Міністерство освіти і науки, молоді та спорту України

Прикарпатський національний університет
імені Василя Стефаника

Володимир Гаврилків

# Регулярні вирази у програмних продуктах

Навчальний посібник
для студентів напрямів підготовки
«математика», «прикладна математика»
вищих навчальних закладів

Івано-Франківськ

2012

УДК 004.423.24
ББК 22.183.492
Г 12

Гаврилків В.М. Регулярні вирази у програмних продуктах: навчальний посібник / В.М. Гаврилків. — Івано-Франківськ: «Сімик», 2012. — 72 с.

В посібнику розглянуто застосування регулярних виразів для пошуку та обробки тексту в програмах `grep`, `sed`, LibreOffice Writer та Total Commander. Наведено велику кількість ілюстративних прикладів, які пояснюють і розширюють теоретичний матеріал. Кожен розділ супроводжується питаннями та завданнями для самостійного розв'язання.

Рекомендовано Вченою радою факультету математики та інформатики як навчальний посібник для студентів напрямів підготовки «математика», «прикладна математика» (протокол № 1 від 6 вересня 2012 р.).

Рецензенти: доктор фізико-математичних наук, професор
**Протасов Ігор Володимирович**, провідний науковий співробітник кафедри дослідження операцій Київського національного університету імені Тараса Шевченка;
кандидат фізико-математичних наук, доцент
**Никифорчин Олег Ростиславович**, завідувач кафедри алгебри та геометрії Прикарпатського національного університету імені Василя Стефаника;
кандидат фізико-математичних наук
**Бандура Андрій Іванович**, доцент кафедри вищої математики Івано-Франківського національного технічного університету нафти і газу.

# Зміст

# Передмова

Даний посібник присвячений регулярним виразам – потужному засобу для пошуку та обробки тексту. З його допомогою читач навчиться застосовувати ці вирази на практиці та максимально задіювати можливості тих програм, в яких вони підтримуються.

Регулярні вирази використовуються багатьма програмами: текстовими редакторами, електронними таблицями, системними утилітами, базами даних, файловими менеджерами. Більшість популярних мов програмування (Perl, PHP, Java, Jscript, C, C++, C#, Visual Basic, VBScript, JavaScript, elisp, Python, Tel, Ruby, awk) підтримують регулярні вирази для роботи з рядками. Їх використання у настільки різноманітних програмах пояснюється в першу чергу тим, що вони володіють унікально багатими можливостями. Регулярні вирази по-різному реалізуються у програмах, але якщо зрозуміти їх загальну концепцію, то освоїти конкретну реалізацію буде не важко. Подана в цій книзі інформація допоможе здобути навички роботи з регулярними виразами.

Посібник вирізняється послідовним викладом тексту з великою кількістю ілюстративних прикладів, які пояснюють і розширюють теоретичний матеріал. Він складається з чотирьох розділів. Перший розділ містить виклад основних теоретичних понять, пов'язаних із регулярними виразами. У другому та третьому розділах розглядається застосування цих виразів для контекстного пошуку тексту з використанням програми-фільтра `grep` та обробки тексту з допомогою неінтерактивного потокового текстового редактора `sed` відповідно. Дані програми одними з перших сприяли популяризації поняття регулярних виразів. Останній розділ присвячений застосуванню регулярних виразів у текстовому редакторі LibreOffice Writer та файловому менеджері Total Commander. Кожен розділ супроводжується питаннями для контролю засвоєння та вправами, які є основою для проведення практичного чи лабораторного заняття з даної теми.

Розділ I

# Регулярні мови та регулярні вирази

В цьому розділі опишемо важливий клас формальних мов – регулярні мови і їх задання з допомогою регулярних виразів, а також розглянемо алгебраїчні закони для регулярних виразів. Ці закони багато в чому подібні до алгебраїчних законів арифметики, проте між ними є і суттєві відмінності. Поданий теоретичний матеріал є основою, на якій базуватиметься розв'язання конкретних практичних задач контекстного пошуку та обробки тексту у наступних розділах.

## 1. Формальні мови

Нехай $X$ – довільна множина, яку назвемо *алфавітом*, а її елементи – *буквами*. *Словом* в алфавіті $X$ називатимемо довільну скінченну послідовність букв. Через $|\omega|$ позначимо довжину слова $\omega$, тобто кількість букв у ньому. Зокрема, довжина порожнього слова $e$, яке не містить жодної букви, дорівнює нулю. *Формальною мовою* в алфавіті $X$ називається довільна множина слів в $X$. Оскільки формальна мова – це множина, то до формальних мов застосовні теоретико-множинні операції об'єднання, перетину, знаходження різниці і доповнення.

Нехай $L_1$ та $L_2$ – формальні мови в алфавітах $X_1$ та $X_2$ відповідно. *Конкатенацією (добутком) мов $L_1$ і $L_2$* називається формальна мова $L_1L_2 = \{\omega_1\omega_2 : \omega_1 \in L_1, \omega_2 \in L_2\}$ в алфавіті $X_1 \cup X_2$, яка складається з слів, які утворюються шляхом дописування до будь-якого слова мови $L_1$ довільного слова мови $L_2$.

*Ітерація (замикання Кліні) мови $L$*, яка позначається через $L^*$, визначається наступним чином:

1) $L^0 = \{e\}$,
2) $L^n = LL^{n-1}$ для $n \geq 1$,
3) $L^* = \cup_{n\geq 0} L^n$.

1.1. Приклад. Нехай $X = \{0, 1\}$ – алфавіт, який містить дві букви 0 та 1. Словами в даному алфавіті є довільні скінченні послідовності з 0 та 1. Наприклад, 100011100101 – слово довжини 12 в даному алфавіті, а $L = \{00, 1\}$ – формальна мова в алфавіті $X$, яка містить два слова – 00 та 1. Нехай формальна мова $M = \{001, e, 1\}$, тоді $L \cup M = \{00, 1, 001, e\}$, а $LM = \{00001, 00, 001, 1001, 1, 11\}$. Слід зауважити, що конкатенація формальних мов не є комутативною операцією. Дійсно, $ML = \{00100, 0011, 00, 1, 100, 11\} \neq LM$.

Знайдемо ітерацію мови $L$. Мова $L^0 = \{e\}$ містить єдине порожнє слово незалежно від самої мови $L$ (слід не плутати мову $\{e\}$ з порожньою мовою $\emptyset$, яка не містить жодного слова). $L^1 = L$ містить всі слова мови $L$. Таким чином, перші два члени в розкладі $L^*$ утворюють мову $\{e, 00, 1\}$. Далі утворимо мову $L^2 = LL = \{0000, 001, 100, 11\}$. Аналогічно $L^3 = LL^2$ є множиною слів, утворених трикратним вибором із двох слів мови $L$. Для обчислення $L^*$ необхідно знайти $L^i$ для кожного $i \in \mathbb{N} \cup \{0\}$ і об'єднати всі ці мови. Хоча кожна множина $L^i$ є скінченною, об'єднання нескінченної кількості множин, як правило, утворює нескінченну мову. Це, зокрема, стосується і мови $L$, яка містить нескінченну кількість слів, які не мають підслів з непарною максимальною кількістю послідовних нулів.

1.2. Приклад. Покажемо, що формальна мова $\{0, 10\}^*$ складається з усіх слів, які не містять підслова 11 і закінчуються на 0.

Очевидно, що конкатенація довільної кількості 0 та 10 закінчується на 0. Вона не може породити підслово 11, оскільки останні букви обох слів 0 та 10 розділяють довільні дві 1 в їхній конкатенації.

Нехай $\omega$ – слово в алфавіті $\{0, 1\}$, яке не містить підслів 11 і закінчується на 0. Якщо $\omega$ не містить входжень 1, то $\omega$ є конкатенацією $|\omega|$ букв 0, і, отже, $\omega \in \{0, 10\}^{|\omega|} \subset \{0, 10\}^*$. Нехай слово $\omega$ містить $n \geq 1$ входжень букв 1. Тоді після кожного входження букви 1 в $\omega$ мусить бути буква 0, бо інакше буква 1 або слідує за 1, або є останньою буквою слова $\omega$, що суперечить вибору слова $\omega$. Таким чином, слово $\omega$ запишеться як

$$0 \ldots 0(10)0 \ldots 0(10)0 \ldots 0(10)0 \ldots 0,$$

де вираз $0 \dots 0$ означає нуль або більше входжень букви 0. Отже, $\omega$ є конкатенацією слів 0 та 10, тобто $\omega \in \{0, 10\}^*$.

## 2. Регулярні мови

Визначимо *регулярну мову (множину) в алфавіті $X$* рекурсивно наступним чином:

1) порожня множина $\emptyset$ є регулярною мовою;
2) для кожної букви $a \in X$ множина $\{a\}$ є регулярною мовою;
3) якщо $L_1$ і $L_2$ є регулярними мовами, то множини $L_1 \cup L_2$, $L_1 L_2$ і $L_1^*$ також є регулярними мовами;
4) інших регулярних мов не існує.

2.1. Приклад. Безпосередньо з означення випливає:

1) множина $\{e\}$ є регулярною мовою, бо $\{e\} = \emptyset^*$;
2) множина $\{001, 110\}$ – регулярна мова над бінарним алфавітом, бо $\{001, 110\} = (\{0\}\{0\}\{1\}) \cup (\{1\}\{1\}\{0\})$;
3) аналогічно, як і в пункті 2), можна показати, що будь-яка скінченна мова є регулярною.

Щоб спростити запис регулярних мов, визначимо поняття *регулярного виразу над алфавітом $X$* наступним чином:

1) порожня множина $\emptyset$ є регулярним виразом, який позначає порожню множину;
2) $e$ є регулярним виразом, який позначає мову $\{e\}$;
3) $a$ – регулярний вираз, що позначає мову $\{a\}$ для кожної букви $a \in X$;
4) якщо $r_A$ і $r_B$ є регулярними виразами, що позначають мови $A$ і $B$ відповідно, то $(r_A)|(r_B)$, $(r_A)(r_B)$ і $(r_A)^*$ є регулярними виразами, що позначають мови $A \cup B$, $AB$ і $A^*$ відповідно;
5) інших регулярних виразів над алфавітом $X$ не існує.

Якщо $r$ – регулярний вираз, то через $L(r)$ позначатимемо відповідну йому регулярну мову. Кажемо, що *слово $\omega$ задане регулярним виразом $r$*, якщо $\omega$ належить регулярній мові $L(r)$. Щоб зменшити кількість дужок в регулярних виразах, будемо вважати, що найвищий пріоритет має операція ітерації $*$, потім виконується конкатенація і останньою – операція об'єднання $|$. Наприклад, запис регулярного виразу $((0)(1))|((1)(0))$, який позначає мову $\{01, 10\}$, спрощується до $01|10$, вираз $0^*$ позначає $\{0\}^*$, $(0|1)^*$ – $\{0, 1\}^*$, а

$(0|1)^*001$ позначає множину всіх слів, які складаються з нулів і одиниць і закінчуються словом 001.

Зрозуміло, що для кожної регулярної мови можна знайти принаймні один регулярний вираз, який її позначає. І навпаки, для кожного регулярного виразу можна побудувати регулярну мову, що записується цим виразом. На жаль, для регулярної мови існує багато виразів. Наприклад, вирази $0^*1|\emptyset$ і $0^*1$ позначають одну і ту ж мову $\{0\}^*\{1\}$. Кажемо, що два регулярних вирази рівні ($=$), якщо вони позначають одну і ту ж регулярну мову.

2.2. Твердження. *Нехай $\alpha$, $\beta$ і $\gamma$ – регулярні вирази. Тоді:*

- $\alpha|\beta = \beta|\alpha$;
- $\emptyset^* = e$;
- $\alpha|(\beta|\gamma) = (\alpha|\beta)|\gamma$;
- $\alpha(\beta\gamma) = (\alpha\beta)\gamma$;
- $\alpha(\beta|\gamma) = \alpha\beta|\alpha\gamma$;
- $(\alpha|\beta)\gamma = \alpha\gamma|\beta\gamma$;
- $\alpha e = e\alpha = \alpha$;
- $\emptyset\alpha = \alpha\emptyset = \emptyset$;
- $\alpha^* = \alpha|\alpha^*$;
- $(\alpha^*)^* = \alpha^*$;
- $\alpha|\alpha = \alpha$;
- $\alpha|\emptyset = \alpha$.

Доведення. Нехай $\alpha$ і $\beta$ позначають мови $A$ і $B$ відповідно. Тоді $\alpha|\beta$ позначає $A \cup B$, а $\beta|\alpha$ позначає $B \cup A$. Але $A \cup B = B \cup A$ за означенням об'єднання, тому $\alpha|\beta = \beta|\alpha$. Решта рівностей доводиться аналогічно. □

Для зручності запису введемо позначення: $r^+ = rr^* = r^*r$.

Розглянемо деякі приклади.

2.3. Приклад. Довести рівності:

а) $a^*(a|b)^* = (a|ba^*)^*$;
б) $(ba)^+(a^*b^*|a^*) = (ba)^*ba^+b^*$.

а) Покажемо, що обидві частини рівності дорівнюють $(a|b)^*$. Дійсно, обидві частини рівності є підмножинами $(a|b)^*$, бо $(a|b)^*$

позначає множину всіх слів в алфавіті $\{a, b\}$. Таким чином, достатньо показати, що обидві частини містять $(a|b)^*$. Оскільки $e \in a^*$, то $a^*(a|b)^* \supset (a|b)^*$. З того, що $b \in ba^*$ випливає потрібне включення $(a|b)^* \subset (a|ba^*)^*$.

б) $(ba)^+(a^*b^*|a^*)=(ba)^*(ba)a^*(b^*|e)=(ba)^*ba^+b^*$.

2.4. Приклад. Регулярний вираз

$$(+38|e)0(67|9(6|7|8))(0|1|2|3|4|5|6|7|8|9)^7$$

позначає множину всіх номерів мобільних телефонів абонентів „Київстар”, а регулярний вираз

$$1((0|1|2|3|4)(0|1|2|3|4|5|6|7|8|9)|5(0|1|2))$$

– множину цілих чисел відрізка $[100, 152]$.

2.5. Приклад. Знайти регулярний вираз для множини всіх слів в алфавіті $\{0, 1, 2\}$, які є записами в трійковій системі чисел невід'ємних цілих степенів числа 9.

В трійковій системі числення степені $9^n$ запишуться як $1\underbrace{00\ldots0}_{2n}$. Таким чином, шуканий регулярний вираз має вигляд $1(00)^*$.

2.6. Приклад. Записати регулярний вираз для множини всіх слів у двійковому алфавіті, які містять підслово 001.

Кожне таке слово запишеться у вигляді $\alpha 001\beta$, де $\alpha$ і $\beta$ – довільні слова в двійковому алфавіті. Таким чином, одержуємо шуканий регулярний вираз:

$$(0|1)^*001(0|1)^*.$$

2.7. Приклад. Записати регулярний вираз для множини всіх слів у двійковому алфавіті, які не містять підслова 001.

Якщо слово $\omega$ належить шуканій множині, то воно не містить підслова 00, за винятком того випадку, коли воно має закінчення $0^k$ для $k \geq 2$. Аналогічно, як і у прикладі 1.2, можна показати, що множина слів, які не містять підслова 00, записується з допомогою регулярного виразу $(01|1)^*(e|0)$. Таким чином, множина всіх слів, які не містять підслова 001, записується наступним регулярним виразом:

$$(01|1)^*(e|0|000^*) = (01|1)^*0^*.$$

## Питання та вправи до розділу 1.

**1.1.** Які операції над формальними мовами вам відомі?

**1.2.** Дайте означення регулярної мови.

**1.3.** Чи може мова $L^*$ або $L^+ = LL^*$ бути порожньою? За яких умов мови $L^*$ і $L^+$ є скінченними?

**1.4.** Нехай $A = \{\text{пра}, e\}$, $B = \{\text{дід}, \text{баба}\}$. Знайти $AB$ і $A^*B$.

**1.5.** Знайти слово найменшої довжини в алфавіті $\{0\}$, яке не належить формальній мові $\{e, 0, 0^2, 0^5\}^3$.

**1.6.** Знайти слово найменшої довжини в кожній з наступних регулярних мов, записаній регулярним виразом. Які непорожні слова є найкоротшими в даних мовах?

а) $10|(0|11)0^*1$;
б) $(00|11|(01|10)(00|11)^*(01|10))^*$;
в) $((00|11)^*|(001|110)^*)^*$.

**1.7.** Визначити, чи належить рядок 1011 регулярним мовам, записаним регулярними виразами:

а) $10^*1$;
б) $1(01)^*1^*$;
в) $(10)^+11$;
г) $(10)^+1011$;
д) $1(00)^*(11)^*$;
е) $(1|00)(01|0)1^*$.

**1.8.** Чи є мова $\{0^{3m+4n} \mid m, n \geq 0\}$ регулярною? Відповідь обґрунтуйте.

**1.9.** Спростити наступні регулярні вирази:

а) $(00)^*0|(00)^*$;
б) $(0|1)(e|00)^+|(0|1)$;
в) $(0|e)0^*1$.

**1.10.** Доведіть рівність $(0^2|0^3)^* = (0^20^*)^*$.

**1.11.** Побудувати регулярний вираз для формальної мови всіх слів довжини менше шести в алфавіті $\{0, 1, m, n\}$, які починаються з букв m або n.

**1.12.** Визначити алфавіт та побудувати регулярний вираз для наступних мов:

а) множина всіх непарних натуральних чисел, записаних в четвірковій системі числення;
б) множина всіх непарних натуральних чисел, записаних в трійковій системі числення.

**1.13.** Побудувати регулярний вираз для множини всіх цілих чисел з відрізка $[100, 2012]$.

**1.14.** Побудувати регулярні вирази для мови всіх слів в алфавіті $\{0, 1\}$, для яких виконується наступна умова:

а) містять підслово 000 або 111;
б) в яких п'ятою буквою справа є 0;
в) містять не більше однієї пари послідовних одиниць;
г) містять непарну кількість букв 0;
ґ) кількість нулів кратна п'яти;
д) не містять підслова 000;
е) не містять ні підслова 000, ні 111;
є) не містять підслова 011;
ж) містять непарну кількість входжень підслова 011;
з) кількість одиниць ділиться на п'ять, а кількість нулів парна.

**1.15.** Нехай $L$ – мова над алфавітом $\{0\}$. Довести, що мова $L^*$ є регулярною. [Підказка: доведіть і використайте той факт, що якщо $a$ та $b$ є взаємно простими натуральними числами, то для кожного натурального $n \geq ab$ існують такі невід'ємні цілі числа $u$ та $v$, що виконується рівність $n = ua + vb$.]

Розділ II

# Розширені регулярні вирази і фільтр `grep`

В даному розділі розглядається застосування регулярних виразів для розширеного контекстного пошуку тексту. Ми покажемо, як завдання пошуку тексту, які зазвичай займають години, можуть бути виконані без труднощів за декілька секунд.

Для пошуку тексту, який задається з допомогою регулярного виразу, використовується програма-фільтр `grep`. За початковий алфавіт береться множина всіх символів коду ASCII (American Standard Code for Information Interchange), крім $\backslash n$ і $\backslash 0$. ASCII – система кодів, у якій числа від 0 до 127 включно поставлені у відповідність керуючим функціям (наприклад, введення чи стирання попереднього символу), літерам, цифрам і знакам пунктуації. Перші 32 коди використовують для керуючих функцій, решта 96 – графічних символів.

Програма `grep` викликається командою:

`grep` [опції] "регулярний вираз" [шлях до вхідних файлів] [> шлях до файла вихідних даних]

`Grep` шукає у вказаному файлі чи файлах (якщо файлів декілька, то вони записуються через пропуск) рядки, що містять слово, яке належить мові, заданій регулярним виразом. За відсутності в команді імен файлів дані вибираються з `stdin` (стандартний файл введення, який відкривається операційною системою) і копіюються в `stdout` (стандартний файл виведення), які за погодженням пов'язуються з терміналом. Якщо `grep` переглядає кілька файлів, то на початку кожного вивідного рядка вміщується ім'я файла, з якого він вибраний. Типовий випадок застосування `grep` – це пошук імен змінних та функцій, використаних у кількох файлах, що містять початкову програму. Так, за виклику `grep` main *.c в `stdout` виведуться рядки, що містять слова main, з усіх файлів, імена яких закінчуються на .c

Опції – необов'язковий елемент, і їх детальніше розглянемо в наступному пункті.

Вивести коротку інформацію про програму можна з допомогою команди `grep −−help`.

Сучасні версії програми `grep` (як і редактора `sed`, який вивчатиметься у наступному розділі) підтримують кирилицю. Однак, щоб охопити всі версії, в посібнику розглядається робота програми тільки з англійськими текстами.

Вміст файлів Company.txt, Metalist.txt та People.txt показаний в додатках А, Б та В відповідно.

## 1. Опції фільтра `grep`

Основні опції програми grep:

1) -i: ігнорує відмінності в регістрі символів у словах, заданих регулярним виразом.

1.1. Приклад. Вивести рядки з файла Metalist.txt, які містять слово Brazil або BRAZIL.

Даний пошук здійснюється з допомогою команди:

`grep` -i "brazil" Disk:\...\Metalist.txt,

де Disk:\...\Metalist.txt – шлях до файла Metalist.txt на локальному диску Disk.

В результаті на екран буде виведено рядок:

```
 10        MF     Cleiton Xavier                Brazil
```

2) -c: виводить кількість рядків, які містять слово, задане регулярним виразом.

1.2. Приклад. Вивести кількість рядків з файла Metalist.txt з інформацією про футболістів, які є громадянами України.

`grep` -i -c "ukraine" Disk:\...\Metalist.txt
14

3) -n: виводить перед кожним рядком його порядковий номер у файлі (нумерація починається з 1).

1.3. Приклад. Вивести у файл Brazil.txt рядки з їх порядковими номерами з файла Metalist.txt, які містять слово Brazil.

`grep` -n "Brazil" Disk:\...\Metalist.txt > Disk:\...\Brazil.txt

В результаті у файл Brazil.txt, який створюється відповідно до шляху Disk:\...\Brazil.txt, буде записано рядок:

```
9:10      MF      Cleiton Xavier                Brazil
```

4) -v: виводить усі рядки, за винятком тих, які містять збіг із регулярним виразом.

1.4. Приклад. Вивести з файла Metalist.txt рядки з інформацією про закордонних футболістів.

```
grep -i -v "ukraine" Disk:\...\Metalist.txt
3         DF      Cristian Villagra             Argentina
6         DF      Marco Torsiglieri             Argentina
10        MF      Cleiton Xavier                Brazil
11        MF      Jose Ernesto Sosa             Argentina
19        MF      Juan Manuel Torres            Argentina
21        FW      Jonathan Cristaldo            Argentina
22        DF      Milan Obradovic               Serbia
23        MF      Sebastian Blanco              Argentina
30        DF      Papa Gueye                    Senegal
37        DF      Vitalie Bordian               Moldova
71        MF      Sergei Tkachyov               Russia
80        DF      Lukas Stetina                 Slovakia
```

5) -w: виводить лише ті рядки, в яких слово, задане регулярним виразом, обмежується зліва і справа будь-яким символом, відмінним від букви, цифри та риски знизу (_).

1.5. Приклад. Вивести рядки з файла Metalist.txt, які містять абревіатуру GK.

```
grep -w "GK" Disk:\...\Metalist.txt
1         GK      Maksym Startsev               Ukraine
29        GK      Oleksandr Goryainov           Ukraine
81        GK      Vladimir Disljenkovic         Ukraine
```

6) -x: обирає лише точні збіги регулярного виразу з цілим рядком і виводить його.

1.6. Приклад. Вивести рядок „Alina Moroz 0975631635 1994” з файла People.txt.

```
grep -n -x "Alina Moroz 0975631635 1994" Disk:\...\People.txt
1:Alina Moroz 0975631635 1994
```

7) -H: виводить назву файла перед кожним рядком, який містить слово, задане регулярним виразом.

1.7. Приклад. Вивести назву файла і рядок, в якому міститься слово Andriy з файла People.txt.

```
grep -w -H "Andriy" Disk:\...\People.txt
Disk:\...\People.txt:Andriy Fedoriv 0971635583 1985
```

8) -h: не виводить назви файла поряд з кожним рядком, який містить слово, задане регулярним виразом.

1.8. Приклад. Вивести рядки з файлів People.txt і Company.txt, які містять число 1989.

```
grep -w -h "1989" Disk:\...\People.txt Disk:\...\Company.txt
Anna Alnikina 0505053455 1989
Hire_date: 1989-10-12
```

9) -o: виводить тільки ту частину рядка, яка збіглася з регулярним виразом.

1.9. Приклад. Вивести число 1989 з тих рядків файлів People.txt і Company.txt, в яких це число міститься.

```
grep -w -o "1989" Disk:\...\People.txt Disk:\...\Company.txt
Disk:\...\People.txt:1989
Disk:\...\Company.txt:1989
```

10) -B N: разом з кожним рядком, який містить слово, задане регулярним виразом, виводить N рядків, розміщених перед ним.

1.10. Приклад. З файла Company.txt вивести кожні три послідовні рядки, в останньому з яких є число 1986.

```
grep -w -B 2 "1986" Disk:\...\Company.txt
Name: Sue Smith
Age:   48
Hire_date: 1986-12-10

Name: Dan Roberts
Age:   45
Hire_date: 1986-10-20
```

11) -A N: разом з кожним рядком, який містить слово, задане регулярним виразом, виводить N рядків, розміщених після нього.

1.11. Приклад. З файла Company.txt вивести кожні чотири послідовні рядки, в першому з яких записано ім'я Bob Smith.

```
grep -A 3 "Bob Smith" Disk:\...\Company.txt
Name: Bob Smith
Age:   33
Hire_date: 1987-05-19
Sales: 142594.00
```

12) -C N: разом із кожним рядком, який містить слово, задане регулярним виразом, виводить N рядків перед і після нього.

1.12. Приклад. З файла Company.txt вивести кожні три послідовні рядки, в другому з яких є число 52.

```
grep -w -C 1 "52" Disk:\...\Company.txt
Name: Sam Clarkson
Age:   52
Hire_date: 1988-06-14
```

13) -m N: виводить перших N рядків, які містять збіг з регулярним виразом.

1.13. Приклад. Вивести з файла Metalist.txt перші три рядки з інформацією про закордонних футболістів.

```
grep -v -m 3 "Ukraine" Disk:\...\Metalist.txt
 3         DF     Cristian Villagra          Argentina
 6         DF     Marco Torsiglieri          Argentina
 10        MF     Cleiton Xavier             Brazil
```

14) -r: рекурсивно аналізує все дерево вказаного каталогу, тобто підкаталоги, підкаталоги підкаталогів, і т. д. аж до файлів.

1.14. Приклад. Вивести назви файлів і рядки з підкаталогу DODATKY, в яких зустрічається 1986 рік.

```
grep -r -w "1986" Disk:\...\DODATKY
Disk:\...\DODATKY/Company.txt:Hire_date: 1986-12-10
Disk:\...\DODATKY/Company.txt:Hire_date: 1986-10-20
Disk:\...\DODATKY/People.txt:Olia Vuykova 0974306550 1986
```

15) -l: виводить тільки імена файлів, які містять слово, задане регулярним виразом, а самі рядки не виводить.

1.15. Приклад. Вивести назви файлів, які містять 45.

```
grep -l 45 Disk:\...\Company.txt Disk:\...\Metalist.txt
Disk:\...\Company.txt
```

16) -L: навпаки, виводить імена файлів, які не містять слова, заданого регулярним виразом.

1.16. Приклад. Вивести назви файлів, які не містять 45.

```
grep -L 45 Disk:\...\Company.txt Disk:\...\Metalist.txt
Disk:\...\Metalist.txt
```

## 2. Метасимволи

Щоб використовувати регулярні вирази для пошуку тексту, їх доведеться розширити. Розширені регулярні вирази складаються зі звичайних символів та метасимволів. Метасимвол – це символ або комбінація символів, яка має спеціальне значення, якщо вона зустрічається у регулярному виразі. Розглянемо наступний регулярний вираз: ^\(Ivan\|Ihor\) is a student. У ньому символи ^, \(, \), та \| відносяться до метасимволів, усі інші частини регулярного виразу є звичайними символами. Якщо у регулярному виразі не використовуються метасимволи, то він перетворюється на „засіб простого пошуку”.

Одними із найбільш простих метасимволів є метасимволи ^ (шапка) і $ (долар). Вони відповідають початку і кінцю рядка, що підлягає перевірці. Наприклад, з допомогою регулярного виразу big шукаються послідовно розташовані букви b і g у будь-якій позиції рядка, але вираз ^big описує лише ті рядки, в яких букви b і g знаходяться на початку рядка. Метасимвол ^ фактично прив’язує наступну частину регулярного виразу до початку рядка. Аналогічно вираз big$ описує тільки ті рядки, в яких букви b і g містяться в кінці рядка.

2.1. Приклад. Вивести рядки з файла People.txt, першим символом яких є буква I.

```
grep "^I" Disk:\...\People.txt
Iryna Boychuk 0990729125 1996
```

2.2. Приклад. Вивести рядки з файла Metalist.txt, які закінчуються словом Slovakia.

```
grep "Slovakia$" Disk:\...\Metalist.txt
 80        DF      Lukas Stetina                 Slovakia
```

Метасимвол . (крапка) позначає будь-який символ, крім символу нового рядка \n. Він використовується тоді, коли в декількох позиціях регулярного виразу може знаходитися будь-який символ. Наприклад, ви хочете знайти дату, яка може бути записана як 12/02/2012 або 12-02-2012. За допомогою метасимволу . можна записати регулярний вираз 12.02.2012, який включатиме два попередні варіанти запису дати. Для того, щоб використовувати символ, який зазвичай інтерпретується як метасимвол, перед ним ставлять зворотній слеш \. Наприклад, з допомогою виразу @yahoo\.com можна знайти всі електронні адреси, зареєстровані на Yahoo.

2.3. Приклад. Вивести рядки з файла Metalist.txt з інформацією про гравців клубу, другою цифрою номера футболки яких є 7.

```
grep -w "^.7" Disk:\...\Metalist.txt
 17        DF      Serhiy Pshenychnykh           Ukraine
 27        MF      Yuriy Chonka                  Ukraine
 37        DF      Vitalie Bordian               Moldova
```

Дуже зручний та корисний метасимвол \|, який означає „або”. Цей метасимвол дозволяє об’єднати декілька регулярних виразів в один, з допомогою якого шукатимуться всі рядки, що містять слово, задане будь-яким із підвиразів, з яких він складається. Наприклад, subgroup і subring – два різні регулярні вирази, а вираз subgroup\|subring – один регулярний вираз, що описує всі рядки, які містять subgroup або subring. Підвирази, об’єднанні даним способом, називаються альтернативами. Слід зауважити, що цей вираз можна записати як sub\(group\|ring\). В останньому варіанті круглі дужки відділяють конструкцію вибору від решти виразу (круглі дужки разом зі слешами також є метасимволами). У виразі sub\(group\|ring\) круглі дужки необхідні, оскільки без них subgroup\|ring буде означати subgroup або ring, а це не те що нам потрібно. Конструкція вибору діє лише всередині круглих дужок.

2.4. Приклад. Вивести рядки з файла People.txt, які містять 314 або 432.

```
grep "314\|432" Disk:\...\People.txt
Dmytro Shopher 0964143217 1987
```

```
Olia Sushaylo 0967314022 1988
```

Для пошуку декількох регулярних виразів, які містяться в одному і тому ж рядку, використовується команда такого вигляду:

`grep` [опції] "регулярний вираз 1" [шлях до файлів] | `grep` [опції] "регулярний вираз 2" | . . . | `grep` [опції] "регулярний вираз n"

2.5. Приклад. Вивести рядки з файла People.txt, які містять 66 та 3 одночасно.

```
grep "66" Disk:\...\People.txt| grep "3"
Lesya Ostropol 0966943076 1976
Natalia Slyvchuk 0976683158 1996
```

## 3. Визначення інтервалів та кількості екземплярів

Якщо символ або частина регулярного виразу, який ми шукаємо, повинен повторитися певну кількість разів, то можна просто записати його у регулярний вираз декілька раз. Наприклад, вираз baaac буде шукати символ b, за яким слідує три однакових символи a, після яких є символ c. Якщо кількість повторень символу або частини регулярного виразу буде великою, то використовувати такий спосіб недоречно. Тим більше, трапляються ситуації, коли точна кількість повторень певного символу наперед невідома.

Синтаксис регулярних виразів включає набір спеціальних метасимволів, які успішно вирішують задачі із повторенням символів. Наступний список містить три метасимволи (квантифікатори), які описують кількісну характеристику символу, що знаходиться безпосередньо перед метасимволом:

1) * – метасимвол ітерації, який вказує на те, що попередній символ повинен повторюватися нуль або більше разів підряд.

3.1. Приклад. З файла People.txt вивести рядки, в яких зустрічається послідовність цифр у порядку 9, 7, 5, 2 в одному слові і, можливо, деякі символи між ними.

```
grep -w ".*9.*7.*5.*2.*" Disk:\...\People.txt
Artur Pylyp 0967752148 1964
Lesya Gaevska 0979453892 1987
```

2) \+ – вказує на те, що попередній символ має повторюватися один або більше разів.

3.2. Приклад. Вивести рядки з файла Metalist.txt, в яких символ r повторюється підряд принаймні 2 рази.

```
grep "rr\+" Disk:\...\Metalist.txt
 19        MF      Juan Manuel Torres          Argentina
```

3) \? – вказує на те, що попередній символ присутній один раз або відсутній.

3.3. Приклад. Вивести рядки з файла Metalist.txt, в яких записані футболісти з номером, в якому перша цифра 1, а друга – або 1, або її немає зовсім.

```
grep -w "11\?" Disk:\...\Metalist.txt
 1         GK      Maksym Startsev             Ukraine
 11        MF      Jose Ernesto Sosa           Argentina
```

Щоб поширити дію квантифікаторів більш, ніж на один символ або на окрему частину регулярного виразу, потрібно виділити цю частину регулярного виразу у круглі дужки з слешами. Наприклад, вираз \(from:\)\?sender@yandex\.ru вказує на те, що послідовність from: передує емейл-адресу відправника або є відсутньою.

В регулярних виразах, крім квантифікаторів, які описують відносні кількісні характеристики символів, є спосіб явно вказати, яку кількість разів має бути присутнім певний символ або частина регулярного виразу. Вираз \{n\} означає, що попередній символ (група символів) має повторитися рівно n разів. Конструкція \{n, m\} називається інтервальним квантифікатором. Вона вказує на те, що попередній символ має повторитися не менше, ніж n, але не більше, ніж m разів. Якщо в інтервальному квантифікаторі відсутній максимум m, то в даному випадку максимальна кількість повторень не передбачається і може бути довільною. Запис \{0,1\} рівносильний метасимволу \?, а запис \{0,\} – метасимволу *.

3.4. Приклад. Вивести всі рядки з файла People.txt, в яких шостим символом справа є цифра 9.

```
grep "9.\{5\}$" Disk:\...\People.txt
Lyuda Bazyta 0979428059 1987
Olia Shveya 0976353459 1987
Taras Saras 0503999999 1990
```

3.5. Приклад. Вивести всі рядки з файла People.txt, які містять 31 або 27 символів.

```
grep -x "\(.\{4\}\)\?.\{27\}" Disk:\...\People.txt
Alina Moroz 0975631635 1994
Artur Pylyp 0967752148 1964
Lesya Gogol 0636200646 1931
Lesya Rybak 0978936545 1987
Lyubchyk Kitral 0633545735 1991
Makar Korok 0666000066 1988
Natalia Pavliuh 0977178827 1998
Oksana Hlymkova 0979157143 1974
Olia Shveya 0976353459 1987
Taras Saras 0503999999 1990
Vasyl Loboz 0965306862 1988
```

3.6. Приклад. Вивести рядки з файла People.txt, в яких цифра 2 зустрічається від двох до п'яти разів.

```
grep "2\{2, 5\}" Disk:\...\People.txt
Olia Sushaylo 0967314022 1988
```

3.7. Приклад. Вивести ті рядки з файла People.txt, які містять більше 30 символів.

```
grep ".\{31,\}" Disk:\...\People.txt
Lyubchyk Kitral 0633545735 1991
Natalia Pavliuh 0977178827 1998
Natalia Slyvchuk 0976683158 1996
Oksana Hlymkova 0979157143 1974
```

## 4. Символьні класи

Основним будівельним блоком регулярних виразів є односимвольний шаблон. Він відповідає просто вказаному символу. Символи можуть бути згруповані разом у клас. Символьний клас записується як пара квадратних дужок зі списком символів між ними. Клас також є односимвольним шаблоном. Один і лише один із цих символів повинен бути присутнім у тексті, який співпаде з регулярним виразом. Наприклад, ви хочете знайти слово begin, яке може бути записане в трьох часових формах: begin, began, begun. За допомогою конструкції [...] можна перелічити усі символи, що можуть бути у даній позиції тексту. Таким чином, вираз beg[iau]n означає: „Знайти слово beg, за яким слідує одна з букв i, a або u,

після яких слідує буква n". У символьному класі передбачається вибір будь-якого символу, незалежно від його розміщення.

Не плутайте конструкцію вибору із символьними класами. У символьних класах передбачається співпадання із одним символом цільового тексту. В конструкції вибору кожна альтернатива може бути повноцінним регулярним виразом.

Кількість символів у символьному класі є довільною. Наприклад, клас [123456]=1\|2\|3\|4\|5\|6 співпаде з будь-якою із цих цифр. Всередині символьного класу метасимвол – (дефіс) позначає інтервал символів. Наприклад, клас [1-6] рівносильний класу [123456]. Класи [0-9] та [a-z] зазвичай використовуються для пошуку цифр та букв нижнього регістру. Інтервали можна успішно комбінувати зі звичайними символами. Наприклад, вираз [0-9A-Z:!] співпаде із цифрою, буквою верхнього регістру, символом двох крапок (:) або знаком оклику (!).

4.1. Приклад. Вивести рядки з файла People.txt, які починаються символом I чи G.

```
grep "^[IG]" Disk:\...\People.txt
Galyna Koziyka 0674538354 1983
Iryna Boychuk 0990729125 1996
```

Діапазони символів вказуються згідно з кодуванням ASCII.

4.2. Приклад. Клас [>?@*abcde*012345] може бути записаний у вигляді $[> -e0-5]$ або $[>?@a-e0-5]$, оскільки кодами в ASCII для символів $>, ?, @, a, b, c, d, e$ відповідно є 62, 63, 64, 65, 66, 67, 68, 69. Цифри 0, 1, 2, 3, 4, 5 мають кодами 48, 49, 50, 51, 52, 53.

4.3. Приклад. Регулярний вираз $\([0-9]\|[1-9][0-9]\|1[0-9][0-9]\|2[0-4][0-9]\|25[0-5]\)$ позначає множину всіх цілих чисел з відрізка [0, 255].

Якщо замість конструкції [...] поставити конструкцію [^...], то символьний клас буде описувати всі символи, що не входять у нього. Наприклад, вираз [^1-6] описує всі символи, що не належать інтервалу від 1 до 6. Слід зауважити, що для інвертування класу використовується той самий символ ^, що й для позначення початку рядка, але всередині символьного класу у нього зовсім інша дія.

4.4. Приклад. Вивести усі рядки з файла People.txt, в яких дев'ятим символом не є мала буква англійського алфавіту.

```
grep "^.\{8\}[^a-z]" Disk:\...\People.txt
Lyubchyk Kitral 0633545735 1991
Nanashka Oks 0968030837 1999
Natalia Pavliuh 0977178827 1998
Natalia Slyvchuk 0976683158 1996
```

Для деяких символьних класів є спеціально введені позначення:

1) [[:digit:]] – клас цифр.

4.5. Приклад. Вивести рядки з файла Metalist.txt, в яких номер футболіста складається з двох цифр, причому другою цифрою є 3.

```
grep -w "[[:digit:]]3" Disk:\...\Metalist.txt
 23        MF     Sebastian Blanco            Argentina
 33        FW     Marko Devich                Ukraine
```

2) [[:upper:]] – клас великих англійських букв.

4.6. Приклад. Вивести ті рядки з файла People.txt, в яких десятим символом є велика буква англійського алфавіту.

```
grep "^.\{9\}[[:upper:]]" Disk:\...\People.txt
Lyubchyk Kitral 0633545735 1991
Nanashka Oks 0968030837 1999
```

3) [[:lower:]] – клас малих англійських букв.

4.7. Приклад. Вивести рядки з файла People.txt, в яких зустрічається 8 букв нижнього регістру підряд.

```
grep "[[:lower:]]\{8\}" Disk:\...\People.txt
Dima Nykyforuk 0979735771 1987
Dima Volovchuk 0969558087 1985
```

4) [[:alpha:]] – клас великих і малих англійських букв.

4.8. Приклад. Вивести рядки з файла Metalist.txt, які містять слова, що складаються з 12 англійських букв верхнього або нижнього регістру.

```
grep -w "[[:alpha:]]\{12\}" Disk:\...\Metalist.txt
 17        DF     Serhiy Pshenychnykh         Ukraine
 81        GK     Vladimir Disljenkovic       Ukraine
```

4.9. Приклад. З файла Metalist.txt вивести всі слова, які містять 11 або 12 букв.

```
grep -o -w "[[:alpha:]]\{11,12\}" Disk:\...\Metalist.txt
Berezovchuk
Torsiglieri
Pshenychnykh
Disljenkovic
```

5) [[:alnum:]] – клас великих і маленьких англійських букв та цифр.

4.10. Приклад. Вивести рядки з файла People.txt, в яких перший і останній символ належить класу [:alnum:], а передостанній – цифра 3.

```
grep "^[[:alnum:]].*3[[:alnum:]]$" Disk:\...\People.txt
Lesia Gogol 0636200646 1931
```

6) [[:punct:]] – клас знаків пунктуації: ! " # $ % & ’ ( ) * + , – . / : ; < = > ? [ \ ] ^ _ ‘ { | } ~

4.11. Приклад. Вивести рядки з файла Company.txt, в яких присутній знак пунктуації і останнє слово складається з 8 букв нижнього або верхнього регістру.

```
grep ".*[[:punct:]].*[[:alpha:]]\{8\}$" Disk:\...\Company.txt
Name: Sam Clarkson
```

7) [[:graph:]] – клас графічних символів (він є об’єднанням попередніх двох класів).

4.12. Приклад. Вивести рядки з файла Metalist.txt, в яких другий символ не належить класу [:graph:].

```
grep -v "^.[[:graph:]]" Disk:\...\Metalist.txt
 1        GK     Maksym Startsev            Ukraine
 2        DF     Oleksandr Romanchuk        Ukraine
 3        DF     Cristian Villagra          Argentina
 4        DF     Andriy Berezovchuk         Ukraine
 5        MF     Oleh Shelayev              Ukraine
 6        DF     Marco Torsiglieri          Argentina
 7        MF     Serhiy Valyayev            Ukraine
 9        FW     Andriy Vorobey             Ukraine
```

8) [[:blank:]] – клас символів пропуску і табуляції.

4.13. Приклад. Вивести рядки з файла People.txt, в яких першою буквою другого слова є A або B.

```
grep "^[[:alpha:]]\+[[:blank:]]\+[AB]" Disk:\...\People.txt
Anna Alnikina 0505053455 1989
Iryna Boychuk 0990729125 1996
Lyuda Bazyta 0979428059 1987
```

9) [[:space:]] – попередній клас в об'єднанні з символами вертикального пропуску і горизонтальної лінії.

## 5. Групи та посилання

Ситуація, коли потрібно використовувати не весь знайдений за допомогою регулярного виразу рядок, а лише деяку його частину, доволі часто зустрічається на практиці. Будь-яку частину регулярного виразу можна виділити в групу, щоб потім звертатися не до всього знайденого рядка одразу, а окремо до конкретної раніше виділеної групи. Задати групу неважко: для цього потрібно лише виділити частину регулярного виразу конструкцією: \( частина регулярного виразу \). Групи можуть бути вкладені одна в одну. Отримати доступ до групи можна за допомогою порядкового номера. Вони нумеруються зліва направо. Найлівіша група має порядковий номер 1, за нею слідує група 2 і т.д. Їх може бути не більше дев'яти. Нульовий номер зарезервований для всього рядка, який відповідає регулярному виразу.

З групами пов'язана ще одна тема – посилання. Її просто не можна обійти стороною, оскільки при використанні посилань „розширені" регулярні вирази виходять далеко за рамки того, що називається регулярними виразами в теорії, і ці відмінності треба розуміти. Посилання дозволяють використовувати у регулярному виразі вміст уже знайденої групи. Якщо група має порядковий номер $k$, то посилання на неї виконується з допомогою виразу $\backslash k$. Наприклад, у регулярному виразі \(.\)\(.\)\1\2 перший та другий символи виділені в групи, які одержують порядкові номери 1 та 2 відповідно, а далі використовуються посилання на першу та другу групи. Таким чином, даний регулярний вираз описує всі слова, в яких третя буква співпадає з першою, а четверта – з другою.

5.1. Приклад. Вивести рядки з файла People.txt, в яких перша та остання букви в першому слові співпадають.

```
grep -i "^\([[:alpha:]]\)[[:alpha:]]*\1)[[:blank:]]" Disk:\...\People.txt
Alina Moroz 0975631635 1994
Anna Alnikina 0505053455 1989
```

5.2. Приклад. З файла People.txt вивести всі слова-поліндроми довжини 4 або 5 букв разом з порядковими номерами рядків, в яких вони знаходяться.

```
grep -n -o -i "\([[:alpha:]]\)\([[:alpha:]]\)[[:alpha:]]\?\2\1" Disk:\...\
People.txt
3:Anna
3:nikin
4:Pylyp
6:Volov
21:Korok
32:Saras
```

5.3. Приклад. Вивести рядки з файла Metalist.txt, в яких в останньому слові є подвоєння будь-якої букви.

```
grep -x ".*[[:blank:]][[:alpha:]]*\([[:alpha:]]\)\1[[:alpha:]]*$"
Disk:\...\Metalist.txt
 71        MF      Sergei Tkachyov                Russia
```

5.4. Приклад. Вивести всі рядки з файла a.txt, в яких зустрічаються два послідовно записані однакові слова, розділені пропусками.

```
grep -w -i "\([[:alpha:]]\+\)[[:blank:]]\+\1" a.txt
```

## Питання та вправи до розділу 2.

**2.1.** Перерахуйте основні опції команди grep.

**2.2.** Як утворюються символьні класи?

**2.3.** Для чого використовують групи та посилання?

**2.4.** Вивести всі рядки з інформацією про форвардів з файла Metalist.txt.

**2.5.** У файлі Metalist.txt підрахувати кількість закордонних футболістів.

**2.6.** Вивести рядки з інформацією про футболістів із Бразилії та Аргентини з файла Metalist.txt.

**2.7.** Видати інформацію про усіх закордонних захисників файла Metalist.txt.

**2.8.** Вивести назви файлів додатку, які містять двоцифрові числа.

**2.9.** Вивести назви файлів додатку, які не містять слів довжини 3.

**2.10.** Вивести всі рядки з файлів Metalist.txt, People.txt та Company.txt, в яких восьмим символом є буква a.

**2.11.** Вивести всі рядки з файлів Metalist.txt та Company.txt, в яких міститься принаймні одне слово з третьою буквою n.

**2.12.** Видати всі рядки з файла Metalist.txt, в яких 12 символом є велика буква.

**2.13.** Підрахувати кількість рядків з файла Metalist.txt, які містять числа в інтервалі від 10 до 35.

**2.14.** Вивести рядки з файла Metalist.txt, в яких другим символом є цифра 7, а п'ятим – буква f (регістр букв ігнорувати).

**2.15.** З файла Metalist.txt вивести всі рядки (з порядковими номерами), передостаннім символом в яких є буква n.

**2.16.** Підрахувати кількість рядків файла Metalist.txt, які не містять букв m та o одночасно.

**2.17.** Видати усі рядки з файла People.txt, другими буквами яких не є букви i, a або e.

**2.18.** Вивести інформацію про тих людей з файла People.txt, в яких номер телефону закінчується на цифру з діапазону 0-5, а рік народження не закінчується на жодну з цих цифр.

**2.19.** З файла People.txt вивести усі рядки, останньою буквою імені яких є a, а першою буквою прізвища – K.

**2.20.** Видати всі рядки з файлів Metalist.txt, People.txt та Company.txt, в яких 8 символ не належить діапазону k-z.

**2.21.** Вивести усі рядки з файла People.txt, в яких ім'я не містить 5 букв, а прізвище містить 6 букв.

**2.22.** Вивести усі рядки з файла People.txt, які містять числа 74 та 47 одночасно.

**2.23.** Видати інформацію про людей із файла People.txt, номери телефонів яких містять цифри 9 і 4, між якими міститься довільна кількість цифр 7 (їх може не бути зовсім).

**2.24.** Видати інформацію про усіх людей 1991 року народження із файла People.txt, номери телефонів яких містять цифру 8.

**2.25.** Підрахувати кількість абонентів „Київстар" із файла People.txt.

**2.26.** З файла Company.txt вивести всі рядки, в яких є принаймні два знаки пунктуації.

**2.27.** Видати всі рядки з файла People.txt, в яких число 06 зустрічається двічі.

**2.28.** Вивести інформацію про перших п'ятьох людей з файла People.txt, які народилися в 80-х роках.

**2.29.** Вивести інформацію про усіх людей із файла People.txt, в яких прізвище або ім'я закінчується буквою k.

**2.30.** Видати інформацію про тих людей з файла People.txt, в яких прізвище містить більше шести літер.

**2.31.** Видати дані про тих людей з файла People.txt, в яких ім'я містить від 5 до 8 літер.

**2.32.** Вивести всі слова непарної довжини файла Company.txt.

**2.33.** Видати всі рядки з файла Company.txt, в яких першим символом є велика буква, а останнім – цифра.

**2.34.** Вивести всі рядки файла Metalist.txt, довжина яких при діленні на 4 дає остачу 1.

**2.35.** З файла Company.txt вивести всі рядки, які містять принаймні два пропуски підряд та довжина яких не ділиться на 6.

**2.36.** Вивести інформацію про першого футболіста з іменем Andriy з файла Metalist.txt та наступних трьох футболістів по списку.

**2.37.** Видати кожні чотири послідовні рядки з файла People.txt, в другому з яких передостанньою цифрою номера телефону є 3.

**2.38.** Вивести рядки з інформацією про прізвища, вік та дату прийняття на роботу тих людей з файла Company.txt, ім'я яких містить три букви.

**2.39.** Вивести повну інформацію про людей з файла Company.txt, вік яких від 30 до 35 років.

**2.40.** З файла People.txt вивести всі слова довжини 5 разом з порядковими номерами рядків, в яких вони містяться.

**2.41.** Вивести тільки номери телефонів абонентів „Київстар" з файла People.txt.

**2.42.** Видати всі ті рядки з файла People.txt, в яких номери телефонів містять поліндроми довжини 2 або 3.

**2.43.** Вивести ті рядки з файла People.txt, в яких третя цифра номера телефону співпадає з останньою цифрою.

**2.44.** Видати інформацію про тих людей 1990 року народження з файла People.txt, в яких номер телефону містить три послідовні однакові цифри.

**2.45.** Підрахувати кількість тих людей з файла People.txt, в яких друга літера імені співпадає з останньою.

**2.46.** Вивести дані про тих людей з файла People.txt, в яких перша літера прізвища співпадає з передостанньою літерою.

**2.47.** Вивести інформацію про усіх людей з файла People.txt, в яких 2 та 3 літера імені співпадають з 1 та 4 літерою прізвища відповідно (регістр букв ігнорувати).

**2.48.** Видати всі ті рядки з файла People.txt, в яких ще раз зустрічається перша буква (регістр букв ігнорувати).

**2.49.** Вивести всі ті рядки з файла People.txt, в яких останні чотири символи утворюють поліндром.

Розділ III

# Обробка тексту потоковим редактором sed

Програма `sed` – це потоковий редактор (`s`tream `ed`itor) для автоматичного редагування тексту. „Потоковий редактор" в тому сенсі, що він може редагувати вхідний потік даних неперервно. Автоматично – означає те, що після задання вами правил редагування, далі все відбувається без вашої участі. Іншими словами, редактор `sed` не є інтерактивним. Він здатен виконувати складні завдання, але треба витратити час, щоб навчитися їх формулювати.

Редактор `sed` викликається наступним чином:

`sed` [-n] [-i] [-e команда 1]...[-e команда n] [-f файл команд] [шлях до вхідних файлів] [> шлях до файла вихідних даних]

`Sed` циклічно зчитує рядки з вхідних текстових файлів або зі стандартного вводу `stdin` в робочий буфер, змінює їх у залежності від команд редагування або від файла команд і виводить результат (за відсутності опції -n) на стандартний вивід `stdout` або в файл вихідних даних. Якщо використовується опція -i, то зміни будуть виконуватися у вхідних файлах. Після кожного циклу робочий буфер очищається. Опція -n відміняє вивід (який відбувається за замовчуванням). Якщо виконується тільки одна команда, то опція -e необов'язкова. У випадку, коли потрібно виконати значну кількість команд, зручніше записати їх у файл і застосувати опцію -f. Наприклад, команда `sed` -f cmd.txt list.txt буде застосовувати команди з файла cmd.txt до даних файла list.txt. Кожна команда файла команд має міститися в окремому рядку. Важливо не плутати файл команд із файлом вхідних даних.

Формат команди:

[адреса1[, адреса2]][!]функція[аргументи]

Адресами можуть бути: порожнє поле; десяткові числа, що задають номер вхідних рядків (за кількох файлів припускається наскрізна нумерація); знак долара ($), який вказує на останній рядок в останньому файлі; контекстна адреса, яка складається з регулярного виразу, поміщеного між двома косими рисками. Адреси команди повністю визначають її застосовність до вмісту робочого буфера. Командний рядок, який не містить адреси, застосовується до кожного рядка вхідного файла. Командний рядок, що містить одну контекстну адресу застосовується до кожного рядка з вказаним регулярним виразом. Дві адреси в команді вказують діапазон її застосовності: початок – до вмісту робочого буфера з моменту його відповідності першій адресі (зіставляється з регулярним виразом або має рядок з вказаним номером), кінець – його відповідність другій адресі. Потім процес виділення діапазону повторюється. Починаючи з першого рядка, який супроводжується обраним діапазоном, `sed` починає пошук першої адреси знову.

Наприклад, команда з адресою /big/ застосовується до всіх рядків вхідного файла, які містять слово big. Якщо вказана адреса 3, $, то команда застосовуватиметься до всіх рядків, починаючи з третього. Команда з адресою /big/,/^$/ застосовується до всіх рядків, починаючи з першого рядка, який містить слово big, до наступного першого порожнього рядка, потім знову шукається вказаний діапазон і т.д. Якщо ж деякий рядок містить слово big, але після нього немає більше порожніх рядків, то команда застосовується до останнього рядка включно.

Варто зауважити використання конструкції виду „!функція”. У цьому випадку команда застосовується до всіх рядків, крім тих, які вказані в адресі. Якщо адреса відсутня, то команда не застосовується до жодного рядка.

У `sed` немає змінних, однак є дві області пам’яті, з якими можна працювати фактично як із змінними: робочий буфер (pattern space) і допоміжний буфер (hold space). На початку роботи обидва буфери є порожніми. Робочий буфер містить останній зчитаний з файла рядок. Саме з цією областю пам’яті працюють основні команди. Деякі команди використовують допоміжну пам’ять, щоб запам’ятати всю робочу область або її частину для подальшого використання. Hold space – це допоміжній буфер, в якому можуть міститися рядки протягом всієї роботи редактора. Між робочим і допоміжним

буферами можна проводити обмін даними (додавання, заміщення, обмін). Слід зауважити, що обидві області можуть містити не тільки один рядок, а й декілька рядків, розділених символом нового рядка $\backslash n$(newline). Робоча і допоміжна області пам'яті здатні зберігати не менше як 8192 байт.

## 1. Список функцій редактора sed

Усі функції `sed` – це букви a b c d D g G h H i n N p P q Q r s t w x y =. Кожна визначає деяку дію. Як правило, більшість команд `sed` застосовуються до робочого буфера і вихідного потоку.

Для вхідного потоку застосовують тільки дві команди:

- n – взяти наступний рядок з `stdin` і замінити вміст робочого буфера цим рядком;
- N – взяти наступний рядок і додати його до вмісту робочого буфера (не очищаючи цей буфер), розділивши рядки символом нового рядка $\backslash n$.

На допоміжний буфер впливають ще три команди:

- h – замінити вміст допоміжного буфера вмістом робочого буфера;
- H – додати вміст робочого буфера до допоміжного буфера, розділивши рядки символом нового рядка $\backslash n$;
- x – поміняти місцями вміст робочого та допоміжного буферів.

На вихідний потік впливають команди:

- = – надрукувати номер рядка;
- a\text – додати text після рядка виводу;
- i\text – додати text перед рядком виводу;
- c\text – замінити текст виводу на зазначений text;
- p – вивести поточний вміст робочого буфера;
- P – вивести перший рядок (до символу $\backslash n$) з робочого буфера;
- r rfile – вивести вміст rfile на `stdout` перед читанням наступного рядка;

- w wfile – записати вміст робочого буфера в файл wfile (максимально можна використовувати до 10 відкритих файлів).

До робочого буфера можна застосувати команди:

- d – видалити вміст робочого буфера і почати новий цикл;
- D – видалити перший рядок з робочого буфера і почати новий цикл, але новий вхідний рядок не брати, доки робочий буфер не буде порожнім;
- g – замінити вміст робочого буфера вмістом допоміжного буфера;
- G – додати вміст допоміжного буфера до робочого, розділивши рядки символом нового рядка $\backslash n$;
- s/str1/str2/ – виконати заміну str1 на str2;
- y/str1/str2/ – виконати транслітерацію, тобто замінити всі входження символів з str1 на відповідні з str2. Довжини рядків повинні бути рівними.

Є ще умовні і керуючі команди:

- { } – визначити блок команд;
- : label – встановлює мітку label для переходу за b і t командами;
- b label – перейти на мітку, що встановлюється за допомогою символу :, якщо label порожній, то перейти в кінець скрипту;
- t – перейти на мітку, яка встановлюється за допомогою символу :, якщо для цього рядка була виконана заміна з допомогою команди s (прапорець здійснення заміни відновлюється при читанні наступного рядка або виконанні команди s);
- T – перейти на мітку, яка встановлюється з допомогою символу :, якщо для цього рядка не була виконана заміна з допомогою команди s;
- q – завершити команди, але перед цим додрукувати те, що залишилося на вході;
- Q – завершити команди негайно;
- !func – застосовувати функцію func (або групу функцій в {}) до рядків, які не потрапляють у вказані адреси.

В наступних пунктах зупинимося на основних командах детальніше.

## 2. Функція контекстної заміни

Програма `sed` має безліч власних команд, проте більшість користувачів знають тільки команду s, і цього цілком вистачає, щоб працювати з редактором `sed`.

Формат:

"*s*/регулярний вираз/заміна/прапори"

Функція s замінює слово, яке належить формальній мові, записаній регулярним виразом, на заміну.

Наприклад, якщо задати команду

`echo` sunday | `sed` s/day/night/ (Enter),

то в результаті отримаємо:

sunnight

Ми не брали вираз s/day/night/ в лапки, бо даний приклад не потребує їх, але якби в ньому були присутні метасимволи, то лапки були б обов'язкові.

2.1. Приклад. Нехай файл know.txt містить текст:

```
Knowledge is so simple
just look around any activity
gain knowledge with any
and try to give your life a big turn.
```

В результаті дії команди

`sed` -e "s/simple/hard/" -e "s/big/small/" know.txt > new.txt

створиться файл new.txt, який міститиме текст:

```
Knowledge is so hard
just look around any activity
gain knowledge with any
and try to give your life a small turn.
```

Редактор `sed` працює з регулярними виразами так, як і фільтр `grep`. Зокрема, при їх побудові можна використовувати групи та посилання.

2.2. Приклад. З допомогою команди

```
echo They have three books | sed "s/\([[:alpha:]]\+\).*/\1/"
```

буде виведено тільки перше слово рядка:
They

2.3. Приклад. Для того щоб поміняти слова місцями, потрібно спершу їх запам'ятати, а потім записати посилання на них у протилежному порядку:

```
echo one two | sed "s/\([a-z]\+\)[[:blank:]]\+\([a-z]\+\)/\2 \1/"
two one
```

Щоб видалити дублікати слів, можна використати команду:

```
echo one one | sed "s/\([[:alpha:]]\+\)[[:blank:]]\+\1/\1/"
one
```

2.4. Приклад. У всіх рядках файла know.txt, які містять слово any, видалити перше слово.

Для цього, як відомо, потрібно використати команду з адресою /any/. Шукана команда має вигляд:

```
sed "/any/s/^[[:alpha:]]\+//" know.txt
Knowledge is so simple
 look around any activity
 knowledge with any
and try to give your life a big turn.
```

2.5. Приклад. Подвоїти перше слово, витерши решту тексту, у всіх рядках, які не містять слова just:

```
sed "/just/!s/\([[:alpha:]]\+\).*/\1 \1/" know.txt
Knowledge Knowledge
just look around any activity
gain gain
and and
```

2.6. Приклад. Розглянемо роботу редактора `sed` з вкладеними групами:

```
echo abcde | sed "s/\([a-z]\([a-z]\)\([a-z]\)\)\(\([ a-z]\)\([a-z]\)\)/
\1 \2 \3 \4 \5 \6/"
abc b c de d e
```

2.7. Приклад. Видалити всі пропуски та символи табуляції на початку і вкінці кожного рядка файла file.txt можна командою:

```
sed -e "s/^[[:blank:]]*//" -e "s/[[:blank:]]*$//" file.txt
```

Прапори заміни ставляться після останнього роздільника. Вони визначають дію команди у випадку, коли в рядку знайшлося більше одного збігу із регулярним виразом. Якщо не вказано жодного прапора, то програма замінить лише перше входження регулярного виразу в кожному рядку. Якщо потрібно замінити кожне слово, задане регулярним виразом, то слід використати прапор глобальної заміни g.

2.8. Приклад. Без прапора g

```
echo 0989895779 | sed "s/9/a/"
0a89895779
```

редактор замінив тільки перше входження цифри 9. А тепер з прапором глобальної заміни:

```
echo 0989895779 | sed "s/9/a/g"
0a8a8a577a
```

Усі збіги в цьому рядку були замінені.

Якщо не застосовувати прапорів, то редактор sed замінить тільки перше входження регулярного виразу. Якщо ж використати прапор g, то програма замінить кожен збіг. А як вибрати один зі збігів, якщо їх декілька в рядку? Це можна зробити з допомогою числового прапора, який ставиться після останнього роздільника і вказує, яке по порядку входження підлягає заміні.

2.9. Приклад. Щоб замінити третє входження цифри 9, треба використати команду:

```
echo 0989895779 | sed "s/9/a/3"
09898a5779
```

Числовий прапор може бути будь-яким цілим числом від 1 до 512. Можна комбінувати числовий прапор з прапором g.

2.10. Приклад. Якщо потрібно залишити незмінним перше входження, а друге і наступні видалити, то команда буде така:

```
echo 0989895779 | sed "s/9//2g"
0988577
```

Редактор видалив всі входження цифри 9, починаючи з другого.

Для виводу змінених рядків використовується прапор p (print). Слід зауважити, що за замовчуванням програма `sed` видає на стандартний вивід (наприклад, на екран монітора) всі рядки, тому з прапором p деякі рядки виводитимуться двічі. Зазвичай прапор p використовують з опцією -n, яка блокує стандартне виведення результату. Як наслідок, виводяться тільки ті рядки, в яких зроблено зміни.

2.11. Приклад. Видалити зайві пропуски в усіх рядках файла Metalist.txt (додаток Б), які знаходяться між рядками, що містять цифри 4 та 7 відповідно. Змінені рядки вивести на екран монітора.

Як згадувалось вище, команда з адресою /4/,/7/ застосовується до всіх рядків, починаючи з першого рядка, який містить цифру 4, до наступного першого рядка з цифрою 7, потім знову шукається вказаний діапазон і т.д. Тому дана команда матиме вигляд:

```
sed -n "/4/,/7/s/[[:blank:]]\+/ /gp" Disk:\...\Metalist.txt
```

В результаті на екран монітора будуть виведені рядки:

```
4 DF Andriy Berezovchuk Ukraine
5 MF Oleh Shelayev Ukraine
6 DF Marco Torsiglieri Argentina
7 MF Serhiy Valyayev Ukraine
24 FW Yevhen Budnik Ukraine
27 MF Yuriy Chonka Ukraine
```

Прапори I та i роблять процес заміни нечутливим до регістру символів.

2.12. Приклад. Без прапора i команда

`echo` Night | `sed` s/night/day/

виведе на екран монітора слово Night, а результатом дії команди

`echo` Night | `sed` s/night/day/i

буде слово day.

Символ & (амперсанд), включений до складу регулярного виразу заміни, замінюється на знайдений регулярний вираз. Наприклад, з допомогою команди `sed` "s/[;:?]/& sign &/g" кожен символ ; заміниться на „; sign ;”, : – на „: sign :”, а ? – на „? sign ?”.

2.13. Приклад. Для того щоб взяти в дужки друге слово у кожному рядку файла know.txt, застосовуємо команду:

```
sed "s/[[:alpha:]]\+/(&)/2" know.txt
Knowledge (is) so simple
just (look) around any activity
gain (knowledge) with any
and (try) to give your life a big turn.
```

2.14. Приклад. Щоб подвоїти 50 символ в усіх рядках, починаючи з п'ятого, можна скористатися командою:

```
sed "5,$s/./&&/50" ім'я файла
```

Зауважимо також, що можна використовувати спеціальну послідовність зі слешу і однієї з букв L, l, U, u чи E. Вони мають наступні значення:

- \L – переводить букви заміни в нижній регістр доти, доки не зустріне \U чи \E;
- \l – переводить наступний символ заміни в нижній регістр;
- \U – переводить букви заміни в верхній регістр доти, доки не зустріне \U чи \E;
- \u – переводить наступний символ заміни в верхній регістр;
- \E – відміняє перевід, розпочатий \L чи \U.

2.15. Приклад. Розглянемо наступні команди:

```
echo clever student | sed "s/clever/\u&/"
Clever student
echo clever student | sed "s/[a-z]\+/\u&/2"
clever Student
echo clever student | sed "s/.*/\U&/"
CLEVER STUDENT
echo Clever Student | sed "s/\([[:alpha:]]\+\)\(.*\)/\U\1\E\2/"
CLEVER Student
```

## 3. Функції видалення та друку

Команда d (delete) видаляє зі стандартного виводу зазначені в адресі рядки. Якщо адреса відсутня, то не виводиться жоден рядок. Все те, що було сказано раніше про адресацію рядків, має місце і для команди d (як і для майже всіх команд редактора sed).

3.1. Приклад. Команда

`sed` 2d know.txt

видалить з файла know.txt другий рядок:

```
Knowledge is so simple
gain knowledge with any
and try to give your life a big turn.
```

З допомогою команди

`sed` /now/d know.txt

будуть видалені всі рядки, які містять слово now:

```
just look around any activity
and try to give your life a big turn.
```

3.2. Приклад. Видалити перші 10 рядків з файла file.txt можна командою:

`sed` "1,10d" file.txt

А видалити останній рядок:

`sed` "$d" file.txt

3.3. Приклад. Видалити всі порожні рядки з файла file.txt можна з допомогою будь-якої з двох команд:

`sed` "/^$/d" file.txt або `sed` "/./!d" file.txt

А видалити всі порожні рядки на початку файла можна так:

`sed` "/./,$!d" file.txt

3.4. Приклад. Друкувати всі рядки, вилучивши групи рядків, починаючи від рядка зі словом look до рядка, що містить with:

`sed` "/look/,/with/d" know.txt

```
Knowledge is so simple
and try to give your life a big turn.
```

Команда p (print) виводить вміст робочого буфера. Англійське слово „print” перекладається як „друкувати”, що в українській мові асоціюється з принтером, або, принаймні, з клавіатурою. Насправді ж це слово в англійському контексті найчастіше означає просто виведення на екран монітора. Так що команда p нічого не друкує, а просто виводить на стандартний вивід зазначені рядки. Дана команда подвоює рядки у виводі, бо програма `sed` за замовчуванням виводить рядок, а команда p виводить той же рядок вдруге.

3.5. Приклад. Подвоїти порожні рядки для поліпшення вигляду тексту можна з допомогою команди:

`sed "/^$/p" ім'я файла`

Але „справжнє своє обличчя" команда p розкриває в поєднанні з опцією -n, яка, як ви пам'ятаєте, забороняє виведення рядків на екран. Комбінуючи опцію -n з командою p, можна отримати на виводі тільки потрібні рядки.

3.6. Приклад. Переглянути рядки з першого по десятий можна з допомогою команди:

`sed -n "1,10p" file.txt`

А друкувати останній рядок файла:

`sed -n "$p" file.txt` або `sed "$!d" file.txt`

3.7. Приклад. Друкувати лише ті рядки, які містять слово, задане регулярним виразом:

`sed -n "/регулярний вираз/p"` або `sed "/регулярний вираз/!d"`

А друкувати ті рядки, які не містять вказаного слова:

`sed -n "/регулярний вираз/!p"` або `sed "/регулярний вираз/d"`

3.8. Приклад. Друкувати частину файла між двома регулярними виразами so та any (включно):

```
sed -n "/so/,/any/p" know.txt
Knowledge is so simple
just look around any activity
```

## 4. Функції вставки нових рядків

Команда a\text додає рядок text після вказаних в адресі рядків. Якщо ж адреса не вказана, то виводить даний text після кожного рядка.

Розглянемо дію команди на прикладах:

4.1. Приклад. Нехай вхідні дані заходяться в файлі know.txt.

```
sed "a\it is easy" know.txt
Knowledge is so simple
it is easy
just look around any activity
it is easy
```

```
gain knowledge with any
it is easy
and try to give your life a big turn.
it is easy

sed "/now/a\it is easy" know.txt
Knowledge is so simple
it is easy
just look around any activity
gain knowledge with any
it is easy
and try to give your life a big turn.

sed "/so/,/with/a\it is easy" know.txt
Knowledge is so simple
it is easy
just look around any activity
it is easy
gain knowledge with any
it is easy
and try to give your life a big turn.
```

Команда c\text видаляє вказані рядки і замінює їх на text.

4.2. Приклад. Замінити другий рядок на it is easy можна з допомогою команди:

```
sed "2c\it is easy" know.txt
Knowledge is so simple
it is easy
gain knowledge with any
and try to give your life a big turn.
```

Якщо вказаний діапазон, то команда виводить тільки один екземпляр тексту, а не замінює кожен рядок діапазону:

```
sed "2,3c\it is easy" know.txt
Knowledge is so simple
it is easy
and try to give your life a big turn.
```

Слід зазначити таку особливість: якщо перед командою c була виконана команда a, то текст, який вставляється командою c, вставляється раніше тексту, вставленого з допомогою a.

4.3. Приклад. Розглянемо таку команду:

```
sed -e "2,3a\it is easy" -e "3c\new line" know.txt
Knowledge is so simple
just look around any activity
it is easy
new line
it is easy
and try to give your life a big turn.
```

Розглянемо інший варіант:

```
sed -e "3a\it is easy" -e "2,3c\new line" know.txt
Knowledge is so simple
new line
it is easy
and try to give your life a big turn.
```

Команда i\text виводить на `stdout` рядок text перед вказаними рядками. Вона діє аналогічно, як і команда a:

4.4. Приклад. Вставити перед другим і третім рядком файла know.txt рядок it is easy:

```
sed "2,3i\it is easy" know.txt
Knowledge is so simple
it is easy
just look around any activity
it is easy
gain knowledge with any
and try to give your life a big turn.
```

При використанні команди i разом з командою c, слід враховувати особливість, вказану нами вище при описі команди a.

```
sed -e "2,3i\it is easy" -e "3c\new line" know.txt
Knowledge is so simple
it is easy
just look around any activity
```

```
it is easy
new line
and try to give your life a big turn.
```

## 5. Приклади використання інших функцій

Функція y/str1/str2/ виконує транслітерацію, тобто заміну всіх входженнь символів з str1 на відповідні з str2. Довжини рядків мають бути рівними.

5.1. Приклад. Команда

`echo` acbbcca |`sed` "y/abc/012/"

виконає заміни букв a, b, c на цифри 0, 1, 2 відповідно. В результаті на екран монітора буде виведено число 0211220.

Виконаємо шифрування тих рядків файла know.txt, які містять слово now, шифром зсуву малих букв на дві позиції праворуч:
`sed` /no/y/abcdefghiklmnopqastuvwxyz/cdefghiklmnopqastuvwxyzab/ know.txt

```
Kpqyngf ig lu uq uloang
just look around any activity
itlp mpqyngfig ylvk tpa
and try to give your life a big turn.
```

Функція q припиняє роботу програми `sed` після зазначеного рядка. Це зручно, якщо потрібно припинити редагування після досягнення певного місця в тексті. Наприклад, команда `sed` "11q" закінчить роботу після досягнення 11-го рядка. Функція q – одна з небагатьох функцій `sed`, які не працюють з діапазонами рядків. Не може ж команда припинити роботу 10 раз поспіль, якщо ми введемо `sed` "1,10q"!

Функція r не тільки прочитає вказаний файл, але і вставить його вміст у потрібне місце редагованого файла. Для вибору „потрібного місця” використовується вже знайома нам адресація (за номерами рядків, за виразами, та ін.).

5.2. Приклад. Команда

`echo` KNOWLEDGE|`sed` "r know.txt"

виведе на екран монітора текст:

```
KNOWLEDGE
Knowledge is so simple
```

```
just look around any activity
gain knowledge with any
and try to give your life a big turn.
```

Функція = видає номер зазначеного рядка. Вона не працює з діапазонами рядків.

5.3. Приклад. Перед кожним рядком, який містить слово no, вивести його порядковий номер:

```
sed "/no/=" know.txt
1
Knowledge is so simple
just look around any activity
3
gain knowledge with any
and try to give your life a big turn.
```

Або тільки номери рядків:

```
sed -n "/no/=" know.txt
1
3
```

5.4. Приклад. Для того щоб підрахувати кількість рядків у вхідному файлі, достатньо просто вивести номер останнього рядка:

```
sed -n "$=" know.txt
4
```

Функція G додає вміст допоміжного буфера до робочого буфера, розділивши рядки символом нового рядка $\backslash n$. Оскільки на початку роботи допоміжний буфер містить порожній рядок, то дану функцію зручно використовувати для додавання порожніх рядків у текст.

5.5. Приклад. Вставити порожній рядок після кожного рядка файла know.txt, що містить слово no:

```
sed "/no/G" know.txt
Knowledge is so simple

just look around any activity
gain knowledge with any
```

```
and try to give your life a big turn.
```

Два порожні рядки після кожного рядка вставляються з допомогою команди:

`sed "{G; G}" know.txt`

А команда

`sed "/^$/d; G" know.txt`

вставить порожні рядки після кожного непорожнього рядка.

Функція x міняє місцями вміст допоміжного та робочого буферів.

5.6. Приклад. Вставити порожній рядок перед кожним рядком, який містить no:

`sed "/no/{x; p; x;}" know.txt`

```
Knowledge is so simple
just look around any activity

gain knowledge with any
and try to give your life a big turn.
```

Дана команда спершу міняє вміст буферів: вхідний рядок переноситься в допоміжний буфер, а порожній – в робочий. Потім функція p виводить вміст робочого буфера (порожній рядок). Наостанок, функція x знову міняє вміст буферів і `sed` виводить вміст робочого буфера.

Вставити порожній рядок перед і після кожного рядка, який містить no, можна з допомогою команди:

`sed "/no/{x; p; x; G;}" know.txt`

Функція n виводить вміст робочого буфера, бере наступний рядок з `stdin` і замінює вміст робочого буфера цим рядком.

5.7. Приклад. Видалити кожен другий рядок:

`sed "n; d" know.txt`

```
Knowledge is so simple
gain knowledge with any
```

Дана команда діє наступним чином: програма `sed` заносить перший рядок в робочий буфер, потім функція n виводить вміст буфера (перший рядок) і заносить другий рядок, а функція d видаляє його і т.д.

5.8. Приклад. Видалити кожен восьмий рядок у файлі:

```
sed "n; n; n; n; n; n; n; d;" file.txt
```

Додати порожній рядок через кожні 5 рядків:

```
sed "n; n; n; n; G;" file.txt
```

Починаючи з третього рядка, друкувати кожний сьомий рядок файла People.txt:

```
sed -n "3,${p; n; n; n; n; n; n;}" Disk:\...\People.txt
Anna Alnikina 0505053455 1989
Lesya Gogol 0636200646 1931
Lyuba Nazarova 0974364068 1991
Natalia Slyvchuk 0976683158
Olia Vuykova 0672151856 1965
```

5.9. Приклад. Вивести наступний рядок після рядка, вказаного в адресі, не друкуючи самого рядка з адресою, можна з допомогою команди:

```
sed -n "1{n; p}" know.txt
just look around any activity
```

Функція g замінює вміст робочого буфера вмістом допоміжного, а функція h навпаки – вміст допоміжного буфера вмістом робочого буфера.

5.10. Приклад. Вивести рядок перед рядком, вказаним адресою, не друкуючи самого рядка з адресою, можна з допомогою команди:

```
sed -n "/look/{g; 1!p}; h" know.txt
Knowledge is so simple
```

5.11. Приклад. Видалити останній непорожній рядок в кожному абзаці (порожній рядок розділяє абзаци):

```
sed -n "/^$/{p; h;}; /./{x; /./p;}" file.txt
```

5.12. Приклад. Друкувати передостанній рядок у файлі (для однорядкового файла друкувати порожній рядок):

```
sed -e "$!{h; d;}" -e x know.txt
```

```
gain knowledge with any
```

Функція H додає вміст робочого буфера до допоміжного, розділивши рядки символом нового рядка $\backslash n$.

5.13. Приклад. Вивести рядки вхідного файла в оберненому порядку:

```
sed -n "1h; 1n; x; H; $x; $p" know.txt
```

```
and try to give your life a big turn.
gain knowledge with any
just look around any activity
Knowledge is so simple
```

Розглянемо дію даної команди детальніше. Оскільки на початку роботи допоміжний буфер порожній, то при обробці першого рядка переміщуємо вміст робочого буфера в допоміжний. Далі виводимо в `stdout` вміст робочого буфера (після попередньої команди він порожній), бо якщо цього не зробити, то перший рядок продублюється. Після цього для всіх рядків міняємо місцями вміст робочого і допоміжного буферів та додаємо новий вміст робочого буфера до нового вмісту допоміжного буфера. Після того, як опрацюємо останній рядок, переносимо в робочий буфер вміст допоміжного буфера і виводимо отриманий результат в `stdout`.

5.14. Приклад. Друкувати абзац, якщо він містить регулярний вираз "1986" (порожній рядок розділяє абзаци):

```
sed -e "/./{H; $!d;}" -e "x; /1986/!d;" Disk:\...\Company.txt
Name: Sue Smith
Age:   48
Hire_date: 1986-12-10
Sales: 474050.00

Name: Dan Roberts
Age:   45
Hire_date: 1986-10-20
Sales: 305673.00
```

Функція N бере наступний рядок і додає його до вмісту робочого буфера (не очищаючи його), розділивши рядки символом нового рядка \n.

5.15. Приклад. Об'єднати пари сусідніх рядків у один:

```
sed "$!N; s/\n//" know.txt
Knowledge is so simple just look around any activity
gain knowledge with any and try to give your life a big
turn.
```

Функція D видаляє перший рядок з робочого буфера і починає новий цикл, але новий вхідний рядок не береться, доки робочий буфер не буде порожнім.

5.16. Приклад. Друкувати два останні рядки файла:

```
sed "$!N; $!D" know.txt
gain knowledge with any
and try to give your life a big turn.
```

Видалити два останні рядки файла:

```
sed "N; $!P; $!D; $d" know.txt
Knowledge is so simple
just look around any activity
```

Пригадаємо функції, які працюють з мітками:

- : label – встановлює мітку label для переходу за b і t командами;
- b label – перейти на мітку, що встановлюється за допомогою символу :, якщо label порожній, то перейти в кінець скрипту;
- t label – перейти на мітку, яка встановлюється за допомогою символу :, якщо для цього рядка була виконана заміна з допомогою команди s (прапорець здійснення заміни відновлюється при читанні наступного рядка або виконанні команди s);
- T label – перейти на мітку, яка встановлюється з допомогою символу :, якщо для цього рядка не була виконана заміна з допомогою команди s.

5.17. Приклад. Проаналізуємо наступні команди:

```
sed "b lb; 2d; : lb; 3d" know.txt
Knowledge is so simple
just look around any activity
and try to give your life a big turn.
```

В даній команді з допомогою функції b відбувається перехід на мітку lb, а далі видаляється третій рядок функцією d.

```
sed "1s/a/A/g; t lb; 2d; : lb; 3d" know.txt
Knowledge is so simple
and try to give your life a big turn.
```

Оскільки в першому рядку не виконується заміна (в ньому немає букви a), то перехід з допомогою функції t не відбувається, а тому спершу видаляється другий, а потім третій рядки.

```
sed "1s/a/A/g; T lb; 2d; : lb; 3d" know.txt
Knowledge is so simple
just look around any activity
and try to give your life a big turn.
```

Оскільки в першому рядку не виконується заміна, то відбувається перехід з допомогою функції T і видаляється третій рядок.

5.18. Приклад. Розглянемо команду:

```
sed -n -e "1{h; b eof}" -e "${=; x; p; Q}" -e "H; : eof" know.txt
4
Knowledge is so simple
just look around any activity
gain knowledge with any
```

Вона виводить на екран кількість рядків файла, а потім всі рядки, крім останнього. Алгоритм простий:

- якщо опрацьовуємо перший рядок, то переносимо його в допоміжний буфер (h) і виходимо через мітку eof (b);
- опрацьовуючи останній рядок, виводимо його номер (=), дістаємо все, що є в допоміжному буфері (x), виводимо (p) і закінчуємо (Q);
- по замовчуванню додаємо наступний рядок в допоміжний буфер (H).

5.19. Приклад. Вивести вісім останніх рядків файла People.txt можна з допомогою команди:

```
sed -e ": a" -e "$q; N; 9,$D; b a" Disk:\...\People.txt
Oksana Kotsur 0967105158 1965
Olia Dulevych 0977477058 1983
```

```
Olia Sushaylo 0967314022 1988
Olia Shveya 0976353459 1987
Olia Vuykova 0974306550 1986
Olia Vuykova 0672151856 1965
Taras Saras 0503999999 1990
Vasyl Loboz 0965306862 1988
```

А видалити десять останніх рядків файла:

`sed` -e ": a" -e "$d; N; 2,10b a" -e "P; D" file.txt або

`sed` -n -e ": a" -e "1,10!{P; N; D;}; N; b a" file.txt

## Питання та вправи до розділу 3.

**3.1.** Опишіть дію основних опцій програми `sed`.

**3.2.** Перерахуйте основні функції редактора `sed`.

**3.3.** Які прапори функції контекстної заміни вам відомі? Для чого їх використовують?

**3.4.** Яким чином можна перевести всі букви текстового файла у верхній (нижній) регістр?

**3.5.** З допомогою якої функції редактора `sed` можна виконувати ті ж дії, що й фільтром `grep`?

**3.6.** Чи можна шифрувати інформацію, використовуючи програму `sed`?

**3.7.** Як підрахувати кількість рядків у текстовому файлі?

**3.8.** Яким чином можна вставити певну кількість порожніх рядків у потрібному місці файла?

**3.9.** Замінити друге слово у кожному рядку файла Metalist.txt на своє ім'я і вивести всі рядки у файл name.txt.

**3.10.** Замінити третє входження букви а на ааа у 5-15 рядках файлів Metalist.txt та People.txt.

**3.11.** В кожному рядку, який містить слово Andriy, файлів Metalist.txt та People.txt замінити всі входження букви о на ОО, починаючи з 2 входження.

**3.12.** У всіх рядках файла Company.txt, які містяться між рядком з словом Age і порожнім рядком (включаючи ці рядки), видалити всі пропуски.

**3.13.** Замінити у всіх рядках, починаючи з третього, файла People.txt кожну цифру 6 на 7.

**3.14.** Видалити всі входження цифр 0, 1, 2 та 3 в файлі People.txt та вивести змінені рядки на екран.

**3.15.** Вивести у файл mobile.txt всі номери мобільних телефонів з файла People.txt.

**3.16.** Створити файл, який містить тільки прізвища тих людей з файла People.txt, ім'я яких починається з букви A або N.

**3.17.** У файлі People.txt поміняти місцями прізвище та ім'я людей.

**3.18.** Поміняти місцями 2 та 4 букви прізвища у файлі Metalist.txt.

**3.19.** У файлі People.txt замінити всі послідовності з дев'яток на нуль.

**3.20.** Вивести прізвища тих українців з файла Metalist.txt, ім'я яких містить п'ять літер.

**3.21.** Видалити всі імена людей з файла People.txt, перша літера яких співпадає з останньою.

**3.22.** Вивести номери телефонів тих людей з файла People.txt, в яких прізвище містить більше семи літер.

**3.23.** У файлі People.txt подвоїти імена тих людей, прізвища яких починаються з букв M або N.

**3.24.** Вивести номери футболок всіх форвардів (FW) і захисників (DF) з файла Metalist.txt

**3.25.** В перших десяти рядках файла People.txt видалити всі букви прізвища, починаючи з третьої.

**3.26.** Якщо ім'я у файлі People.txt містить чотири букви, то подвоїти кожну його букву.

**3.27.** У файлі People.txt записати всі номери мобільних телефонів у міжнародному стандарті (додати +38).

**3.28.** Вивести у файл kyivstar.txt прізвища абонентів „Київстар" з файла People.txt.

**3.29.** У файлі People.txt після другого слова у кожному рядку поставити кому.

**3.30.** Потроїти двадцятий символ кожного рядка файла Metalist.txt, який містить натуральне число з відрізка $[15, 99]$.

**3.31.** У кожному рядку файла Metalist.txt видалити зайві пропуски.

**3.32.** Дописати вкінці кожного непорожнього рядка файла Company.txt слово end.

**3.33.** Взяти перші три цифри всіх номерів телефонів з файла People.txt в дужки.

**3.34.** У файлі People.txt записати останні сім цифр всіх номерів телефонів, у яких друга цифра співпадає з останньою, в протилежному порядку.

**3.35.** У файлі Metalist.txt незалежно від регістру букв замінити всі входження ar та se на своє ім'я.

**3.36.** Вивести останні слова тих рядків файла People.txt, які містять поліндроми довжини чотири або п'ять (регістр букв ігнорувати).

**3.37.** Вивести прізвища та імена людей з файла People.txt великими літерами.

**3.38.** Перевести всі символи файла Company.txt в нижній регістр.

**3.39.** Записати прізвища всіх форвардів файла Metalist.txt великими літерами.

**3.40.** У файлі Metalist.txt перевести всі символи, починаючи з десятого, у верхній регістр.

**3.41.** Видалити у файлі People.txt всі рядки, починаючи з третього.

**3.42.** Вивести з файла Metalist.txt рядки з 3 до 8.

**3.43.** Вивести у файл m.txt всі рядки з файлів Company.txt, Metalist.txt та People.txt, у яких третім символом є m.

**3.44.** У файлі People.txt видалити всі рядки, п'ятим символом яких є буква a.

**3.45.** Видалити рядки з інформацією про тих людей з файла People.txt, в яких друга літера прізвища співпадає з передостанньою літерою.

**3.46.** Вивести з файла People.txt у файл lm.txt всі рядки друга буква яких є l або m.

**3.47.** Вивести з файла People.txt всі рядки, у яких п'ятнадцятим символом є цифра 8 або 9.

**3.48.** Вивести ті рядки з файла People.txt, в яких ім'я містить від 6 до 8 літер.

**3.49.** Видалити всі порожні рядки з файла Company.txt.

**3.50.** Видалити всі рядки до першого порожнього включно з файла Company.txt.

**3.51.** Видалити всі рядки з файла Company.txt, які не містять поліндромів довжини три.

**3.52.** Перед кожним рядком з інформацією про закордонного футболіста файла Metalist.txt вставити рядок foreign.

**3.53.** Після кожного рядка файла People.txt, який містить принаймні дві послідовні цифри 6, вставити рядок зі своїми даними.

**3.54.** Перед першим рядком файла Metalist.txt вставити рядок METALIST.

**3.55.** Замінити всі рядки з третього по десятий включно файла People.txt інформацією про себе.

**3.56.** У файлі People.txt поміняти цифри n на 9-n, де n=0,1,..,9.

**3.57.** Зашифрувати всі слова файла People.txt шифром зсуву малих літер на дві позиції ліворуч, а великих – на три праворуч.

**3.58.** Після п'ятого рядка файла People.txt вставити вміст файла Metalist.txt.

**3.59.** Підрахувати кількість рядків у файлі Metalist.txt.

**3.60.** Вивести номери рядків з файла Metalist.txt, у яких міститься інформація про українців.

**3.61.** Після кожного рядка з інформацією про форварда файла Metalist.txt вставити порожній рядок.

**3.62.** Після кожного рядка файла People.txt, який закінчується цифрою 1, вставити порожній рядок.

**3.63.** Виділити кожен рядок файла Metalist.txt, який містить слово Ukraine, порожніми рядками.

**3.64.** Після кожного рядка файла Metalist.txt, який містить цифру 7, вставити два порожніх рядки.

**3.65.** Видалити кожен четвертий рядок файла Metalist.txt, починаючи з третього рядка.

**3.66.** Після кожного п'ятого рядка файла People.txt вставити інформацію про себе.

**3.67.** Виділити кожен шостий рядок файла Metalist.txt порожніми рядками.

**3.68.** Вивести передостанній рядок файла Metalist.txt.

**3.69.** Вивести наступний рядок після кожного рядка, який містить слово Age (не друкуючи самого рядка з Age) в файлі Company.txt.

**3.70.** Вивести рядок перед рядком, який містить слово Age в файлі Company.txt.

**3.71.** Видалити останній непорожній рядок в кожному абзаці файла Company.txt.

**3.72.** Вивести ті абзаци файла Company.txt, які містять цифру 7.

**3.73.** Вивести тільки ті рядки файла People.txt, які містять не менше тридцяти символів.

**3.74.** Вивести десять останніх рядків файла People.txt.

**3.75.** Видалити два останні рядки файла Metalist.txt.

**3.76.** Вивести рядки файла Metalist.txt в протилежному порядку.

**3.77.** Поміняти місцями кожні два рядки файла Metalist.txt.

**3.78.** В чому полягає робота наступних команд:

- `sed` "s/\([a-z]\+\)[[:blank:]]\+\([a-z]\+\)/\2 \1/g" file.txt
- `sed` "/./,/^$/!d" file.txt
- `sed` -e :a -e "$!N; s/\n=/ /; ta" -e "P; D" file.txt
- `sed` -e : a -e "s/\(.* [0-9]\)\([0-9]\{3\}\)/\1,\2/; t a" file.txt ?

Розділ IV

# Регулярні вирази в програмних продуктах LibreOffice Writer та Total Commander

Як ми вже з'ясували в попередніх розділах, на практиці регулярні вирази найчастіше застосовуються для пошуку та обробки тексту. Автори програм, які так чи інакше пов'язані з пошуком і редагуванням тексту, оцінили великі можливості регулярних виразів. І вже більшість таких програм їх підтримують. Оскільки єдиного стандарту для синтаксису регулярних виразів немає, то, на жаль, в більшості випадків навряд чи вдасться обійтися без читання документації до програми. В цьому розділі ми зупинимося на застосуванні регулярних виразів в програмних продуктах LibreOffice Writer та Total Commander.

## 1. Регулярні вирази в редакторі LibreOffice Writer

У текстовому редакторі LibreOffice Writer регулярні вирази застосовуються для пошуку та обробки тексту. Для пошуку та редагування тексту в деякій частині документа потрібно виділити цю частину і відкрити діалогове вікно „Знайти і замінити" з пункту меню „Редагування". Якщо пошук проводиться у всьому документі, то текст можна не виділяти. У діалоговому вікні обрати „Додаткові параметри" і встановити прапорець у поле „Регулярний вираз". Далі у текстове поле „Знайти" ввести регулярний вираз для тексту, який потрібно знайти, а у поле „Замінити на" – текст, який має замінити знайдений текст. Потім потрібно натискати кнопки „Знайти", „Знайти все", „Замінити" або „Замінити все" для пошуку і заміни тексту.

Після натискання кнопки „Знайти" LibreOffice Writer вказує на наступний фрагмент тексту, що відповідає регулярному виразу в

текстовому полі „Знайти”. Текст можна переглядати і редагувати, після чого слід знову натиснути „Знайти” для переходу до наступного знайденого фрагмента. Якщо діалогове вікно було закрите, то можна натиснути комбінацію клавіш (Ctrl+Shift+F) для переходу до наступного фрагмента тексту без повторного виклику діалогового вікна. Як альтернативу для переходу до наступного фрагмента тексту або до будь-якого іншого об’єкта документа, можна використовувати значки в нижній правій частині документа. Після натискання на кнопку „Знайти все”, редактор вказує на всі фрагменти тексту, що відповідають регулярному виразу в полі „Знайти”.

Після клацання на кнопці „Замінити”, LibreOffice Writer шукає заданий у полі пошуку текст у всьому документі, починаючи з поточної позиції курсора. Якщо текст знайдений, то редактор виділяє його і чекає вашої відповіді. Слід натиснути кнопку „Замінити”, щоб замінити виділений в документі текст текстом поля „Замінити на”. Щоб перейти до наступного знайденого тексту без заміни поточного вибору, потрібно клацнути кнопку „Знайти”. Після натискання на кнопку „Замінити все”, редактор замінює весь текст, який відповідає заданому в полі „Знайти” регулярному виразу.

Синтаксис регулярних виразів такий же, як і для програм `grep` і `sed`, за винятком деяких позначень. Для зручності наведемо список всіх метасимволів і вкажемо їх дію:

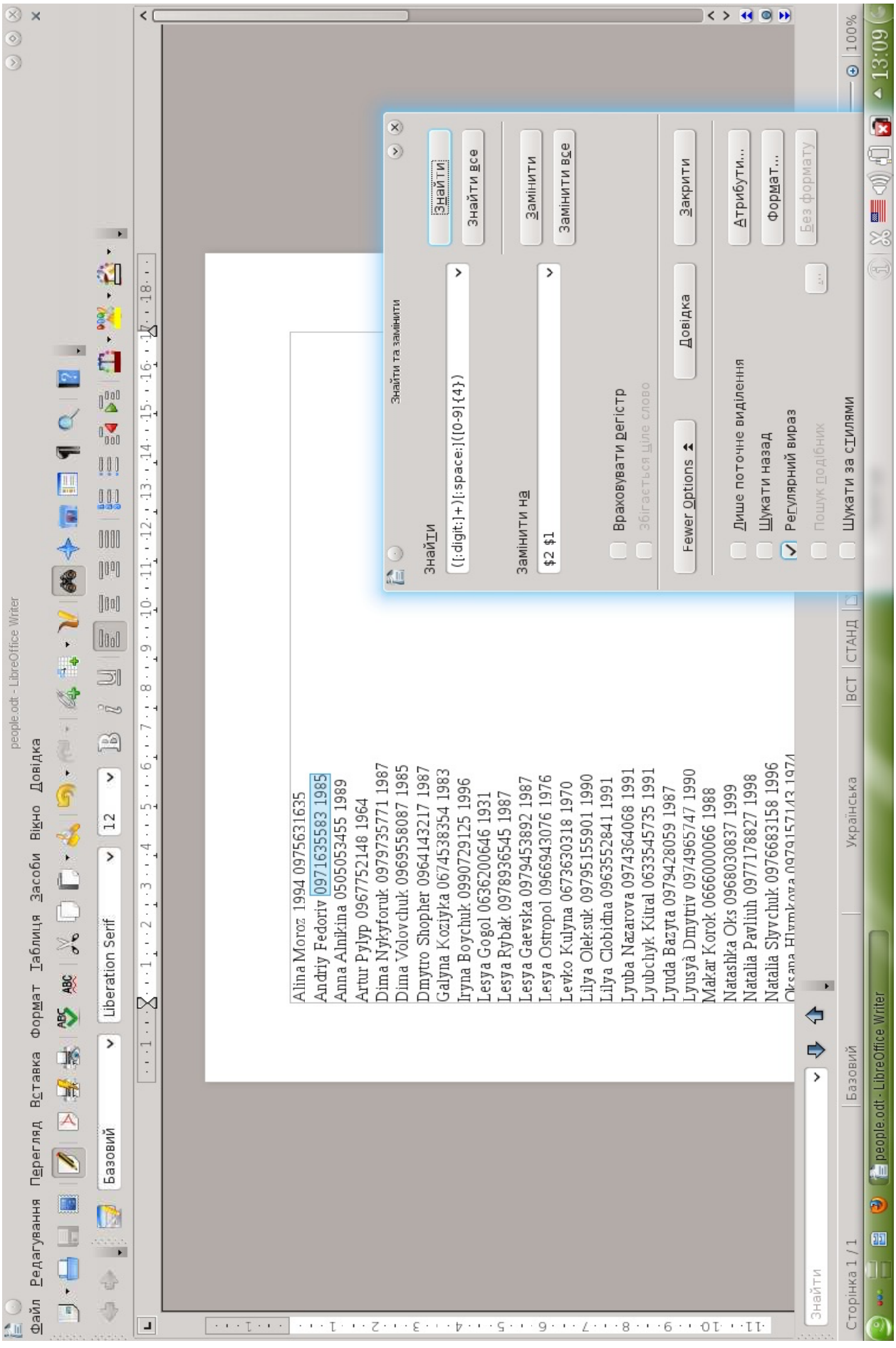
- . – позначає довільний символ, крім символу нового рядка \n;
- ^ – позначає початок абзаца;
- $ – позначає кінець абзаца;
- * – метасимвол ітерації, який вказує на те, що попередній символ або група символів має повторитися нуль або більше разів підряд;
- + – вказує на те, що попередній символ або група символів має повторюватися один або більше разів;
- ? – вказує на те, що попередній символ або група символів міститься рівно один раз або відсутня;
- \ – ставлять перед символом, який зазвичай інтерпретується як метасимвол, щоб використовувати його як звичайний символ (за винятком \n, \t, \> та \<). Наприклад, регулярний вираз „сон\.” знайде „сон.”, а не „соня”;

- \n – позначає символ нового рядка, який був вставлений комбінацією клавіш Shift+Enter. Щоб замінити розрив рядка на розрив абзацу, треба ввести \n в поля „Знайти" і „Замінити на", а потім виконати пошук та заміну. В полі пошуку \n позначає розрив рядка, який був вставлений комбінацією клавіш Shift+Enter, а у полі „Замінити на" – розрив абзацу, який створюється з допомогою клавіші вводу Enter або повернення Return;
- \t – символ табуляції, який можна вживати як у полі пошуку, так і в полі заміни;
- \> – позначає кінець слова. Наприклад, вираз „група\>" знайде слово „напівгрупа", але не знайде – „групах";
- \< – позначає початок слова;
- & або $0 – додає знайдений за допомогою поля „Знайти" текст у вказане у полі „Замінити на" місце. Наприклад, якщо ввести „група" в поле пошуку та „напів&" або „напів$0" в поле заміни, то редактор замінить слово „група" на „напівгрупа";
- | – використовують для пошуку декількох регулярних виразів, розділених |;
- {k} – попередній символ (група символів) має повторитися рівно k разів;
- {k,m} – вказує на те, що попередній символ (група символів) має повторитися не менше ніж k, але не більше, ніж m разів;
- {k,} – вказує на те, що попередній символ (група символів) має повторитися не менше, ніж k разів;
- ( ) – використовують для виділення частини регулярного виразу в групу. Групи номеруються зліва направо, починаючи з номера 1. В текстовому полі „Знайти" можна послатися на групу з номером $k$ з допомогою виразу $\backslash k$. Наприклад, якщо текст містить число 13487889, то з використанням регулярного виразу (8)7\1\1 буде знайдено 8788. Також ( ) можна використовувати для групування символів, наприклад, 1(23)?4 знаходить 14 або 1234. У полі „Замінити на" замість $\backslash k$ використовується $\$k$ для вставки вмісту групи з номером $k$.

Як і в програмах `grep` та `sed`, при написанні регулярних виразів в LibreOffice можна використовувати класи. Символьний клас записується як пара квадратних дужок зі списком символів між ними. Клас є односимвольним шаблоном. Один і лише один з цих символів повинен бути присутнім в тексті, який співпаде з регулярним виразом. Наприклад, ви хочете знайти слова „сон”, „син” та „сян”. За допомогою конструкції [...] можна перелічити усі символи, що можуть бути у даній позиції тексту. Таким чином, вираз „с[оия]н” означає: «Знайти букву „с”, за якою слідує одна з букв „о”, „и” або „я”, а далі записана буква „н” ». У символьному класі передбачається вибір будь-якого символу незалежно від його розміщення. Всередині символьного класу метасимвол – (дефіс) позначає інтервал символів, наприклад, клас [1-6] рівносильний класу [123456]. Інтервали можна успішно комбінувати із звичайними символами. Наприклад, вираз [0-9А-Я:!] співпаде із цифрою, буквою верхнього регістру, символом двох крапок (:) або знаком оклику (!). Якщо замість конструкції [...] поставити конструкцію [^...], то символьний клас буде описувати всі символи, що не входять у нього. Наприклад, вираз [^1-6] описує всі символи, що не належать інтервалу від 1 до 6. Слід зауважити, що для інвертування класу використовується той самий символ ^, що й для позначення початку абзацу, але всередині символьного класу у нього зовсім інша дія.

Для деяких символьних класів використовують спеціально введені позначення:

- [:lower:] – позначає клас малих букв (прапорець у полі „Враховувати регістр” має бути встановлений);
- [:upper:] – клас великих букв (прапорець у полі „Враховувати регістр” має бути встановлений);
- [:alpha:] – клас всіх букв (об’єднання двох попередніх класів);
- [:digit:] – клас цифр;
- [:alnum:] – клас цифр та букв ([:alpha:] та [:digit:]);
- [:space:] – позначає пропуск;
- [:print:] – клас графічних символів;
- [:cntrl:] – клас керуючих символів.


people.odt - LibreOffice Writer
Файл Редагування Перегляд Вставка Формат Таблиця Засоби Вікно Довідка
Базовий
Liberation Serif
12
Alina Moroz 1994 0975631635
Andriy Fedoriv 0971635583 1985
Anna Alnikina 0505053455 1989
Artur Pylyp 0967752148 1964
Dima Nykyforuk 0979735771 1987
Dima Volovchuk 0969558087 1985
Dmytro Shopher 0964143217 1987
Galyna Koziyka 0674538354 1983
Iryna Boychuk 0990729125 1996
Lesya Gogol 0636200646 1931
Lesya Rybak 0978936545 1987
Lesya Gaevska 0979453892 1987
Lesya Ostropol 0966943076 1976
Levko Kulyna 0673630318 1970
Lilya Oleksuk 09795155901 1990
Lilya Clobidna 0963552841 1991
Lyuba Nazarova 0974364068 1991
Lyubchyk Kitral 0633545735 1991
Lyuda Bazyta 0979428059 1987
Lyusyà Dmytriv 0974965747 1990
Makar Korok 0666000066 1988
Natashka Oks 0968030837 1999
Natalia Pavliuh 0977178827 1998
Natalia Slyvchuk 0976683158 1996
Oksana Hlymkova 0979157143 1974
Знайти та замінити
Знайти
([:digit:]+)([:space:])([0-9]{4})
Замінити на
$2 $1
Враховувати регістр
Збігається ціле слово
Знайти
Знайти все
Замінити
Замінити все
Fewer Options
Довідка
Закрити
Дише поточне виділення
Шукати назад
Регулярний вираз
Пошук подібних
Шукати за стилями
Атрибути...
Формат...
Без формату
Знайти
Базовий
Українська
ВСТ
СТАНД
Сторінка 1 / 1
100%
13:09
people.odt - LibreOffice Writer

1.1. Приклад. У кожному рядку файла People.txt (див. додаток В) поміняти місцями номер телефону та рік народження.

Спершу треба відкрити програму LibreOffice Writer, обрати „Відкрити” в меню „Файл” і на локальному диску знайти файл People.txt. Далі відкрити діалогове вікно „Знайти і замінити” з меню „Редагування” (див. мал. вище), обрати „Додаткові параметри” і встановити прапорець у поле „Регулярний вираз”. У текстове поле „Знайти” слід ввести регулярний вираз

([:digit:]+)[:space:]([0-9]{4})

для тексту, який потрібно знайти (тут у першій групі запам’ятовується номер телефону, а в другій – рік народження). Потім у поле „Замінити на” ввести вираз $2 $1, який міняє місцями вміст першої та другої груп. Далі треба натискати кнопки „Знайти”, „Знайти все”, „Замінити” або „Замінити все” для пошуку та заміни тексту.

1.2. Приклад. Після кожного абзацу файла People.txt, який закінчується цифрою 5, 6 або 7, вставити порожній рядок.

Аналогічно, як у попередньому прикладі, треба відкрити діалогове вікно „Знайти і замінити” з меню „Редагування”, обрати „Додаткові параметри” і встановити прапорець у поле „Регулярний вираз”. У текстове поле „Знайти” потрібно ввести регулярний вираз [5-7]$, який позначає всі абзаци, що закінчуються цифрами 5, 6 або 7. Потім у поле „Замінити на” ввести вираз &\n, який вкінці знайдених абзаців вставляє символ нового рядка \n. Далі треба натиснути кнопку „Замінити все” для пошуку та заміни тексту.

1.3. Приклад. Ім’я кожної людини з файла People.txt, прізвище якої містить 5 букв, замінити своїм ім’ям.

У текстове поле „Знайти” діалогового вікна „Знайти і замінити” меню „Редагування” можна ввести регулярний вираз

[:alpha:]+ ([:alpha:]{5})[:space:]+,

з допомогою якого редактор знаходить потрібні імена та прізвища. Потім у поле „Замінити на” слід ввести вираз Volodymyr $1\t і натиснути кнопку „Замінити все” для пошуку та заміни тексту.

## 2. Регулярні вирази в менеджері файлів Total Commander

Синтаксис регулярних виразів у Total Commander такий же, як і в LibreOffice Writer, за винятком того, що він не підтримує стандартних символьних класів та метасимволів \> і \<, натомість підтримує наступні метасимволи:

- \w – алфавітно-цифровий символ або знак підкреслення „_";
- \W – доповнення до \w, тобто будь-який символ, крім алфавітно-цифрових символів і підкреслення;
- \d – цифра;
- \D – доповнення до \d (не цифра);
- \s – роздільник між словами (пропуск, табуляція та ін.);
- \S – доповнення до \s (будь-який символ, крім роздільників);
- \b – вказує межу слова;
- \B – вказує, що задана позиція не є межею слова.

Метасимвол межі слова \b означає, що в тому місці, де він знаходиться, обов'язково повинен бути початок або кінець слова. Метасимвол \B, навпаки, означає, що у вказаному місці межі слова не має бути. Наприклад, регулярний вираз abc\b знайде підслово abc в слові xabc def і не знайде нічого в слові xabcdef. Вираз abc\B, навпаки, нічого не знайде в xabc def і виявить підслово abc в слові xabcdef.

Total Commander підтримує регулярні вирази в наступних функціях:

- при пошуку файлів (в імені та вмісті файла);
- при виділенні файлів за маскою;
- у внутрішній програмі перегляду;
- в інструменті групового перейменування.

Зупинимося на перших двох пунктах детальніше.

Для пошуку файлів в Total Commander з допомогою регулярних виразів потрібно відкрити діалогове вікно „Пошук файлів" з меню „Інструменти" і встановити прапорець у поле „Регулярні вирази". Далі у текстовому полі „Шукати файли" ввести регулярний вираз для імен файлів, які потрібно знайти. У полі „Місце пошуку" вказується каталог, у якому потрібно робити пошук.

Опція „Глибина вкладеності підкаталогів” дозволяє проводити пошук тільки до вказаної глибини вкладення підкаталогів. Слід вибрати потрібне значення в випадаючому списку:

- всі (необмежений) – пошук проводиться по всіх підкаталогах без обмеження глибини;
- тільки поточний – пошук проводиться тільки в поточному каталозі, але не в його підкаталогах;
- кількість рівнів $n$ – пошук проводиться в поточному каталозі та його підкаталогах до $n$-го рівня включно.

Якщо відмічена опція „З текстом”, то в дане текстове поле можна ввести текст, який має міститися в шуканих файлах. При цьому можна також встановити прапорець у відвідне цій опції поле „Регулярні вирази” і задати потрібний вміст файла з допомогою регулярних виразів. Також тут можна вказати опції „Враховувати регістр символів”, „Файли, що не містять цей текст”, „Тільки слова цілком” та ін.

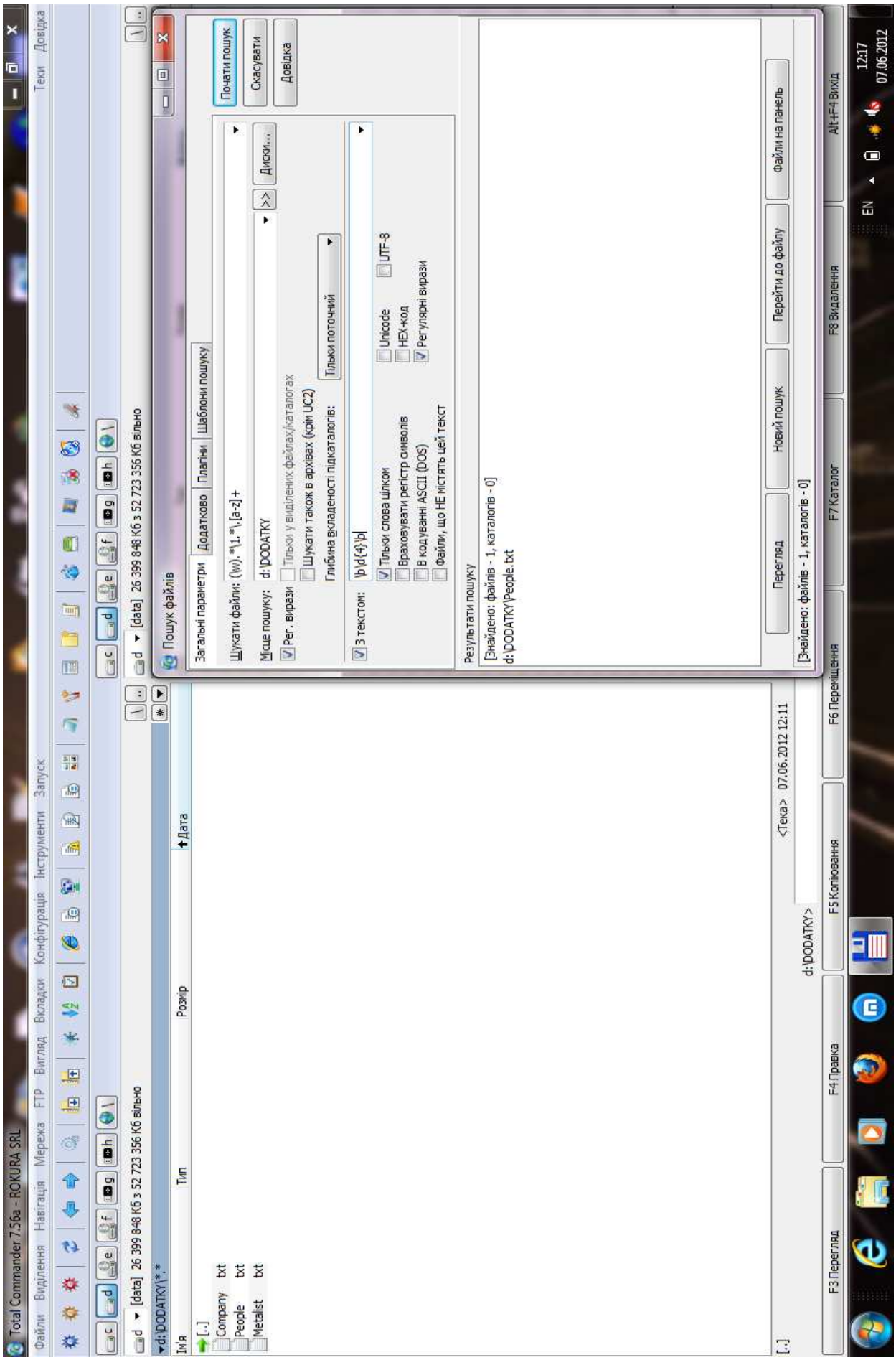
Після того, як всі необхідні опції відмічені, потрібно натиснути кнопку „Почати пошук” і у вікно „Результати пошуку” будуть виведені шукані імена файлів.

2.1. Приклад. Знайти всі файли каталогу DODATKY, які містять чотирицифрові числа, і в їхньому імені деяка буква, цифра або знак підкреслення зустрічається двічі.

У текстове поле „Шукати файли” діалогового вікна „Пошук файлів” меню „Інструменти” потрібно ввести регулярний вираз

$$(\backslash w).*\backslash 1.*\backslash .[a\text{-}z]+,$$

який задає потрібні імена файлів, та в полі „Місце пошуку” вказати шлях до каталогу DODATKY. Для підтримки регулярних виразів при заданні імен файлів треба встановити прапорець „Регулярні вирази”. Далі в текстове поле „З текстом” слід ввести регулярний вираз \b\d{4}\b, який шукатиме у файлах чотирицифрові числа, відмітивши при цьому опції „Тільки слова цілком” та „Регулярні вирази”. Наостанок натиснути кнопку „Почати пошук” і у вікно „Результати пошуку” буде виведений шлях до файла People.txt, який задовольняє вимоги пошуку (див. мал. нижче).


Total Commander 7.56a - ROKURA SRL
Файли Виділення Навігація Мережа FTP Вигляд Вкладки Конфігурація Інструменти Запуск Теки Довідка
Пошук файлів
Загальні параметри
Додатково
Плагіни
Шаблони пошуку
Шукати файли: (\w).*1.*\.[a-z]+
Місце пошуку: d:\DODATKY
Рег. вирази
Тільки у виділених файлах/каталогах
Шукати також в архівах (крім UC2)
Глибина вкладеності підкаталогів:
Тільки поточний
З текстом: \b\d{4}\b
Тільки слова цілком
Враховувати регістр символів
В кодуванні ASCII (DOS)
Файли, що НЕ містять цей текст
Unicode
HEX-код
Регулярні вирази
UTF-8
Диски...
Почати пошук
Скасувати
Довідка
Результати пошуку
[Знайдено: файлів - 1, каталогів - 0]
d:\DODATKY\People.txt
Перегляд
Новий пошук
Перейти до файлу
Файли на панель
[Знайдено: файлів - 1, каталогів - 0]
d:\DODATKY*.*
Ім'я
Тип
Розмір
Дата
[..]
Company txt
People txt
Metalist txt
<Тека> 07.06.2012 12:11
d:\DODATKY>
F3 Перегляд
F4 Правка
F5 Копіювання
F6 Перемещення
F7 Каталог
F8 Видалення
Alt+F4 Вихід
EN
12:17
07.06.2012

Для виділення файлів за маскою в Total Commander з допомогою регулярних виразів потрібно обрати „Виділити групу” у меню „Виділення”, при цьому відкриється діалогове вікно „Додати виділення”. Натиснувши на кнопку „Шаблон”, відкриємо діалогове вікно „Вибір шаблону”, яке ідентичне вікну „Пошук файлів” меню „Інструменти”. У ньому потрібно задати регулярні вирази для імен та вмісту файлів, натиснути кнопку „Записати” та задати ім’я шаблону. Насамкінець у вікні „Додати виділення” обрати потрібну назву шаблону та натиснути кнопку OK.

2.2. Приклад. Виділити всі файли каталогу DODATKY, які містять слово з п’яти букв, і в їхньому імені є не менше семи символів.

Спершу потрібно обрати „Виділити групу” у меню „Виділення”, при цьому відкриється діалогове вікно „Додати виділення”. Натиснувши на кнопку „Шаблон”, відкриємо діалогове вікно „Вибір шаблону”. У текстове поле „Шукати файли” діалогового вікна „Вибір шаблону” потрібно ввести регулярний вираз \b\w{7,}\.[a-z]+, який задає потрібні імена файлів, та для підтримки регулярних виразів при заданні імен файлів треба встановити прапорець „Регулярні вирази”. Далі в текстове поле „З текстом” слід ввести регулярний вираз \b[a-zA-Z]{5}\b, який шукатиме у файлах п’ятибуквені слова, відмітивши при цьому опцію „Регулярні вирази”. Потім натиснути кнопку „Записати” і назвати шаблон у полі „Ім’я шаблону”. У вікні „Додати виділення” обрати потрібну назву шаблону та натиснути кнопку OK. В результаті файли Company.txt та Metalist.txt будуть виділені.

## Питання та вправи до розділу 4.

**4.1.** Вкажіть основні відмінності в синтаксисі регулярних виразів у програмах `grep`, LibreOffice Writer та Total Commander.

**4.2.** Опишіть, як застосовуються регулярні вирази для пошуку та заміни тексту в текстовому редакторі LibreOffice Writer.

**4.3.** Які функції підтримують регулярні вирази в файловому менеджері Total Commander?

Використовуючи текстовий редактор LibreOffice Writer:

**4.4.** Замінити друге слово у кожному абзаці файла Metalist.txt на своє ім'я.

**4.5.** У файлі People.txt подвоїти кожне третє слово.

**4.6.** Знайти всі пари однакових послідовних слів у файлі People.txt та видалити зайві слова.

**4.7.** Після кожного абзацу, який закінчується цифрою 5 або 7, файла Metalist.txt вставити порожній рядок.

**4.8.** У файлі Metalist.txt незалежно від регістру букв замінити всі входження ar та se на своє ім'я.

**4.9.** У другому та третьому абзаці файла Metalist.txt видалити всі пропуски.

**4.10.** У файлі People.txt поміняти місцями прізвище та ім'я людей.

**4.11.** Поміняти місцями другу та четверту букви прізвищ у файлі Metalist.txt.

**4.12.** У файлі People.txt замінити всі послідовності з дев'яток на нуль.

**4.13.** Знайти всі слова файла People.txt, перша літера яких співпадає з останньою.

**4.14.** У файлі People.txt подвоїти імена тих людей, прізвища яких починаються з букв M або N.

**4.15.** В перших десяти рядках файла People.txt видалити всі букви прізвища, починаючи з третьої.

**4.16.** У файлі People.txt записати всі номери мобільних телефонів у міжнародному стандарті (додати +38).

**4.17.** Дописати вкінці кожного непорожнього абзацу файла Company.txt слово end.

**4.18.** Взяти перші три цифри всіх номерів телефонів з файла People.txt в дужки.

**4.19.** У файлі People.txt записати перші 5 цифр всіх номерів телефонів, у яких друга цифра співпадає з останньою, в протилежному порядку.

**4.20.** Знайти всі поліндроми довжини чотири або п'ять (регістр букв ігнорувати) у файлі People.txt.

Використовуючи програму Total Commander:

**4.21.** Знайти всі файли каталогу DODATKY, третя або четверта буква імені яких є голосною.

**4.22.** Знайти всі файли каталогу DODATKY, які містять числа 74 та 47 одночасно.

**4.23.** Знайти всі файли каталогу DODATKY, які містять чотирицифрові поліндроми, і в їхньому імені ще раз міститься перша буква.

**4.24.** Виділити всі файли каталогу DODATKY, четверта буква імені яких не є голосною.

**4.25.** Виділити всі файли каталогу DODATKY, які містять слово довжини 11.

**4.26.** Виділити всі файли каталогу DODATKY, що місятять хоча б один абзац, в якому число 19 зустрічається двічі.

**4.27.** Виділити всі файли каталогу DODATKY, які містять ціле число з відрізка [10-35], і в їхньому імені п'ятою або шостою буквою є а.

# Додатки

## Додаток А: Company.txt

```
Name: Mary Jones
Age:   31
Hire_date: 1989-10-12
Sales: 392725.00

Name: Sue Smith
Age:   48
Hire_date: 1986-12-10
Sales: 474050.00

Name: Sam Clark
Age:   52
Hire_date: 1988-06-14
Sales: 299912.00

Name: Bob Smith
Age:   33
Hire_date: 1987-05-19
Sales: 142594.00

Name: Dan Roberts
Age:   45
Hire_date: 1986-10-20
Sales: 305673.00
```

## Додаток Б: Metalist.txt

| | | | |
|---|---|---|---|
| 1 | GK | Maksym Startsev | Ukraine |
| 2 | DF | Oleksandr Romanchuk | Ukraine |
| 3 | DF | Cristian Villagra | Argentina |
| 4 | DF | Andriy Berezovchuk | Ukraine |
| 5 | MF | Oleh Shelayev | Ukraine |
| 6 | DF | Marco Torsiglieri | Argentina |
| 7 | MF | Serhiy Valyayev | Ukraine |
| 9 | FW | Andriy Vorobey | Ukraine |
| 10 | MF | Cleiton Xavier | Brazil |
| 11 | MF | Jose Ernesto Sosa | Argentina |
| 17 | DF | Serhiy Pshenychnykh | Ukraine |
| 18 | MF | Dmytro Yeremenko | Ukraine |
| 19 | MF | Juan Manuel Torres | Argentina |
| 21 | FW | Jonathan Cristaldo | Argentina |
| 22 | DF | Milan Obradovic | Serbia |
| 23 | MF | Sebastian Blanco | Argentina |
| 24 | FW | Yevhen Budnik | Ukraine |
| 27 | MF | Yuriy Chonka | Ukraine |
| 29 | GK | Oleksandr Goryainov | Ukraine |
| 30 | DF | Papa Gueye | Senegal |
| 33 | FW | Marko Devich | Ukraine |
| 37 | DF | Vitalie Bordian | Moldova |
| 71 | MF | Sergei Tkachyov | Russia |
| 80 | DF | Lukas Stetina | Slovakia |
| 81 | GK | Vladimir Disljenkovic | Ukraine |
| 99 | MF | Artem Radchenko | Ukraine |

## Додаток В: People.txt

```
Alina Moroz 0975631635 1994
Andriy Fedoriv 0971635583 1985
Anna Alnikina 0505053455 1989
Artur Pylyp 0967752148 1964
Dima Nykyforuk 0979735771 1987
Dima Volovchuk 0969558087 1985
Dmytro Shopher 0964143217 1987
Galyna Koziyka 0674538354 1983
Iryna Boychuk 0990729125 1996
Lesya Gogol 0636200646 1931
Lesya Rybak 0978936545 1987
Lesya Gaevska 0979453892 1987
Lesya Ostropol 0966943076 1976
Levko Kulyna 0673630318 1970
Lilya Oleksuk 09795155901 1990
Lilya Clobidna 0963552841 1991
Lyuba Nazarova 0974364068 1991
Lyubchyk Kitral 0633545735 1991
Lyuda Bazyta 0979428059 1987
Lyusya Dmytriv 0974965747 1990
Makar Korok 0666000066 1988
Nanashka Oks 0968030837 1999
Natalia Pavliuh 0977178827 1998
Natalia Slyvchuk 0976683158 1996
Oksana Hlymkova 0979157143 1974
Oksana Kotsur 0967105158 1965
Olia Dulevych 0977477058 1983
Olia Sushaylo 0967314022 1988
Olia Shveya 0976353459 1987
Olia Vuykova 0974306550 1986
Olia Vuykova 0672151856 1965
Taras Saras 0503999999 1990
Vasyl Loboz 0965306862 1988
```

# Список літератури


1. Ахо А. Теория синтаксического анализа, перевода и компиляции / А. Ахо, Дж. Ульман. – М.: Мир, 1978. – Т. 1. – 611 с.
2. Бєлов Ю.А. Інструментальні засоби програмування: навчальний посібник / Ю.А. Бєлов, В.С. Проценко, П.Й. Чаленко. – К.: Либідь, 1993. – 248 с.
3. Бондаренко М.Ф. Комп'ютерна дискретна математика: підручник / М.Ф. Бондаренко, Н.В. Білоус, А.Г. Руткас. – Харків: «Компанія СМІТ», 2004. – 480 с.
4. Гаврилків В.М. Формальні мови та алгоритмічні моделі: навчальний посібник / В.М. Гаврилків. – Івано-Франківськ: «Сімик», 2012. – 172 с.
5. Мозговой М.В. Классика программирования: алгоритмы, языки, автоматы, компиляторы. Практический подход / М.В. Мозговой. – СПб.: Наука и Техника, 2006. – 320 с.
6. Нікольський Ю.В. Дискретна математика / Ю.В. Нікольський, В.В. Пасічник, Ю.М. Щербина. – К.: Видавнича група BHV, 2007. – 368 с.
7. Смит Б. Методы и алгоритмы вычислений на строках / Б. Смит. – М.: Издательство «Вильямс», 2006. – 496 с.
8. Форта Б. Освой самостоятельно регулярные выражения. 10 минут на урок / Б. Форта. – М.: Издательство «Вильямс», 2005. – 184 с.
9. Фридл Дж. Регулярные выражения, 3-е издание / Дж. Фридл. – СПб.: Символ-Плюс, 2008. – 608 с.
10. Холзнер С. Perl: специальный справочник / С. Холзнер. – СПб: Издательство «Питер», 2000. – 496 с.
11. Bambenek J. grep Pocket Reference / John Bambenek and Agnieszka Klus. – O'Reilly Media, 2009 – 75 p.
12. Goyvaerts J. Regular Expressions Cookbook / Jan Goyvaerts and Steven Levithan. – O'Reilly Media, 2009 – 494 p.

Навчальне видання

**Володимир Михайлович Гаврилків**

# РЕГУЛЯРНІ ВИРАЗИ У ПРОГРАМНИХ ПРОДУКТАХ

Підписано до друку 12.09.2012. Формат 60×84/16.
Папір офсетний. Друк цифровий.
Гарнітура CM Roman. Умовн. друк. арк. 4,2.
Тираж 100. Зам. № 86 від 12.09.2012.

Віддруковано: Приватний підприємець О.М. Голіней
76000, м. Івано-Франківськ,
вул. Галицька,128,
тел.: (0342) 58-04-32.