Лаба 3 варіант 12

Мій код реалізує лексичний аналізатор (лексер) для мови програмування Pascal. Лексичний аналізатор обробляє вхідний текстовий файл, що містить вихідний код на Pascal, і розбиває його на окремі лексеми (токени), кожна з яких має тип (наприклад, ключове слово, оператор, число тощо) і значення. Ці токени записуються в інший файл, де вони зберігаються разом з їхнім типом.

Опис роботи коду:

Структури даних:

- **`TokenType`** — перелік можливих типів токенів: ключові слова, ідентифікатори, числа, шістнадцяткові числа, рядки, символи, коментарі, оператори, розділові знаки, невідомі символи, помилки та кінець файлу.
- **`Token`** — структура, що містить тип токена (`TokenType`) і його значення (рядок).

Набори даних:

- **`keywords`** — набір ключових слів Pascal, таких як `begin`, `end`, `if`, `then` тощо.
- **`operators`** — набір операторів, таких як `+`, `-`, `*`, `/`, `:`, `=`, `<`, `>`, `%`, `^`.
- **`delimiters`** — набір розділових знаків, таких як круглі, квадратні, фігурні дужки, крапка з комою тощо.

Клас `Lexer`:

- Основний клас, що виконує лексичний аналіз. Має методи для обробки різних типів токенів: коментарів, директив препроцесора, рядкових або символьних констант, чисел, ідентифікаторів, ключових слів, операторів та розділових знаків.

Методи:

- `getNextToken()` — отримує наступний токен із вихідного коду. Визначає тип токена на основі символів і передає його для подальшої обробки.
- `skipWhitespace()` — пропускає пропуски і порожні символи.
- `peek()` — повертає наступний символ для перевірки без зсуву позиції.

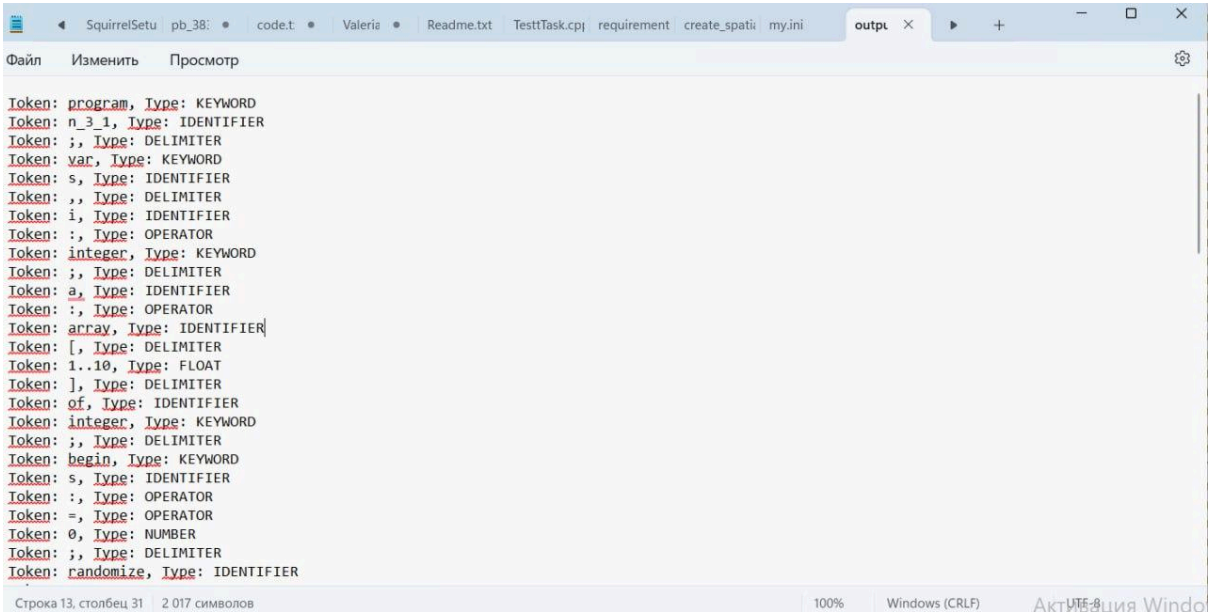
- `handleComment()` — обробляє коментарі (починаються з `{` або `(*`).
- `handlePreprocessor()` — обробляє директиви препроцесора (починаються з `#`).
- `handleStringOrChar()` — обробляє рядкові або символьні константи (між `'` або `"`).
- `handleNumber()` — обробляє числа (десяткові, числа з плаваючою крапкою, шістнадцяткові).
- `handleIdentifierOrKeyword()` — обробляє ідентифікатори або ключові слова.

Функція `tokenTypeToString(TokenType type)` — перетворює тип токена в текстовий формат для виведення.

Головна функція (`main`):

- Читає ім'я файлу від користувача і відкриває цей файл для зчитування вихідного коду Pascal.
- Якщо файл не вдалося відкрити, виводить повідомлення про помилку.
- Читає вміст файлу в змінну `sourceCode` і створює об'єкт лексера для аналізу цього коду.
- Послідовно отримує токени за допомогою лексера і записує їх у файл `output.txt`.
- Якщо файл для запису результатів не відкрився, також виводить помилку.
- По завершенню виводить повідомлення про успішне завершення аналізу.

Ось як виглядає результат:


SquirrelSetu
pb_38:
code.t
Valeria
Readme.txt
TesttTask.cp
requirement
create_spati
my.ini
outpu
Файл
Изменить
Просмотр
Token: program, Type: KEYWORD
Token: n_3_1, Type: IDENTIFIER
Token: ;, Type: DELIMITER
Token: var, Type: KEYWORD
Token: s, Type: IDENTIFIER
Token: ,, Type: DELIMITER
Token: i, Type: IDENTIFIER
Token: :, Type: OPERATOR
Token: integer, Type: KEYWORD
Token: ;, Type: DELIMITER
Token: a, Type: IDENTIFIER
Token: :, Type: OPERATOR
Token: array, Type: IDENTIFIER
Token: [, Type: DELIMITER
Token: 1..10, Type: FLOAT
Token:], Type: DELIMITER
Token: of, Type: IDENTIFIER
Token: integer, Type: KEYWORD
Token: ;, Type: DELIMITER
Token: begin, Type: KEYWORD
Token: s, Type: IDENTIFIER
Token: :, Type: OPERATOR
Token: =, Type: OPERATOR
Token: 0, Type: NUMBER
Token: ;, Type: DELIMITER
Token: randomize, Type: IDENTIFIER
Строка 13, столбец 31
2 017 символов
100%
Windows (CRLF)
UTF-8