سیستم‌عامل نیست اما زبان کامپایلری به سیستم‌عامل وابسته است. در زبان مفسری کدها چون در هر اجرای برنامه ترجمه می‌شوند، در هر سیستم‌عاملی قابل اجرا هستند. اما خروجی زبان کامپایلری، در همان سیستم‌عاملی قابل اجرا است که در همانجا کامپایل شده است. اگر بخواهیم در سیستم‌عامل دیگری استفاده کنیم، باید در همان سیستم‌عامل، کدها دوباره کامپایل شوند.[2]

منابع

https://developersho.com/blog/compiler-vs-interpreter

https://maktabkhooneh.org/mag/difference-between-compiler-and-interpreter/