Одеський національний політехнічний університет

Інститут комп’ютерних систем

Кафедра системного програмного забезпечення

Лабораторна робота №5

з дисципліни «Проектний практикум»

Виконала:

ст. гр. АС-142

Маковій Т.Ю.

Перевірив:

доц. Блажко О. А.

Одеса – 2018 рік

Було виконано модульне тестування методом **чорного ящику** функції формування переліку лексем, що була описана на схемі алгоритмів на рис. 1.

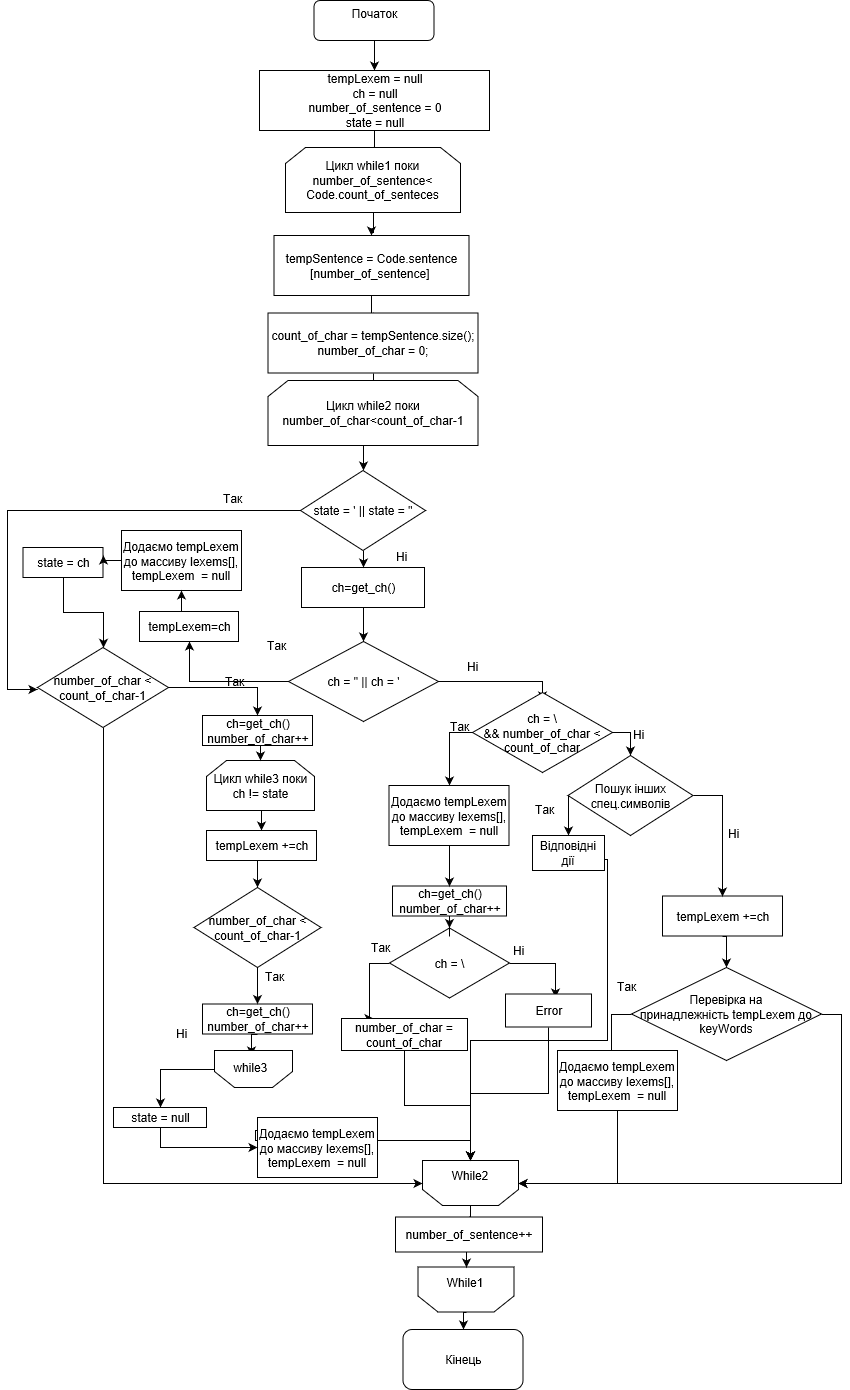


Рисунок 1 - Схема алгоритму формування переліку лексем

По-перше були виділені класи еквівалентності для функції, що тестується. Їх надано в таблиці 1.

Таблиця 1 - Класи еквівалентності для функції формування переліку лексем

|  |  |  |
| --- | --- | --- |
| Вхідна умова | Правильні класи  еквівалентності | Неправильні класи  еквівалентності |
| state== “ || state == ’  Стан аналізатора дорівнює одинарним або двійним лапкам | Шукаємо закриваючи лапки | Зчитуємо новий символ |
| ch== “ || ch == ’  Зчитаний символ дорівнює одинарним або двійним лапкам | Додаємо лапки до лексем. Зчитуємо слідуючий символ | Переходимо до наступної перевірки |
| Number\_of\_char < count\_of\_char-1  Номер зчитаного символу не перевищує їх кількість - 1 | Зчитаний символ – не останній | Цикл з символами завершується |
| Перевірка на приналежність tempLexem до keyWords  tempLexem належить до keyWords | Додаємо нову лексему | Зчитуємо новий символ |
| Цикл while1  Number\_of\_sentence >= Code.count\_of\_sentence | Цикл завершується | Цикл не переривається |
| Цикл while2  Number\_of\_char >= count\_of\_char | Цикл завершується | Цикл не переривається |
| Цикл while3  Ch = state | Цикл завершується | Цикл не переривається |
| Пошук інших спец символів  Символ є спецсимволом (‘=’, ‘+’, ‘-’, ‘/’, ‘\*’ та інші) | Додається нова лексема та спецсимвол до переліку лексем | Символ додається до tempLexem |

По-друге, пишуться тести для покриття еквівалентних класів. Тестові випадки описані в таблиці 2

Таблиця 2 - тестові випадки

|  |  |  |  |
| --- | --- | --- | --- |
| № | Вхідні дані | Очікуваний  результат | Результат тестування |
| 1 | Речення1 – string s = “hi  Речення 2 – ”; \\say hi | Заходимо в цикл while1, проходимо по циклу while2  Знаходиться ключове слово «string» та додається до переліку лексем. Продовжуємо пошук та знаходимо спецсимвол «=», додаємо до переліку лексем. Продовжуємо пошук та знаходимо подвійні лапки, додаємо їх до переліку, у циклі while3 заповнюємо tempLexem, переходимо на нове речення, та продовжуємо заповнення tempLexem при знайдені закриваючих лапок додаємо tempLexem до переліку лексем, заходимо спецсимвол «;», додаємо, знаходимо послідовність «\\», не заносимо її до лексем та завершуємо роботу з цим реченням. | В результаті отримуємо масив лексем  lexems[0] = String  lexems[1] = s  lexems[2] = =  lexems[3] = “  lexems[4] = hi  lexems[5] = ”  lexems[6] = ; |

Цими тестовими випадками були покриті всі правильні еквівалентні класи. Вхідних даних для отримання результатів неправильних еквівалентних класів не існує.

Також було виконано модульне тестування методом **білого ящика** функції додавання лексеми до переліку на рис. 2.

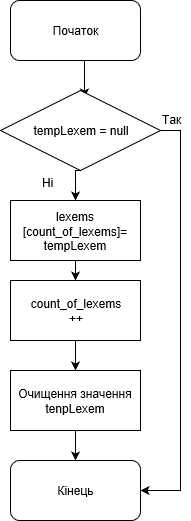


Рисунок 2 - Схема алгоритму додавання лексеми до переліку

void addLexemToMassive(){

if(tempLexem != null){//a

lexems[count\_of\_lexems] = tempLexem;//b

count\_of\_lexems ++;//c

tempLexem =null;//d

}

}

Для тестування використовувався метод тестування умов. В нашій функції є всього одна умова, яку можна покрити наступними тестами:

1. tempLexem = null --- a
2. tempLexem = “hi” --- a---b---c---d