

ПРАКТИЧНА РОБОТА №1

Робота з формальними мовами та граматиками

Мета роботи. Закріпити теоретичні знання з теми «Формальні мови та граматики» шляхом програмної реалізації базових алгоритмів: побудови алфавіту, генерації слів, реалізації теоретико-множинних операцій над мовами, а також задання формальної граматики та побудови виведення слова разом з деревом виведення. Робота виконується самостійно на одній із мов програмування: Python, C# або C++.

Завдання. У межах практичної роботи необхідно реалізувати програмно такі завдання:

1. Створення алфавіту.
2. Генерація слів заданої довжини.
3. Реалізація операцій над мовами.
4. Задання граматики.
5. Побудова виведення слова.
6. Побудова дерева виведення.

Кожне із завдань має 15 варіантів. Номер варіанта відповідає порядковому номеру студента у списку групи; для всіх шести завдань використовується один і той самий номер варіанта.

Загальні вимоги до виконання

1. Мова програмування - на вибір студента: Python, C# або C++.
2. Програма повинна мати консольний інтерфейс (текстове меню або послідовний вивід результатів для кожного із завдань).
3. Вхідні дані (алфавіт, довжина слова n , мови L_1 та L_2 , граMATика, слово для виведення) можна задавати як константи у коді відповідно до свого варіанта - не обов'язково реалізовувати довільне введення користувачем, якщо це не передбачено ускладненням.
4. Код повинен бути розбитий на функції/методи (окрема функція чи клас для кожного із завдань, а не оформлений одним суцільним блоком).

5. Результати роботи кожного завдання повинні виводитися на екран у зрозумілому, підписаному вигляді (наприклад: «Завдання 3. Об'єднання $L1 \cup L2 = \{\dots\}$ »).

6. Забороняється використовувати готові сторонні бібліотеки, які повністю реалізують теорію формальних мов; дозволені лише стандартні бібліотеки мови (робота з рядками, множинами, колекціями).

Хід роботи

Завдання 1. Створення алфавіту

Реалізувати структуру даних для представлення алфавіту Σ (множина унікальних символів) та функцію перевірки належності символу алфавіту. Передбачити функцію перевірки, чи є довільний рядок словом над заданим алфавітом Σ (тобто чи складається він лише із символів Σ).

Вимоги:

- Алфавіт зберігати як множину символів, без дублікатів.
- Реалізувати функцію - перевірку належності рядка w множині слів над Σ .
- Передбачити коректну обробку порожнього рядка (порожнє слово ϵ) та порожнього алфавіту як межових випадків.

Варіанти :

№	Алфавіт для реалізації
1	$\Sigma = \{a, b\}$
2	$\Sigma = \{0, 1\}$
3	$\Sigma = \{a, b, c\}$
4	$\Sigma = \{x, y\}$
5	$\Sigma = \{0, 1, 2\}$
6	$\Sigma = \{p, q, r\}$
7	$\Sigma = \{a, b, c, d\}$
8	$\Sigma = \{0, 1, 2, 3\}$
9	$\Sigma = \{m, n\}$
10	$\Sigma = \{a, e, i, o, u\}$
11	$\Sigma = \{A, B, C\}$

12	$\Sigma = \{+, -, 1\}$
13	$\Sigma = \{ (,) \}$
14	$\Sigma = \{0, 1, \dots, 9\}$
15	$\Sigma = \{a, b, c, d, e\}$

Завдання 2. Генерація слів заданої довжини

Реалізувати функцію, яка генерує (перебором або рекурсивно) усі слова довжини n над алфавітом свого варіанта, що задовольняють задану умову (властивість мови). Довжину n задати як параметр (наприклад, $n = 3$ або $n = 4$ залежно від варіанта і швидкодії алгоритму).

Вимоги:

- Реалізувати генерацію без явного дублювання коду для різних n (рекурсивний або ітеративний генератор загального вигляду).
- Відфільтрувати згенеровані слова за умовою свого варіанта та вивести результат на екран.
- Порахувати і вивести загальну кількість слів, що задовольняють умову, для заданого n .

Варіанти:

№	Умова генерації слів	Приклад
1	усі слова довжини n , що починаються символом a	$n=3$: aab, aba, abb, ...
2	усі слова довжини n із парною кількістю одиниць	$n=4$: 0011, 0110, 1001, ...
3	усі слова довжини n без двох однакових символів поспіль	$n=3$: aba, bab, аса, ...
4	усі слова довжини n , що закінчуються символом u	$n=3$: хху, хуу, ухх, ...
5	усі слова довжини n , сума цифр яких кратна 3	$n=3$: 012, 021, 111, ...
6	усі слова довжини n , що містять символ r рівно один раз	$n=3$: pqr, qpr, rpq, ...
7	усі слова довжини n , які є паліндромами	$n=4$: abba, baab, ...
8	усі слова довжини n без підрядка «00»	$n=3$: 011, 101, 111, ...
9	усі слова довжини n , у яких кількість m дорівнює кількості n	$n=4$: mmnn, mnmn, ...
10	усі слова довжини n без двох однакових голосних	$n=2$: ae, io, ua, ...

	поспіль	
11	усі слова довжини n , що містять принаймні одну літеру С	$n=2$: AC, BC, CC, ...
12	усі слова довжини n вигляду запису знакового цілого числа	$n=3$: +11, -11 (лише ці два)
13	усі слова довжини n зі збалансованими дужками	$n=4$: (()), ()(), ...
14	усі слова довжини n , що є десятковим записом числа, кратного 5	$n=3$: 100, 105, 110, ...
15	усі слова довжини n , символи яких ідуть у неспадному алфавітному порядку	$n=3$: abc, abd, bcd, ...

Завдання 3. Реалізація операцій над мовами

Для двох заданих (скінченних) мов L_1 та L_2 свого варіанта реалізувати базові теоретико-множинні операції, а також одну додаткову (ускладнену) операцію, індивідуальну для варіанта.

Базові операції:

- об'єднання $L_1 \cup L_2$;
- перетин $L_1 \cap L_2$;
- різниця $L_1 \setminus L_2$;
- конкатенація $L_1 L_2 = \{ uv : u \in L_1, v \in L_2 \}$;
- доповнення $\bar{L}_1$ відносно множини всіх слів над Σ довжиною $\leq k$ (k обрати самостійно, наприклад $k = 3$).

Варіанти мови та додаткова операція:

№	L_1	L_2	Додаткова операція
1	$L_1 = \{a, ab, abb\}$	$L_2 = \{b, ab, bb\}$	симетрична різниця $L_1 \Delta L_2$
2	$L_1 = \{0, 01, 011\}$	$L_2 = \{1, 10, 110\}$	конкатенація $L_1 L_2$ з підрахунком $ L_1 L_2 $
3	$L_1 = \{a, ab, abc\}$	$L_2 = \{c, bc, abc\}$	квадрат мови L_1^2
4	$L_1 = \{x, xy, xyy\}$	$L_2 = \{y, yx, yyy\}$	L_1^* обмежена довжиною слів ≤ 4
5	$L_1 = \{0, 01, 012\}$	$L_2 = \{2, 21, 210\}$	L_1^+ обмежена довжиною слів ≤ 4

6	$L_1 = \{p, pq, pqr\}$	$L_2 = \{r, qr, pqr\}$	перевірка $L_1 \subseteq L_2$
7	$L_1 = \{a, ab, abc, abcd\}$	$L_2 = \{d, cd, bcd, abcd\}$	перевірка $L_1 \cap L_2 = \emptyset$
8	$L_1 = \{0, 01, 012, 0123\}$	$L_2 = \{3, 23, 123, 0123\}$	реверс усіх слів мови L_1 (L_1^R)
9	$L_1 = \{m, mn, mnm\}$	$L_2 = \{n, nm, nm n\}$	декартів добуток $L_1 \times L_2$ як мова пар « $u\#v$ »
10	$L_1 = \{a, ae, aei\}$	$L_2 = \{u, ou, oiu\}$	підрахунок $ L_1 \cup L_2 $ та $ L_1 \cap L_2 $
11	$L_1 = \{A, AB, ABC\}$	$L_2 = \{C, BC, ABC\}$	різниця $L_2 \setminus L_1$
12	$L_1 = \{+1, +11, +111\}$	$L_2 = \{-1, -11, -111\}$	перевірка $(L_1 L_2)^R = L_2^R L_1^R$
13	$L_1 = \{(), (()), ()()\}$	$L_2 = \{((())), ()(), ()()()\}$	перевірка комутативності $L_1 L_2 = L_2 L_1$
14	$L_1 = \{1, 12, 123\}$	$L_2 = \{9, 98, 987\}$	ступінь мови L_2^3
15	$L_1 = \{a, ab, abc\}$	$L_2 = \{e, de, cde\}$	симетрична різниця та підрахунок $ L_1 \Delta L_2 $

Завдання 4. Задання граматики

Для мови L свого варіанту побудувати формальну граматику $G = (V, T, P, S)$, яка породжує саме цю мову ($L(G) = L$), і реалізувати її представлення у програмі (наприклад, як список правил вигляду «ліва частина $\rightarrow$ права частина» або словник/структуру відповідних об'єктів).

Вимоги:

- Явно визначити у кодї множини V , T , початковий символ S та список правил P .
- Передбачити перевірку коректності побудованої граматики: множини V і T не перетинаються (їх спільна частина порожня), символ S належить множині V , а всі символи правих і лівих частин правил належать об'єднанню $V \cup T$.
- Вивести граматику на екран у зрозумілому вигляді (наприклад, по одному правилу в рядку: $S \rightarrow aSb$).

Варіанти (мова, для якої потрібно побудувати граматику)

№	Мова L , яку потрібно задати граматикою
1	$L = \{ a^n b^n : n \geq 1 \}$

2	$L = \{ w \in \{0,1\}^* : w \text{ починається і закінчується однаковим символом} \}$
3	$L = \{ w \in \{a,b,c\}^* : w \text{ містить рівно одну літеру } c \}$
4	$L = \{ x^n y^n : n \geq 0 \}$
5	$L = \{ w \in \{0,1,2\}^* : \text{символи } w \text{ йдуть у неспадному порядку} \}$
6	$L = \{ p^n q : n \geq 0 \}$
7	$L = \{ w \in \{a,b,c,d\}^* : w \text{ — паліндром парної довжини} \}$
8	$L = \{ w \in \{0,1,2,3\}^* : w \text{ починається символом } 0 \}$
9	$L = \{ m^n n^n : n \geq 1 \}$
10	$L = \{ w \in \{a,e,i,o,u\}^* : w \text{ парна} \}$
11	$L = \{ A^n B : n \geq 0 \}$
12	$L = \{ +1^n : n \geq 1 \} \cup \{ -1^n : n \geq 1 \}$
13	$L = \text{мова правильних дужкових послідовностей (мова Дика)}$
14	$L = \{ w \in \{0,\dots,9\}^* : \text{останній символ } w \text{ — парна цифра} \}$
15	$L = \{ a^n b^n c^m : n \geq 1, m \geq 1 \}$

Завдання 5. Побудова виведення слова

Використовуючи граматику, побудовану в завданні 4, реалізувати функцію, яка для заданого слова w свого варіанта знаходить і виводить послідовність кроків виведення $S \Rightarrow \gamma_1 \Rightarrow \gamma_2 \Rightarrow \dots \Rightarrow w$ (ліве виведення).

Вимоги:

- На кожному кроці виводити поточний ланцюжок символів і назву застосованого правила.
- Реалізувати виведення саме як ліве виведення (заміна найлівішого нетермінала на кожному кроці).
- Додатково (за бажанням, для підвищеної оцінки) реалізувати автоматичний пошук послідовності правил, що виводить задане слово, методом повного перебору або пошуку в ширину (BFS) по ланцюжках, а не лише відтворення заздалегідь відомого виведення.

Варіанти (15) - слово для виведення:

№	Слово w для побудови виведення
1	$w = aaabbb$

2	w = 0110
3	w = abcab
4	w = xxyy
5	w = 0122
6	w = ppq
7	w = abba
8	w = 0123
9	w = mmnn
10	w = aeio
11	w = AAB
12	w = +111
13	w = (())
14	w = 1234
15	w = aabbccc

Завдання 6. Побудова дерева виведення

Для того самого слова w і граматики свого варіанта (див. завдання 4–5) побудувати дерево виведення та вивести його структуру.

Вимоги:

- Реалізувати дерево виведення як структуру даних (вузол з міткою-символом і списком дочірніх вузлів).
- Вивести дерево у текстовому вигляді з відступами, що відображають рівні вкладеності (наприклад, кожен наступний рівень зсунутий на 2–4 пробіли або символ «`|`»).
- Перевірити, що послідовність листків дерева зліва направо збігається із заданим словом w .
- Додатково (за бажанням): згенерувати графічне зображення дерева — засобами Graphviz/matplotlib (Python), System.Drawing (C#) або власною ASCII-графікою (C++).

Варіант використовує той самий номер, граматику та слово, що й у завданнях 4 та 5 — окрема таблиця варіантів для цього завдання не потрібна.

Додаткові (підвищені) вимоги - ускладнення

Для отримання підвищеної оцінки студент можна виконати одну або декілька з наведених нижче ускладнених вимог:

- Генерація слів (завдання 2): замінити повний перебір усіх Σ^n слів із подальшою фільтрацією на рекурсивний генератор із відсіканням, який одразу будує лише слова, що задовольняють умову, без побудови «зайвих» варіантів.

- Операції над мовами (завдання 3): реалізувати мову як окремий клас/тип даних із перевантаженими операторами (наприклад, у Python - `__or__`, `__and__`, `__sub__` для об'єднання, перетину, різниці; у C# - перевантаження operator `|`, `&`, `-`; у C++ - перевантаження operator `+`, operator `*`).

- Граматика (завдання 4): реалізувати автоматичну перевірку коректності введеної користувачем граматики та повідомляти про конкретну помилку (наприклад, «символ 'X' не належить $V \cup T$ у правилі №3»).

- Виведення слова (завдання 5): реалізувати пошук виведення методом обходу в ширину (BFS) по всіх можливих застосуваннях правил, а не лише відтворення наперед заданого ланцюжка кроків.

- Дерево виведення (завдання 6): додати генерацію графічного зображення дерева (Graphviz для Python, аналогічні засоби для C#/C++) замість текстового відображення.

- Неоднозначність граматики: для граматик, що допускають декілька виведень одного слова (наприклад, варіант 12 - граматика знакових чисел, або довільна граматика арифметичних виразів на вибір), реалізувати пошук двох різних дерев виведення для одного й того самого слова і продемонструвати неоднозначність.

Контрольні питання

1. Чим формальний спосіб задання мови відрізняється від задання переліком слів?
2. Які обмеження має алфавіт Σ як математичний об'єкт?
3. Доведіть (на прикладі), що конкатенація слів у загальному випадку некомутативна.
4. Чим відрізняються операції об'єднання та конкатенації мов?
5. Що таке зірка Кліні L^* і чим вона відрізняється від L^+ ?
6. Поясніть призначення кожного з елементів четвірки $G = (V, T, P, S)$.
7. У чому різниця між лівим і правим виведенням? Чи завжди вони дають однакове дерево виведення?
8. Що означає неоднозначність граматики і чому це небажана властивість для мов програмування?
9. Як за деревом виведення відновити виведене слово?