Міністерсто освіти і науки України
Івано-Франківський національний технічний університет нафти і газу

Кафедра КСМ

Лабораторна робота №4
Тема “Графи, дерева, списки, стьоки і черги (робота зі структурою даних транслятора) ”

Виконав студент
групи КІ-18-1
Марчук О. Р.

Перевірила
Кропивницька В. Б.

м.Івано-Франківськ
2021р.

Мета: Навчитися формувати та обробляти основні компонентні структури трансляторів (компіляторів): графи, дерева, списки, стьоки і черги при.

1. Завдання:

Згідно заданого варіанту (таблиця 1.1) виконати відповідні операції з файлом, ім’я якого задається в командному рядку, а результат зберегти у файлі із такою ж назвою, як у вхідного але розширенням ”.tmp”. Мова програмування С++ (чи інша за згодою викладача).

Таблиця 1.1 - Вихідні дані для виконання лабораторної роботи

Варіант	Завдання
10	Провести перевірку правильності розстановки дужок наступних типів: (), [], {} в заданому файлі.

2. Хід роботи

2.1 Розробляю програму згідно з завднанням(*рис. 2.1*). Основна сновна логіка винесена сервіс BracketService(*рис. 2.2*), а формування повідомлення помилки в BracketError(*рис. 2.3*).

```
1   const fs = require('fs');
2   const BracketService = require('./bracket-service')
3
4   const inputFile = process.argv[2];
5
6   fs.readFile(inputFile, 'utf8', (err, data) => {
7     if (err) {
8       console.error(err);
9     }
10    try {
11      BracketService.checkBrackets(data);
12    } catch (error) {
13      console.error('\x1b[1m%s', error.message);
14    }
15  });
16
```

Рисунок 2.1 - вміст файлу index.js

```
1   const BracketError = require('./bracket-error');
2       You, an hour ago • Finish lab3 program
    You, an hour ago | 1 author (You)
3   class BracketService {
4     checkBrackets(sourceString) {
5       const openedBracketsPairs = {
6         '(': ')',
7         '[': ']',
8         '{': '}',
9       };
10
11      const bracketsSequence = sourceString.match(/[\(\){}\[\]]/g);
12      let openedBlocks = [];
13      let errorBracketIndex = -1;
14
15      iFor: for (let i = 0; i < bracketsSequence.length; i++) {
16        if (bracketsSequence[i] in openedBracketsPairs) {
17          openedBlocks.push({ bracket: bracketsSequence[i], position: i });
18        } else {
19          for (let j = i - 1; j > -1; j--) {
20            if (openedBlocks.find((block) => block.position === j)) {
21              if (
22                openedBracketsPairs[bracketsSequence[j]] === bracketsSequence[i]
23              ) {
24                openedBlocks.pop();
25                break;
26              } else {
27                errorBracketIndex = i;
28                break iFor;
29              }
30            }
31          }
32        }
33      }
```

```
    return this.handleCheckResult({
      errorBracketIndex,
      openedBlocks,
      sourceString,
      bracketsSequence,
      openedBracketsPairs,
    });
  }

  findErrorBracketOrder(bracketsSequence, errorBracketIndex) {
    const errorBracketOrder = bracketsSequence
      .slice(0, errorBracketIndex + 1)
      .filter(
        (bracket) => bracket === bracketsSequence[errorBracketIndex]
      ).length;
    return errorBracketOrder;
  }

  findErrorLine(sourceString, ErrorBracket, errorBracketOrder) {
    const errorLine = sourceString
      .split('\n')
      .filter((line) => line.includes(ErrorBracket))[errorBracketOrder - 1];
    return errorLine;
  }

  findErrorLineIndex(sourceString, errorLine) {
    const errorLineIndex = sourceString.split('\n').indexOf(errorLine);
    return errorLineIndex;
  }

  handleCheckResult({
    errorBracketIndex,
    openedBlocks,
    sourceString,
    bracketsSequence,
    openedBracketsPairs,
  }) {
    if (errorBracketIndex === -1 && openedBlocks.length > 0) {
      const { bracket, position } = openedBlocks.pop();
      const errorBracketOrder = this.findErrorBracketOrder(
        bracketsSequence,
        position
      );
      const errorLine = this.findErrorLine(
        sourceString,
        bracket,
        errorBracketOrder
      );
      const errorLineIndex = this.findErrorLineIndex(sourceString, errorLine);

      throw BracketError.unclosedBracket({
        errorLine,
        errorLineIndex,
        unclosedBracket: bracket,
      });
    } else if (errorBracketIndex !== -1) {
      const errorBracketOrder = this.findErrorBracketOrder(
        bracketsSequence,
        errorBracketIndex
      );
```

```
95          const errorLine = this.findErrorLine(
96            sourceString,
97            bracketsSequence[errorBracketIndex],
98            errorBracketOrder
99          );
100         const errorLineIndex = this.findErrorLineIndex(sourceString, errorLine);
101         const errorBracket = bracketsSequence[errorBracketIndex];
102         const expectedBracket = openedBracketsPairs[openedBlocks.pop().bracket];
103
104         throw BracketError.unexpectedBracket({
105           errorLine,
106           errorLineIndex,
107           errorBracket,
108           expectedBracket,
109         });
110       }
111     }
112   }
113
114   module.exports = new BracketService()
115
```

Рисунок 2.2 - вміст файлу bracket-service.js

```
1   module.exports = class BracketError {
2     static unexpectedBracket({
3       errorLine,
4       errorLineIndex,
5       errorBracket,
6       expectedBracket,
7     }) {
8       const ErrorMessage = `unexpected '${errorBracket}' at line ${errorLineIndex}, expected '${expectedBracket}'
9       ${errorLine}`;
10      return new Error(ErrorMessage);
11    }
12
13    static unclosedBracket({ errorLine, errorLineIndex, unclosedBracket }) {
14      const ErrorMessage = `unclosed '${unclosedBracket}' at line ${errorLineIndex}
15      ${errorLine}`;
16      return new Error(ErrorMessage);
17    }
18  };
19
```

Рисунок 2.3 - вміст файлу bracket-service.js

2.2 Результат виконання:

Використовую текст програми на c++ з попередньої лабораторної, навмисно не закривше блок int main() (*рис. 2.4*). І отримую помилку про незакритий блок(*рис. 2.5*). Закриваю блок неправильною дужкою(*рис 2.6*), і отримую повідомлення помилки(*рис. 2.7*)

```
33      #ifdef VARIANT3
34        cout << "VARIANT3 really works" << endl;
35      #else
36        cout << "Hello World";
37      #endif
38    }
39
40    return 0;
41
```

Рисунок 2.4 - частина файлу app.cpp з незакритим блоком

```
oleh@MacBook-Pro-Oleh ~/Projects/system-software2/lab3 main ± npm start app.cpp

> lab3@1.0.0 start
> node index "app.cpp"

unclosed '{' at line 7
    {
```

Рисунок 2.5 - Результат виконання програми з помилкою про незакритий блок

```
33      #ifdef VARIANT3
34        cout << "VARIANT3 really works" << endl;
35      #else
36        cout << "Hello World";
37      #endif
38    )
39
40    return 0;
41  }
42
```

Рисунок 2.6 - частина файлу app.cpp з неправильною дужкою

```
oleh@MacBook-Pro-Oleh ~/Projects/system-software2/lab3 main ± npm start app.cpp

> lab3@1.0.0 start
> node index "app.cpp"

unexpected ')' at line 37, expected '}'
      )
```

Рисунок 2.5 - Результат виконання програми з помилкою про неочікувану дужку

Висновок: На цій лабораторній роботі я написав програму згідно завданням, але обійшовши основну мету роботи.