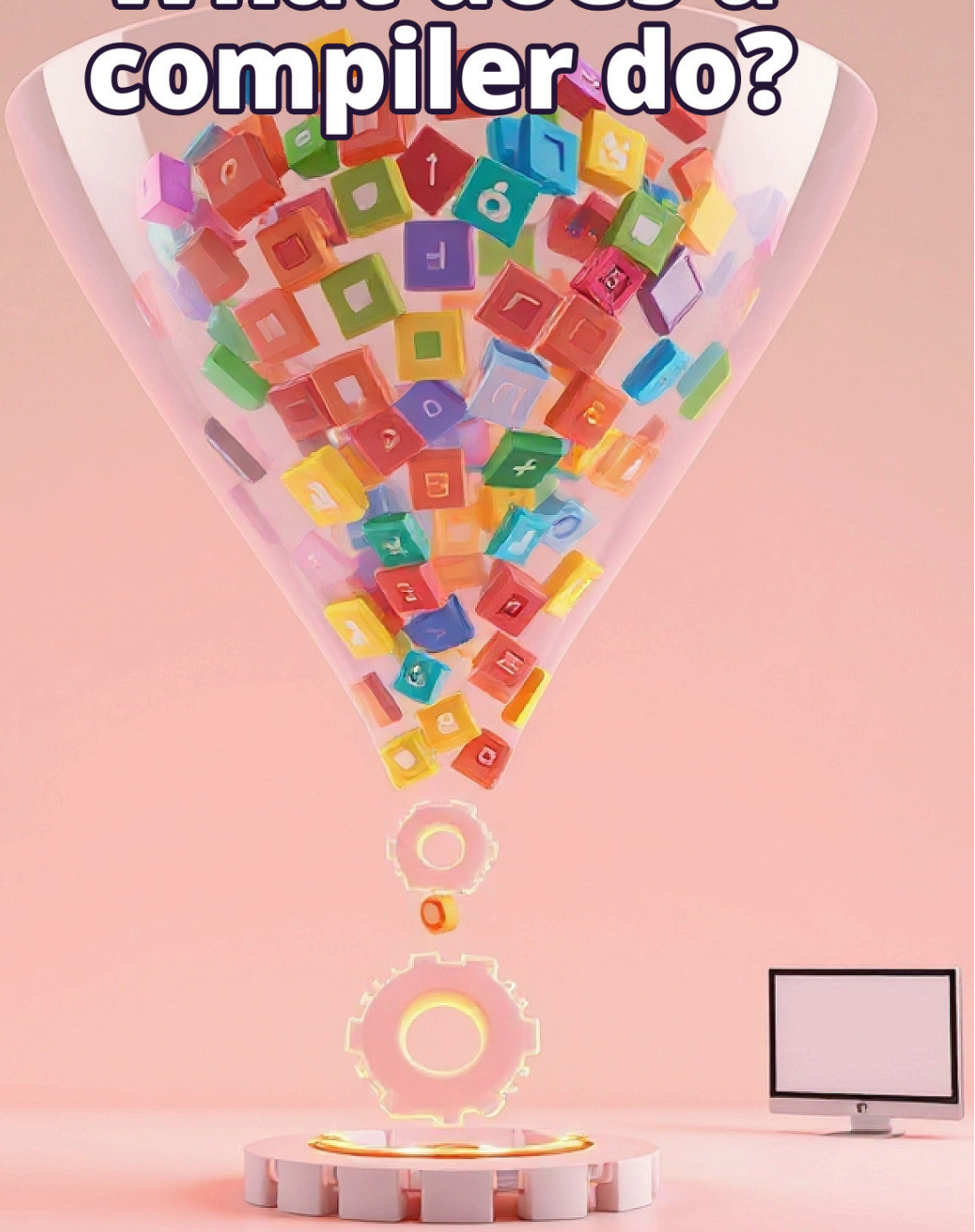
What does a compiler do?



www.yzokul.com.tr

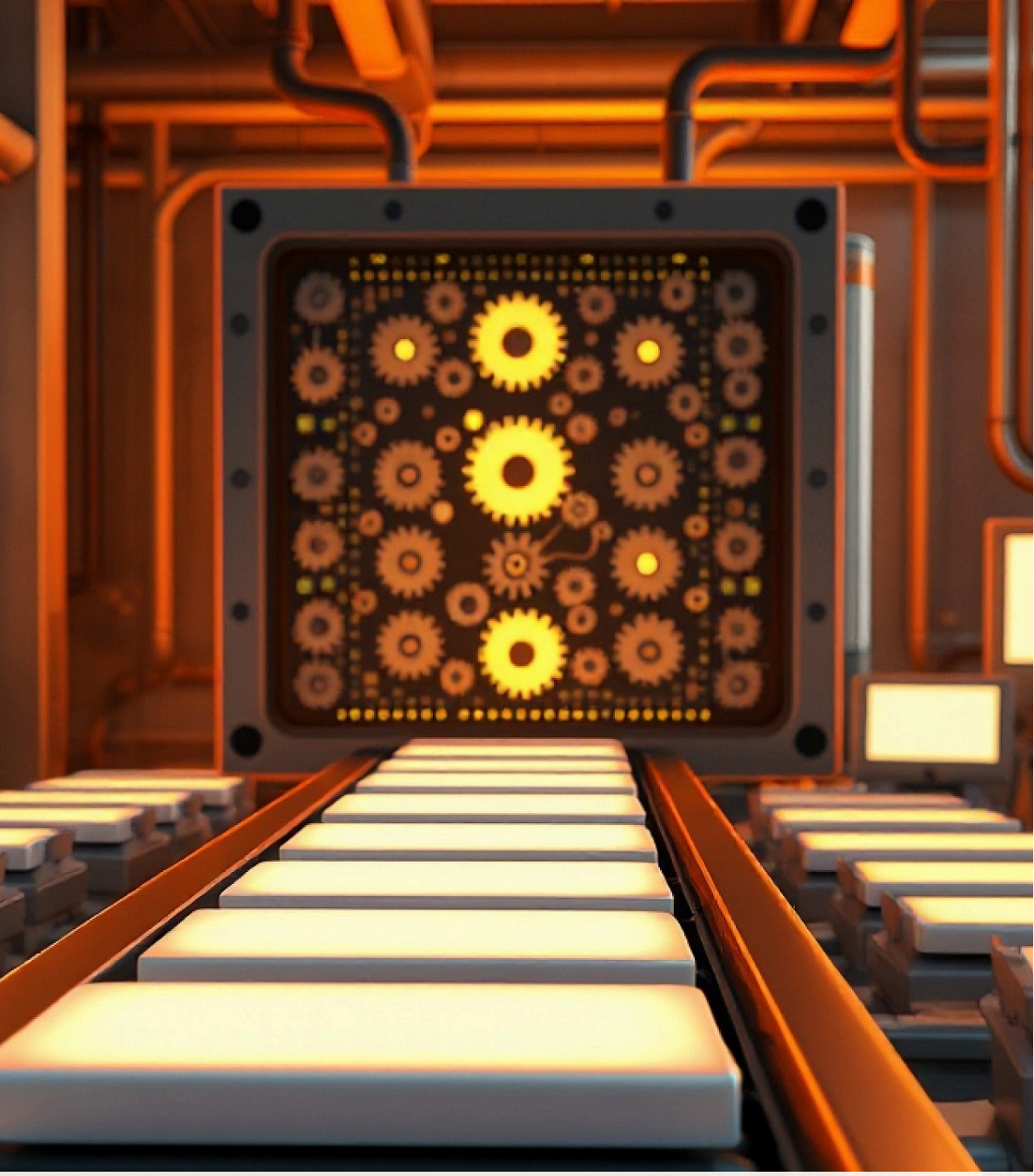
What does a compiler do?



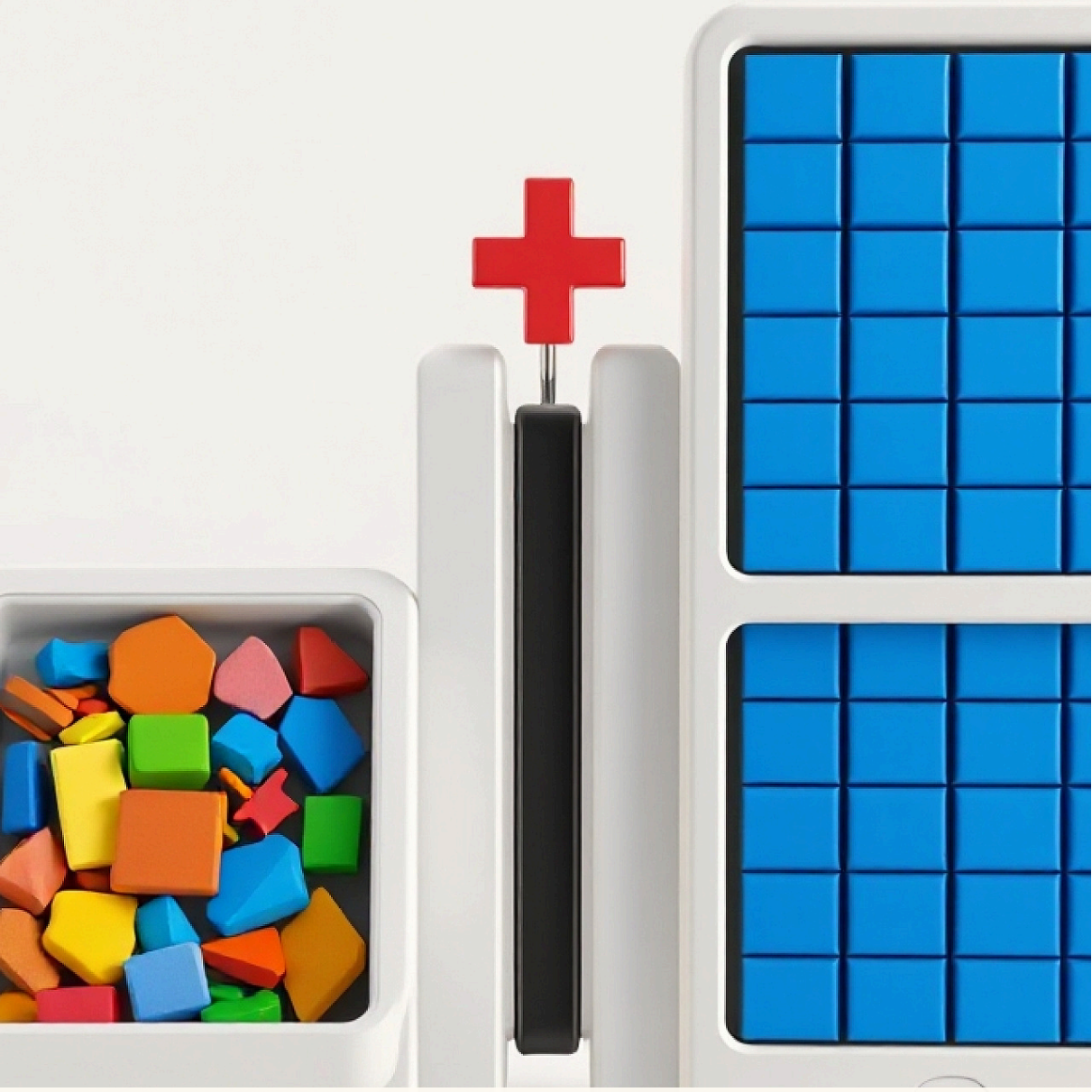
When you write a program, you write it in a programming language. A compiler is a program that translates that code into another form. The short answer is: a compiler converts code from one programming language into another representation, such as machine code, intermediate code, or even another programming language. This translation process also checks for certain errors.



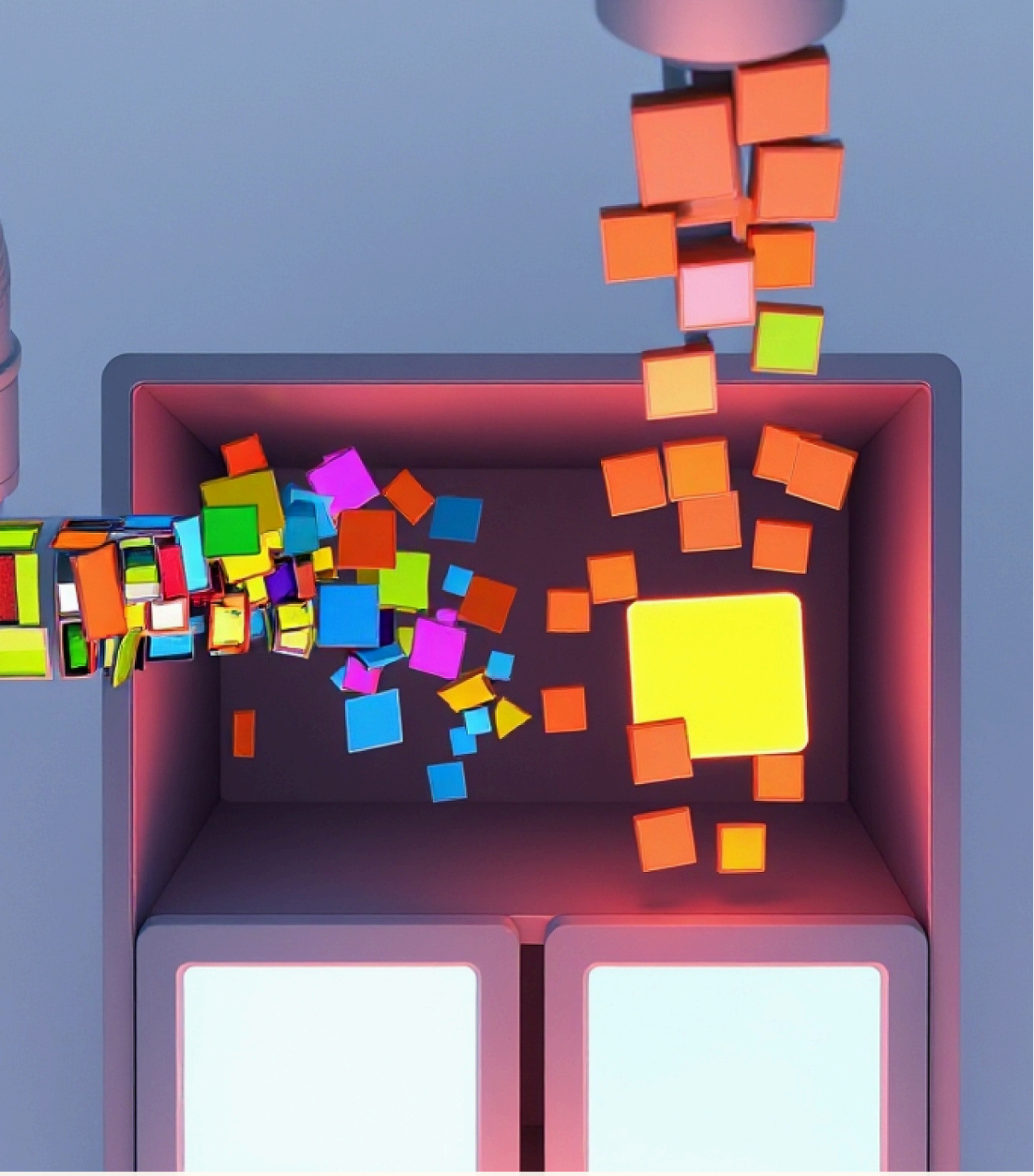
A program starts as plain text that you type into a file. This text is written in a programming language, like Python or C. The computer cannot run this text directly because it only understands its own machine instructions. The text must be turned into a form the computer can process. That is where a compiler comes in.



A computer follows instructions in a form it can process, called machine code. Machine code consists of numbers that tell the computer exactly what to do. Your written code is not in that form yet. Before the computer can run your program, your code must be translated into machine code or another executable form. A compiler performs this translation.



Think of a compiler as a translator between two forms of code. It reads your code, which is the input, and produces a new form, which is the output. It does not run your program; it only converts it. The compiler checks the code for errors during translation. If it finds errors, it reports them and does not produce the final output.



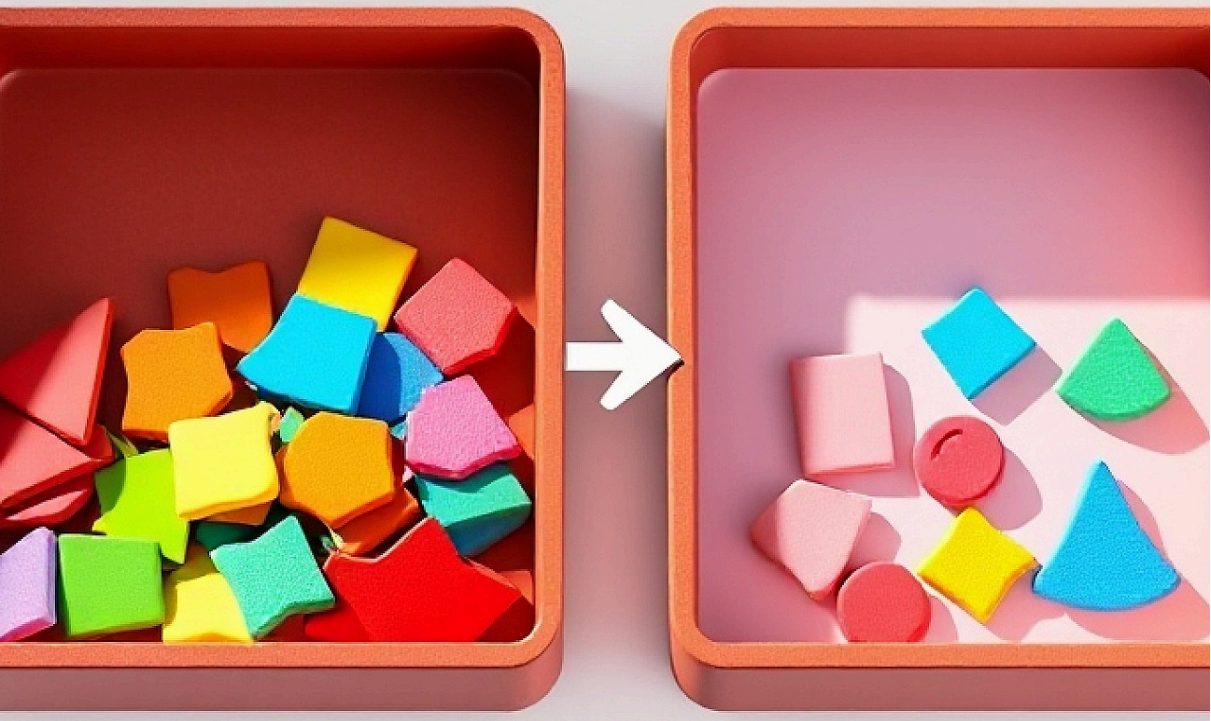
Picture the compiler as a box. On one side, your code goes in. On the other side, a new form comes out. The input is the code you wrote in a programming language. The output is the translated code, which could be machine code, intermediate code, or code in another language. The compiler does not execute the code; it only transforms it.



One possible output of a compiler is machine code. Machine code is the set of instructions that the computer's processor can execute directly. When the compiler produces machine code, the program is ready to run on that specific type of computer. This is the most direct kind of result because no further translation is needed. The machine code is often saved in an executable file.



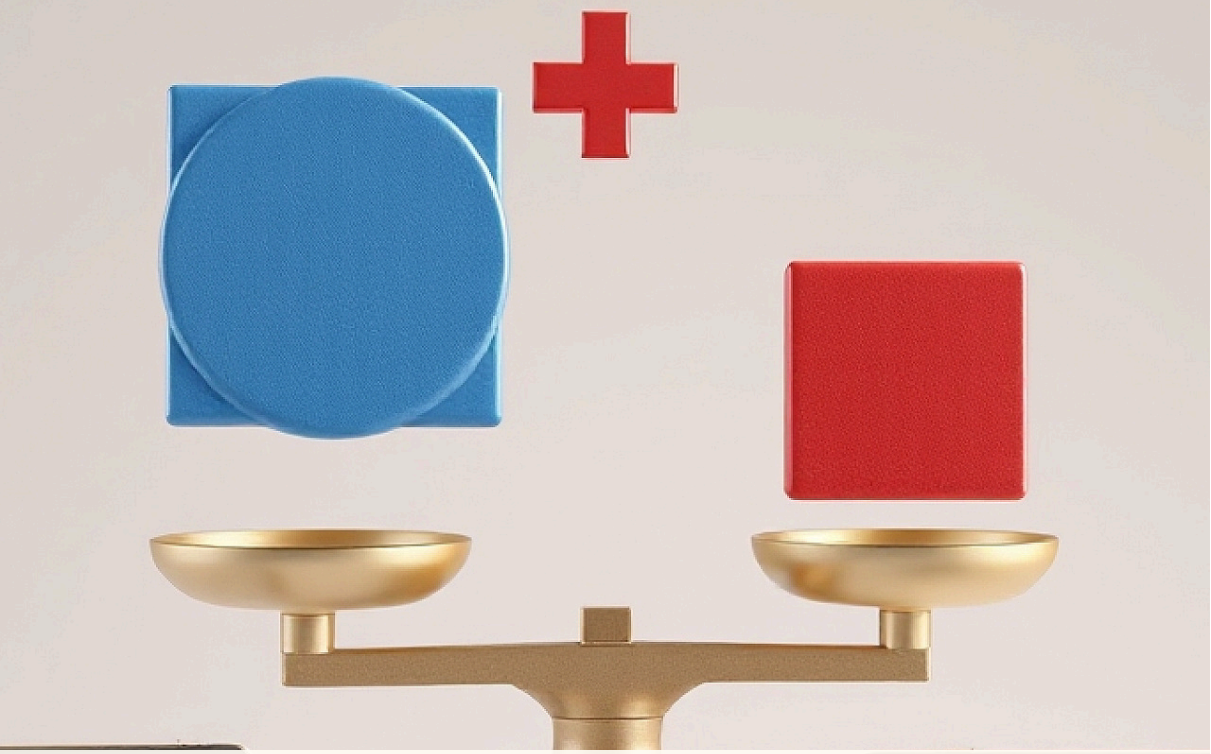
Another possible output is intermediate code. Intermediate code is a middle form between your source code and machine code. It is not the original programming language and not yet machine code. A virtual machine or a second compiler often translates intermediate code into machine code later. This approach can make programs easier to port to different computers.



A compiler can also produce code in another programming language. For example, a compiler might translate code from one language into another language like C or JavaScript. The output is still source code, just in a different language. This is useful when you want to use features of one language but run the program in an environment that supports another language.



The compiler checks your code for spelling errors, such as misspelled keywords or variable names. If you type 'prnit' instead of 'print', the compiler will notice that 'prnit' is not a recognized word. It reports these errors and stops. You must fix the spelling and compile again. This error detection helps you catch simple mistakes early.



The compiler also checks for type errors. A type error happens when you use a value with the wrong kind of data. For example, trying to add a number to a word might be a type error. The compiler knows the rules for combining different types. If you break those rules, it reports an error and does not produce output.



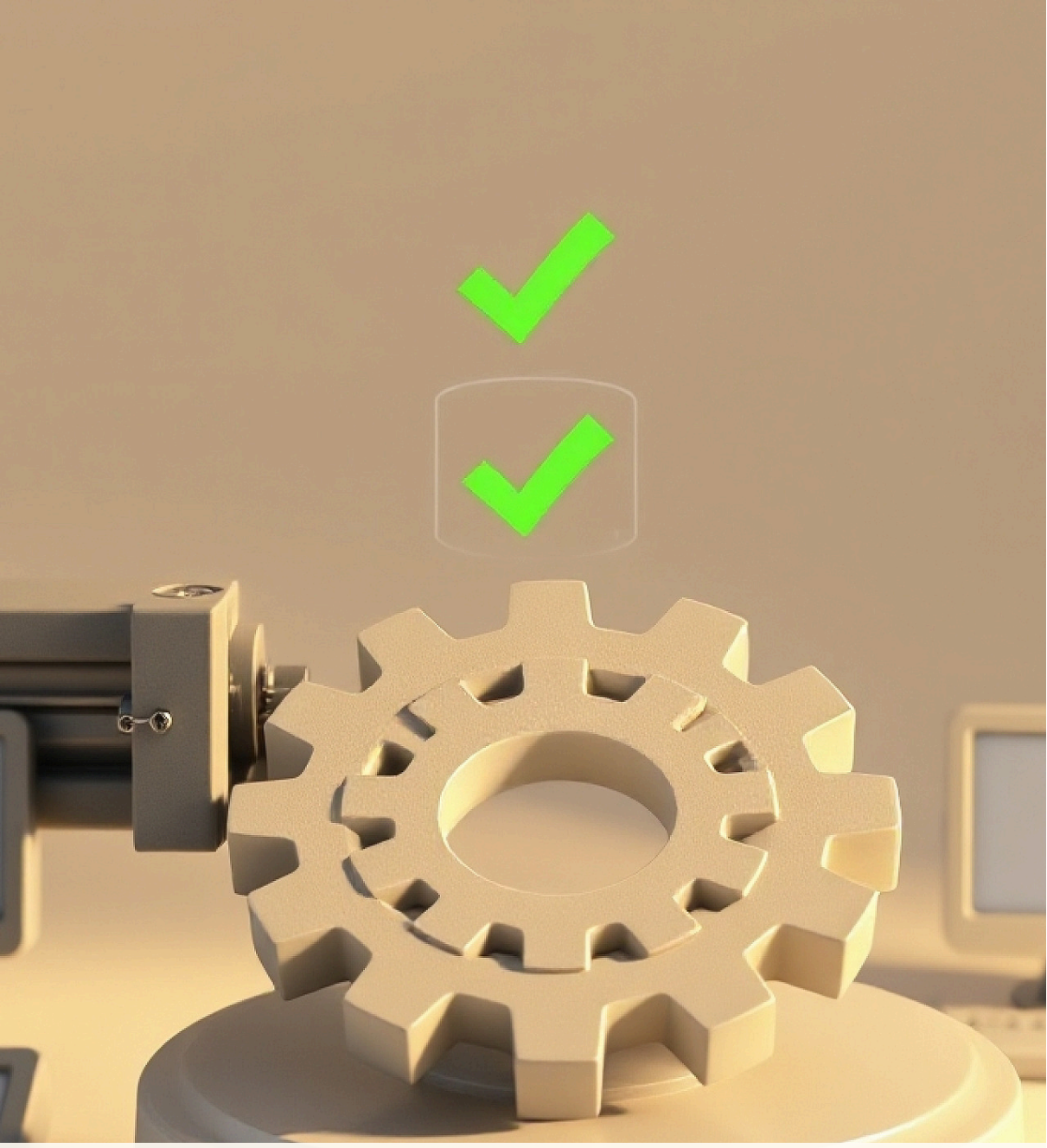
When the compiler finds errors, it does not finish its job. It stops and reports the errors to you. No output file is produced. You must fix the errors in your code and then compile again. This cycle repeats until the code has no errors that the compiler can detect. Only then does the compiler produce the final output.



Some compilers produce an executable file. An executable file is a file that you can run to start the program. On Windows, it often has the extension .exe. On other systems, it may have no extension or a different one. You create the executable file by compiling your code, and then you can run it by double-clicking or using a command.



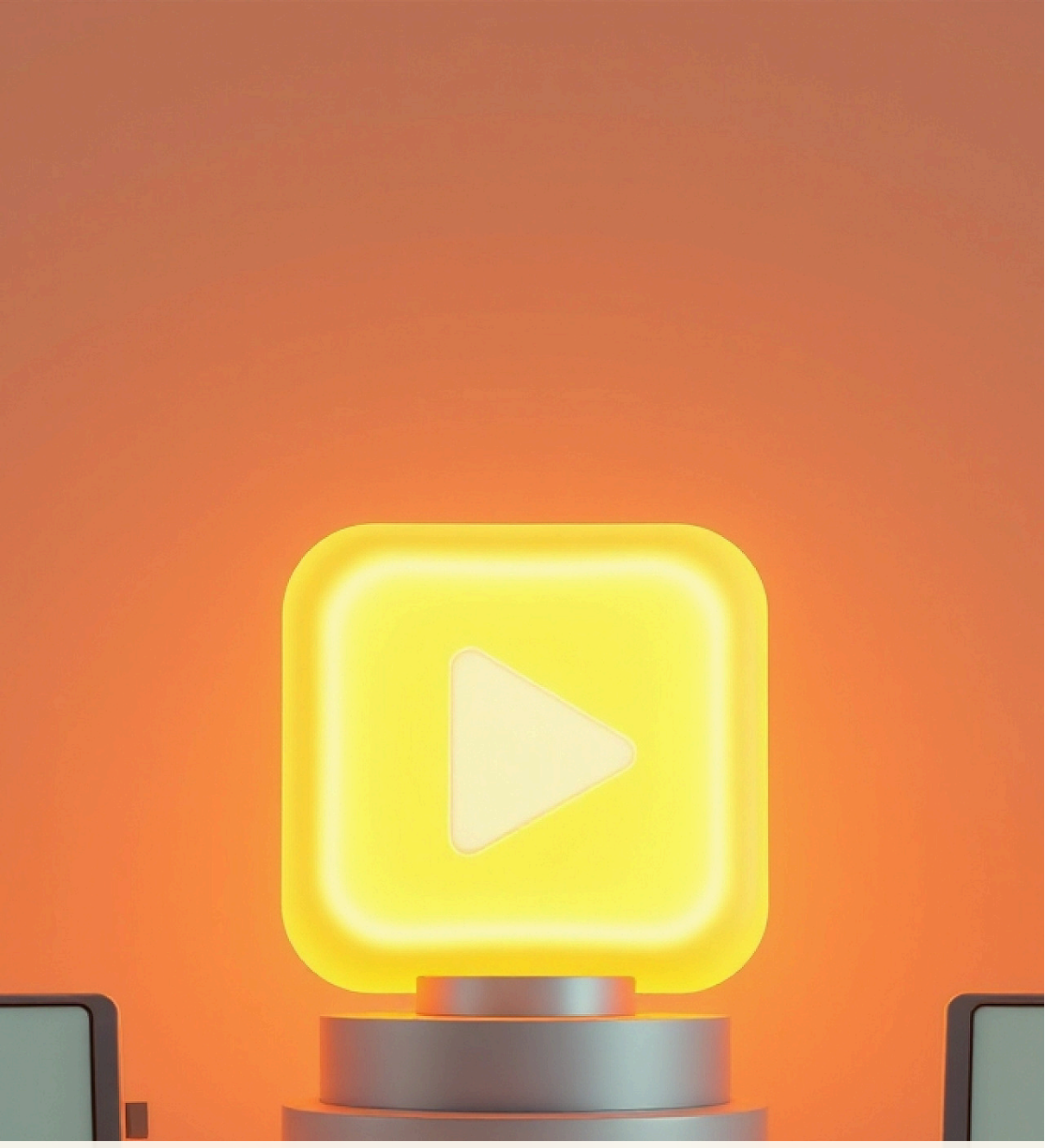
Compiling and running are two separate steps. The compiler prepares your program by translating it into another form. Running the program means executing it on the computer. You compile first, and then you run the result. The compiler does not run the program for you; it only produces the translated code or executable file. Running comes later.



The compiler checks the form of your code, not its meaning. For example, if you write code that adds two numbers when you meant to subtract, the compiler will not catch that. The code is syntactically correct and type-correct, so it compiles successfully. But the result will be wrong. The compiler does not know what you intended; it only follows the rules of the language.



A common mistake is to think that if a program compiles, it is correct. Compiling only means the code follows the grammar and type rules of the language. It does not mean the logic is right. A program can compile perfectly and still give wrong answers. Always test your program with different inputs to check that it behaves as expected.



Remember: a compiler translates code from a programming language into another form, such as machine code, intermediate code, or another language. It catches spelling and type errors and can produce an executable file. But it does not guarantee that the logic is correct. Code with wrong calculations can still compile. Always test your program after compiling.