

## Домашнє завдання №1

З використанням `std::regex` розробити програму, яка шукає в тексті лексеми заданого формату.

### Вибір варіанту

$$(N_{\text{ж}} + N_{\text{г}} + 1) \% 30 + 1$$

де:  $N_{\text{ж}}$  – порядковий номер студента в групі, а  $N_{\text{г}}$  – номер групи (1,2,3,4,5,6,7 або 8)

### Варіанти завдання

№	Формат лексем
1	Up-Low2, перший символ Up
2	Low-Up2, перший символ Low
3	Up2
4	Low2
5	Up-Low4, перший символ Up
6	Low-Up4, перший символ Low
7	Up4
8	Low4
9	Up-Low6, перший символ Up
10	Low-Up6, перший символ Low
11	Up6
12	Low6
13	Up-Low8, перший символ Up
14	Low-Up8, перший символ Low
15	Up8
16	Low8
17	Up-Low2, перший символ _
18	Low-Up2, перший символ _
19	Up4, перший символ _
20	Low4, перший символ _
21	Up-Low4, перший символ _
22	Low-Up4, перший символ _
23	Up6, перший символ _
24	Low6, перший символ _
25	Up-Low6, перший символ _
26	Low-Up6, перший символ _
27	Up8, перший символ _
28	Low8, перший символ _
29	Up-Low8, перший символ _
30	Low-Up8, перший символ _

## Приклад коду

Лістинг

```
#include <fstream>
#include <iostream>
#include <algorithm>
#include <iterator>
#include <regex>

int main()
{
    std::string text =
        "Sir, in my heart there was a kind of fighting "
        "That would not let me sleep. Methought I lay "
        "Worse than the mutines in the bilboes. Rashly- "
        "And prais'd be rashness for it-let us know "
        "Our indiscretion sometimes serves us well ... "
        ; // – Hamlet, Act 5, Scene 2, 4-8

    std::regex token_re("(^)[s][a-z']+|[\\s\\.\\,\\:\\;][s][a-z']+",
std::regex::icase);
    std::copy(std::sregex_token_iterator(text.begin(), text.end(), token_re, 0),
        std::sregex_token_iterator(),
        std::ostream_iterator<std::string>(std::cout, "\n"));

    return 0;
}
```

## Регулярні фрази

**Метасимвол** (англ. metacharacter) – це символ або комбінація символів, яка має спеціальне значення, якщо вона зустрічається у складі регулярного виразу. Найпростіший приклад метасимволів – символи узагальнення.

**Символ узагальнення** (англ. wildcard character) – символ, який дає змогу робити у рядку підстановку інших символів, щоб в одному запиті шукати, наприклад, одразу багато файлів. Зазвичай з цією метою вживають символи «\*» (заміняє довільне число інших символів) та «?» (заміняє один довільний символ).

**Регулярний вираз** (англ. regular expression, regexp) – це зразок (англ. pattern) (послідовність символів та метасимволів), який відповідає (або не відповідає) послідовностям символів у тексті. Як правило, регулярні вирази використовуються для того, щоб задати правило пошуку підрядка у тексті.

Регулярні вирази використовуються не самі по собі, а разом з якоюсь мовою програмування або застосуванням. Синтаксис регулярних виразів може дещо відрізнятися, в залежності від того, як саме вони використовуються.

**Метасимволи:**

**.** (крапка) – один довільний символ (крім символу переходу на новий рядок)

**\t** – табуляція

**\n** – новий рядок

**\\** – власне \ (backslash)

**\s** – один довільний пробільний символ (пробіл, табуляція, новий рядок)

**\S** – один довільний символ, який не входить у перелічені для \s

**\d** – одна довільна цифра (digit)

**\D** – один довільний символ, який не може бути цифрою

**\w** – одна довільна літера або цифра або знак підкреслювання (word character), те саме що [A-Za-z0-9\_]

**\W** – один довільний символ, який не входить у перелічені для \w

**-** – позначає діапазон(у класі символів) або власне цей символ, якщо це перший символ у класі ([ -abc])

**\$** – кінець рядка

**^** – початок рядка або заперечення, якщо це перший символ у класі символів ([^abc])

**^\$** – пустий рядок

**|** - логічне АБО (використовується у групі символів)

**\<** – початок слова

**\>** – кінець слова

**\b** – межа слова (початок або кінець) або символ backspace, якщо він знаходиться у класі символів

**\B** – позиція, що не є межею слова

**\.** – власне крапка

**\\$** – власне символ \$

**\[, \]** – власне квадратні дужки

**\(, \)** – власне круглі дужки

**\{, \}** – власне фігурні дужки

**Групування та повторення:**

**\*** – нуль або більше разів повторений попередній символ (або група символів)

**+** – один або більше разів повторений попередній символ (або група символів)

**?** – нуль або один раз повторений попередній символ (або група символів)

**()** – групують символи (всі, що присутні у дужках, можливе застосування логічного АБО (символ |))

**[]** – визначають клас (або множину) символів – неупорядковану групу символів, з якої для відповідного регулярного виразу обирається один довільний

**{n}** – рівно n разів повторений попередній символ (або група символів)

**{n, m}** – попередній символ (або група символів) повторений від n до m разів

**{n, }** – n або більше разів повторений попередній символ (або група символів)

У випадках типу (a|ab) можливо, що другий варіант взагалі не знаходиться, хоча він насправді є (тому що алгоритм пошуку використовує скорочену форму АБО – як тільки значення логічного виразу встановлене, подальша обробка не проводиться – тобто знайшли перший (коротший) варіант і заспокоїлися). Вихід: «довший» варіант треба писати на початку. При визначенні діапазонів символів слід враховувати, що вважається, що символи беруться з таблиці ASCII або Unicode (з якої саме, буде визначатися конкретним випадком використання).

### Приклади

**[A-Za-z]** – один довільний символ латинської абетки, незалежно від регістру (тут не можна писати [A-z], бо між літерами у великому та малому регістрах знаходяться інші символи)

**[0-9]** або **(0|1|2|3|4|5|6|7|8|9)** або **\d** – одна довільна цифра

**\(\d{3}\) \d{3}-\d{4}** – номер телефону у форматі (044) 123-4567

**#[0-9a-fA-F]{6}** – шістнадцятковий код кольору (наприклад, #12CCAA)

**([0-9]|[1-9][0-9]|1[1-9][0-9]|2[0-4][0-9]|25[0-5])** – довільне число з діапазону 0-255 (цей вираз можна скоротити)