

## Citation paper:

### 1. Chatgpt prompt that helped me get the structure of the lexer code:

hey using typescript, can you help me make a compiler by writing a lexer that validates the input source code against the grammar found on our class web site at <https://www.labouseur.com/courses/compilers/grammar.pdf>. The source code will be read from a file for command-line compilers or an HTML text area element for browser-based compilers.

#### Answer:

```
class Token {
  constructor(public type: string, public value: string) { }
}

class Lexer {
  private sourceCode: string;
  private currentPosition: number;

  constructor(sourceCode: string) {
    this.sourceCode = sourceCode;
    this.currentPosition = 0;
  }

  private isEOF(): boolean {
    return this.currentPosition >= this.sourceCode.length;
  }

  private match(pattern: RegExp): string | null {
    const match = this.sourceCode.substr(this.currentPosition).match(pattern);

    if (match && match.index === 0) {
      const value = match[0];
      this.currentPosition += value.length;
      return value;
    }

    return null;
  }

  private skipWhitespace(): void {
    this.match(/^s+/);
  }

  private tokenizeNumber(): Token | null {
```

```

const value = this.match(/^d+/);

if (value !== null) {
    return new Token('NUMBER', value);
}

return null;
}

private tokenizeIdentifier(): Token | null {
    const value = this.match(/^a-zA-Z_\w*/);

    if (value !== null) {
        return new Token('IDENTIFIER', value);
    }

    return null;
}

private tokenizeOperator(): Token | null {
    const operators = ['+', '-', '*', '/', '=', '<', '>', '=='];

    for (const operator of operators) {
        const value = this.match(new RegExp(`^${operator}`));

        if (value !== null) {
            return new Token('OPERATOR', value);
        }
    }

    return null;
}

public tokenize(): Token[] {
    const tokens: Token[] = [];

    while (!this.isEOF()) {
        this.skipWhitespace();

        const token =
            this.tokenizeNumber() ||
            this.tokenizeIdentifier() ||
            this.tokenizeOperator();

```

```
    if (token === null) {  
        throw new Error(`Invalid token at position ${this.currentPosition}`);  
    }  
  
    tokens.push(token);  
}  
  
return tokens;  
}  
}  
  
// Example usage  
const sourceCode = "x = 42 + y";  
const lexer = new Lexer(sourceCode);  
const tokens = lexer.tokenize();  
  
console.log(tokens);
```

2. I also used a public repository of a Marist student that went to Marist in the past for assistance in debugging and creating new necessary functions that I could not figure out.

**Link to GitHub repo:** <https://github.com/liamhealy/CMPT432N-JSCompiler>