1.2.2 Stages of Compilation

The compilation process can be divided into several stages. The key stages covered here are lexical analysis, syntax analysis and code generation.

Lexical Analysis

Lexical analysis breaks the source code into tokens and creates a symbol table containing information about identifiers.
Whitespace and comments are removed.
Information about variables, subroutines and other identifiers is stored in a symbol table.
The tokens and symbol table are passed to the syntax analysis stage.

The source code is broken into tokens, and whitespaces and comments are removed

Source Code

// calculate area
int width = 10;
int height = 20;
 	
int area;
 	
area = width * height;

→

Tokens		
Type	**Lexeme**	**Line**
KEYWORD	int	2
IDENTIFIER	width	2
OPERATOR	=	2
NUMBER	10	2
SEPARATOR	;	2
KEYWORD	int	3
…	…	…

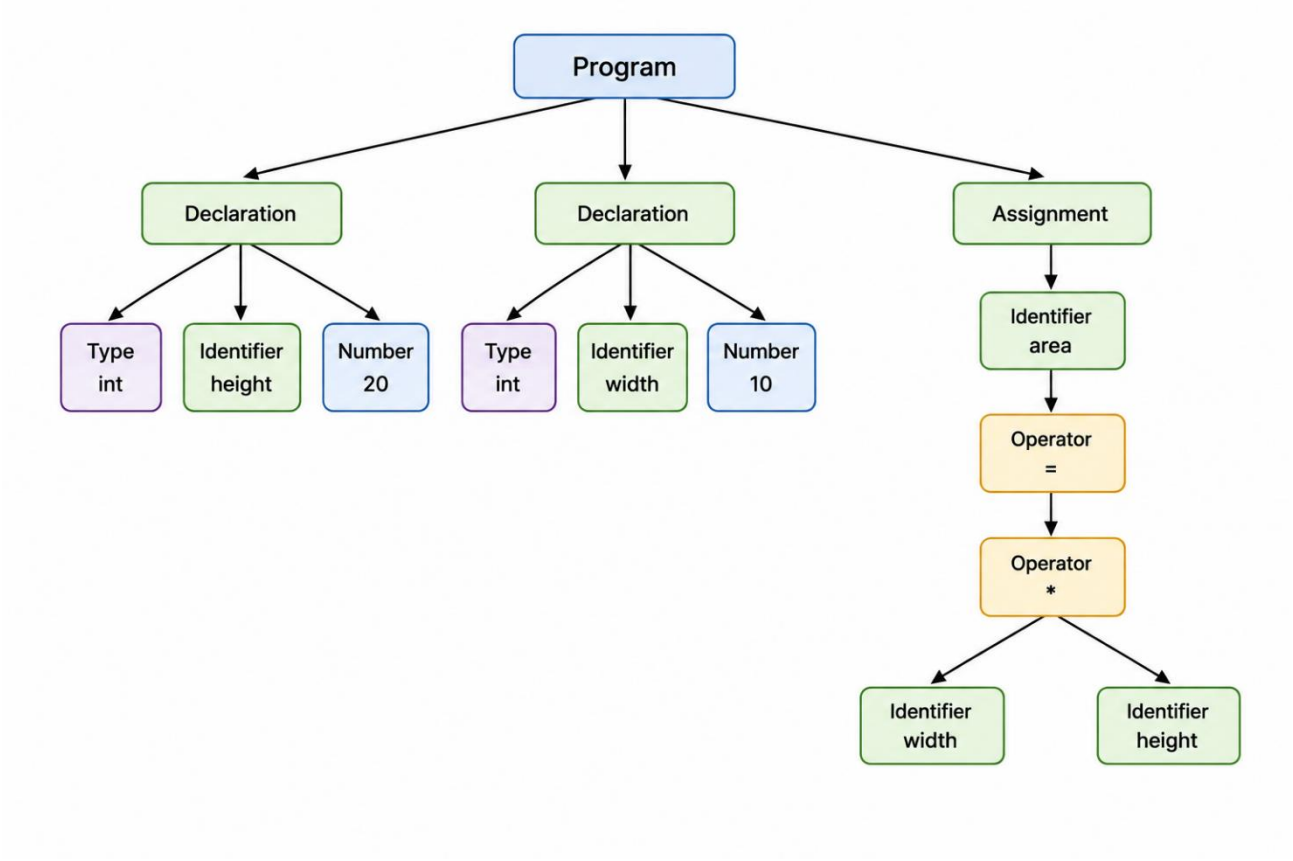
The symbol table stores information about identifiers

Symbol Table		
Name	**Type**	**Scope**
width	int	global
height	int	global
area	int	global

Syntax Analysis

Checks whether the sequence of tokens follows the grammar rules of the programming language. The tokens are arranged into an abstract syntax tree (AST), which represents the structure of the program.
A syntax error is generated if the code does not follow the rules of the language.
If all structural rules are followed pass on to the next stage, code generation, to be turned into machine code.

The tokens are arranged into an abstract syntax tree



Code Generation

The compiler generates object code from the processed source code so that it can be executed by the processor.

The generated code may then be optimised to improve performance, such as by reducing execution time or memory usage.