

Структура згідно методичних рекомендацій	«Чекліст» компонентів, які обов'язково мають бути продемонстровані (позначені як [_]), а опціональні — як [X] (також усі блок-схеми алгоритмів)	Допоміжні матеріали (ВНС або назва теки на репозиторії), надані протягом семестру на практичних заняттях
ТИТУЛЬНА СТОРІНКА		
ЗАВДАННЯ НА КУРСОВЕ ПРОЄКТУВАННЯ	«Скріншот» варіанту завдання (а) та текстовий опис (б): а[_], б[_].	варіанти завдань (ВНС)
АНОТАЦІЯ		
ЗМІСТ		
ВСТУП		
1. ОГЛЯД МЕТОДІВ ТА СПОСОБІВ ПРОЄКТУВАННЯ ТРАНСЛЯТОРІВ		
2. ФОРМАЛЬНИЙ ОПИС ВХІДНОЇ МОВИ ПРОГРАМУВАННЯ 2.1. Деталізований опис вхідної мови в термінах розширеної нотації Бекуса-Наура 2.2. Опис термінальних символів та ключових слів	EBNF: [_]. Код (boost::spirit), що <u>пострічково відповідає EBNF</u> та коректно опрацьовує тестові програми: [_].	EBNFVerify+GrammarVerify EBNFVerify+GrammarVerify verify_syntax_by_EBNF__2025
3. РОЗРОБКА ТРАНСЛЯТОРА З ВХІДНОЇ МОВИ ПРОГРАМУВАННЯ		
3.1. Вибір технології програмування 3.2. Проектування таблиць транслятора та вибір структур даних		

<p>3.3. Розробка лексичного аналізатора</p> <p>3.3.1. Розробка блок-схеми алгоритму</p> <p>3.3.2. Опис програми реалізації лексичного аналізатора</p>	<p style="text-align: right;">Початковий етап:</p> <p style="text-align: right;">Регулярний вираз для токенизатора: [_].</p> <p style="text-align: right;">Регулярний вираз підмови ключових слів: [_].</p> <p style="text-align: right;">Регулярний вираз підмови ідентифікаторів: [_].</p> <p style="text-align: right;">Регулярний вираз підмови беззнакових літералів із фіксованою комою: [_].</p> <p style="text-align: right;">Код на основі регулярних виразів коректно опрацьовує тестові програми: [_].</p> <p style="text-align: right;">Безпосередній етап:</p> <p style="text-align: right;">Детальний опис процесу перетворення регулярних виразів у недетерміновані скінченні автомати (NFA): [X].</p> <p style="text-align: right;">Детальний опис процесу видалення λ-переходів для отриманих NFA: [X].</p> <p style="text-align: right;">Детальний опис процесу перетворення NFA (після видалення λ-переходів) у детермінований скінченний автомат (DFA): [X].</p>	<p>lexica__2025 lexica_part_dfa__2025</p> <p>lexica__2025 lexica_part_dfa__2025</p> <p>lexica__2025 lexica_part_dfa__2025</p> <p>lexica__2025 lexica_part_dfa__2025</p> <p>lexica__2025 lexica_part_dfa__2025</p> <p>lexica__2025 lexica_part_dfa__2025</p> <p>(KN1/)sp2019_soft__12345/ addon__09_11_2024/1</p> <p>(KN1/)sp2019_soft__12345/ addon__09_11_2024/2</p> <p>(KN1/)sp2019_soft__12345/ addon__09_11_2024/3</p>
---	--	---

	<p>Детальний опис процесу мінімізації (ключовий етап) DFA: [X].</p> <p>Результуючий етап:</p> <p>Повністю детермінований скінченний автомат для токенизації (а: двовимірний масив для програмного моделювання; б: орієнтований граф з усіма переходами, окрім тих, що ведуть у <u>«мертвий» стан, для графічного відображення</u>): a[_], б[_].</p> <p>Модель токенизатора на основі детермінованого скінченного автомату, оскільки це нетривіальне завдання теорії автоматів- акцепторів: [_].</p> <p>Повністю детермінований скінченний автомат для розпізнавання підмови ключових слів у вигляді таблиці переходів (а: двовимірний масив для програмного моделювання; б: орієнтований граф з усіма переходами, окрім тих, що ведуть у <u>«мертвий» стан, для графічного відображення</u>): a[_], б[_].</p> <p>Повністю детермінований скінченний автомат для розпізнавання підмови ідентифікаторів у вигляді таблиці переходів (а: двовимірний масив для програмного моделювання; б: орієнтований граф з усіма переходами, окрім тих, що ведуть у <u>«мертвий» стан, для графічного відображення</u>): a[_], б[_].</p> <p>Повністю детермінований скінченний автомат для розпізнавання підмови беззнакових літералів із фіксованою комою (а: двовимірний масив для програмного моделювання; б: орієнтований граф з усіма переходами, окрім тих, що ведуть у</p>	<p>(KN1/)sp2019_soft__12345/ addon__09_11_2024/4</p> <p>dfa_generator__2025</p> <p>(проаналізувати програмну модель у файлі matcher_by_dfa.hpp) bult_src</p> <p>dfa_generator__2025</p> <p>dfa_generator__2025</p>
--	---	---

	«мертвий» стан, для графічного відображення): a[_], б[_].	dfa_generator__2025
3.4. Розробка синтаксичного та семантичного аналізатора 3.4.1. Розробка дерев граматичного розбору 3.4.2. Розробка блок-схеми алгоритму 3.4.3. Опис програми реалізації синтаксичного та семантичного аналізатора	<i>Початковий етап посилається на попередній розділ (2.1), у якому формувалися EBNF та еквівалентний (пострічково) код для його перевірки.</i> Безпосередній етап: Синтез граматики відповідно до EBNF: [_]. Синтез множин FIRST та FOLLOW для отриманої граматики: [X]. Специфікація lookahead у правилах виведення граматики для моделі LL(2)-аналізатора: [_]. Форммування параметрів моделі LL(2)-аналізатора: [_]. Результуючий етап: Формування моделі LL(2)-аналізатора у вигляді недетермінованого автомата з магазинною пам'яттю (NPDA) (а: формальний опис; б: програмна модель): a[_], б[_]. Форммування моделі LL(2)-аналізатора у вигляді детермінованого автомату з магазинною пам'яттю (DPDA): [X].	EBNFVerify+GrammarVerify EBNFVerify+GrammarVerify EBNFVerify+GrammarVerify EBNFVerify+GrammarVerify dpda1_for_ll2_generator__2025 dpda1_for_ll2_generator__2025

3.5. Розробка генератора коду 3.5.1. Розробка блок-схеми алгоритму 3.5.2. Опис програми реалізації генератора коду	<p>У цьому розділі виконується опис:</p> <p>алгоритму формування польського інверсного запису [_],</p> <p>алгоритму виконання польського інверсного запису з використанням стеку [_],</p> <p><i>Сам код генерації універсальний. Додатково можна описати процес формування SSA та інші аспекти [X].</i></p>	cw_sp2__2025_2026/ src/implementation/preparer cw_sp2__2025_2026/ src/implementation/generator 28_11_2025 (+інші теки)
4. НАЛАГОДЖЕННЯ ТА ТЕСТУВАННЯ РОЗРОБЛЕНОГО ТРАНСЛЯТОРА 4.1. Опис інтерфейсу та інструкції користувачу 4.2. Виявлення лексичних і синтаксичних помилок 4.3. Перевірка роботи транслятора за допомогою тестових задач	<p>Описати консольний інтерфейс компілятора [_]</p> <p>Три тестові програми згідно варіанту [_]</p> <p>Коректна компіляція трьох тестових програм* [_]</p> <p><i>*у записці наводиться тестування лише трьох заданих тестових програм, але під час захисту потрібно вміти написати вашою мовою програмування будь-яку просту тестову програму</i></p>	cw_sp2__2025_2026/ src/implementation/cli base_test_programs_2025 cw_sp2__2025_2026 (запуск)
ВИСНОВКИ СПИСОК ЛІТЕРАТУРНИХ ДЖЕРЕЛ ДОДАТКИ А. Таблиці лексем для тестових прикладів Б. С код (або код на асемблері), отриманий на виході транслятора для тестових прикладів; Б. Документований текст програмних модулів (лістинги)	<p>Оскільки це курсовий проект (а не курсова робота), то окремо у форматі А3 (що має бути оформлено як стандартне креслення) подається один графічний матеріал. Це може бути приміром загальна блок-схема роботи транслятора:</p> <p>[_]</p>	