Підручник з KatePart

Thad McGinnis
Anne-Marie Mahfouf
Anders Lund
T.C. Hollingsworth
Christoph Cullmann
Lauri Watts
Переклад українською: Юрій Чорноіван

Зміст

Анотація

KatePart — повноцінний компонент редактора для програм, створений розробниками KDE.

Розділ 1

Вступ

KatePart — повноцінний компонент текстового редактора, що використовується у багатьох програмах на основі бібліотек Qt™ та KDE. KatePart — не просто текстовий редактор; компонент призначено для програмістів, його можна розглядати принаймні як часткову альтернативу потужнішим редакторам. Однією з основних можливостей KatePart є можливості розфарбовування синтаксичних конструкцій, налаштовані на багато різних мов програмування, зокрема: C/C++, Java™, Python, Perl, Bash, Modula 2, HTML та Ada.

KWrite — простий текстовий редактор, заснований на KatePart. У програмі передбачено однодокументний інтерфейс (SDI), за допомогою якого можна одночасно редагувати лише один файл у кожному з вікон програми. Оскільки KWrite є дуже простою реалізацією KatePart, власна документація програмі не потрібна. Якщо ви можете користуватися KWrite, ви можете використовувати KatePart будь-де!

Розділ 2

Дещо з основ роботи

Програма KWrite та багато інших програм, де використовується KatePart, є дуже простою у користуванні. Всі, хто колись користувався текстовим редактором, не матимуть з цією програмою ніяких проблем.

2.1 Перетягування і скидання

KatePart використовує протокол перетягання і скидання KDE. Ви можете перетягувати і скидати файли у вікно KatePart зі стільниці, інструменту для роботи з файлами Dolphin або деякого віддаленого сайта FTP, відкритому у одному з вікон Konqueror.

2.2 Клавіатурні скорочення

Більшість клавіатурних скорочень можна налаштувати за допомогою меню Параметри. Типово KatePart використовує такі клавіатурні скорочення:

Ins	Перемикає програму між режимами вставки і перезапису. У режимі вставки редактор додає будь-які введені символи до тексту, одночасно посуваючи дані, що знаходяться праворуч від курсора. У режимі перезапису введення кожного символу призводить до того, що цей символ заміщує символ, який знаходився праворуч від курсора.
←	Пересуває курсор на одну позицію ліворуч.
→	Пересуває курсор на одну позицію праворуч.
↑	Пересуває курсор на один рядок вгору.
↓	Пересуває курсор на один рядок вниз.
Ctrl+**E**	Перейти до попереднього місця редагування у документі.
Ctrl+**Shift**+**E**	Перейти до наступного місця редагування у документі.

Alt+**Shift**+↑	Пересунути курсор до попереднього відповідного відступу.
Alt+**Shift**+↓	Пересунути курсор до попереднього відповідного відступу.
Ctrl+**6**	Перейти до відповідної дужки.
PgUp	Пересуває курсор на одну сторінку вгору.
PgDn	Пересуває курсор на одну сторінку вниз.
Home	Пересуває курсор на початок рядка.
End	Пересуває курсор у кінець рядка.
Ctrl+**Home**	Перейти на початок документа.
Ctrl+**End**	Перейти до кінця документа.
Ctrl+↑	Прокрутити на один рядок вгору.
Ctrl+↓	Прокрутити на один рядок вниз.
Ctrl+→	Пересунути слово праворуч.
Ctrl+←	Пересунути слово ліворуч.
Ctrl+**Shift**+↑	Пересунути рядки вище.
Ctrl+**Shift**+↓	Пересунути рядки нижче.
Ctrl+**.**	Здублювати позначені рядки нижче.
Ctrl+**B**	Встановити закладку.
Alt+**PgUp**	Попередня закладка.
Alt+**PgDn**	Наступна закладка.
Del	Вилучає символ праворуч від курсора (або будь-який виділений текст).
Backspace	Вилучає символ ліворуч від курсора.
Ctrl+**Del**	Вилучити слово праворуч.
Ctrl+**Backspace**	Вилучити слово ліворуч.
Ctrl+**K**	Вилучити рядок.
Shift+**Enter**	Вставити новий рядок з початковими символами поточного рядка, які не є літерами або цифрами. Корисно, наприклад, якщо ви пишете коментарі до коду: в кінці рядка «// певний текст» вам просто слід натиснути цю комбінацію клавіш і новий рядок почнеться з вже готового «// ». Отже, вам не потрібно буде вводити символи коментарів на початку кожного нового рядка з коментарями.
Ctrl+**Shift**+**Enter**	Створити рядок під поточним рядком.
Ctrl+**Alt**+**Enter**	Створити рядок над поточним рядком.
Shift+←	Додає до виділеного тексту символ ліворуч від курсора.
Shift+→	Додає до виділеного тексту символ праворуч від курсора.
Ctrl+**F**	Знайти.
F3	Знайти наступне.
Shift+**F3**	Знайти попереднє.
Ctrl+**H**	Знайти вибране.
Ctrl+**Shift**+**H**	Знайти вибране позаду.
Ctrl+**Shift**+→	Вибрати слово праворуч.
Ctrl+**Shift**+←	Вибрати слово ліворуч.
Shift+**Home**	Вибрати до початку рядка.
Shift+**End**	Вибрати до кінця рядка.
Shift+↑	Вибрати до попереднього рядка.

Shift+↓	Вибрати наступний рядок.
Ctrl+**Shift**+**6**	Вибрати до відповідної дужки.
Ctrl+**Shift**+**PgUp**	Вибрати до початку перегляду.
Ctrl+**Shift**+**PgDn**	Вибрати до кінця перегляду.
Shift+**PgUp**	Вибрати сторінку вгору.
Shift+**PgDn**	Вибрати сторінку вниз.
Ctrl+**Shift**+**Home**	Вибрати до початку документа.
Ctrl+**Shift**+**End**	Вибрати до кінця документа.
Ctrl+**Home**	Вибрати все.
Ctrl+**Shift**+**A**	Скасувати вибір.
Ctrl+**Shift**+**B**	Прямокутний режим вибору.
Ctrl+**C** / **Ctrl**+**Ins**	Копіювати виділений текст до буфера.
Ctrl+**D**	Коментувати.
Ctrl+**Shift**+**D**	Розкоментувати.
Ctrl+**G**	Перейти до рядка...
Ctrl+**I**	Додати відступ до виділеного тексту.
Ctrl+**Shift**+**I**	Прибрати відступ для виділеного тексту.
Ctrl+**J**	Об'єднати рядки.
Ctrl+**P**	Надрукувати.
Ctrl+**R**	Замінити.
Ctrl+**S**	Викликає команду **Зберегти**.
Ctrl+**Shift**+**S**	Зберегти з новою назвою.
Ctrl+**U**	Верхній регістр.
Ctrl+**Shift**+**U**	Нижній регістр.
Ctrl+**Alt**+**U**	З великої літери.
Ctrl+**V** / **Shift**+**Ins**	Вставляє текст з буфера до рядка, що редагується.
Ctrl+**X** / **Shift**+**Ins**	Копіює виділений текст до буфера і вилучає його.
Ctrl+**Z**	Скасувати.
Ctrl+**Shift**+**Z**	Повторити.
Ctrl+**-**	Зменшити шрифт.
Ctrl++**Ctrl**+=	Збільшити шрифт.
Ctrl+**Shift**+**-**	Згорнути вузли найвищого рівня.
Ctrl+**Shift**++	Розгорнути вузли найвищого рівня.
Ctrl+**Пробіл**	Активізувати автозавершення коду.
F5	Перезавантажити.
F6	Показувати або ховати рамку піктограм.
F7	Перемикнутись до командного рядка.
F9	Показувати або ховати мітки згортання.
F10	Динамічне перенесення рядків.
F11	Показати або сховати номери рядків.
Ctrl+**T**	Транспонувати символи.
Ctrl+**Shift**+**O**	Автоматична перевірка правопису.
Ctrl+**Shift**+**V**	Перемкнутися на наступний режим введення.
Ctrl+**8**	Повторно використати слово вище.
Ctrl+**9**	Повторно використати слово нижче.
Ctrl+**Alt**+#	Розгорнути скорочення.
Ctrl+**Alt**+↑	Додати курсор над поточним курсором.
Ctrl+**Alt**+↓	Додати курсор під поточним курсором.
Shift+**Alt**+**I**	Створити курсор наприкінці кожного рядка у позначеному.

Alt+**J**	Знайти наступне входження слова під курсором і позначити його.
Ctrl+**Alt**+**Shift**+**J**	Знайти усі входження слова під курсором і позначити їх.

Розділ 3

Робота у редакторі KatePart

Anders Lund
Dominik Haumann
Переклад українською: Юрій Чорноіван

3.1 Огляд

Редактор KatePart — це область редагування у вікні KatePart. Цей редактор також використовується у Kate і KWrite, його також можна використовувати у Konqueror для показу текстових файлів, які зберігаються на вашому комп'ютері або у мережі.

Вікно редактора складається з таких компонентів:

Області редактора

Тут знаходитиметься текст вашого документа.

Смужки гортання

На смужках гортання позначається позиція видимої частини тексту документа, ви можете скористатися нею для пересування області перегляду документом. Пересування області перегляду за допомогою смужки гортання не призводитиме до пересування курсора у документі.

Смужки гортання може бути сховано або показано за вимогою користувача.

Смужка піктограм

Смужка піктограм — це невеличка панель з лівого боку редактора, на якій буде показано маленьку піктограму поряд з кожним з позначених рядків.

Ви можете встановити або вилучити закладку на одному з видимих рядків наведенням вказівника миші на позицію смужки піктограм навпроти цього рядка з наступним клацанням лівою кнопкою миші.

Увімкнути або вимкнути показ смужки піктограм можна за допомогою пункту меню **Перегляд** → **Показувати смужку піктограм**.

Панель номерів рядків

На панелі номерів рядків буде показано номери рядків всіх видимих рядків документа.

Увімкнути або вимкнути показ панелі номерів рядків можна за допомогою пункту меню **Перегляд** → **Показати номери рядків**.

Панель згортання

За допомогою панелі згортання ви зможете згортати або розгортати блоки рядків документа. Визначення у документі рядків, придатних для згортання, відбуватиметься відповідно до правил визначення підсвічування синтаксису у цьому документі.

Крім того, у цій главі ви познайомитеся з:

- Тим, як пересуватися текстом
- Тим, як працювати з позначеним текстом
- Тим, як копіювати і вставляти текст
- Тим, як шукати і замінювати текст
- Тим, як користуватися закладками
- Тим, як автоматично розбивати текст на рядки
- Тим, як користуватися автоматичним встановленням відступів

3.2 Пересування текстом

Навігація текстом документа у KatePart подібна до навігації у більшості текстових редакторів з графічним інтерфейсом. Пересувати курсор можна за допомогою клавіш зі стрілками та клавіш **Page Up**, **Page Down**, **Home** і **End**, ці клавіші можна комбінувати з клавішами-модифікаторами, **Ctrl** і **Shift**. Клавіша-модифікатор **Shift** завжди використовується для позначення тексту, а клавіша **Ctrl** має різне призначення у комбінаціях з іншими клавішами:

- Для клавіш ↑ і ↓ додавання **Ctrl** призводить до гортання без пересування курсора.
- Для клавіш ← і → додавання **Ctrl** призводить до пересування курсора кроками у слово, а не у символ.
- Для клавіш **Page Up** і **Page Down** додавання **Ctrl** призводить до пересування до видимого краю перегляду, а не навігації текстом.
- Для клавіш **Home** і **End** додавання **Ctrl** призводить до переходу на початок або кінець документа, а не на початок або кінець рядка.

Крім того, у KatePart передбачено можливість переходу до відповідної круглої або фігурної дужки: розташуйте курсор поряд з однією з дужок всередині фрагмента тексту, який обмежують ці дужки, і натисніть комбінацію клавіш **Ctrl**+**6**, щоб перевести курсор до парної цій дужці дужки.

Ви також можете скористатися закладками для пришвидшення переходу до вказаних вами власноруч місць у документі.

3.3 Робота з позначеним текстом

Існує два основних способи позначення тексту у KatePart: за допомогою миші і за допомогою клавіатури.

Щоб виділити фрагмент тексту за допомогою миші, натисніть і утримуйте ліву кнопку миші з одночасним перетягування вказівника миші від місця, де має розпочинатися позначений фрагмент, до бажаного місця завершення позначення, після завершення позначення відпустіть кнопку. Фрагмент позначеного тексту буде підсвічуватися під час позначення.

Наведення вказівника миші на слова з наступним подвійним клацанням лівою кнопкою миші призведе до позначення цього слова.

Наведення вказівника миші на рядок з наступним потрійним клацанням лівою кнопкою миші призведе до позначення цілого рядка.

Якщо під час клацання утримувати натиснутою клавішу **Shift**, буде позначено текст:

- від позиції курсора до позиції вказівника миші, якщо нічого ще не було позначено.

- від початку позначеного тексту до позиції вказівника миші, якщо вже було позначено певний текст

> **Примітка**
> За використання позначення тексту за допомогою перетягування вказівника миші, позначений вами текст буде скопійовано до буфера обміну інформацією, — ви зможете вставити цей текст клацання середньою кнопкою миші у потрібному місці редактора або будь-якої іншої програми, куди ви бажаєте вставити цей текст.

Щоб обрати фрагмент тексту за допомогою клавіатури, натисніть і утримуйте клавішу **Shift**, а потім скористайтеся клавішами навігації (клавішами зі стрілочками, **Page Up**, **Page Down**, **Home** і **End**, можливо у сполученні з клавішею **Ctrl** для використання додаткових можливостей пересування текстового курсора) для позначення фрагмента тексту.

Ознайомтеся також з розділом Навігація текстом цієї глави.

Щоб скопіювати поточний позначений фрагмент тексту, скористайтеся пунктом меню **Зміни** → **Копіювати** або відповідним клавіатурним скороченням (типовим скороченням є комбінація клавіш **Ctrl**+**C**).

Щоб скасувати вибір поточного фрагмента тексту, скористайтеся пунктом меню **Зміни** → **Скасувати вибір** або відповідним клавіатурним скороченням (типовим скороченням є **Ctrl**+**Shift**+**A**), ви також можете просто навести вказівник миші на довільне місце у області редагування і один раз клацнути лівою кнопкою миші.

3.3.1 Користування режимом прямокутного вибору

Якщо увімкнено режим прямокутного вибору, ви зможете виконувати «вибір тексту по вертикалі», тобто обирати символи на певних позиціях (певних стовпчиках) у декількох рядках. Цим режимом вибору зручно користуватися, наприклад, у рядках з роздільниками-табуляціями.

Увімкнути режим прямокутного вибору можна за допомогою пункту меню **Зміни** → **Режим прямокутного вибору**. Типовим клавіатурним скороченням для цієї дії є комбінація клавіш **Ctrl**+**Shift**+**B**.

3.3.2 Використання перезапису позначеного

Якщо увімкнено режим перезапису позначеного фрагмента, введення або вставлення тексту у позначений фрагмент призведе до заміни позначеного фрагмента тексту введеним або вставленим. Якщо цей режим буде вимкнено, новий текст буде додано за поточною позицією курсора.

Типово, режим перезапису позначеного увімкнено.

Увімкнути або вимкнути цей режим можна за допомогою сторінки Курсор та вибір діалогового вікна налаштування програми.

3.3.3 Використання стійкого вибору

Якщо буде увімкнено режим постійного позначення, введення символів або пересування курсора не призводитиме до скасування позначення фрагмента тексту. Це означає, що ви зможете вивести курсор з позначеного фрагмента і ввести до документа текст з клавіатури.

Типово, режим постійного позначення вимкнено.

Увімкнути або вимкнути режим постійного вибору можна за допомогою сторінки Курсор та вибір діалогового вікна налаштування програми.

> Застереження
> Якщо буде одночасно увімкнено режими постійного вибору і перезапису, введення або вставлення тексту у той момент, коли курсор знаходитиметься всередині позначеного фрагмента тексту призведе до заміни і скасування вибору відповідного фрагмента.

3.4 Копіювання і вставка тексту

Щоб скопіювати фрагмент тексту, виділіть цей фрагмент і скористайтеся пунктом меню **Зміни** → **Копіювати**. Крім того, фрагмент тексту, позначений за допомогою миші, автоматично копіюється до буфера обміну інформацією сервера X.

Щоб вставити фрагмент тексту, який знаходиться у буфері обміну інформацією, скористайтеся пунктом меню **Зміни** → **Вставити**.

Крім того, текст, позначений за допомогою миші, може бути вставлено наведенням вказівника миші на місце, куди слід вставити фрагмент, з наступним клацанням середньою кнопкою миші.

> Підказка
> Якщо ви користуєтеся стільничним середовищем KDE, ви можете отримати доступ до списку фрагментів тексту, скопійованих у інших програмах, за допомогою піктограми системного лотка для програми Klipper.

3.5 Пошук і заміна тексту

3.5.1 Панелі пошуку і заміни

У KatePart передбачено покрокову панель пошуку і потужну панель пошуку з заміною, за допомогою якої ви зможете ввести рядок замінник і отримати доступ до деяких додаткових параметрів.

На панелях ви знайдете такі загальні пункти:

Знайти
: У це поле ви можете ввести рядок, який слід знайти. Те, яким чином програма сприйматиме цей рядок, залежатиме від параметрів, які описано далі за текстом.

Враховувати регістр
: Якщо буде позначено цей пункт, пошук обмежуватиметься рядками, регістр символів яких (верхній або нижній), збігатиметься з регістром символів у шаблоні пошуку.

На панелях пошуку і заміни з додатковими можливостями ви знайдете декілька додаткових пунктів:

Простий текст
: Буквальний пошук всіх відповідників вказаного вами рядка пошуку.

Цілі слова
: Якщо буде позначено цей пункт, збіг шаблону з рядком у тексті буде зареєстровано лише у випадку, коли рядок у тексті не межує з будь-якої зі сторін з буквенно-цифровим символом, тобто межує з будь-якими іншими символами або символом завершення рядка.

Керівні послідовності

Якщо буде позначено цей пункт, програма увімкне пункт меню **Додати**, розташований у нижній частині контекстного меню текстових панелей, і надасть вам змогу давати керівні послідовності з наперед визначеного списку до шаблону пошуку.

Формальний вираз

Якщо буде позначено цей пункт, рядок шаблону пошуку буде вважатися рядком формального виразу. Програма увімкне пункт меню **Додати**, розташований у нижній частині контекстного меню текстових панелей, і надасть вам змогу додати елементи формального виразу з попередньо визначеного списку до шаблону пошуку.

Докладніші відомості можна отримати з розділу Формальні вирази.

Шукати лише у позначеному фрагменті

Якщо буде позначено цей пункт, пошук з заміною буде виконано лише у межах позначеного фрагмента тексту.

Знайти всі

Натискання цієї кнопки призведе до підсвічування всіх відповідників у документі та показу їх кількості у невеличкій контекстній підказці.

3.5.2 Пошук тексту

Щоб знайти фрагмент тексту, відкрийте панель покрокового пошуку за допомогою комбінації клавіш **Ctrl**+**F** або пункту меню **Зміни** → **Пошук...**.

Цей пункт відкриває додаткову панель пошуку, розташовану внизу вікна редактора. Ліворуч на цій панелі ви побачите піктограму для закриття панелі, за якою розташовано невеличке поле для введення шаблону пошуку.

Пошук буде розпочато негайно після того, як ви почнете вводити символи вашого шаблону пошуку. Якщо буде знайдено відповідник у тексті, його буде підсвічено, а колір тла поля запису буде змінено на світло-зелений. Якщо шаблон пошуку не відповідає жодному з рядків тексту, програма продемонструє це зміною кольору тла поля запису на світло-червоний.

Скористайтеся кнопкою  or для переходу до наступного або попереднього відповідника пошуку у документі.

Відповідники у документі буде підсвічено навіть після закриття панелі пошуку. Щоб зняти підсвічування, натисніть клавішу **Esc**.

Ви можете визначитися з тим, чи слід виконувати пошук з врахуванням регістру символів.

Позначення пункту  обмежить варіанти відповідності записами з точною відповідністю регістру для кожного з символів у ключі пошуку.

Натисніть кнопку  на правому краю додаткової панелі пошуку, щоб перемкнути панель у стан потужного пошуку і заміни.

Щоб повторити останню дію з пошуку, якщо така була, без виклику панелі покрокового пошуку, скористайтеся пунктом меню **Зміни** → **Знайти далі** (**F3**) або **Зміни** → **Знайти позаду** (**Shift**+**F3**).

3.5.3 Заміна тексту

Щоб замінити текст, відкрийте панель пошуку з заміною за допомогою пункту меню **Зміни** → **Замінити** або комбінації клавіш **Ctrl**+**R**.

У лівій верхній частині панелі ви побачите кнопку-піктограму, за допомогою якої можна закрити панель, за нею можна побачити невеличкий рядок для введення шаблону пошуку. У рядку передбачено запам'ятовування використаних шаблонів.

Ви можете керувати режимом пошуку вибором одного з режимів **Простий текст**, **Цілі слова**, **Керівні послідовності** і **Формальний вираз** за допомогою спадного списку.

Якщо позначено принаймні один з пунктів **Керівні послідовності** або **Формальні вирази**, програма увімкне пункт меню **Додати...** у нижній частині контекстного меню текстових панелей. Це надасть вам змогу додавати керівні послідовності або формальні вирази у шаблони пошуку або заміни за допомогою списку шаблонів.

Скористайтеся кнопкою  or  для переходу до наступного або попереднього відповідника пошуку у документі.

Текст, на який слід буде замінити ключ пошуку, слід ввести у поле для введення тексту з міткою **Замінити**, після чого слід натиснути кнопку **Замінити**, щоб замінити лише поточний підсвічений елемент, або кнопку **Замінити всі**, щоб замінити ключ пошуку у всьому документі.

Ви можете змінити поведінку програми під час пошуку з заміною позначенням відповідних пунктів у правій частині панелі. За допомогою кнопки  можна обмежити пошук елементами, у яких регістр (верхній або нижній) кожного символу збігається з ключем пошуку. За допомогою кнопки  можна обмежити пошук з заміною лише позначеним фрагментом тексту. Якщо ви позначите пункт **Знайти всі**, програма позначить кольором всі відповідники ключа пошуку у документі і покаже кількість знайдених відповідників на невеличкій контекстній панелі.

Натисніть кнопку  , розташовану у правій частині панелі потужного пошуку і заміни, щоб перемкнутися на звичайну нагромаджувальну панель пошуку.

> Підказка
> Якщо ви використовуєте формальний вираз для пошуку тексту, який слід замінити, ви можете використати зворотні посилання для повторного використання знайденого тексту в вигляді виразів підшаблонів у круглих дужках.
> Докладніше про це можна дізнатися з розділу Формальні вирази.

> Підказка
> Ви можете виконати команди **find** (знайти), **replace** (замінити) і **ifind** (покроковий пошук) за допомогою командного рядка.

3.6 Використання закладок

За допомогою закладок ви зможете позначити певні рядки, щоб їх було легше знайти згодом.

Вилучити або встановити закладку на певному рядку можна у два способи:

- Пересунути курсор вставки на відповідний рядок і скористатися пунктом меню **Закладки** → **Встановити закладку** (**Ctrl**+**B**) .
- Навести вказівник миші на місце на смужці піктограм, розташоване поряд з відповідним рядком, і клацнути лівою кнопкою миші.

Доступ до закладок можна отримати у меню **Закладки**. Окремі закладки виглядатимуть у меню як пункти, позначені номером рядка, на який вказує закладка і декількома першими літерами тексту цього рядка. Щоб пересунути курсор на початок рядка закладки, вам достатньо буде відкрити меню і обрати у ньому пункт цієї закладки.

Для пришвидшення руху між позиціями закладок або переходу до наступної/попередньої закладки ви можете скористатися пунктами меню **Закладки** → **Наступна** (**Alt**+**PgDown**) or **Закладки** → **Попередня** (**Alt**+**PgUp**).

3.7 Автоматичне розбиття тексту на рядки

За допомогою цього пункту ви зможете наказати програмі виконати елементарне форматування тексту: текст буде розбито на рядки так, щоб довжина кожного з рядків, у яких присутні пробіли, не перевищувала встановленої максимальної кількості символів у рядку.

Щоб увімкнути або вимкнути цю можливість, позначте або зніміть позначку з пункту **Статичне перенесення слів** на сторінці редагування діалогового вікна налаштування програми.

Щоб встановити максимальну довжину рядка (максимальну кількість символів у рядку), скористайтеся пунктом **Переносити слова на** на сторінці Редагування діалогового вікна налаштування програми.

Якщо увімкнено таке розбиття, буде виконано такі дії:

- Під час набору тексту редактор автоматично розриватиме рядок після останнього символу пробілу, якщо загальна кількість символів у рядку перевищить вказану максимальну довжину рядка.
- Під час завантаження документа редактор розіб'є рядки тексту подібним же чином: у тексті не залишиться рядків, довжина яких перевищуватиме максимальну довжину рядка, якщо, звичайно, у цих рядках міститимуться пробіли, які надаватимуть можливість такого розбиття.

> **Примітка**
> У програмі поки що не передбачено встановлення окремого способу перенесення рядків для кожного з типів документів і вмикання або вимикання цієї можливості для окремого документа у програмі. Цей недолік буде виправлено у наступній версії KatePart.

3.8 Використання автоматичних відступів

Компонент текстового редактора KatePart підтримує декілька режимів автоматичного встановлення відступів для текстів у різних форматах. Ви можете обрати один з можливих режимів за допомогою меню **Інструменти** → **Відступ**. Модулі встановлення відступів забезпечують функціонування пункту меню **Інструменти** → **Форматований відступ**, за допомогою якого буде повторно встановлено відступи для позначених або поточного рядка. Таким чином, ви можете переобчислити всі відступи у документі, якщо виконаєте позначення всього тексту у документі, а потім скористаєтеся останнім пунктом меню.

Всі режими відступу використовують відповідні параметри, встановлені для активного документа.

> **Підказка**
> Ви можете встановлювати значення всіх змінних налаштування, зокрема тих зі змінних, які стосуються відступів, за допомогою налаштування Змінних документа і Типів файлів.

Можливі режим автоматичного відступу

Немає
: Вибір цього пункту призведе до повного вимикання автоматичного відступу.

Звичайний
: За використання цього режиму відступів програма зберігатиме у кожному наступному рядку відступ, який був у попередньому рядку, для всього вмісту цього рядка, окрім початкових пробілів. Ви можете скористатися цим режимом, разом з діями зі встановлення і скасування відступів, для створення відступів за вашим смаком.

C Style

Інструмент встановлення відступів для C-подібних мов програмування, таких як C++, C#, java, javascript тощо. Цей інструмент встановлення відступів не працюватиме з кодами скриптових мов, на зразок Perl або PHP.

Haskell

Інструмент додавання відступів для функціональної мови програмування Haskell.

Lilypond

Інструмент додавання відступів для нотних записів у позначеннях Lilypond.

Lisp

Інструмент додавання відступів для скриптової мови Lisp та її діалектів.

Python

Інструмент додавання відступів для скриптової мови Python.

Стиль XML

Інструмент додавання відступів для XML-подібних мов.

3.9 Індикатори змін у рядках

За допомогою індикаторів змінених рядків у KatePart вам буде простіше стежити за зміненими рядками у файлі. Типово, збережені зміни позначатимуться зеленими прямокутниками на лівому полі документа, а незбережені — жовтогарячим прямокутником.

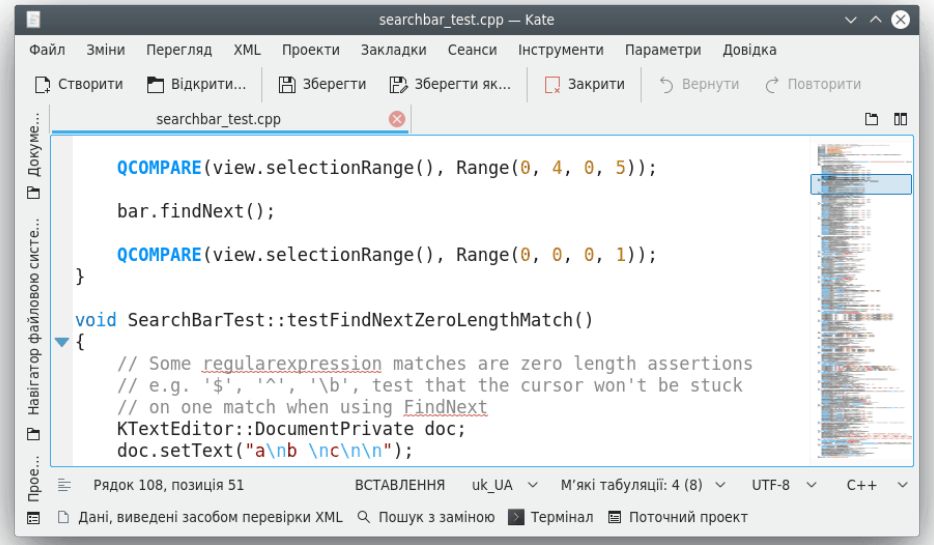
```
// new block for empty buffer I write new stuff HERE
andNewStuffHere = 1;
if (newLineChangedAgain)
  lala ();
more = 1;
afterSaveEdited ();

TextBlock *newBlock = new TextBlock (this, 0);
newBlock->appendLine (TextLine (new TextLineData()));
```

Індикатори змін у рядках.

Змінити використані кольори можна за допомогою панелі налаштовування **Шрифти та кольори**. Крім того, цю можливість можна повністю вимкнути за допомогою вкладки **Межі** сторінки налаштовування **Вигляд**.

3.10 Мінікарта на смужці гортання

На мінікарті на смужці гортання у KatePart буде показано попередній перегляд вмісту документів замість самої смужки гортання. Поточну видиму частину документа буде позначено кольором.



На мінікарті показано попередній перегляд коду Kate.

Тимчасово увімкнути або вимкнути мінікарту можна за допомогою пункту меню **Перегляд → Показувати мінікарту на смужці гортання**. Остаточне налаштовування можна виконати за допомогою сторінки «Вигляд» налаштувань KatePart.

3.11 Декілька курсорів

Підтримку декількох курсорів було впроваджено у версії 5.93 katepart.

3.11.1 Створення декількох курсорів

- Щоб створити курсори за допомогою миші, скористайтеся комбінацією **Alt** + клацання лівою кнопкою миші. Модифікатор можна налаштувати, див. налаштовування модифікатор мультикурсора.
- Щоб створити курсор за допомогою клавіатури, натисніть **Ctrl**+**Alt**+↑, щоб створити курсор над основним курсором, або **Ctrl**+**Alt**+↓, щоб створити курсор під ним. Ці скорочення також можна налаштування
- Щоб створити курсори з позначеного, спочатку позначте якийсь фрагмент тексту, а потім натисніть комбінацію **Shift**+**Alt**+**I**. У результаті буде створено курсор наприкінці кожного з рядків у позначеному фрагменті.
- Скористайтеся комбінацією клавіш **Alt**+**J** для пошуку наступного входження слова під курсором, його позначення та створення курсора. Якщо ви хочете пропустити поточне слово під курсором, натисніть комбінацію клавіш **Alt**+**K**, і програма позначить поточне слово як пропущене. Якщо ви натиснете комбінацію клавіш **Alt**+**J** ще раз, програма зніме позначення з поточного слова і перейде до наступного слова.
- Скористайтеся комбінацією клавіш **Ctrl**+**Alt**+**Shift**+**J**, щоб знайти усі входження слова під курсором і позначити їх курсором наприкінці кожного позначеного фрагмента. Для циклічного переходу між позначеними словами ви можете скористатися комбінацією клавіш **Alt**+**J**, а для зняття позначення з будь-якого зі слів — комбінацією клавіш **Alt**+**K**.

3.11.2 Робота з декількома курсорами

Коли ви створити декілька курсорів, ви зможете виконувати більшість дій з редагування з ними так, як ви виконуєте дії з одним курсором. Наприклад, введення літери призведе до вставляння цієї літери для кожного з курсорів. Так само, ви зможете виконувати перетворення тексту, наприклад, перетворення літер на великі в усіх позиціях або позначених фрагментах.

Іноді, може виникнути потреба у вилученні курсорів. Для цього ви можете скористатися комбінацією **Alt** + ліва кнопка миші на курсорі, який ви хочете вилучити. Якщо ви хочете вилучити курсори на порожніх рядках, для цього вже є готова дія. Щоб викликати цю дію, відкрийте панель команд за допомогою комбінації клавіш **Ctrl**+**Alt**+**I**, знайдіть пункт **Вилучити курсори з порожніх рядків** і натисніть **Enter**. Ви також можете налаштувати клавіатурне скорочення для цієї дії.

Розділ 4

Пункти меню

4.1 Меню «Файл»

Файл → Створити (Ctrl+N)

За допомогою цього пункту можна створити новий документ у новому незалежному вікні редактора.

Файл → Нове вікно

Створює ще одне вікно з поточним документом. Всі зміни у документі, що редагується у одному з вікон, повторюватимуться у іншому і навпаки.

Файл → Відкрити... (Ctrl+O)

Відкриває стандартне діалогове вікно KDE **Відкрити файл**. Скористайтеся областю перегляду файлів, щоб обрати потрібний вам файл, а потім натисніть кнопку **Відкрити**, щоб відкрити його.

Файл → Відкрити недавні

Це клавіатурне скорочення призначено для відкриття документів, які ви зберігали нещодавно. Натискання цього пункту відкриє список, розташований збоку від меню, у якому ви побачите назви декількох збережених нещодавно файлів. Натискання позначки одного з цих файлів відкриє відповідний файл у KatePart, якщо файл все ще знаходиться за старою адресою.

Файл → Зберегти (Ctrl+S)

За допомогою цієї дії можна зберегти поточний документ. Якщо цей документ вже було збережено, ця дія призведе до перезапису попереднього збереженого файла без погодження з користувачем. Якщо документ зберігається вперше, буде відкрито діалогове вікно збереження (його описано нижче).

Файл → Зберегти як... (Ctrl+Shift+S)

За допомогою цього пункту можна зберегти файл з новою назвою. Ця дія виконується за посередництвом діалогового вікна, описаного раніше у розділі цього підручника щодо дії Відкрити.

Файл → Зберегти з іншим кодуванням

Зберегти документ із новою назвою файла і іншим кодуванням.

Файл → Зберегти копію як

Зберегти копію документа у файлі з новою назвою та продовжити редагування початкового документа.

Файл → Перезавантажити (F5)

Перезавантажує поточний файл з диска. Цією командою зручно користуватися, якщо інша програма або процес змінили файл, який ви відкрили у KatePart.

Файл → Друкувати... (Ctrl+P)

Відкриває просте діалогове вікно друку, яке надає користувачеві можливість вказати що, де і як друкувати.

Файл → Експортувати як HTML

Зберегти поточний відкритий документ як файл HTML, який буде форматовано з використанням поточного режиму підсвічування синтаксичних конструкцій та параметрів схеми кольорів.

Файл → Закрити (Ctrl+W)

За допомогою цієї команди можна закрити вікно редагування активного файла. Якщо виконані вами зміни не було збережено, програма запитає вас про те, чи слід зберегти файл, перш ніж KatePart закриє його.

Файл → Вийти (Ctrl+Q)

Цей пункт меню закриє вікно редактора, якщо запущено декілька екземплярів KatePart за допомогою пунктів меню **Створити** або **Нове вікно**, ці екземпляри закрито не буде.

4.2 Меню «Зміни»

Зміни → Вернути (Ctrl+Z)

Скасувати останню команду редагування (введення тексту, копіювання, вирізання тощо)

> Примітка
> За допомогою цього пункту можна скасувати одразу декілька команд редагування, які належать до одного типу, наприклад введення символів.

Зміни → Повторити (Ctrl+Shift+Z)

Цей пункт дозволить вам повторити найостаннішу зміну (якщо така була) виконану за допомогою пункту «Вернути».

Зміни → Вирізати (Ctrl+X)

Ця команда копіює поточний вибраний фрагмент до буфера і вилучає його з тексту. Буфер — елемент системи, який працює як фоновий процес для переносу даних між програмами.

Зміни → Копіювати (Ctrl+C)

Цей пункт меню призводить до копіювання вибраного тексту до буфера, отже, ви зможете вставити його у іншому місці. Буфер — елемент системи, який працює як фоновий процес для переносу даних між програмами.

Зміни → Копіювати як HTML

Копіювати позначений фрагмент тексту як код HTML, який буде форматовано з використанням поточного режиму підсвічування синтаксичних конструкцій та параметрів схеми кольорів.

Зміни → Вставити (Ctrl+V)

Цей пункт меню дасть вам змогу вставити перший пункт буфера обміну даними у позицію курсора. Буфер — елемент, який працює як фоновий процес для переносу даних між програмами.

> **Примітка**
> Якщо буде увімкнено перезапис виділеного тексту, вставлений текст замінить собою будь-який виділений текст, якщо такий існує у вашому документі.

Зміни → Вставити позначене (Ctrl+Shift+Ins)

За допомогою цього пункту можна вставити до тексту раніше позначений вміст буфера позначення мишею. Позначте який текст за допомогою вказівника миші і вставте його до поточного відкритого файла, скориставшись цим пунктом.

Зміни → Поміняти місцями із вмістом буфера обміну даними

За допомогою цього пункту можна поміняти місцями позначений фрагмент текст із вмістом буфера обміну даними.

Зміни → Вставлення з журналу буфера

Цей пункт відкриває вікно для вибору і вставлення запису з журналу буфера обміну даними.

Зміни → Режими введення

Перемикає програму між звичайним і подібним до vi модальним режимом редагування. У режимі введення vi передбачено підтримку найвживаніших команд і пересування курсора зі звичайного і візуального режимів vim, також передбачено можливість вмикання панелі стану режиму vi. На цій панелі стану буде показано команди під час їх введення, вивід команд і поточний режим. Поведінку програми у цьому режимі можна налаштувати на вкладці Режим вводу Vi на сторінці **Редагування** діалогового вікна параметрів KatePart.

Зміни → Режим перезапису (Ins)

Перемикає ввід у програмі між режимами Вставки і Заміни. Якщо програма перебуває у режимі **ВСТ**, символ, який ви введете буде записано за позицією курсора. У режимі **ЗАМ** введені символи перезапишуть поточні символи, якщо курсор було розташовано всередині тексту. На панелі стану поточний режим введення буде показано у вигляді записів **ВСТ** або **ЗАМ**.

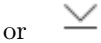
Зміни → Режим лише для читання

Переводить поточний документ у режим «лише для читання». Цей режим призначено для запобігання будь-якому додаванню тексту та будь-якій зміні форматування документа.

Зміни → Пошук... (Ctrl+F)

Цей пункт відкриває додаткову панель пошуку, розташовану внизу вікна редактора. Ліворуч на цій панелі ви побачите піктограму для закриття панелі, за якою розташовано невеличке поле для введення шаблону пошуку.

Пошук буде розпочато негайно після того, як ви почнете вводити символи вашого шаблону пошуку. Якщо буде знайдено відповідник у тексті, його буде підсвічено, а колір тла поля запису буде змінено на світло-зелений. Якщо шаблон пошуку не відповідає жодному з рядків тексту, програма продемонструє це зміною кольору тла поля запису на світло-червоний.

Скористайтеся кнопкою  or  для переходу до наступного або попереднього відповідника пошуку у документі.

Відповідники у документі буде підсвічено навіть після закриття панелі пошуку. Щоб зняти підсвічування, натисніть клавішу **Esc**.

Ви можете визначитися з тим, чи слід виконувати пошук з врахуванням регістру символів. Позначення пункту  обмежить варіанти відповідності записами з точною відповідністю регістру для кожного з символів у ключі пошуку.

Натисніть кнопку  на правому краю додаткової панелі пошуку, щоб перемкнути панель у стан потужного пошуку і заміни.

Зміни → Знайти варіанти → Знайти далі (F3)

Повторює останню операцію з пошуку, якщо така виконувалася, без відкриття панелі нагромаджувального пошуку, пошук виконується у напрямку до кінця документа, починаючи з позиції курсора.

Зміни → Знайти варіанти → Знайти позаду (Shift+F3)

Повторює останню операцію з пошуку, якщо така виконувалася, без відкриття панелі нагромаджувального пошуку, пошук виконується у напрямку до початку документа, а не до його кінця.

Зміни → Знайти варіанти → Знайти вибране (Ctrl+H)

Знаходить наступний елемент у виділеному тексті.

Зміни → Знайти варіанти → Знайти вибране позаду (Ctrl+Shift+H)

Знаходить попереднє елемент у виділеному тексті.

Зміни → Замінити... (Ctrl+R)

Цей пункт відкриє панель потужного пошуку і заміни. У лівій верхній частині цієї панелі ви побачите піктограму, призначену для закриття панелі, поряд з якою буде невеличке поле для введення шаблону тексту.

Ви можете керувати режимом пошуку вибором одного з режимів **Простий текст**, **Цілі слова**, **Керівні послідовності** і **Формальний вираз** за допомогою спадного списку.

Якщо позначено принаймні один з пунктів **Керівні послідовності** або **Формальні вирази**, програма увімкне пункт меню **Додати...** у нижній частині контекстного меню текстових панелей. Це надасть вам змогу додавати керівні послідовності або формальні вирази у шаблони пошуку або заміни за допомогою списку шаблонів.

Скористайтеся кнопкою  or для переходу до наступного або попереднього відповідника пошуку у документі.

Текст, на який слід буде замінити ключ пошуку, слід ввести у поле для введення тексту з міткою **Замінити**, після чого слід натиснути кнопку **Замінити**, щоб замінити лише поточний підсвічений елемент, або кнопку **Замінити всі**, щоб замінити ключ пошуку у всьому документі.

Ви можете змінити поведінку програми під час пошуку з заміною позначенням відповідних пунктів у правій частині панелі. За допомогою кнопки  можна обмежити пошук елементами, у яких регістр (верхній або нижній) кожного символу збігається з ключем пошуку. За допомогою кнопки  можна обмежити пошук з заміною лише позначеним фрагментом тексту. Якщо ви позначите пункт **Знайти всі**, програма позначить кольором всі відповідники ключа пошуку у документі і покаже кількість знайдених відповідників на невеличкій контекстній панелі.

Натисніть кнопку  , розташовану у правій частині панелі потужного пошуку і заміни, щоб перемкнутися на звичайну нагромаджувальну панель пошуку.

4.3 Меню «Позначення»

Позначення → Вибрати все (Ctrl+A)

Використання цього пункту меню призведе до виділення всього документа. Це буває дуже корисно, якщо ви бажаєте скопіювати весь файл до іншої програми.

Позначення → Скасувати вибір (Ctrl+Shift+A)

Скасовує виділення (якщо таке було) будь-якого тексту у редакторі.

Позначення → Режим прямокутного вибору (Ctrl+Shift+B)

Перемикає режим вибору. У режимі вибору **БЛК**, коли на панелі стану показано рядок **[БЛК]**, ви зможете робити вибір по вертикалі, наприклад, вибирати стовпчики з 5 по 10 у рядках з 9 по 15.

Позначення → Закоментувати (Ctrl+D)

За допомогою цього пункту можна додати один пробіл на початку рядка, де знаходить курсор, або на початку всіх вибраних рядків.

Позначення → Розкоментувати (Ctrl+Shift+D)

За допомогою цього пункту можна прибрати один пробіл (якщо він є) на початку рядка, де знаходить курсор, або на початку всіх вибраних рядків.

Позначення → Об'єднати рядки (Ctrl+J)

За допомогою цього пункту меню можна об'єднати вибрані рядки або поточний рядок з рядком, розташованим нижче. Старі рядки в межах нового об'єднаного рядка буде відокремлено пробілом. Початкові і кінцеві пробіли у рядках, що об'єднуються буде прибрано.

Позначення → З великої літери (Ctrl+Alt+U)

Переводить першу літеру вибраного тексту або поточного слова у верхній регістр.

Позначення → Верхній регістр (Ctrl+U)

Переводить вибраний текст або літеру, розташовану одразу за курсором у верхній регістр.

Позначення → Нижній регістр (Ctrl+Shift+U)

Переводить вибраний текст або літеру, розташовану одразу за курсором у нижній регістр.

Позначення → Очистити відступ

За допомогою цього пункту можна прибрати відступ у поточній виділеній ділянці документа або у рядку, де зараз знаходиться курсор. За допомогою прибирання відступу можна встановити обраний вами режим відступу для всього тексту.

Позначення → Форматований відступ

За допомогою цього пункту можна вирівняти поточний рядок або вибрані рядки відповідно до режиму відступу і параметрів відступу, встановлених для документа.

Позначення → Вирівняти до...

Ця команда вирівнює рядки позначеного блоку або усього документа до позиції, яку вказано формальним виразом, який ви маєте вказати у відповідь на запит програми.

Якщо буде вказано порожній взірець, вирівнювання типово відбудеться за першим непорожнім символом.

Якщо у взірці буде вказано блок захоплення, вирівнювання відбудеться за захопленим блоком.

Приклади:

Якщо вказати «-», програма вставить пробіли до першого «-» у кожному з рядків для вирівнювання усіх дефісів за вказаною позицією.

Якщо вказати «:
s+(.)» програма вставить пробіли до першого непорожнього символу, який стоїть після двокрапки, для вирівнювання усіх таких символів за однаковою позицією.

Позначення → Застосувати перенесення слів

Застосовує статичне перенесення слід до всього тексту документа. Це означає, що редактор автоматично починатиме новий рядок тексту після того, як кількість символів у поточному рядку перевищить довжину рядка, вказану у параметрі **Переносити слова на** вкладки «Редагування» вікна, яке відкривається за допомогою пункту меню **Параметри → Налаштувати редактор...**

Позначення → Додати каретку над поточним курсором (Ctrl+Alt+↑)

Додає ще одну каретку над поточним курсором. Каретка розташовується у рядку вище на тій самій позиції, що і курсор у поточному рядку.

Позначення → Додати каретку під курсором (Ctrl+Alt+↓)

Додає ще одну каретку під поточним курсором. Каретка розташовується у рядку вище на тій самій позиції, що і курсор у поточному рядку.

Позначення → Додати курсори на кінцях рядків (Alt+Shift+I)

Додає курсор до кожного поточного позначеного рядка.

Позначення → Знайти і позначити наступний збіг (Alt+J)

Знаходить наступний відповідник слова, у якому зараз перебуває курсор, позначає його і додає курсор.

Позначення → Знайти і позначити усі збіги (Ctrl+Alt+Shift+J)

Знаходить усі відповідники слова, у якому зараз перебуває курсор, позначає їх і додає курсор до кожного з них.

4.4 Меню «Перегляд»

Перегляд → Збільшити шрифт (Ctrl++)

Буде збільшено розмір шрифту.

Перегляд → Зменшити шрифт (Ctrl+-)

Буде зменшено розмір шрифту.

Перегляд → Перенесення слів → Динамічне перенесення слів (F10)

Вмикає або вимикає динамічне перенесення рядків у поточному перегляді документа. За допомогою динамічного перенесення рядків можна зробити видимим весь текст документа: вам не потрібно буде використовувати горизонтальну смужку гортання, — рядки документа, за потреби, буде розбито на декілька рядків для перегляду.

Перегляд → Перенесення слів → Помітки динамічного перенесення слів

За допомогою цього пункту ви зможете обрати те, за яких умов і яким чином слід показувати помітки динамічного перенесення слів. Цей пункт доступний, лише якщо позначено пункт **Динамічне перенесення слів**.

Перегляд → Перенесення слів → Показувати помітки статичного перенесення слів

Якщо позначено цей пункт, у стовпчику перенесення слів буде намальовано вертикальну лінію, її розташування визначатиметься параметрами, вказаними у вікні, яке відкривається пунктом **Параметри → Налаштувати редактор...**, на вкладці Редагування. Будь ласка, зауважте, що помітку статичного перенесення слів буде показано, лише якщо ви використовуєте моноширинний шрифт.

Перегляд → Межі → Показувати рамку для піктограм (F6)

Це пункт-перемикач. Якщо його позначено у лівій частині активного редактора буде показано рамку для піктограм, якщо позначку знято — рамки для піктограм показано не буде. На рамці для піктограм буде показано всі позначені рядки у редакторі.

Перегляд → Межі → Показати номери рядків (F11)

Це пункт-перемикач. Якщо його буде позначено у лівій частині вікна активного редактора буде показано панель з номерами рядків, якщо позначку буде знято — цю панель буде сховано.

Перегляд → Межі → Показувати позначки на смужці гортання

Якщо позначено цей пункт, у поточному перегляді на вертикальній смужці гортання буде показано позначки. Цими позначками буде відмічено, наприклад, закладки. Перелік показаних позначок збігається з переліком позначок рамки для піктограм.

Перегляд → Межі → Показувати мінікарту на смужці гортання

За допомогою цього пункту можна замінити смужку гортання візуальним представленням вмісту поточного документа. Докладніший опис мінікарти на смужці гортання можна знайти у розділі Розділ 3.10.

Перегляд → Згортання коду

Пункти цього меню стосуються згортання коду:

Перегляд → Межі → Згортання коду → Показувати маркери згортання

Вмикає або вимикає показ панелі маркерів згортання у лівій частині області перегляду.

Перегляд → Згортання коду → Згорнути поточний вузол

Згортає область, у якій перебуває курсор.

Перегляд → Згортання коду → Розгорнути поточний вузол

Розгортає область, у якій перебуває курсор.

Перегляд → Згортання коду → Згорнути вузли найвищого рівня (Ctrl+Shift+-)

Згортає усі області верхнього рівня у документі. Натисніть стрілочку-трикутник, спрямовану праворуч, щоб розгорнути усі області верхнього рівня.

Перегляд → Згортання коду → Розгорнути вузли найвищого рівня (Ctrl+Shift++)

Розгортає всі області верхнього рівня у документі.

Показувати недруковані пробіли

Показати або приховати рамку навколо непридатних до друку пробілів.

4.5 Меню «Перехід»

Перехід → Перейти до рядка... (Ctrl+G)

За допомогою цього пункту можна відкрити панель переходу внизу вікна. Цією панеллю можна скористатися для переведення курсора на певний рядок (який визначається номером) у документі. Номер рядка можна ввести безпосереднім набором на клавіатурі або за допомогою стрілочок вгору і вниз у керуванні лічильника збоку від поля для введення тексту. Маленька стрілочка вгору збільшує значення номера рядка, а стрілочка вниз збільшує його. Закрити панель можна за допомогою натискання піктограми у лівій частині панелі.

Перехід → Перейти до попереднього рядка редагування (Ctrl+E)

У результаті використання цього пункту можна перейти до попереднього рядка редагування у режимі декількох курсорів.

Перехід → Перейти до наступного рядка редагування (Ctrl+Shift+E)

У результаті використання цього пункту можна перейти до наступного рядка редагування у режимі декількох курсорів.

Перехід → Перейти до попереднього зміненого рядка

Рядки, які було змінено з часу відкриття файла, називаються зміненими рядками. За допомогою цього пункту можна перейти до попереднього зміненого рядка.

Перехід → Перейти до наступного зміненого рядка

Рядки, які було змінено з часу відкриття файла, називаються зміненими рядками. За допомогою цього пункту можна перейти до наступного зміненого рядка.

Перехід → Перейти до відповідної дужки (Ctrl+6)

Пересунути курсор до відповідної парної початкової чи кінцевої дужки.

Перехід → Вибрати до відповідної дужки (Ctrl+Shift+6)

Позначити текст між відповідними початковою і кінцевою дужками.

Перехід → Закладки (Ctrl+Shift+6)

Під описаними тут пунктами буде по одному пункту для кожної з закладок у поточному документі. Текст цих пунктів визначатиметься декількома першими словам позначеного рядка. Вибір одного з таких пунктів призведе до переведення курсора на початок відповідного рядка. Область перегляду редактора буде прогорнуто так, щоб зробити належний рядок видимим.

Перехід → Закладки → Встановити закладку (Ctrl+B)

Встановлює або вилучає закладку з поточного рядка активного документа (якщо там уже була закладка, її буде вилучено, якщо ж закладки не було, буде встановлено).

Перехід → Закладки → Очистити всі закладки

За допомогою цього пункту можна вилучити з документа всі позначки, а також список позначок, який буде показано внизу цього меню.

Перехід → Закладки → Попередня (Alt+PgUp)

За допомогою цього пункту можна перевести курсор на початок першого ж рядка вище за текстом, на якому встановлено закладку. У пункті меню буде показано номер рядка і перші символи тексту у рядку з закладкою. Цей пункт меню стане доступним, лише якщо вище за текстом від рядка з курсором існує закладка.

Перехід → Закладки → Наступна (Alt+PgDown)

За допомогою цього пункту можна перевести курсор на початок першого ж рядка нижче за текстом, на якому встановлено закладку. У пункті меню буде показано номер рядка і перші символи тексту у рядку з закладкою. Цей пункт меню стане доступним, лише якщо нижче за текстом від рядка з курсором існує закладка.

4.6 Меню «Інструменти»

Інструменти → Режим

Тут ви зможете обрати схему типу файла, якою ви бажаєте скористатися у активному документі Вказаний вами режим перевизначить для поточного документа загальний режим типу файлів, встановлений за допомогою вікна, що відкривається пунктом меню **Параметри → Налаштувати Kate**, на вкладці «Режими і типи файлів».

Інструменти → Підсвічування

За допомогою цього пункту ви можете обрати схему підсвічування, яка потрібна для вашого активного документа. Цей параметр перевизначить глобальний режим підсвічування, встановлений у вікні, яке відкривається за допомогою пункту меню **Параметри → Налаштувати редактор...**, але лише для поточного документа.

Інструменти → Відступ

За допомогою цього пункту ви можете обрати стиль відступу, яку ви бажаєте використати для вашого активного документа. Цей параметр перевизначить глобальний режим відступів, встановлений у вікні, яке відкривається за допомогою пункту меню **Параметри → Налаштувати редактор...**, але лише для поточного документа.

Інструменти → Кодування

За допомогою цього пункту ви можете перевизначити типове кодування, встановлене у вікні, яке відкривається за допомогою пункту меню **Параметри → Налаштувати редактор...**, на сторінці **Відкрити/зберегти**, тобто встановити інше кодування для вашого поточного документа. Кодування, яке ви встановите за допомогою цього пункту, буде діяти лише у межах поточного документа.

Інструменти → Додати позначку порядку байтів (BOM)

Після позначення цього пункту ви зможете явним чином додати позначку порядку байтів до документів у кодуванні Unicode. Позначка порядку байтів (BOM) — це символ Unicode, що використовується для визначення порядку байтів текстового файла або потоку даних. Докладніше про неї можна дізнатися зі статті Позначка порядку байтів.

Інструменти → Кінець рядка

За допомогою цього пункту ви можете обрати режим завершення рядків, який ви бажаєте використати для вашого активного документа. Цей параметр перевизначить глобальний режим завершення рядків, встановлений у вікні, яке відкривається за допомогою пункту меню **Параметри → Налаштувати редактор...**, але лише для поточного документа.

Інструменти → Скрипти

У цьому підменю ви знайдете список всіх дій, пов'язаних з виконанням скриптів. Внести зміни до списку доволі просто: достатньо створити власні скрипти. За допомогою цих скриптів користувач може удосконалити KatePart власними інструментами.

Інструменти → Скрипти → Навігація

Інструменти → Скрипти → Навігація → Пересунути курсор до попереднього відповідного відступу (Alt+Shift+↑)

Пересуває курсор на перший рядок над поточним рядком, відступ якого збігається з рівнем відступу поточного рядка.

Інструменти → Скрипти → Навігація → Пересунути курсор до наступного відповідного відступу (Alt+Shift+↓)

Пересуває курсор на перший рядок під поточним рядком, відступ якого збігається з рівнем відступу поточного рядка.

Інструменти → Скрипти → Редагування

Інструменти → Скрипти → Редагування → Впорядкувати позначений текст

Впорядкувати позначений фрагмент тексту або цілий документ за зростанням.

Інструменти → Скрипти → Редагування → Пересунути рядки нижче (Ctrl+Shift+↓)

Пересунути позначені рядки нижче.

Інструменти → Скрипти → Редагування → Пересунути рядки вище (Ctrl+Shift+↑)

Пересунути позначені рядки вище.

Інструменти → Скрипти → Редагування → Здублювати позначені рядки нижче (Ctrl+Alt+↓)

Дублювати позначені рядки нижче.

Інструменти → Скрипти → Редагування → Здублювати позначені рядки вище (Ctrl+Alt+↑)

Дублювати позначені рядки вище.

Інструменти → Скрипти → Редагування → Закодувати адресу у позначеному фрагменті

Закодувати позначений фрагмент тексту так, щоб ним можна було скористатися як частиною адреси URL із запитом, замінивши позначений фрагмент тексту на закодований.

Інструменти → Скрипти → Редагування → Декодувати адресу у позначеному фрагменті

Якщо позначено частину рядка запиту URL, за допомогою цього пункту можна її декодувати і замінити початковим фрагментом тексту.

Інструменти → Скрипти → Emmet

Інструменти → Скрипти → Emmet → Розгорнути скорочення

Перетворює позначений фрагмент тексту на пару з початкового та завершального теґу HTML або XML. Наприклад, якщо було позначено `div`, позначений фрагмент буде замінено на `<div></div>`.

Інструменти → Скрипти → Emmet → Згорнути цей теґ

За допомогою цієї команди можна огорнути позначений фрагмент тексту у теґ, вказаний у командному рядку.

Інструменти → Скрипти → Emmet → Пересунути курсор до парного теґу
Якщо курсор перебуває на початковому тезі HTML/XML, за допомогою цього пункту можна пересунути його до завершального теґу. Якщо ж курсор перебуватиме на завершальному тезі, курсор буде пересунуто до початкового теґу.

Інструменти → Скрипти → Emmet → Позначити вміст, обмежений теґами HTML/XML
Якщо курсор перебуває у парі теґів HTML/XML, вибір цього пункту змінить позначення фрагмента так, щоб до нього було включено вміст цих теґів HTML/XML без позначення самих теґів.

Інструменти → Скрипти → Emmet → Позначити зовнішню частину, обмежену теґами HTML/XML
Якщо курсор перебуває у парі теґів HTML/XML, вибір цього пункту змінить позначення фрагмента так, щоб до нього було включено вміст цих теґів HTML/XML разом з цими теґами.

Інструменти → Скрипти → Emmet → Додати або вилучити коментування
Якщо позначений фрагмент тексту не є коментарем, за допомогою цього пункту можна огорнути цей фрагмент у позначення коментарів HTML/XML (наприклад `<!-- позначений текст -->`). Якщо позначений фрагмент вже закоментовано, мітки коментування буде вилучено.

Інструменти → Скрипти → Emmet → Вилучити теґ під курсором
Якщо курсор перебуває у тезі HTML/XML, за допомогою цього пункту можна вилучити весь теґ.

Інструменти → Скрипти → Emmet → Зменшити число на 1
За допомогою цього пункту можна відняти одиницю від позначеного тексту, якщо позначено число. Наприклад, якщо позначено `5`, отримаєте `4`.

Інструменти → Скрипти → Emmet → Зменшити число на 10
За допомогою цього пункту можна відняти десять від позначеного тексту, якщо позначено число. Наприклад, якщо позначено `15`, отримаєте `5`.

Інструменти → Скрипти → Emmet → Зменшити число на 0.1
За допомогою цього пункту можна відняти 0.1 від позначеного тексту, якщо позначено число. Наприклад, якщо позначено `4.5`, отримаєте `4.4`.

Інструменти → Скрипти → Emmet → Збільшити число на 1
За допомогою цього пункту можна додати одиницю до позначеного тексту, якщо позначено число. Наприклад, якщо позначено `5`, отримаєте `6`.

Інструменти → Скрипти → Emmet → Збільшити число на 10
За допомогою цього пункту можна додати десять до позначеного тексту, якщо позначено число. Наприклад, якщо позначено `5`, отримаєте `15`.

Інструменти → Скрипти → Emmet → Збільшити число на 0.1
За допомогою цього пункту можна додати 0.1 до позначеного тексту, якщо позначено число. Наприклад, якщо позначено `4.5`, отримаєте `4.6`.

Інструменти → Перемкнутися до командного рядка (F7)

Показує командний рядок KatePart внизу вікна. У цьому командному рядку можна ввести `help`, щоб переглянути довідкову інформацію, або `help list`, щоб переглянути список команд. Докладніші відомості щодо командного рядка можна знайти у розділі щодо командного рядка компонента редактора.

Інструменти → Увімкнути завершення коду (Ctrl+Пробіл)

Вручну викликати завершення команд за допомогою скорочення, яке прив'язане до цієї дії.

Інструменти → Завершення слів

За допомогою пунктів **Повторно використати слово нижче** (**Ctrl**+**9**) і **Повторно використати слово нижче** (**Ctrl**+**8**) можна наказати програмі завершувати поточні слова на основі подібних слів, розташованих у напрямку кінця чи початку документа відносно поточної позиції курсора. **Завершення оболонки** відкриває панель завершення з відповідними пунктами.

Інструменти → Перевірка правопису → Автоматична перевірка правопису (Ctrl+Shift+O)

Якщо буде позначено пункт **Автоматична перевірка правопису**, помилкові слова у тексті буде підкреслено на льоту.

Інструменти → Перевірка правопису → Перевірка правопису...

Цей пункт запускає програму для перевірки правопису — програму, розроблену з метою допомоги користувачеві у виявленні і усуненні помилок у правописі. Натискання цього пункту запустить перевірку правопису і відкриє діалогове вікно перевірки правопису, за допомогою якого користувач зможе керувати процесом перевірки. Існує чотири параметри, впорядковані вертикально у центрі діалогового вікна, позначені відповідними мітками, розташованими ліворуч. Починаючи згори це:

Невідоме слово:
Тут інструмент перевірки правопису показує слово, на якому було зупинено перевірку правопису. Така зупинка відбувається, якщо програма зустріне слово, якого немає у її словнику — файлі, де міститься список правильних написань слів, за допомогою якого програма перевіряє правопис кожного зі слів у редакторі.

Замінити на:
Якщо інструмент перевірки правопису знайде подібні слова у власному словнику, у цьому полі він покаже їх список. Користувач може погодитися з пропозицією, виконати виправлення самостійно або обрати з наступного списку іншу пропозицію.

Мова:
Якщо ви встановили декілька словників, тут ви можете обрати той з них, який слід використовувати.

У правій частині цього діалогового вікна розташовано 6 кнопок, за допомогою яких користувач може керувати процесом перевірки правопису. Це кнопки:

Додати до словника
Натискання цієї кнопки додає **Невідоме слово** до словника засобу для перевірки правопису. Це означає, що наступного разу, коли під час перевірки трапиться таке слово, засіб перевірки правопису вважатиме, що слово написано правильно.

Запропонувати
Тут засіб для перевірки правопису може показати список можливих правильних варіантів написання невідомого слова. Натискання одного з запропонованих варіантів вказівником миші призведе до введення цього слова до поля **Замінити на**, розташованого вище.

Замінити
За допомогою цієї кнопки можна наказати засобу для перевірки правопису замінити невідоме слово у документі на слово з поля **Замінити на**.

Замінити всі
За допомогою цієї кнопки можна наказати засобу для перевірки правопису замінити не лише поточне **Невідоме слово:**, а й автоматично зробити цю саму заміну для всіх подібних **Невідомих слів** у документі.

Ігнорувати
Натискання цієї кнопки накаже засобу для перевірки правопису рухатися текстом далі, не виконуючи ніяких змін.

Ігнорувати все
За допомогою цієї кнопки можна наказати засобу для перевірки правопису не виконувати ніяких дій з поточним **Невідомим словом**, а також пропускати всі інші випадки, коли у документі зустрічається це невідоме слово.

> Примітка
> Дії, які виконуються у результаті натискання цієї кнопки стосуються лише поточного сеансу запуску перевірки правопису. Якщо пізніше ви запустите перевірку правопису знову, перевірку знову буде зупинено на цьому слові.

Вздовж нижньої частини діалогового вікна перевірки правопису по горизонталі розташовано це три кнопки. Це кнопки:

Довідка

За допомогою натискання цієї кнопки можна скористатися довідковою системою KDE, яку буде відкрито на сторінці відповідного діалогового вікна.

Завершено

Ця кнопка завершує процес перевірки правопису, і повертає фокус на документ.

Скасувати

Ця кнопка скасовує процес перевірки правопису, всі зміни буде скасовано і ви повернетеся до редагування вашого документа.

Інструменти → Перевірка правопису → Перевірка правопису (від курсора)...

За допомогою цього пункту можна розпочати перевірку правопису, але не з початку документа, а з місця, де зараз знаходиться курсор.

Інструменти → Перевірка правопису → Перевірка правопису вибраного...

Перевіряє правопис у вибраному.

Інструменти → Перевірка правопису → Змінити словник

Відкриває у нижній частині вікна редактора спадний список з усіма можливими словниками для перевірки правопису. За допомогою цього списку вам буде простіше перемикатися між словниками перевірки правопису, наприклад, у документах, написаних різними мовами.

4.7 Меню «Параметри» і «Довідка»

Параметри → Тема кольорів редактора

У цьому меню ви знайдете список всіх доступних схем кольорів. За допомогою пунктів меню ви зможете змінити схему для поточного перегляду. Щоб змінити типову схему, вам слід скористатися сторінкою Шрифти та кольори діалогового вікна налаштовування.

У KatePart передбачено типові для KDE пункти меню **Параметри** і **Довідка**. Щоб дізнатися більше, ознайомтеся з розділами щодо меню «Параметри» та меню «Довідка» підручника з основ роботи у KDE.

Розділ 5

Додаткові інструменти редагування

Anders Lund
Dominik Haumann
Переклад українською: Юрій Чорноіван

5.1 Додавання/Вилучення позначок коментаря

За допомогою пунктів **Закоментувати** і **Розкоментувати** меню **Інструменти** ви зможете додавати або вилучати позначки коментарів до позначеного фрагмента або поточного рядка (у разі, якщо нічого не позначено), якщо коментарі підтримуються форматом файла, який ви редагуєте.

Правила коментування визначаються у файлі визначень синтаксису, отже, якщо не використано ніяких правил підсвічування синтаксису, коментування/вилучення коментарів буде неможливим.

У деяких форматах файлів можливе коментування лише окремих рядків, у деяких форматах передбачено коментування одразу декількох рядків парою позначок коментаря, а у деяких форматах можливі обидва варіанти. Якщо коментування парою позначок не передбачено форматом файла, коментування останнього рядка за неповного його позначення у багаторядковому фрагменті тексту буде неможливим.

Якщо форматом файла передбачено позначки для коментування окремого рядка, використанню цих позначок слід надавати перевагу, оскільки за такого варіанта ви уникнете проблем з вкладеними коментарями.

Якщо ви вилучаєте позначки коментарів, вам не слід позначати фрагментів тексту, які не було закоментовано. Під час вилучення коментування парою позначок з позначеного фрагмента тексту всі пробільні рядки поза позначками коментарів у позначеному фрагменті буде проігноровано.

Щоб додати позначки коментарів, скористайтеся пунктом меню **Інструменти** → **Закоментувати** або відповідною комбінацією клавіш, типовою комбінацією є **Ctrl**+**D**.

Щоб вилучити позначки коментарів, скористайтеся пунктом меню **Інструменти** → **Розкоментувати** або відповідною комбінацією клавіш, типовою комбінацією є **Ctrl**+**Shift**+**D**.

5.2 Командний рядок компонента редактора

У компонент редактора KatePart вбудовано інструмент для роботи з командним рядком, за допомогою якого ви зможете виконувати різноманітні дії за мінімального графічного інтерфейсу. Командний рядок — це текстова панель у нижній частині вікна редактора, щоб

увімкнути показ цієї панелі скористайтеся пунктом меню **Перегляд → Перемкнутися до командного рядка** або скористайтеся клавіатурним скороченням (типовим клавіатурним скороченням є **F7**). У редакторі передбачено набір команд, документацію з яких наведено нижче, та додаткові команди, за допомогою яких можна керувати додатками до програми.

Щоб виконати команду, введіть цю команду до командного рядка і натисніть клавішу **Enter**. За командним рядком ви зможете визначити, чи було виконано команду, у ньому ж можна переглянути повідомлення, яке могло бути виведено у відповідь на команду. Якщо ви вводили команду за після натискання клавіші **F7**, за декілька секунд панель командного рядка буде приховано. Щоб витерти виведене командою повідомлення і ввести нову команду, натисніть клавішу **F7** ще раз.

У командний рядок вбудовано довідкову систему. Щоб розпочати роботу з довідкою, введіть команду **help**. Щоб переглянути список можливих команд, введіть команду **help list**. Щоб переглянути довідку з окремої команди, виконайте команду **help** *назва_команди*.

Під час роботи з командним рядком ви можете скористатися вбудованим журналом команд, таким чином ви можете зменшити час доступу до команд, які вже раніше було введено. Для навігації журналом команд скористайтеся клавішами ↑ і ↓. Під час показу команд з журналу буде позначено параметри команди, — це спростить вам перезапис параметрів команди.

5.2.1 Стандартні команди командного рядка

Типи аргументів

BOOLEAN
: Цей тип використовується для команд, які вмикають або вимикаю певні налаштування. Можливими значеннями таких параметрів є `on`, `off`, `true`, `false`, `1` і `0`.

INTEGER
: Ціле число.

STRING
: Рядок, обмежений одинарними лапками (') або подвійними лапками (″), якщо у ньому містяться пробіли.

5.2.1.1 Команди налаштування редактора

Ці команди стосуються компонента редактора, за їх допомогою можна налаштовувати лише параметри показу поточного документа. Цими командами зручно користатися, якщо вам потрібно використати параметри, наприклад параметри відступів, відмінні від типових параметрів.

`set-tab-width` **INTEGER ширина**
: Встановлює ширину табуляції у значення `ширина`.

`set-indent-width` **INTEGER ширина**
: Встановлює відступ у значення кількості пробілів, визначене параметром `ширина`. Використовується, лише якщо увімкнено відступи з пробілів.

`set-word-wrap-column` **INTEGER довжина**
: Встановлює для параметра максимальної довжини рядка для примусового перенесення значення `довжина`. Цей параметр використовується, якщо увімкнено автоматичне розбиття тексту на рядки.

`set-icon-border` **BOOLEAN вмикання**
: Встановлює видимість бічної смужки піктограм.

`set-folding-markers` **BOOLEAN вмикання**
: Встановлює значення видимості позначок панелі згортання.

`set-line-numbers` **BOOLEAN вмикання**
Встановлює значення видимості панелі номерів рядків.

`set-replace-tabs` **BOOLEAN вмикання**

Якщо має значення «true», введені вами знаки табуляції буде замінено пробілами.

`set-remove-trailing-space` **BOOLEAN вмикання**

Якщо цей параметр буде увімкнено, після переведення курсора на інших рядок пробіли наприкінці поточного рядка вилучатимуться.

`set-show-tabs` **BOOLEAN вмикання**
Якщо буде увімкнено цей параметр, символи табуляції і пробіли наприкінці рядка буде показано у вигляді маленьких крапок.

`set-show-indent` **BOOLEAN вмикання**
Якщо буде увімкнено цей параметр, відступи буде показано вертикальною крапчастою лінією.

`set-indent-spaces` **BOOLEAN вмикання**

Якщо буде увімкнено цей параметр, редактор робитиме відступ на `indent-width` пробілів, замість символу табуляції, під час переходу на наступний рівень відступу.

`set-mixed-indent` **BOOLEAN вмикання**
Якщо буде увімкнено цей параметр, KatePart використовуватиме суміш з символів табуляції і пробілів для додавання відступів. Ширина кожного з рівнів відступу визначатиметься значенням параметра `indent-width`, загальний же відступ буде оптимізовано так, щоб у ньому використовувалася максимальна кількість символів табуляції.

Після виконання цієї команди буде увімкнено створення відступів пробілами, якщо ширину відступу не було вказано, буде встановлено значення ширини відступу у половину значення параметра документа `tab-width` на час виконання команди.

`set-word-wrap` **BOOLEAN вмикання**

Встановлює значення для динамічного перенесення слів у `вмикання` (true або false)

`set-replace-tabs-save` **BOOLEAN вмикання**

Якщо увімкнути цей параметр, під час збереження документа символи табуляції буде замінено на пробіли.

`set-remove-trailing-space-save` **BOOLEAN вмикання**

Якщо увімкнути цей параметр, під час збереження документа пробіли наприкінці рядків документа будуть вилучатися.

`set-indent-mode` **STRING назва**
Встановлює режим автоматичного встановлення відступів у значення `назва`. Якщо програмі не вдасться розпізнати параметр `назва`, буде встановлено режим 'none'. Коректними значеннями режиму є «none», «normal», «cstyle», «haskell», «lilypond», «lisp», «python», «ruby» і «xml».

`set-auto-ident` **BOOLEAN скрипт**

Увімкнути або вимкнути автоматичне встановлення відступів.

`set-highlight` **STRING підсвічування**

Визначає систему підсвічування синтаксису для поточного документа. Параметром команди має бути коректна назва підсвічування у тому вигляді, у якому ця система фігурує у меню **Інструменти → Підсвічування**. У цій команді передбачено автоматичне доповнення параметра за списком.

`reload-scripts`

Перезавантажити всі скрипти JavaScript, використані Kate, зокрема скрипти встановлення відступів та скрипти командного рядка.

`set-mode` **STRING режим**

Вибрати схему типів файлів для поточного документа.

`nn[oremap]` **STRING початкова STRING відображена**

Відобразити послідовність символів `початкова` на послідовність `відображена`.

5.2.1.2 Команди редагування

Наступні команди змінюють вміст поточного документа.

`indent`
: Додає відступ у позначені рядки або поточний рядок.

`unindent`
: Вилучає початковий відступ з позначених рядків або поточного рядка.

`cleanindent`
: Вилучає відступи для позначених рядків або поточного рядка відповідно до параметрів відступів поточного документа.

`comment`
: Додає позначки коментаря, які роблять позначений фрагмент тексту, позначені рядки або поточний рядок коментарями, відповідно до формату тексту, який визначено у параметрах підсвічування синтаксису поточного документа.

`uncomment`
: Вилучає позначки коментаря, які роблять позначений фрагмент тексту, позначені рядки або поточний рядок коментарями, відповідно до формату тексту, який визначено у параметрах підсвічування синтаксису поточного документа.

`kill-line`
: Вилучає поточний рядок.

`replace` **STRING шаблон STRING замінник**

: Замінює текст, що відповідає `шаблону`, на текст, вказаний у `замінникуу`. Якщо вам потрібно включити пробіл у `шаблон`, вам слід взяти як `шаблон`, так і `замінник` у одинарні або подвійні лапки. Якщо у команді не буде лапок, перше зі слів-параметрів вважатиметься `шаблоном`, а решта — `замінником`. Якщо параметр `замінник` буде порожнім, всі рядки, які відповідають `шаблону` буде вилучено з тексту.

  Ви можете вказати прапорці налаштування пошуку за допомогою додавання двокрапки, за якою слід вказати одну або декілька літер, кожна з яких відповідатиме за налаштування. Команда викладатиме так: `replace:параметри шаблон замінник`. Можливі параметри:

  `b`
  : Пошук назад.

  `c`
  : Шукати від позиції курсора.

  `e`
  : Шукати лише у позначеному фрагменті.

  `r`
  : Пошук за формальним виразом. Якщо встановлено цей прапорець, ви зможете використовувати параметр `\N`, де N — номер підрядка у рядку «замінник».

  `s`
  : Виконати пошук з врахуванням регістру.

  `p`
  : Запитувати про дозвіл на заміну наступного елемента.

  `w`
  : Шукати лише цілі слова.

`date` **STRING формат**

: Вставляє дату/час у форматі, визначеному параметром `формат`, або у форматі «yyyy-MM-dd hh:mm:ss», якщо рядок формату не буде вказано. Під час обробки рядка `формат` буде використано такі перетворення:

d	Номер дня без початкового нуля (1-31).
dd	Номер дня з початковим нулем (01-31).
ddd	Скорочена локалізована назва дня тижня (наприклад «пн» або «нд»).
dddd	Довга локалізована назва дня (наприклад «понеділок» або «неділя»).
M	Номер місяця без початкового нуля (1-12).
MM	Номер місяця з початковим нулем (01-12).
MMMM	Повна локалізована назва місяця (наприклад, «січень» або «грудень»).
MMM	Скорочена локалізована назва місяця (наприклад «січ» або «гру»).
yy	Рік як двоцифрове значення (00-99).
yyyy	Рік як чотирицифрове значення (1752-8000).
h	Години без початкового нуля (0..23 або 1..12 з показом до опівдня/після опівдня).
hh	Години з початковим нулем (00..23 або 01..12 з показом до опівдня/після опівдня).
m	Хвилини без початкового нуля (0..59).
mm	Хвилини з початковим нулем (00..59).
s	Секунди без початкового нуля (0..59).
ss	Секунди з початковим нулем (00..59).
z	Мілісекунди без початкових нулів (0..999).
zzz	Мілісекунди з початковими нулями (000..999).
AP	Використовувати показ у форматі AM/PM (до опівдня/після опівдня). AP буде замінено на рядок «AM» або рядок «PM».
ap	Використовувати показ у форматі am/pm (до опівдня/після опівдня). ap буде замінено на рядок «am» або рядок «pm».

`char` STRING ідентифікатор

За допомогою цієї команди можна вводити символи за їх числовими ідентифікаторами у десятковій, вісімковій або шістнадцятковій формі. Щоб скористатися нею, відкрийте діалогове вікно редагування команди і введіть команду `char: [число]` у поле для введення, а потім натисніть кнопку **Гаразд**.

Example 5.1 Приклади команди char

Ввід: `char:234`
Вивід: ê
Ввід: `char:0x1234`
Вивід: jué

s///[ig] %s///[ig]

За допомогою цієї команди можна виконати дію з пошуку або заміни у стилі sed у поточному рядку або у всьому файлі (**%s**///).

Якщо коротко, у тексті буде виконано пошук за *шаблоном пошуку*, формальним виразом між першою і другою навскісними рисками, якщо буде знайдено відповідний фрагмент тексту, його буде замінено на вираз між середньою і останньою навскісними рисками у рядку. За допомогою круглих дужок можна створювати *зворотні посилання*: команда запам'ятає частину виразу, яка відповідає рядку між дужками і використає рядки, які будуть у пам'яті для шаблону заміни, — посилання на ці рядки можна вказати як `\1` для першого набору у дужках, `\2` для другого тощо.

Щоб знайти символ `(` або символ `)`, вам слід *екранувати* ці символи за допомогою зворотної навскісної риски: `\(\)`

Якщо ви додасте ключ `i` в кінець виразу, під час пошуку буде враховано регістр символів. Якщо в кінець виразу додати `g`, буде замінено всі рядки, які відповідають шаблону, у іншому випадку буде замінено лише перший з цих рядків.

Example 5.2 Заміна тексту у поточному рядку

Припустімо, компілятор, яким ви збирали вашу програму, повідомив вам про те, що клас `myClass`, який згадується у рядку 3902 коду вашої програми не визначено.
Ви подумаєте: «Оце тобі, то це ж має бути мій клас `MyClass`». Ви переходите до рядка 3902 і, замість спроб знайти потрібне слово у тексті, відкриваєте діалогове вікно редагування команди, вводите у нього `s/myclass/MyClass/i`, натискаєте кнопку **Гаразд**, зберігаєте файл і компілюєте його ще раз, — все, помилки немає.

Example 5.3 Заміна тексту у всьому файлі

Припустімо тепер, що у вас є файл, у якому ви згадуєте «пані Іванченко» декілька разів, і тут вам стає відомо, що ця особа нещодавно вийшла заміж за «пана Петренка». Ну і, звісно ж, вам потрібно тепер замінити всі рядки з «пані Іванченко» на «пані Петренко».
Відкрийте вікно командного рядка і введіть туди команду `%s/пані Іванченко/пані Петренко/` , а потім натисніть клавішу **Enter**, — решту зробить програма.

Example 5.4 Складніший приклад

У цьому прикладі ми познайомилися з використанням *зворотніх посилань*, а також *класу символів* (якщо вам не знайомі ці терміни, будь ласка, зверніться до відповідної документації, посилання на яку ви зможете знайти далі за текстом).
Припустімо, що у вас є такий рядок:

```
void MyClass::DoStringOps( String      &foo, String &bar, String *p, int ↩
     &a, int &b )
```

Тепер вам здається, що цей код виглядає кострубато: ви вирішили скористатися ключовим словом `const` для всіх параметрів «address of», у яких назві аргументу передує оператор &. Ви також бажаєте вилучити зайві пробіли так, щоб між будь-якими двома словами був лише один символ пробілу.
Відкрийте діалогове вікно редагування команди і введіть таку команду: `s/\s+(\w+)\s+(&)/const \1 \2/g`. Натисніть кнопку **Гаразд**. Літера `g` наприкінці виразу наказує програмі перебудовувати формальний вираз для кожного зі знайдених елементів, щоб зберегти *зворотні посилання*.
Вивід: `void MyClass::DoStringOps( const String &foo, const String &bar String *p, const int &a, const int &b )`
Справу зроблено! Що ж сталося? Гаразд, було виконано пошук інтервалу з пробілів (`\s+`), за яким слідує один або декілька символів абетки (`\w+`), за якими слідують ще пробіли (`\s+`), за якими слідує символ амперсанда. У процесі пошуку було збережено ланцюжок символів абетки і амперсанд для подальшого використання під час операції з заміни. Після цього було виконано заміну знайденого відповідника нашого рядка на рядок з пробілом, за яким слідує слово «const», за яким слідує пробіл, за яким слідує наш ланцюжок символів абетки (`\1`), за яким слідує один пробіл, за яким слідує збережений нами символ амперсанда (`\2`)
Крім того, у деяких випадках ланцюжок символів абетки міг мати значення «String», у деяких «int», цього було досягнуто використанням класу `\w` і лічильника `+`.

`sort`
: Впорядковує позначений фрагмент тексту або весь текст у документі.

`natsort`
: Впорядковує позначені рядки або увесь документ у природному порядку.

Example 5.5 sort порівняно з natsort

`sort(a10, a1, a2)` виведе `a1, a10, a2`
`sort(a10, a1, a2)` виведе `a1, a10, a2`

`moveLinesDown`
: Пересунути позначені рядки нижче.

`moveLinesUp`
: Пересунути позначені рядки вище.

`uniq`
: Вилучити рядки-дублікати з позначеного фрагмента тексту або всього документа.

`rtrim`
: Вилучити кінцевий пробіл з позначеного фрагмента тексту або всього документа.

`ltrim`
: Вилучити початковий пробіл з позначеного фрагмента тексту або всього документа.

`join` **[STRING роздільник]**
: Об'єднати позначені рядки або рядки у всьому документі. Додатково можна визначити роздільник, наприклад: `join ', '`

`rmblank`

Вилучити всі пробіли з позначеного фрагмента тексту або всього документа.

`alignon`

Ця команда вирівнює рядки позначеного блоку або усього документа до позиції, яку вказано формальним виразом-аргументом.

Якщо буде вказано порожній взірець, вирівнювання типово відбудеться за першим непорожнім символом.

Якщо у взірці буде вказано блок захоплення, вирівнювання відбудеться за захопленим блоком.

Приклади:

`alignon -` вставить пробіли до першого «-» у кожному з рядків для вирівнювання усіх дефісів за вказаною позицією.

`alignon :\s+(.)` вставить пробіли до першого непорожнього символу, який стоїть після двокрапки, для вирівнювання усіх таких символів за однаковою позицією.

`unwrap`

Скасувати перенесення рядків у позначеному фрагменті тексту або всьому документі.

`each` **STRING скрипт**

Викликати вказану як аргумент функцію JavaScript для обробки списку (позначених) рядків і замінити їх повернутим значенням виклику.

Example 5.6 Об'єднати позначені рядки

```
each 'function(lines){return lines.join(˝, ˝)}'
```

Або коротше:

```
each 'lines.join(˝, ˝)'
```

`filter` **STRING скрипт**

Викликати вказану як аргумент функцію JavaScript для обробки списку (позначених) рядків і вилучити ті з них, для яких буде повернуто false.

Example 5.7 Вилучити порожні рядки

```
filter 'function(1){return 1.length > 0;}'
```

Або коротше:

```
filter 'line.length > 0'
```

`map` **STRING скрипт**

Викликати вказану як аргумент функцію JavaScript для обробки списку (позначених) рядків і замінити рядок значенням, яке буде повернуто у відповідь на виклик.

Example 5.8 Вилучити порожні рядки

```
map 'function(line){return line.replace(/^s+/,˝˝);}'
```

Або коротше:

```
map 'line.replace(/^s+/,˝˝)'
```

`duplicateLinesUp`

Здублювати позначені рядки над поточним позначеним фрагментом.

`duplicateLinesDown`

Здублювати позначені рядки під поточним позначеним фрагментом.

5.2.1.3 Команди навігації

`goto` **INT рядок**

За допомогою цієї команди можна перейти до вказаного рядка.

`grep` **STRING шаблон**

Знайти у документі рядки, що відповідають формальному виразу `шаблон`. Щоб дізнатися більше, зверніться до розділу Додаток A.

`find` **STRING шаблон**

За допомогою цієї команди можна перейти до першого відповідника рядка `шаблон` у тексті, відповідно до налаштувань. Перейти до наступних відповідників можна за допомогою пункту меню **Зміни** → **Знайти далі** (типовим клавіатурним скороченням є **F3**).

Команду пошуку можна налаштувати додаванням двокрапки, за якою можна вказати один або декілька параметрів, у формі `find:параметри шаблон`. Серед можливих параметрів:

`b`
: Пошук назад.

`c`
: Шукати від позиції курсора.

`e`
: Шукати лише у позначеному фрагменті.

`r`
: Пошук за формальним виразом. Якщо встановлено цей прапорець, ви зможете використовувати параметр `\N`, де N — номер підрядка у рядку «замінник».

`s`
: Виконати пошук з врахуванням регістру.

`w`
: Шукати лише цілі слова.

`ifind` **STRING шаблон**

За допомогою цієї команди можна виконати «інтерактивний» пошук. Ви можете налаштувати пошук додаванням одного або декількох параметрів після символу двокрапки, ось так: `ifind:параметри шаблон`. Серед можливих параметрів:

`b`
: Пошук назад.

`r`
: Виконати пошук за вказаним формальним виразом.

`s`
: Виконати пошук з врахуванням регістру.

`c`
: Шукати від позиції курсора.

5.2.1.4 Команди базового керування редактором (залежать від використаного у програмі компонента редактора)

`w`
: Зберегти поточний документ.

`wa`
: Зберегти всі поточні відкриті документи.

`q`
: Закрити поточний документ.

`qa`
: Закрити всі відкриті документи.

`wq`
: Зберегти і закрити поточний документ.

`wqa`
: Зберегти і закрити всі поточні відкриті документи.

`x`
: Зберегти і закрити поточний документ, лише якщо його було змінено.

`x`
: Зберегти і закрити всі поточні відкриті документ, лише якщо до них було внесено зміни.

`bp`
: Перейти до попереднього документа у списку документів.

`bn`
: Перейти до наступного документа у списку документів.

`new`
: Відкрити новий документ на новій панелі, створеній горизонтальним поділом поточної панелі редагування.

`vnew`
: Відкрити новий документ на новій панелі, створеній вертикальним поділом поточної панелі редагування.

`e`
: Перезавантажити поточний документ, якщо його було змінено на диску.

`enew`
: Редагувати новий документ.

`print`
: Відкрити діалогове вікно друку, за допомогою якого ви зможете надрукувати поточний документ.

5.3 Використання обгортки коду

За допомогою згортання коду ви можете приховувати частини документа у вікні редактора: так легше працювати з великими документами. У KatePart діапазони рядків, придатні для згортання, визначаються правилами підсвічування синтаксису, тому згортання можливе не для всіх форматів файлів, — типово, згортання працює у коді програм, розмітки XML та подібних форматів. Під час редагування коду визначення підсвічування також можна вручну визначати діапазони згортання, зазвичай, такі діапазони визначаються за допомогою ключових слів `BEGIN` і `END`.

Щоб скористатися можливостями згортання, слід задіяти позначки згортання за допомогою пункту меню **Перегляд** → **Показувати маркери згортання**, якщо цих позначок ще не показано у вікні редактора. На панелі позначок згортання з лівого боку екрана буде показано у графічному вигляді діапазони рядків, які можна згорнути. Символи-трикутники позначатимуть можливі дії з вказаним діапазоном: трикутник, вершина якого вказує вниз, означатиме розгорнутий діапазон рядків, — натискання позначки трикутника призведе до згортання діапазону, — тепер програма показуватиме трикутник з вершиною, спрямованою праворуч.

Передбачено три пункти меню, за допомогою яких можна керувати станом областей згортання коду, їх перелік можна знайти у розділі, присвяченому меню.

Програма запам'ятовує згортання після закриття файла. Коли ви наступного разу відкриєте файл для редагування, згорнуті вузли залишатимуться згорнутими. Це також стосується і дій з перезавантаження даних файлів.

Якщо ви не бажаєте користуватися можливістю згортання коду, ви можете зняти позначку з пункту **Показувати маркери згортання (якщо наявні)** на сторінці Вигляд налаштування редактора.

Розділ 6

Розширення можливостей KatePart

T.C. Hollingsworth
Переклад українською: Юрій Чорноіван

6.1 Вступ

Подібно до всіх компонентів текстових редакторів з широкими можливостями, у KatePart передбачено певні шляхи до розширення функціональних можливостей компонента. Ви можете створювати прості скрипти, що реалізують додаткові можливості, за допомогою JavaScript. Нарешті, вдосконаливши ваш екземпляр KatePart, ви можете долучитися до розробників і поділитися вашими удосконаленнями з іншими користувачами!

6.2 Як працювати з підсвічуванням синтаксису

6.2.1 Огляд

Підсвічування синтаксису призначено для автоматично показу тексту у різних стилях і кольорах, залежно від призначення відповідного рядка та файла, з якого взято цей рядок. У початковому коді програми, наприклад, оператори керування може бути показано жирним шрифтом, а типи даних і коментарі — кольором, відмінним від кольору решти тексту. За допомогою підсвічування можна значно покращити зручність читання тексту, а отже підвищити ефективність та продуктивність роботи з цим текстом.

```
Theme Repository::defaultTheme(Repository::DefaultTheme t)
{
    if (t == DarkTheme)
        return theme(QLatin1String("Breeze Dark"));
    return theme(QLatin1String("Default"));
}
```

Функція, написана мовою C++, показана з підсвічуванням синтаксису.

```
Theme Repository::defaultTheme(Repository::DefaultTheme t)
{
    if (t == DarkTheme)
        return theme(QLatin1String("Breeze Dark"));
    return theme(QLatin1String("Default"));
}
```

Та сама функція, написана мовою C++, без підсвічування.

Ну що, який з двох прикладів зручніше читати?

У KatePart передбачено гнучку, придатну для налаштування і потужну систему підсвічування синтаксичних конструкцій, у стандартному пакунку ви знайдете визначення для широкого спектру мов програмування, написання скриптів та розмітки, а також текстових файлів у інших форматах. Окрім того, ви можете створювати власні визначення за допомогою простих файлів XML.

KatePart автоматично визначатиме належні правила підсвічування синтаксису під час відкриття файла. Дія з визначення відбуватиметься на основі типу MIME файла, який визначатиметься за суфіксом назви файла або, якщо у назві немає суфіксу, за вмістом файла. Якщо, на вашу думку, програма зробила неправильний вибір, ви можете вручну вказати правила підсвічування за допомогою пункту меню **Інструменти** → **Підсвічування**.

Гарнітури шрифту та кольори, які буде використано визначенням підсвічування синтаксису, можна налаштувати на вкладці Стилі підсвічування тексту діалогового вікна налаштування, типами ж MIME та суфіксами назв файлів, для яких буде використано таке підсвічування керує вкладка Режими та типи файлів.

> **Примітка**
> Підсвічування можна використовувати для покращення візуального сприйняття тексту, але довірятися підсвічуванню під час перевірки коректності синтаксису тексту не слід. Синтаксична розмітка тексту є непростим завданням, складність якого залежить від формату тексту, — у деяких випадках, автори синтаксичних правил вважають успіхом правильний показ 98% тексту, хоча для того, щоб побачити 2% тексту з неправильним підсвічуванням, вам доведеться скористатися не дуже поширеним стилем.

6.2.2 Система підсвічування синтаксису KatePart

У цьому розділі ми зосередимося на механізмах, які використовуються для підсвічування синтаксису у KatePart. Відомості з цього розділу призначено для тих користувачів, яким цікаво дізнатися про роботу системи підсвічування синтаксису, або тих користувачів, які бажають змінити або створити нові визначення підсвічування синтаксису.

6.2.2.1 Як це працює

Під час відкриття файла однією з перших дій, які виконує редактор KatePart, є визначення правил підсвічування синтаксису для цього файла. Під час читання тексту з файла або отримання введених вами рядків система підсвічування синтаксису аналізуватиме текст на основі правил підсвічування синтаксису і позначатиме у показаному тексті позиції початку і завершення різних контекстів і стилів.

Під час введення документа за допомогою клавіатури створений вами текст буде проаналізовано і розмічено на льоту, отже, якщо ви вилучите символ, які система розмітила як початок або завершення певного контексту, стиль сусідніх з поточним фрагментів тексту також змінюватиметься відповідно до зміни контексту.

Визначення синтаксичних правил, які використовуються у системі підсвічування синтаксису KatePart, є файлами XML, у яких містяться

- Правила для визначення ролі тексту, впорядковані у контекстні блоки
- Списки ключових слів
- Визначення елементів стилю

Під час аналізу тексту правила визначення контексту застосовуватимуться у порядку, у якому ці правила було визначено у файлі визначень, — якщо початок поточного рядка відповідає певному правилу, буде використано відповідний контекст. Після цього початкову точку у тексті буде пересунуто у завершальну точку застосування визначеного правила і почнеться новий цикл пошуку відповідників правил в межах контексту, встановленого попереднім правилом.

6.2.2.2 Правила

Правила визначення є основою системи визначення підсвічування. Кожне правило визначається рядком, символом або формальним виразом, з яким порівнюватиметься текст документа. Правилом визначаються відомості, які буде використано під час визначення стилю відповідного фрагмента тексту. Правило може перемкнути поточний контекст системи підсвічування або на явно вказаний у правилі контекст або на попередній контекст, який було використано у тексті.

Правила об'єднуються у контекстні групи. Кожна контекстна група реалізує основні елементи у відповідному форматі файлів, наприклад текстові рядки у лапках або блоки коментарів у файлах кодів програми. За такої структури системи підсвічування можна уникнути потреби у переборі всього набору правил, коли такий перебір не потрібен, а також мати можливість різного трактування деяких послідовностей символів у тексті, залежно від поточного контексту.

Контексти можуть створюватися і динамічно, таким чином забезпечується використання у правилах особливостей даних документа.

6.2.2.3 Контекстні стилі і ключові слова

У деяких мовах програмування цілі числа обробляються компілятором (програмою, яка перетворює початкові коди програми на бінарний файл програми) у спосіб, відмінний від способу обробки чисел з плаваючою крапкою, також у рядках, взятих у лапки, можуть бути символи зі спеціальним призначенням. У таких випадках було б доцільним виокремити подібні символи, щоб їх простіше було виявити під час читання коду. Отже, навіть якщо ці символи не мають окремого контексту, система підсвічування тексту показати їх так, неначе подібний контекст для них існує, тобто виокремити їх з-поміж навколишнього тексту.

У визначенні синтаксису може бути довільна кількість стилів, достатня для визначення всіх елементів формату тексту, для якого це визначення було створено.

У багатьох форматах існують списки слів, які відповідають певному елементу. Наприклад, у мовах програмування керівні команди складають один елемент, назви типів даних — другий, вбудовані функції мови — третій. Система підсвічування синтаксису KatePart здатна використовувати такі списки для виявлення і позначення слів у тексті з метою підкреслення призначення елементів у текстових форматах.

6.2.2.4 Типові стилі

Якщо ви відкриєте файл кодів мовою C++, Java™ або документ HTML у KatePart, ви переконаєтеся у тому, що, хоча формати цих файлів є різними, а отже у них використовуються різні слова для позначень елементів тексту, кольори, які буде використано програмою будуть тими самими. Причиною цього є існування у KatePart наперед визначеного списку типових стилів, який використовується у окремих визначеннях синтаксису.

Типові стилі спрощують розпізнавання подібних елементів у різних форматах тексту. Наприклад, коментарі передбачено майже у всіх мовах програмування, написання скриптів або мовах розмітки, отже, якщо їх буде показано у однаковому стилі у всіх форматах мов, вам не потрібно буде зупинятися і розмірковувати над тим, як виглядають коментарі у тексті документа.

> **Підказка**
> Під час створення всіх стилів визначення синтаксису використовують один з типових стилів. У деякій частині визначень синтаксису використовуються додаткові стилі, яких немає серед типових, отже, якщо ви часто працюєте з файлами у таких форматах, доцільно відкрити діалогове вікно налаштування, щоб подивитися, чи не використовуються для певних елементів однакові стилі. Наприклад, існує лише один типовий стиль для рядків, але, оскільки у мові програмування Perl існує два типи рядків, ви можете налаштувати підсвічування для кожного з цих типів трохи по-різному. Огляд всіх можливих типових стилів буде наведено далі.

6.2.3 Формат XML визначення підсвічування

6.2.3.1 Огляд

У KatePart використовується бібліотека підсвічування синтаксичних конструкцій з KDE Frameworks. Типово, до бібліотеки підсвічування синтаксичних конструкцій KatePart включено засоби підсвічування коду XML.

Цей розділ присвячено огляду формату XML визначення підсвічування. За допомогою невеличкого прикладу у розділі описано основні компоненти, їх призначення і використання. Далі ми докладно розглянемо правила визначення підсвічування синтаксису.

Формальне визначення, також відоме як XSD, зберігається у сховищі підсвічування синтаксису у файлі `language.xsd`.

Нетипові файли `.xml` підсвічування синтаксису зберігаються у `org.kde.syntax-highlighting/syntax/` у теці вашого користувача, у підтеці, назву якої можна визначити за допомогою команди `qtpaths--paths GenericDataLocation`, зазвичай `$HOME /.local/share/` і `/usr/share/` .

Для пакунків Flatpak і Snap місце зберігання даних є різним для різних програм. У пакунках Flatpak нетипові файли XML зберігаються, зазвичай, у `$HOME /.var/app/ назва-пакунка-flatpak /data/org.kde.syntax-highlighting/syntax/` , а у пакунках Snap — у `$HOME /snap/ назва-пакунка-snap /current/.local/share/org.kde.syntax-highlighting/syntax/` .

У Windows® ці файли зберігаються у `%USERPROFILE%\AppData\Local\org.kde.syntax-highlighting\syntax`. *`%USERPROFILE%`* зазвичай є скороченням від `C:\Users\користувач`.

Загалом, для більшості конфігурацій каталогом нетипових файлів XML є такий каталог:

Для окремого користувача:	`$HOME /.local/share/org.kde.syntax-highlighting/syntax/`
Для усіх користувачів?	`/usr/share/org.kde.syntax-highlighting/syntax/`
Для пакунків Flatpak:	`$HOME /.var/app/ назва-пакунка-flatpak /data/org.kde.syntax-highlighting/syntax/`
Для пакунків Snap:	`$HOME /snap/ назва-пакунка-snap /current/.local/share/org.kde.syntax-highlighting/syntax/`
У Windows®:	`%USERPROFILE%\AppData\Local\org.kde.syntax-highlighting\syntax`
У macOS®	`$HOME /Library/Application Support/org.kde.syntax-highlighting/syntax/`

Якщо для якоїсь мови існує декілька файлів, буде завантажено файл із найбільшим значенням атрибута `version` в елементі `language`.

Головні розділи файлів визначення підсвічування KatePart

Файл підсвічування містить заголовок, у якому встановлюється версія XML:

```
<?xml version="1.0" encoding="UTF-8"?>
```

Кореневим елементом файла визначення є елемент `language`. Серед можливих атрибутів:

Обов'язкові атрибути:

`name` визначає назву мови. Цю назву буде згодом показано у пунктах меню та діалогових вікнах програми.

`section` визначає категорію.

`extensions` визначає суфікси назв файлів, на зразок ″*.cpp;*.h″

`version` визначає поточну модифікацію файла визначень у форматі цілого числа. Кожного разу, коли ви вноситимете зміни до файла визначень правил підсвічування, вам слід збільшувати це число.

`kateversion` визначає найостаннішу з підтримуваних версій KatePart.

Необов'язкові атрибути:

`mimetype` прив'язує файли на основі типу MIME.

`casesensitive` визначає, чи розрізнятимуться ключові слова за регістром символів, чи ні.

`priority` потрібно вказати, якщо у іншому файлі визначення підсвічування використовуються ті самі суфікси файлів. Для підсвічування буде використано правила з вищим пріоритетом.

`author` повинен містити ім'я і адресу електронної пошти автора.

`license` повинен містити назву ліцензії нового файла підсвічування, зазвичай, MIT.

`style` має містити дані щодо мови програмування і використовується засобами додавання відступів для атрибута `required-syntax-style`.

`indenter` визначає, який із засобів додавання відступів у рядки буде використано типово. Доступні засоби додавання відступів: *ada, normal, cstyle, cmake, haskell, latex, lilypond, lisp, lua, pascal, python, replicode, ruby* та *xml*.

`hidden` визначає, чи має бути показано назву підсвічування у меню KatePart.

Отже, наступний рядок має виглядати десь так:

```
<language name="C++" version="1" kateversion="2.4" section="Sources ↩
    " extensions="*.cpp;*.h" />
```

Наступним елементом є `highlighting`, у цьому елементі міститься необов'язковий елемент `list` і обов'язкові елементи `contexts` і `itemDatas`.

Елементи `list` містять список ключових слів. У цьому випадку ключовими словами будуть *class* і *const*. У разі потреби ви можете додати довільну кількість списків.

Починаючи з KDE Frameworks 5.53, до списку можна включати ключові слова з іншого списку або мови чи файла за допомогою елемента `include`. Для відокремлення назви списку та назви визначення мови слід використовувати `##` у той самий спосіб, що і у правилі `IncludeRules`. Ця можливість є корисною для уникнення дублювання списків ключових слів, якщо вам потрібно включити ключові слова з іншої мови або файла. Наприклад, у списку *othername* міститься ключове слово *str* і усі ключові слова списку *types*, який належить до мови програмування *ISO C++*.

У елементі `contexts` містяться всі контексти. Типово, першим контекстом є початок діапазону підсвічування. У контексті *Normal Text* існує два правила: перше визначає список ключових слів з назвою *somename* і правило для визначення лапок і перемикання контексту на контекст *string*. Докладніше правила буде розглянуто у наступній главі.

Третя частина складається з елемента `itemDatas`. У цьому елементі містяться всі кольори і гарнітури шрифтів, потрібні для контекстів і правил. У нашому випадку було використано `itemData` *Normal Text*, *String* і *Keyword*.

```
<highlighting>
    <list name="somename">
      <item>class</item>
      <item>const</item>
    </list>
    <list name="othername">
      <item>str</item>
      <include>types##ISO C++</include>
    </list>
    <contexts>
      <context attribute="Normal Text" lineEndContext="#pop" name=" ↩
          Normal Text" >
        <keyword attribute="Keyword" context="#stay" String=" ↩
            somename" />
        <keyword attribute="Keyword" context="#stay" String=" ↩
            othername" />
        <DetectChar attribute="String" context="string" char="&quot ↩
            ;" />
      </context>
      <context attribute="String" lineEndContext="#stay" name=" ↩
          string" >
        <DetectChar attribute="String" context="#pop" char="&quot;" ↩
            />
      </context>
    </contexts>
    <itemDatas>
      <itemData name="Normal Text" defStyleNum="dsNormal" />
      <itemData name="Keyword" defStyleNum="dsKeyword" />
      <itemData name="String" defStyleNum="dsString" />
    </itemDatas>
  </highlighting>
```

Останньою частиною визначення підсвічування є необов'язковий розділ `genera l`. У ньому можуть міститися відомості щодо ключових слів, згортання коду, коментарів, відступів, порожніх рядків та перевірки правопису.

У розділі `comment` визначається послідовність символів, за допомогою якої можна додати однорядковий коментар. Крім того, ви можете визначати багаторядкові коментарі за допомогою елемента *multiLine* з додатковим атрибутом *end*. Ці визначення буде використано за виконання користувачем дій *закоментувати/розкоментувати*.

У розділі `keywords` визначається те, чи слід враховувати регістр символів у списку ключових слів, чи ні. Інші атрибути буде описано далі за текстом.

В інших розділах, `folding`, `emptyLines` і `spellchecking`, зазвичай, потреби немає. Пояснення щодо цих розділів наведено нижче.

```
<general>
    <comments>
      <comment name="singleLine" start="#"/>
      <comment name="multiLine" start="###" end="###" region=" ↩
          CommentFolding"/>
    </comments>
    <keywords casesensitive="1"/>
    <folding indentationsensitive="0"/>
    <emptyLines>
      <emptyLine regexpr="\s+"/>
      <emptyLine regexpr="\s*#.*"/>
    </emptyLines>
    <spellchecking>
```

```
      <encoding char="á" string="\'a"/>
      <encoding char="à" string="\`a"/>
    </spellchecking>
  </general>
</language>
```

6.2.3.2 Докладно про розділи

У цій частині описано всі доступні атрибути для контекстів, itemDatas, ключових слів, коментарів, згортання коду і відступів.

Елемент `context` належить до групи `contexts`. Цей елемент визначає специфічні для контексту правила, на зразок того, що має трапитися, якщо система підсвічування досягне кінця рядка. Серед можливих атрибутів:

`name` — назва контексту. За допомогою цієї назви правила визначатимуть контекст, на який слід перемкнутися у випадку виявлення відповідника правила.

`attribute` визначає стиль, який слід використати для символу, якщо не виявлено відповідності жодному правилу або якщо правило не визначає атрибута. У другому випадку буде використано *атрибут* контексту, який вказано у *контексті* правила.

`lineEndContext` визначає контекст, на який має перемкнутися система підсвічування у разі досягнення кінця рядка. Значенням може бути назва іншого контексту, `#stay` означатиме, що контекст не слід перемикати (тобто нічого не робити), або `#pop` означатиме, що слід вийти з контексту. Наприклад, можна скористатися значенням `#pop#pop#pop` для того, щоб система після виходу піднялася на три контексти вище, або навіть `#pop#pop!ІншийКонтекст`, щоб піднятися на два контексти вище і перемкнутися на контекст з назвою `ІншийКонтекст`. Також можна перемкнутися на контекст, який належить визначенню іншої мови, так само, як у правилах `IncludeRules`, наприклад, `ЯкийсьКонтекст##JavaScript`. Перемикачі контексту також описано у розділі Розділ 6.2.4. Типове значення: #stay.

`lineEmptyContext` визначає контекст, якщо буде знайдено порожній рядок. Номенклатура перемикань контексту є такою самою, як і раніше описано для *lineEndContext*. Типове значення: #stay.

`fallthroughContext` вказує наступний контекст для перемикання, якщо жодне з правил не є відповідним. Номенклатура перемикань контексту є такою самою, як і раніше описано для *lineEndContext*. Типове значення: #stay.

`fallthrough` визначає, чи перемикатиметься система підсвічування на контекст, визначений у `fallthroughContext`, якщо жодне з правил не буде визнано відповідним. Зауважте, що з версії KDE Frameworks 5.62 цей атрибут є застарілим. Замість нього слід використовувати `fallthroughContext`, оскільки якщо вказано атрибут `fallthroughContext`, неявним чином припускатиметься значення `fallthrough` рівне *true*. Типове значення: *false*.

`noIndentationBasedFolding` вимикає згортання на основі відступів у контексті. Якщо згортання на основі відступів не увімкнено, цей атрибут не має сенсу. Цей атрибут визначається у елементі *folding* групи *general*. Типове значення: *false*.

Елемент `itemData` знаходиться у групі `itemDatas`. За його допомогою визначаються тип шрифту і кольори. Таким чином можна визначати власні типи шрифтів і кольори, але ми рекомендуємо, за можливості, використовувати лише типові стилі так, щоб користувач завжди бачив однакові кольори у файлах різних форматів. Все ж іноді іншого способу не існує, і вам доведеться змінити колір і атрибути шрифту. Обов'язковими атрибутами є name і defStyleNum, інші атрибути є необов'язковими. Серед можливих атрибутів:

`name` визначає назву itemData. Цю назву буде використано у контекстах і правилах для посилання на itemData у атрибуті *attribute*.

`defStyleNum` визначає, який тип слід використовувати типово. Докладний перелік можливих типових стилів ви знайдете нижче.

`color` визначає колір. Можливі формати: '#rrggbb' і '#rgb'.

`selColor` визначає колір виділеного тексту.

`italic`, якщо має значення *true*, шрифт буде курсивним.

`bold`, якщо має значення *true*, шрифт тексту буде напівжирним.

`underline`: якщо цей атрибут має значення *true*, текст буде підкреслено.

`strikeOut`, якщо цей параметр має значення *true*, текст буде перекреслено.

`spellChecking`: якщо цей атрибут має значення *true*, правопис тексту буде перевірено.

Елемент `keywords` у групі `general` визначає властивості ключових слів. Серед можливих атрибутів:

`casesensitive`, може приймати значення *true* або *false*. Якщо має значення *true*, пошук ключових слів відбуватиметься з врахуванням регістру символів.

`weakDeliminator` — це список символів, які не є символами відокремлення слів. Наприклад, крапка «.» є символом відокремлення слів. Припустімо тепер, що ключове слово у `list` містить крапку, тоді це слово буде задіяно, лише якщо ви вкажете крапку серед значень цього аргументу.

`additionalDeliminator` визначає додаткові символи розмітки.

`wordWrapDeliminator` визначає символи, після яких можна розривати рядок.

Типовими символами розмітки і символами відокремлення слів є символи `.():!+,-<=>%&*/;?[]^{|}~\`, пробіл (`' '`) і символ табуляції (`'\t'`).

Елемент `comment` у групі `comments` визначає властивості коментаря, які буде використано для дій за пунктами меню Інструменти → Закоментувати, Інструменти → Розкоментувати and Інструменти → Додати або вилучити коментування. Серед доступних атрибутів:

`name`: може мати значення *singleLine* або *multiLine*. Якщо буде обрано варіант *multiLine*, слід буде також вказати атрибути *end* і *region*. Якщо ви виберете *singleLine*, ви зможете додати необов'язковий атрибут *position*.

`start` визначає рядок, який використовується для позначення початку коментаря. У C++ цим рядком для багаторядкових коментарів є ″/*″. Цей атрибут є обов'язковим для типів *multiLine* і *singleLine*.

`end` визначає рядок, яким завершуватиметься коментар. У C++ цим рядком буде ″*/″. Цей атрибути є доступним лише для коментарів типу *multiLine* і є для них обов'язковим.

Атрибут `region` повинен мати значення назви придатного для згортання багаторядкового коментаря. Припустімо, що у правилах вказано *beginRegion=″Comment″ ... endRegion=″Comment″*, тоді вам слід взяти значення *region=″Comment″*. За допомогою цього атрибута можна користуватися дією з розкоментування, навіть якщо було виділено не весь текст багаторядкового коментаря. Потрібно лише, щоб курсор знаходився всередині цього багаторядкового коментаря. Цей атрибут є доступним лише для типу *multiLine*.

`position` визначає місце вставлення однорядкового коментування. Типово, однорядкове коментування буде вставлено на початку рядка у позиції 0, але якщо ви скористаєтеся записом *position=″afterwhitespace″*, коментування буде вставлено праворуч після першого пробільного блоку, перед першим непробільним символом. Ця можливість є корисною для мов, де важливими є відступи у рядках, зокрема Python та YAML. Цей атрибут є необов'язковим, його єдиним можливим значенням є *afterwhitespace*. Цей атрибут є доступним лише для типу *singleLine*.

Елемент `folding` у групі `general` призначено для визначення властивостей згортання коду. Серед можливих атрибутів:

`indentationsensitive`: якщо має значення *true*, позначки згортання коду буде додано на основі відступів, так, як це робиться у скриптовій мові Python. Зазвичай, вам не потрібно буде встановлювати цей атрибут, оскільки його типовим значенням є *false*.

Елемент `emptyLine` у групі `emptyLines` визначає, які з рядків слід вважати порожніми. За його допомогою можна змінити поведінку атрибута *lineEmptyContext* в елементах `context`. Наявні атрибути:

`regexpr` визначає формальний вираз, відповідник якого вважатиметься порожнім рядком. Типово, порожні рядки не містять жодних символів, тому цей формальний вираз додає «порожні» рядки, наприклад, якщо ви хочете вважати рядки з пробілів порожніми. Втім, здебільшого, для визначення синтаксису у цьому атрибуті немає потреби.

Елемент `encoding` у групі `spellchecking` визначає кодування символів для перевірки правопису. Наявні атрибути:

`char` — кодований символ.

`string` — послідовність символів, яку буде закодовано як символ *char* при перевірці правопису. Наприклад, якщо виконується обробка коду мовою LaTeX, рядок `\¨{A}` відповідатиме символу `Ä`.

6.2.3.3 Можливі типові стилі

Ви вже обговорювали типові стилі у короткому резюме: типовими стилями є наперед визначені набори з гарнітур шрифтів та кольорів.

Загальні типові стилі:

`dsNormal`, якщо непотрібне спеціальне підсвічування.

`dsKeyword`, вбудовані ключові слова мови.

`dsFunction`, виклики і визначення функцій.

`dsVariable`, якщо застосовне: назви змінних (наприклад $someVar у PHP/Perl).

`dsControlFlow`, ключові слова керування обробкою, зокрема if, else, switch, break, return, yield, ...

`dsOperator`, оператори, зокрема + - * / :: < >

`dsBuiltIn`, вбудовані функції, класи і об'єкти.

`dsExtension`, загальні розширення, зокрема класи Qt™ та функції і макроси у C++ та Python.

`dsPreprocessor`, інструкції препроцесора або визначення макросів.

`dsAttribute`, анотації, зокрема @override та __declspec(...).

Типові стилі, пов'язані із рядками:

`dsChar`, окремі символи, зокрема 'x'.

`dsSpecialChar`, символи із спеціальним призначенням у рядках, зокрема символи екранування, замінники або оператори формальних виразів.

`dsString`, рядки, зокрема "hello world".

`dsVerbatimString`, буквальні або необроблювані рядки, зокрема «raw \backlash» у Perl, CoffeeScript та командних оболонках, а також r'\raw' у Python.

`dsSpecialString`, SQL, формальні вирази, документація HERE, математичний режим $\LaTeX$...

`dsImport`, імпорт, включення або потреба у модулях.

Пов'язані із числами типові стилі:

`dsDataType`, вбудовані типи даних, наприклад int, void, u64.

`dsDecVal`, десяткові значення.

`dsBaseN`, величини у численні з основою, відмінною від 10.

`dsFloat`, значення із рухомою крапкою.

`dsConstant`, вбудовані та визначені користувачем сталі, наприклад PI.

Типові стилі, пов'язані із коментаріми та документацією:

`dsComment`, коментарі.

`dsDocumentation`, /** Коментарі у документації */ або """docstrings""".

`dsAnnotation`, команди документації, зокрема @param, @brief.

`dsCommentVar`, назви змінних, використаних у попередніх командах, зокрема «foobar» у @param foobar.

`dsRegionMarker`, позначки області, зокрема //BEGIN, //END у коментарях.

Інші типові стилі:

`dsInformation`, нотатки і підказки, наприклад @note у doxygen.

`dsWarning`, попередження, наприклад @warning у doxygen.

`dsAlert`, спеціальні слова, наприклад TODO, FIXME, XXXX.

`dsError`, підсвічування помилок та синтаксичних неточностей.

`dsOthers`, якщо інше не є застосовним.

6.2.4 Правила визначення способу підсвічування

Цей розділ присвячено опису правил визначення синтаксису.

Кожне з правил може відповідати нульовій або більшій кількості символів на початку рядка, у якому шукатиметься відповідник правила. Якщо такий відповідник буде знайдено, знайдені символи визначать стиль або *attribute*, вказані за допомогою правила, правило також може надіслати системі запит на зміну поточного контексту.

Правило виглядає так:

```
<НазваПравила attribute="(ідентифікатор)" context="(ідентифікатор)" [спе ↩
    цифічні для правила атрибути] />
```

Значення *attribute* визначає назву стилю, який буде використано для відповідних символів, а значення *context* визначає контекст, який слід використовувати, починаючи з цього місця.

context може бути визначено за допомогою:

- *Ідентифікатора*, який є назвою іншого контексту.
- Значення *порядку*, яке повідомляє рушієві, чи слід залишатися у поточному контексті (`#stay`), чи слід повернутися до попереднього контексту, використаного у рядку (`#pop`). Порожній або невизначений контекст означає `#stay`.

  Щоб повернутися на декілька рівнів контексту назад, ключове слово #pop можна повторити декілька разів: `#pop#pop#pop`
- Значення *порядку*, після якого вказано знак оклику (*!*), та значення *ідентифікатора*, яке змусить рушій спочатку використати порядок, а потім перемкнутися на інший контекст, наприклад `#pop#pop!OtherContext`.
- *Ідентифікатор*, який є назвою контексту, за яким вказано два символи решітки (`##`) і ще один *ідентифікатор*, який є назвою визначення мови. Таке іменування є подібним до використаного у правилах `IncludeRules`. Воно надає вам змогу перемкнутися на контекст, що належить іншому визначення підсвічування синтаксису, наприклад, `ЯкийсьКонтекст##JavaScript`.

Специфічні для правила атрибути можуть бути різними, їх описано у наступних розділах.

Загальні атрибути

- *attribute*: атрибут, що вказує на визначені *itemData*. Типове значення: *attribute* з контексту, який вказано в атрибуті *context*.
- *context*: визначає контекст, на який слід перемкнути систему підсвічування у разі виявлення відповідника правила. Типове значення: #stay.

- *beginRegion*: почати блок згортання коду. Типове значення: unset.
- *endRegion*: закрити блок згортання коду. Типове значення: unset.
- *lookAhead*: якщо має значення *true*, система підсвічування не оброблятиме довжину відповідника. Типове значення: *false*.
- *firstNonSpace*: відповідність встановлюватиметься, лише якщо рядок є першою відмінною від пробілів послідовністю символів у рядку тексту. Типове значення: *false*.
- *column*: збіг буде зареєстровано, якщо збігатиметься рядок. Типове значення: unset.

Динамічні правила

- *dynamic*: може мати значення *(true/false)*.

`Як це працює:`

У формальних виразах правил `RegExpr` засіб обробки захоплює і запам'ятовує увесь текст у простих круглих дужках — `(ШАБЛОН)`. Захоплені фрагменти тексту можна використовувати у контексті, до якого перемикається засіб, у правилах із атрибутом `dynamic` *true* для заміни *%N* (у *String*) або *N* (у *char*).

Важливо пам'ятати, що фрагмент тексту, який захоплено у правилі `RegExpr`, зберігається лише у перемкнутому контексті, який вказано за допомогою атрибута `context` елемента.

> **Підказка**
>
> - Якщо у запам'ятовуванні фрагментів тексту немає потреби для побудови динамічних правил або самого формального виразу, слід використовувати групування без захоплення: `(?:ШАБЛОН)`
>
> Захоплення груп із *пошуком вперед* і *пошуком назад*, зокрема за допомогою формальних виразів `(?=ШАБЛОН)`, `(?!ШАБЛОН)` або `(?<=ШАБЛОН)`, не відбуватиметься. Щоб дізнатися більше, ознайомтеся з розділом Формальні вирази.
>
> - Захоплені групи можна використовувати у межах того самого формального виразу за допомогою рядка *\N* замість *%N*. Щоб дізнатися про це більше, ознайомтеся із розділом Збереження знайденого тексту (зворотні посилання) у главі Формальні вирази.

Приклад 1:

У цьому простому прикладі обробник захоплює текст, який відповідає формальному виразу `=*`, і вставляє його замість *%1* у динамічному правилі. Таким чином можна визначити коментар, який завершується тією самою кількістю символів `=`, що і починається. Відповідним текстом буде `[[ коментар ]]`, `[=[ коментар ]=]` та `[=====[ коментар ]=====]`.

Крім того, захоплені дані доступні лише у перемкнутому контексті *Multi-line Comment*.

```
<context name="Normal" attribute="Normal Text" lineEndContext="#stay">
  <RegExpr context="Multi-line Comment" attribute="Comment" String ↩
     ="\[(=*)\[" beginRegion="RegionComment"/>
</context>
<context name="Multi-line Comment" attribute="Comment" lineEndContext="# ↩
    stay">
  <StringDetect context="#pop" attribute="Comment" String="]%1]" dynamic ↩
     ="true" endRegion="RegionComment"/>
</context>
```

Приклад 2:

У динамічному правилі *%1* відповідає захопленому фрагменту тексту, який відповідає шаблону `#+`, а *%2* — шаблону `&quot;+`. Отже, відповідний фрагмент тексту буде таким: `#мітка˝˝˝˝у контексті˝˝˝˝#`.

Захопленими даними не можна буде скористатися у інших контекстах, зокрема *OtherContext*, *FindEscapes* або *SomeContext*.

```
<context name="SomeContext" attribute="Normal Text" lineEndContext="# ↩
    stay">
  <RegExpr context="#pop!NamedString" attribute="String" String="(#+) ↩
     (?:[\w-]|[^[:ascii:]])(&quot;+)"/>
</context>
<context name="NamedString" attribute="String" lineEndContext="#stay">
  <RegExpr context="#pop!OtherContext" attribute="String" String ↩
     ="%2(?:%1)?" dynamic="true"/>
  <DetectChar context="FindEscapes" attribute="Escape" char="\"/>
</context>
```

Приклад 3:

Цей запис відповідає тексту, подібному до такого: `Class::function<T>( ... )`.

```
<context name="Normal" attribute="Normal Text" lineEndContext="#stay">
  <RegExpr context="FunctionName" lookAhead="true"
              String="\b([a-zA-Z_][\w-]*)(::)([a-zA-Z_][\w-]*)(?:&lt;[\w ↩
                 \-\s]*&gt;)?(\()"/>
</context>
<context name="FunctionName" attribute="Normal Text" lineEndContext="# ↩
    pop">
  <StringDetect context="#stay" attribute="Class" String="%1" dynamic=" ↩
     true"/>
  <StringDetect context="#stay" attribute="Operator" String="%2" dynamic ↩
     ="true"/>
  <StringDetect context="#stay" attribute="Function" String="%3" dynamic ↩
     ="true"/>
  <DetectChar context="#pop" attribute="Normal Text" char="4" dynamic=" ↩
     true"/>
</context>
```

Локальні роздільники

- *weakDeliminator*: список символів, які не є роздільниками слів.
- *additionalDeliminator*: визначає додаткові роздільники.

6.2.4.1 Докладно про правила

DetectChar

Перевірка на рівність одному певному символу. Зазвичай, використовується для пошуку кінця рядків, взятих у лапки.

```
<DetectChar char="(символ)" (загальні атрибути) (dynamic) />
```

Атрибут `char` визначає символ, з яким відбуватиметься порівняння.

Detect2Chars

Перевірка на рівність двом певним символам у вказаному порядку.

```
<Detect2Chars char="(символ)" char1="(символ)" (загальні атрибути)  ↩
    />
```

Атрибут `char` визначає перший символ для порівняння, `char1` — другий.

Це правило лишилося з історичних міркувань. Для того, щоб зробити читання зручнішим варто користуватися `StringDetect`.

AnyChar

Перевірка на рівність одному з символів вказаного набору.

```
<AnyChar String="(рядок)" (загальні атрибути) />
```

Атрибут `String` визначає набір символів.

StringDetect

Перевірка на рівність вказаному рядку.

```
<StringDetect String="(рядок)" [insensitive="true|false"] (загальні ↩
    атрибути) (dynamic) />
```

Атрибут `String` визначає рядок для порівняння. Типовим значенням атрибута `insensitive` є *false*, цей атрибут передається функції порівняння рядків. Якщо значенням атрибута буде *true*, порівняння відбуватиметься без врахування регістру.

WordDetect

Виявити рядок, але з додатковою вимогою щодо меж слів, зокрема крапки, '.', або пробілу на початку або у кінці слова. Обробка `\b<рядок>\b` відбувається подібно до формального виразу, але виконується швидше за обробку правила `RegExpr`.

```
<WordDetect String="(рядок)" [insensitive="true|false"] (загальні а ↩
   трибути) (локальні роздільники) />
```

Атрибут `String` визначає рядок для порівняння. Типовим значенням атрибута `insensitive` є *false*, цей атрибут передається функції порівняння рядків. Якщо значенням атрибута буде *true*, порівняння відбуватиметься без врахування регістру.

Починаючи з Kate 3.5 (KDE 4.5)

RegExpr

Перевірка на збіг з формальним виразом.

```
<RegExpr String="(рядок)" [insensitive="true|false"] [minimal="true ↩
   |false"] (загальні атрибути) (dynamic) />
```

Атрибут `String` визначає формальний вираз.

Типовим значенням атрибута `insensitive` є *false*, цей атрибут передається рушію пошуку за формальним виразом.

Типовим значенням атрибута `minimal` є *false*, цей атрибут буде передано рушієві пошуку за формальним виразом.

Оскільки пошук відповідників для застосування правила завжди відбувається на початку поточного рядка, формальний вираз, що починається з символу каретки (^) вказує на те, що пошук відповідника правила слід виконувати лише на початку рядка.

Щоб дізнатися більше, ознайомтеся з розділом Формальні вирази.

keyword

Перевірка на рівність ключовому слову з вказаного списку.

```
<keyword String="(назва списку)" (загальні атрибути) (локальні розд ↩
   ільники) />
```

Атрибут `String` визначає назву списку ключових слів. Список з вказаною назвою має існувати.

Система підсвічування обробляє правила ключових слів у дуже оптимізований спосіб. Тому абсолютно необхідно, щоб усі ключові слова, які слід знайти, було обмежено визначеними роздільниками, заздалегідь передбаченими (типовими роздільниками) або явно визначеними у властивості *additionalDeliminator* теґу *keywords*.

Якщо ключове слово, яке слід знайти, має містити символ роздільника, відповідний символ слід додати до властивості *weakDeliminator* теґу *keywords*. Після цього символ втратить властивість роздільності у всіх правилах *keyword*. Також можна скористатися атрибутом *weakDeliminator* теґу *keyword* так, щоб ці конкретні зміни застосовувалися лише до цього конкретного правила.

Int

Виявлення цілого числа (формальний вираз: `\b[0-9]+`).

```
<Int (загальні атрибути) (локальні роздільники) />
```

У цього правила немає особливих атрибутів.

Float

Виявлення числа із рухомою крапкою (формальний вираз: `(\b[0-9]+\.[0-9]*|\.[0-9]+)([eE][-+]?[0-9]+)?`).

```
<Float (загальні атрибути) (локальні роздільники) />
```

У цього правила немає особливих атрибутів.

HlCOct

Виявлення вісімкового представлення числа (формальний вираз: `\b0[0-7]+`).

```
<HlCOct (загальні атрибути) (локальні роздільники) />
```

У цього правила немає особливих атрибутів.

HlCHex

Виявлення шістнадцяткового представлення числа (формальний вираз: `\b0[xX][0-9a-fA-F]+`).

```
<HlCHex (загальні атрибути) (локальні роздільники) />
```

У цього правила немає особливих атрибутів.

HlCStringChar

Перевірка на відповідність символу керівної послідовності.

```
<HlCStringChar (загальні атрибути) />
```

У цього правила немає особливих атрибутів.

Перевірка на відповідність символам, які часто використовуються у коді програм, наприклад `\n` (перехід на новий рядок) або `\t` (табуляція).

Пошук буде виконуватися за переліченими далі символами, якщо ці символи стоять одразу за зворотною навскісною рискою (\): `abefnrtv~'?\`. Крім того, відповідними вважатимуться шістнадцяткові числа, наприклад `\xff`, і екрановані вісімкові числа, наприклад `\033`.

HlCChar

Перевірка на відповідність символу C.

```
<HlCChar (загальні атрибути) />
```

У цього правила немає особливих атрибутів.

Перевірка на відповідність символам C, взятим у одинарні лапки (Приклад: `'c'`). Отже, у таких лапках може бути простий символ або екранований символ. Щоб дізнатися про пошук екранованих послідовностей символів, перегляньте пункт для HlCStringChar.

RangeDetect

Перевірка на відповідність рядку з вказаними початковим і кінцевим символами.

```
<RangeDetect char="(символ)"  char1="(символ)" (загальні атрибути)  ↩
   />
```

`char` визначає символ, який повинен починати діапазон символів, `char1` — символ, який має завершувати діапазон.

Корисно для виявлення, наприклад, невеличких рядків у лапках, але зауважте, що, оскільки рушій підсвічування обробляє за раз лише один рядок, у такий спосіб неможливо знайти рядки у лапках, які розбито між декількома рядками документа.

LineContinue

Перевірка на відповідність вказаному символу наприкінці рядка.

```
<LineContinue (загальні атрибути) [char="\"] />
```

Необов'язковий для встановлення відповідності атрибут `char`, типовим значенням є символ зворотної риски (`'\'`). Впроваджено з KDE 4.13.

Це правило корисне для перемикання контексту наприкінці рядка. Це потрібно, зокрема, у коді мовами C/C++ для продовження макросів або рядків.

IncludeRules

Включити правила з іншого контексту або мови/файла.

```
<IncludeRules context="посилання на контекст" [includeAttrib="true| ↩
   false"] />
```

Атрибут `context` визначає контекст, який слід включити.

Якщо значенням є простий рядок, у поточний контекст буде включено всі визначені правила, наприклад:

```
<IncludeRules context="anotherContext" />
```

Якщо у рядку міститься послідовність символів `##`, система підсвічування шукатиме контекст у іншому визначенні мови з вказаною назвою. Приклад:

```
<IncludeRules context="String##C++" />
```

включить контекст *String* з визначення правил підсвічування для *C++*.

Якщо атрибут `includeAttrib` матиме значення *true*, атрибут призначення буде змінено на атрибут джерела. Це потрібно для того, щоб, наприклад, виконати коментування, якщо текст, що відповідає знайденому контексту, має інше підсвічування, ніж текст у основному контексті.

DetectSpaces

Пошук пробілів.

```
<DetectSpaces (загальні атрибути) />
```

У цього правила немає особливих атрибутів.

Цим правилом можна скористатися, якщо вам точно відомо, що перед текстом рядка має бути декілька пробілів, наприклад, на початку рядків з відступом. За допомогою цього правила можна пропустити одразу всі пробіли, замість послідовної перевірки на основі декількох правил, кожне з яких надаватиме змогу відкидати по одному пробілу за один прийом через невідповідність.

DetectIdentifier

Пошук рядків ідентифікаторів (зокрема формальних виразів: `[a-zA-Z_][a-zA-Z0-9_]*`).

```
<DetectIdentifier (загальні атрибути) />
```

У цього правила немає особливих атрибутів.

Це правило слід використовувати для пропуску рядка з символів, які складають слова, замість послідовної перевірки на основі декількох правил, кожне з яких надаватиме змогу відкидати по одному символу за один прийом через невідповідність.

6.2.4.2 Підказки та поради

- Формальними виразами просто користуватися, але часто існує інший, набагато швидший спосіб досягти результату. Припустімо вам потрібно перевірити, чи є символ `'#'` першим символом рядка. Вирішення на основі формального виразу має виглядати десь так:

```
<RegExpr attribute="Macro" context="macro" String="^\s*#" />
```

  Того самого результату можна досягти набагато швидше за допомогою такого правила:

```
<DetectChar attribute="Macro" context="macro" char="#" firstNonSpace="
    true" />
```

  Якщо вам потрібно знайти формальний вираз `'^#'` ви знову ж таки можете скористатися правилом `DetectChar` з атрибутом `column="0"`. Відлік значення атрибута `column` засновано на кількості символів, отже табуляція вважатиметься лише одним символом.

- У правилах `RegExpr` користуйтеся атрибутом `column="0"`, якщо буде використано шаблон `^ШАБЛОН` для встановлення відповідності тексту на початку рядка. У таких спосіб ви пришвидшите обробку, оскільки у засобу обробки не буде потреби у пошуку у решті позицій рядка.

- У формальних виразах користуйтеся групуванням без захоплення — `(?:ШАБЛОН)`, замість групування із захопленням — `(ШАБЛОН)`, якщо захоплені дані не буде використано у тому самому формальному виразі або у динамічних правилах. Таким чином, ви уникнете непотрібного зберігання даних.

- Ви можете перемикати контексти без обробки символів тексту. Припустімо, що вам потрібно перемкнути контекст у разі виявлення рядка `*/` , але також слід обробити цей рядок у наступному контексті. Ви можете скористатися наведеним нижче правилом, у якому атрибут `lookAhead` призведе до того, що інструмент визначення підсвічування збереже знайдений рядок для обробки у наступному контексті.

```
<StringDetect attribute="Comment" context="#pop" String="*/" lookAhead
    ="true" />
```

- Скористайтеся `DetectSpaces`, якщо вам відома точна кількість пробілів.

- Скористайтеся `DetectIdentifier` замість формального виразу `'[a-zA-Z_]\w*'`.

- За можливості, використовуйте типові стилі. Таким чином, ви полегшите користувачеві призвичаювання до середовища.

- Зазирайте до інших файлів XML, щоб дізнатися як інші люди реалізують складні правила.

- Ви можете перевірити коректність будь-якого файла XML за допомогою команди **validatehl.sh mySyntax.xml**. Файл `validatehl.sh` використовує `language.xsd`. Обидва ці файли зберігаються у сховищі бібліотеки підсвічування синтаксичних конструкцій.

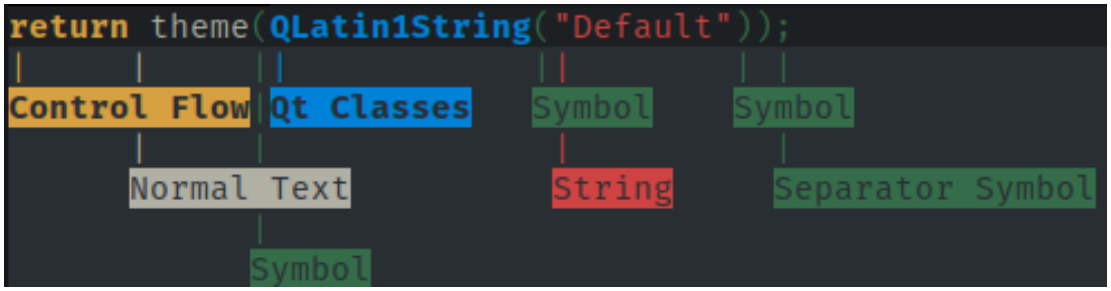
- Якщо у вашому файлі часто вживаються складні формальні вирази, ви можете скористатися визначенням *ENTITIES*. Приклад:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE language
[
        <!ENTITY myref    "[A-Za-z_:][\w.:_-]*">
]>
```

  Після такого визначення ви зможете використовувати *&myref;* замість формального виразу.

- У редакторі Kate ви можете перезавантажити синтаксиси за допомогою вбудованого командного рядка (типове скорочення `F7`) і команди **reload-highlighting**.

- Ви можете скористатися засобом командного рядка із назвою `ksyntaxhighlighter6` (`kate-syntax-highlighter` у старіших версіях) для тестування синтаксису і перегляду стилю та ділянок, які пов'язано із кожною частиною тексту.



Результат виконання ***ksyntaxhighlighter6 --output-format=ansi --syntax-trace=format test.cpp***.

Скористайтеся командою **ksyntaxhighlighter6 -h**, щоб дізнатися більше про параметри керування роботою програми.

6.3 Робота із темами кольорів

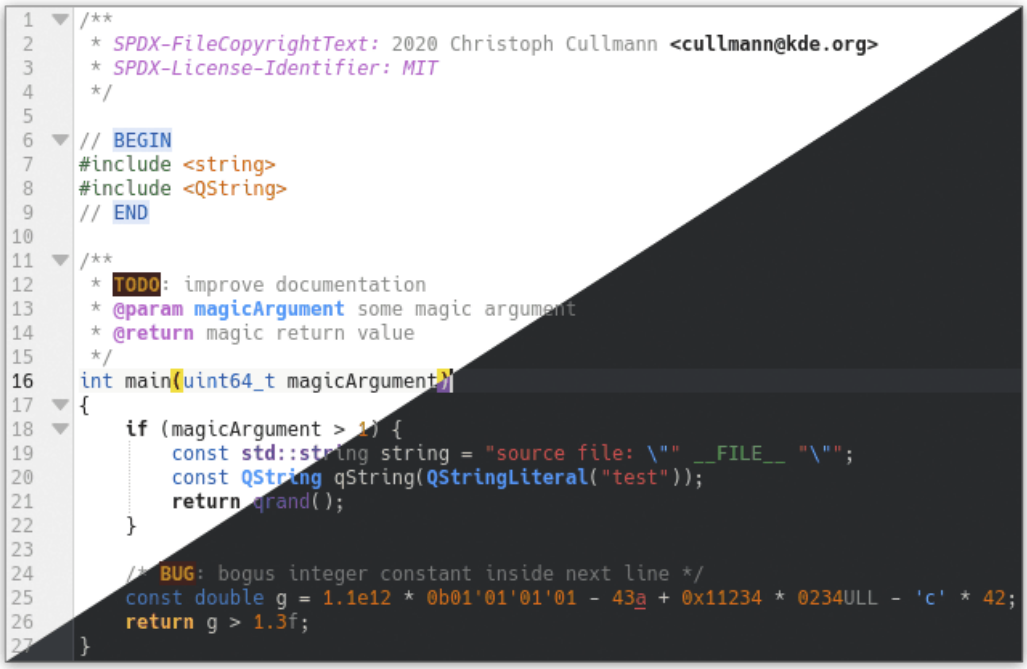
6.3.1 Огляд

Теми кольорів визначають кольори області редагування тексту і підсвічування синтаксису. До теми кольорів включено такі дані:

- Стиль тексту, який використано для підсвічування синтаксису в *атрибутах типових стилів*. Наприклад, колір тексту і колір позначеного тексту.
- Тло області редагування тексту, включно із позначенням тексту та поточним рядком.
- Рамка піктограм для текстової області: тло піктограм, лінія роздільника, номери рядків, позначки перенесення рядків, позначки змінених рядків та згортання коду.
- Декорації тексту, зокрема позначки пошуку, позначки відступів та табуляцій або пробілів, позначення відповідних дужок та розмітка при перевірці правопису.
- Закладки і фрагменти.

Щоб уникнути непорозумінь, це не стосується таких параметрів інтерфейсу:

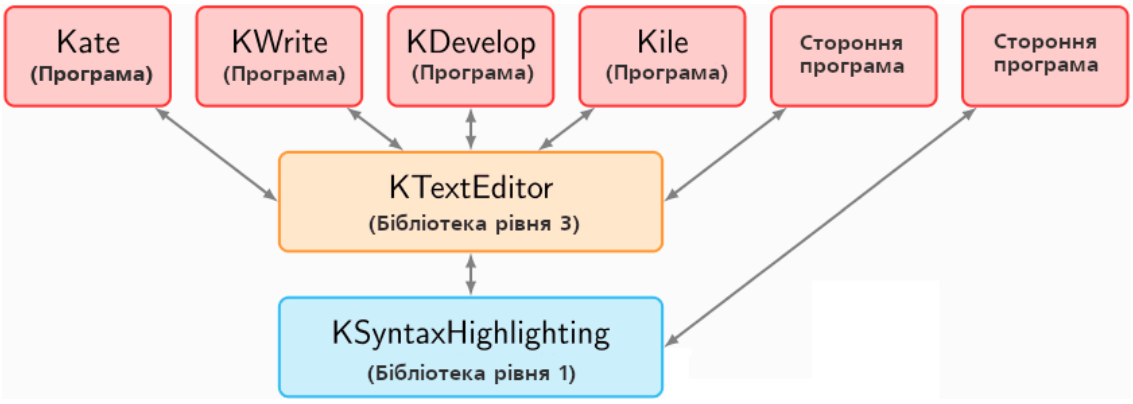
- Гарнітури і розміру символів шрифту.
- Кольорів у програмі для редагування тексту, зокрема на карті смужки гортання, у меню, на панелі вкладок, кольору вікна тощо. У програмах KDE, зокрема Kate або KDevelop, ці кольори визначаються `загальною схемою кольорів Плазми KDE`, яка визначається у модулі «Кольори» програми «Системні параметри», або у самій програмі, у меню **Параметри** → **Схема кольорів**.



Схеми кольорів «Світла Breeze» і «Темна Breeze» із підсвічуванням синтаксису «C++».

6.3.2 Теми кольорів KSyntaxHighlighting

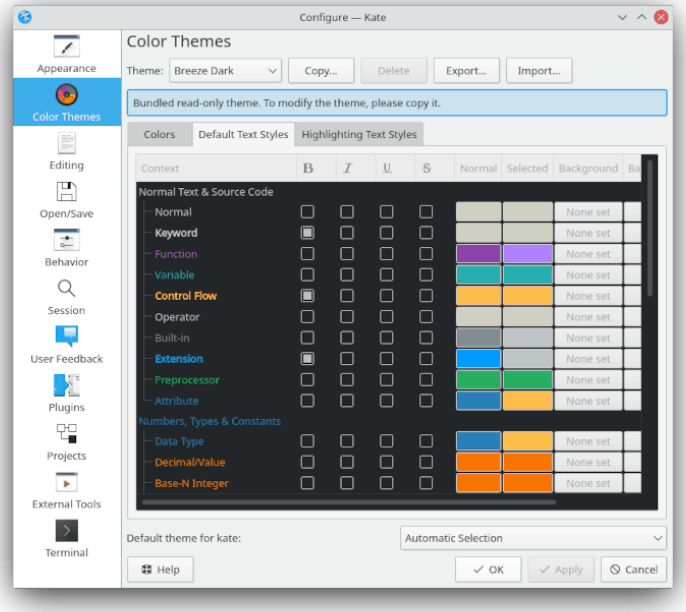
Бібліотека KSyntaxHighlighting, яка є рушієм підсвічування синтаксису — це бібліотека, яка `надає доступ до тем кольорів і керує ними`. Вона є частиною KDE Frameworks, її використовують текстові редактори KDE, зокрема Kate, KWrite, Kile та KDevelop. Ця залежність виглядає так:



Залежність від бібліотек KDE Frameworks у текстових редакторах.

До KSyntaxHighlighting включено діапазон вбудованих тем, список яких показано на сторінці Теми кольорів сайта редактора Kate.

Бібліотека KTextEditor, яка є рушієм текстового редактора, надає у розпорядження користувача інтерфейс для створення та редагування схем кольорів, включно із інструментом для імпортування та експортування тем. Ця бібліотека є найпростішим засобом для створення і редагування тем. Отримати доступ до бібліотеки можна за допомогою діалогового вікна «Налаштувати» текстового редактора. Докладніше про це у розділі Розділ 6.3.5.



Графічний інтерфейс для керування темами кольорів у параметрах Kate.

Варто згадати про те, що у текстових редакторах KDE, зокрема Kate або KDevelop, теми кольорів KSyntaxHighlighting використовують з KDE Frameworks 5.75, випущених 10 жовтня 2020 року. До цієї версії використовували схеми кольорів Kate (налаштування схеми на основі KConfig), які тепер вважаються застарілими. Втім старі схеми Kate можна перетворити на схеми кольорів KSyntaxHighlighting. У сховищі коду KSyntaxHighlighting для виконання цього завдання є скрипт `utils/kateschema_to_theme_converter.py` і допоміжна програма `utils/schema-converter/` .

6.3.3 Формат JSON схем кольорів

6.3.3.1 Огляд

Теми кольорів зберігаються у файлах у форматі JSON із суфіксом назви `.theme`.

У сховищі початкового коду KSyntaxHighlighting файли JSON вбудованих тем зберігаються у каталозі `data/themes/` . Зауважте, що у текстових редакторах вбудовані теми скомпільовано до бібліотеки KSyntaxHighlighting. Тому для доступу до них потрібен початковий код або експортування з графічного інтерфейсу для керування темами у KTextEditor.

Також можна без проблем додавати нетипові теми, які завантажуватимуться з файлової системи комп'ютера. Налаштовані користувачами теми зберігаються у каталозі `org.kde.syntax-highlighting/themes/` теки користувача. Визначити розташування теки можна за допомогою команди `qtpaths --paths GenericDataLocation`. Типовими теками є `$HOME /.local/share/` і `/usr/share/` .

Для пакунків Flatpak і Snap місце зберігання даних є різним для різних програм. У пакунках Flatpak нетипові файли тем зберігаються, зазвичай, у `$HOME /.var/app/ назва-пакунка-flatpak /data/org.kde.syntax-highlighting/themes/` , у пакунках Snap — у `$HOME /snap/ назва-пакунка-snap /current/.local/share/org.kde.syntax-highlighting/themes/` .

У Windows® ці файли зберігаються у `%USERPROFILE%\AppData\Local\org.kde.syntax-highlighting\themes`. *`%USERPROFILE%`*, зазвичай, є каталогом `C:\Users\користувач`.

Загалом, для більшості конфігурацій каталогом нетипових тем є такий каталог:

Для окремого користувача:	`$HOME /.local/share/org.kde.syntax-highlighting/themes/`
Для усіх користувачів?	`/usr/share/org.kde.syntax-highlighting/themes/`
Для пакунків Flatpak:	`$HOME /.var/app/назва-пакунка-flatpak /data/org.kde.syntax-highlighting/themes/`
Для пакунків Snap:	`$HOME /snap/ назва-пакунка-snap /current/.local/share/org.kde.syntax-highlighting/themes/`
У Windows®:	`%USERPROFILE%\AppData\Local\org.kde.syntax-highlighting\themes`
У macOS®	`$HOME /Library/Application Support/org.kde.syntax-highlighting/themes/`

Якщо існує декілька файлів теми із однаковими назвами, буде завантажено файл із найбільшим значенням `revision`.

6.3.3.2 Структура JSON

Структуру файла JSON описано на відповідному сайті. Загалом, файл формату JSON складається з таких елементів:

- Збірки пар ключ-значення, відокремлених комами і згрупованих фігурними дужками `{ }`. Ці збірки ми називатимемо «об'єктами».
- Упорядкованих списків значень, відокремлених комами і згрупованих квадратними дужками `[ ]`, які ми називатимемо «масивами».

У цьому підручнику ми використовуватимемо термінологію «ключ», «значення», «об'єкт» і «масив». Якщо ви вперше працюєте із файлами JSON, зрозуміти цю термінологію можна з наведених нижче прикладів.

6.3.3.3 Основні розділи файлів JSON теми кольорів

Кореневий об'єкт файла JSON теми кольорів містить такі ключі схеми:

- `metadata` — обов'язковий ключ. Значенням є об'єкт із метаданими теми, зокрема назвою, модифікацією та умовами ліцензування.

  Докладний опис наведено у розділі Розділ 6.3.3.4.

- `editor-colors` — обов'язковий ключ. Значенням є об'єкт із кольорами області редагування тексту, зокрема кольором тла, рамки піктограм та декорування тексту.

  Докладний опис наведено у розділі Розділ 6.3.4.1.

- `text-styles` — обов'язковий ключ. Значенням є об'єкт із атрибутами *типового стилю тексту* для підсвічування синтаксичних конструкцій. Кожен атрибути визначає власний *колір тексту*, *колір позначеного тексту*, а також, наприклад, те, чи буде шрифт *напівжирним* або *курсивним*. На стилі тексту можна посилатися з атрибутів файлів XML визначення синтаксису.

  Докладний опис наведено у розділі Розділ 6.3.4.2.

- `custom-styles` — ключ не є обов'язковим. Визначає стилі тексту для атрибутів специфічних визначень підсвічування. Наприклад, у визначення підсвічування, зокрема для `Python` або `Markdown`, ви можете вказати інший стиль тексту, який матиме пріоритет над типовим, який визначено у `text-styles`.

  Докладний опис наведено у розділі Розділ 6.3.4.3.

У мові JSON не передбачено підтримки коментарів. Втім, ви можете скористатися необов'язковим ключем `_comments` у кореневому об'єкті для запису коментарів. Наприклад, якщо ви адаптуєте наявну тему, ви можете додати адресу сховища початкового матеріалу. Найпрактичнішим способом коментування є використання масиву рядків.

Нижче наведено приклад файла для теми «Світла Breeze». Ви можете зауважити, що, з метою зменшення розміру прикладу, об'єкти `editor-colors` і `text-styles` не містять усіх обов'язкових ключів. Переглянути увесь код теми Світла Breeze можна у сховищі KSyntaxHighlighting.

```
{
    "_comments": [
        "This is a comment.",
        "If this theme is an adaptation of another, put the link to the original repository."
    ],
    "metadata": {
        "name" : "Breeze Light",
        "revision" : 5,
        "copyright": [
            "SPDX-FileCopyrightText: 2016 Volker Krause <vkrause@kde.org>",
            "SPDX-FileCopyrightText: 2016 Dominik Haumann <dhaumann@kde.org>"
        ],
        "license": "SPDX-License-Identifier: MIT"
    },
    "editor-colors": {
        "BackgroundColor" : "#ffffff",
        "CodeFolding" : "#94caef",
        "BracketMatching" : "#ffff00",
        "CurrentLine" : "#f8f7f6",
        "IconBorder" : "#f0f0f0",
        "IndentationLine" : "#d2d2d2",
        "LineNumbers" : "#a0a0a0",
        "CurrentLineNumber" : "#1e1e1e",
        The other editor color keys...

    },
    "text-styles": {
        "Normal" : {
            "text-color" : "#1f1c1b",
            "selected-text-color" : "#ffffff",
            "bold" : false,
            "italic" : false,
            "underline" : false,
            "strike-through" : false
        },
        "Keyword" : {
            "text-color" : "#1f1c1b",
            "selected-text-color" : "#ffffff",
            "bold" : true
        },
        "Function" : {
            "text-color" : "#644a9b",
            "selected-text-color" : "#452886"
        },
        "Variable" : {
            "text-color" : "#0057ae",
            "selected-text-color" : "#00316e"
```

```
        },
        Інші ключі стилю тексту...

    },
    "custom-styles": {
        "ISO C++": {
            "Data Type": {
                "bold": true,
                "selected-text-color": "#009183",
                "text-color": "#00b5cf"
            },
            "Keyword": {
                "text-color": "#6431b3"
            }
        },
        "YAML": {
            "Attribute": {
                "selected-text-color": "#00b5cf",
                "text-color": "#00b5cf"
            }
        }
    }
}
```

6.3.3.4 Метадані

Об'єкт JSON ключа `metadata` містить відповідні відомості щодо теми. Цей об'єкт має такі ключі:

- `name` — *рядкове* значення, яке визначає назву мови. Цю назву буде згодом показано у пунктах меню та діалогових вікнах програми. Є обов'язковим.
- `revision` — *ціле* число, яке визначає поточну модифікацію файла теми. Коли ви оновлюєте файл теми, вам слід збільшувати це число. Ключ є обов'язковим.
- `license` — *рядок*, який визначає умови ліцензування теми з використанням ідентифікатора `SPDX-License-Identifier` зі стандартного формату обміну умовами ліцензування SPDX. Ключ є необов'язковим.
  Із повним списком ідентифікаторів умов ліцензування SPDX можна ознайомитися тут.
- `copyright` — *масив рядків*, який визначає авторів теми з використанням ідентифікатора `SPDX-FileCopyrightText` зі стандартного формату обміну даними щодо умов ліцензування SPDX. Ключ є необов'язковим.

```
"metadata": {
    "name" : "Breeze Light",
    "revision" : 5,
    "copyright": [
        "SPDX-FileCopyrightText: 2016 Volker Krause <vkrause@kde.org>",
        "SPDX-FileCopyrightText: 2016 Dominik Haumann <dhaumann@kde.org ↩
            >"
    ],
    "license": "SPDX-License-Identifier: MIT"
}
```

6.3.4 Докладно про кольори

У цьому розділі наведено докладні відомості щодо усіх доступних атрибутів кольорів і доступних параметрів кольорів.

6.3.4.1 Кольори редактора

Відповідає кольорам області редагування тексту.

У файлі JSON теми відповідний ключ, `editor-colors`, має значення *об'єкта*, кожен ключ якого вказує на колір атрибута у текстовому редакторі. Тут `усі доступні ключі є обов'язковими`. Їхніми значеннями є `рядки` шістнадцяткових кодів кольорів, наприклад «#00B5CF».

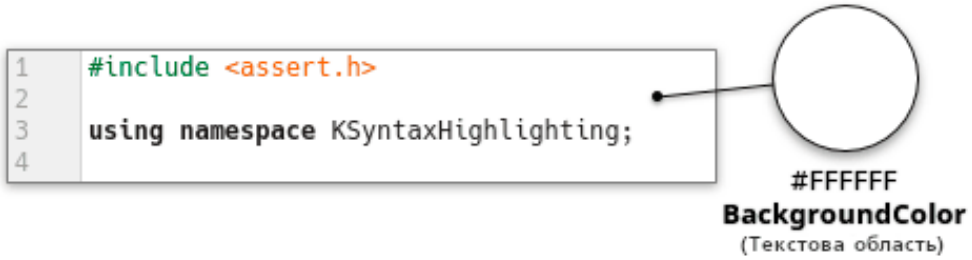
У графічному інтерфейсі керування темами KTextEditor ці атрибути можна змінювати на вкладці `Кольори`.

Нижче наведено доступні ключі. Список ключів, які використано у файлі JSON, наведено *напівжирним* шрифтом. У дужках подано назви, які використано у графічному інтерфейсі.

Кольори тла редактора

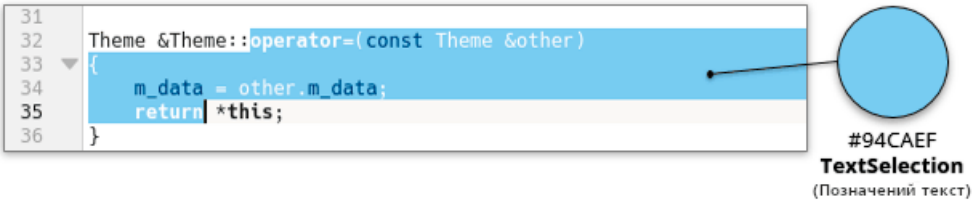
`BackgroundColor` **(Область тексту)**

Це типовий колір тла для області редагування, він буде домінуючим кольором у області редагування.



`TextSelection` **(Позначений текст)**

Це тло позначеного тексту.

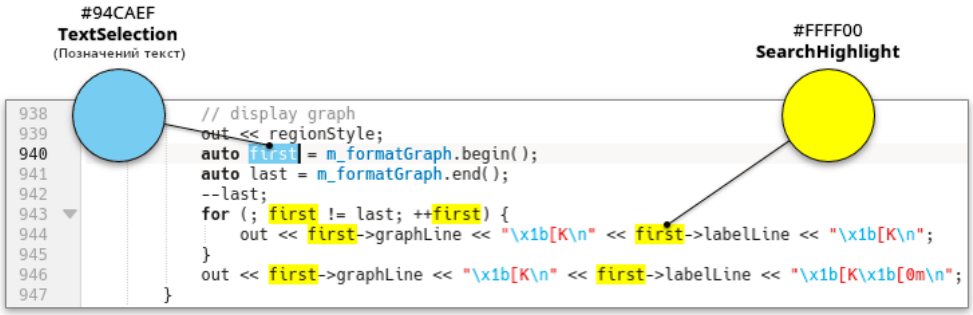


`CurrentLine` **(Поточний рядок)**

Встановлює колір для поточного рядка. Якщо встановити колір, який трошки відрізнятиметься від звичайного кольору тла тексту, вам буде легше фокусуватися на поточному рядку.

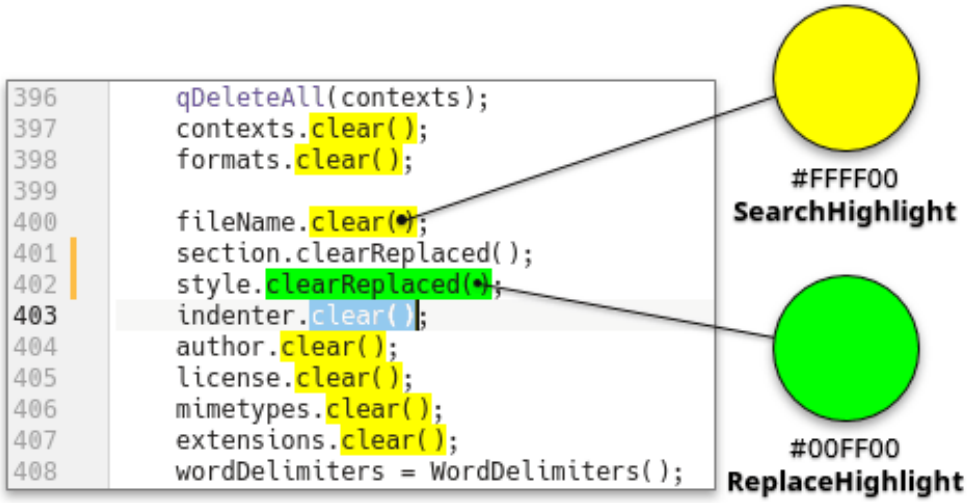
`SearchHighlight` **(Підсвічування результатів пошуку)**

Встановлює колір позначення фрагментів тексту, які відповідають критеріям останнього сеансу пошуку.



`ReplaceHighlight` **(Підсвічування замін)**

Встановлює колір позначення фрагментів тексту, які відповідають критеріям останнього сеансу заміни.



Смужка піктограм

`IconBorder` **(Область тла)**

Цей колір буде використано для позначок, номерів рядків та рамок поміток згортання у лівій частині області перегляду редактора, якщо їх буде показано.

`LineNumbers` **(Номери рядків)**

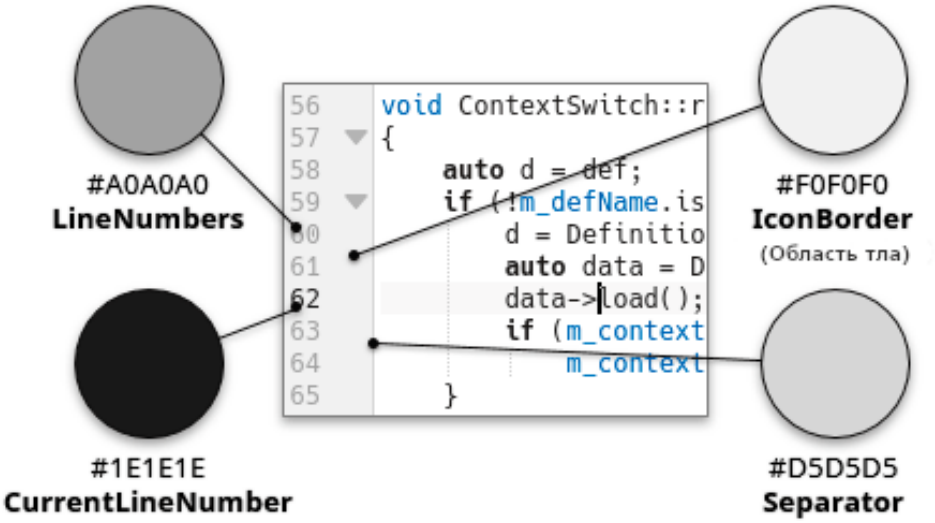
Цей колір буде використано для показу номерів рядків у лівій частині області перегляду, якщо такі номери буде показано.

`CurrentLineNumber` **(Номер поточного рядка)**

Цей колір буде використано для показу номера поточного рядка, якщо його показано у лівій частині області перегляду. Встановлення трохи іншого значення, порівняно із «LineNumbers», допоможе акцентувати поточний рядок.

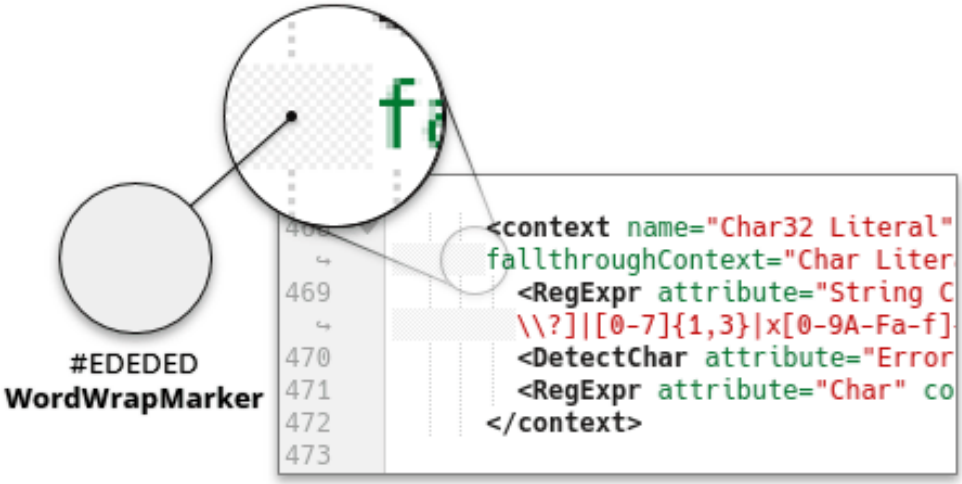
`Separator` **(Роздільник)**

Цей колір буде використано для малювання вертикальної лінії, яка відокремлюватиме рамку піктограм від тла області тексту.



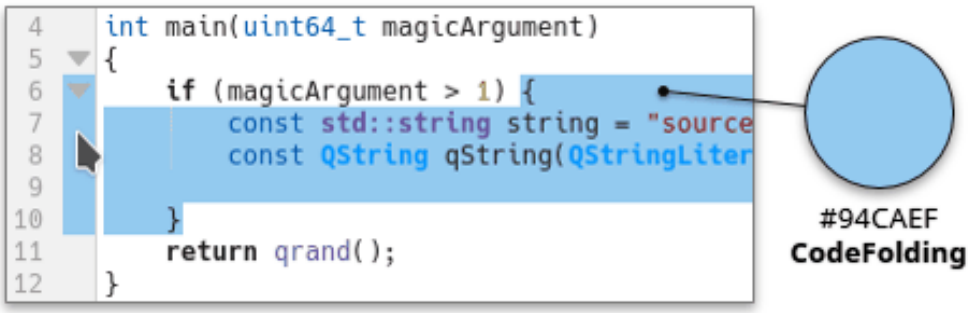
`WordWrapMarker` **(Позначки перенесення слів)**

Цей колір буде використано для візерунка у лівій частині динамічно перенесених рядків, якщо такі рядки вирівняно вертикально, а також для позначення статичного перенесення слів.



`CodeFolding` **(Згортання коду)**

Цей колір використовується для підсвічування розділу коду, який буде згорнуто, якщо ви натиснете стрілочку згортання коду у лівій частині документа Щоб дізнатися більше, зверніться до розділу, присвяченого згортанню коду.

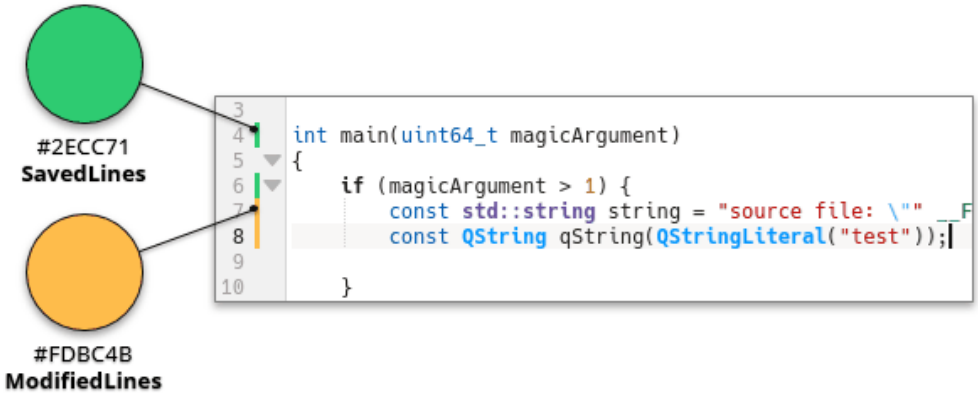


`ModifiedLines` **(Змінені рядки)**

Встановлює колір, який буде використано для позначення на лівій панелі документа рядків, у які було внесено ще не збережені зміни. Щоб дізнатися більше, зверніться до розділу Розділ 3.9.

`SavedLines` **(Збережені рядки)**

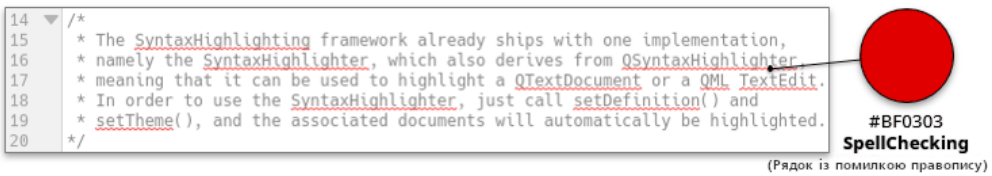
Встановлює колір, який буде використано для позначення на лівій панелі документа рядків, у які було внесено вже збережені зміни. Щоб дізнатися більше, зверніться до розділу Розділ 3.9.



Декорування тексту

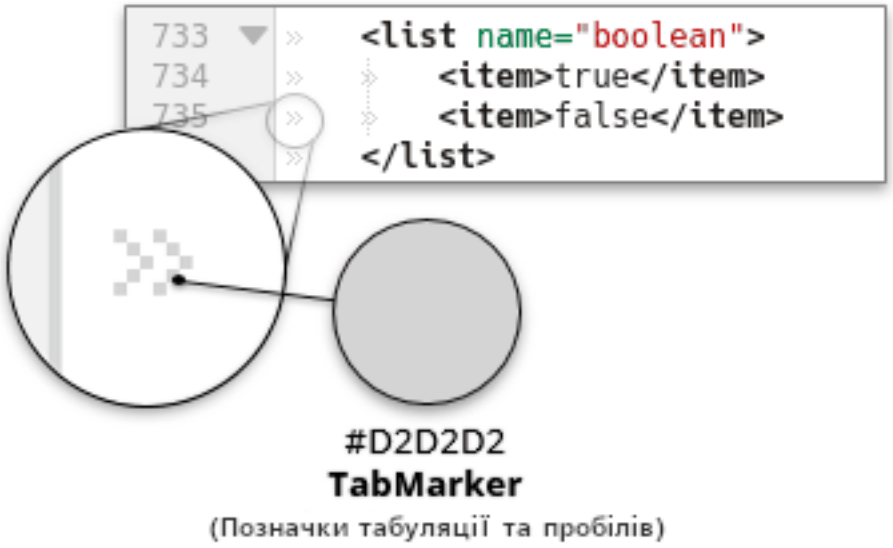
`SpellChecking` **(Рядок з помилкою правопису)**

Встановлює колір позначення помилок правопису.



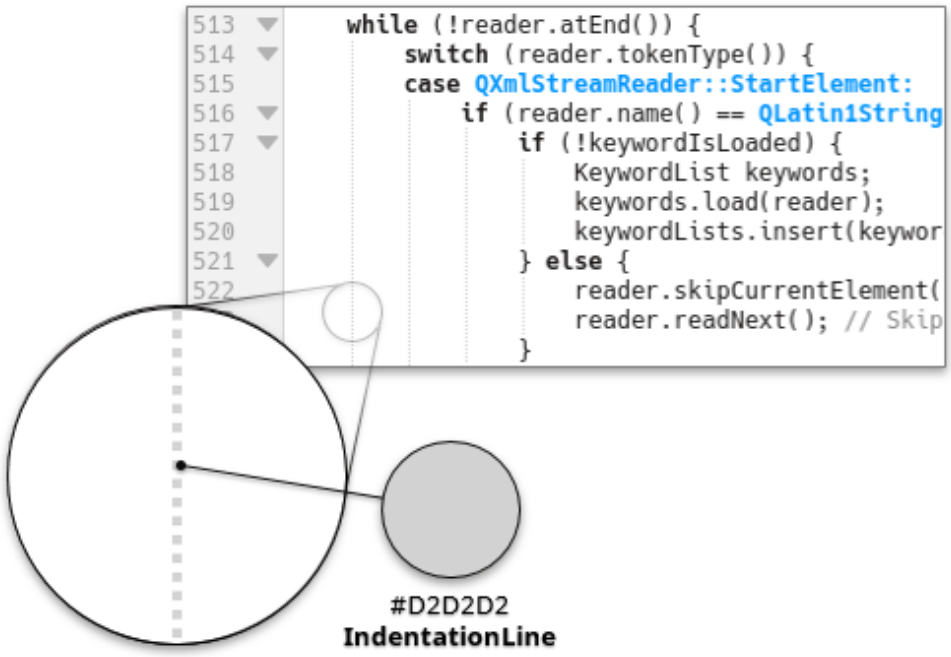
`TabMarker` **(Позначки табуляції та пробілів)**

Цей колір буде використано для показу позначок пробілів, якщо такі позначки увімкнено.



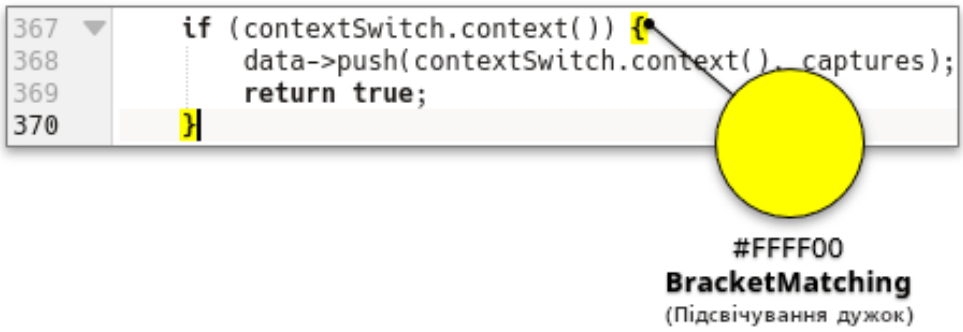
`IndentationLine` **(Лінія відступу)**

Цей колір використовується для малювання лінії ліворуч від блоків з відступом, якщо увімкнено відповідну можливість.



`BracketMatching` **(Підсвічування дужок)**

Цей колір буде використано для показу тла відповідних одна одній дужок.



Кольори позначок

`MarkBookmark` **(Закладка)**

Встановлює колір, який буде використано для позначення закладок. Зауважте, що цей колір має рівень непрозорості щодо тла у 22% (і у 33% для поточного рядка). Щоб дізнатися більше, зверніться до розділу Розділ 3.6.

`MarkBreakpointActive` **(Активна точка зупинки)**

Цей колір використовується додатком GDB для позначення активної точки зупинки. Зауважте, що цей колір має рівень непрозорості щодо тла. Щоб дізнатися більше, зверніться до документації з додатка GDB.

`MarkBreakpointReached` **(Досягнута точка зупинки)**

Цей колір використовується додатком GDB для позначення точки зупинки, якої було досягнуто під час діагностики. Зауважте, що цей колір має рівень непрозорості щодо тла. Щоб дізнатися більше, зверніться до документації з додатка GDB.

`MarkBreakpointDisabled` **(Вимкнена точка зупинки)**

Цей колір використовується додатком GDB для позначення неактивної точки зупинки. Зауважте, що цей колір має рівень непрозорості щодо тла. Щоб дізнатися більше, зверніться до документації з додатка GDB.

`MarkExecution` **(Виконання)**

Цей колір використовується додатком GDB для позначення рядка коду, який виконується. Зауважте, що цей колір має рівень непрозорості щодо тла. Щоб дізнатися більше, зверніться до документації з додатка GDB.

`MarkWarning` **(Попередження)**

Цей колір використовується додатком збирання для позначення рядка, щодо якого засобом збирання видано попередження. Зауважте, що цей колір має рівень непрозорості щодо тла. Щоб дізнатися більше, зверніться до документації з додатка збирання.

`MarkError` **(Помилка)**

Цей колір використовується додатком збирання для позначення рядка, щодо якого засобом збирання видано повідомлення про помилку. Зауважте, що цей колір має рівень непрозорості щодо тла. Щоб дізнатися більше, зверніться до документації з додатка збирання.

Шаблони і фрагменти тексту

`TemplateBackground` **(Тло)**

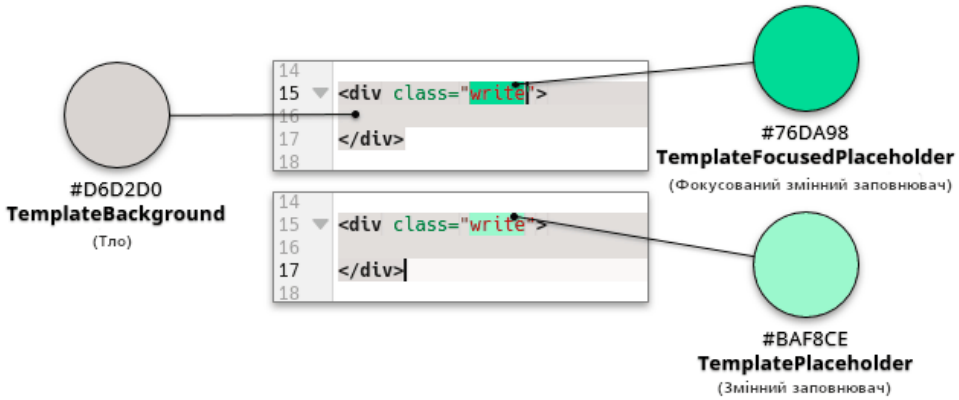
Цей колір використовується додатком фрагментів Kate для позначення тла фрагмента. Щоб дізнатися більше, ознайомтеся із документацією щодо фрагментів Kate.

`TemplatePlaceholder` **(Змінний заповнювач)**

Цей колір використовується додатком фрагментів Kate для позначення замінника, на якому ви можете клацнути з метою внесення змін вручну. Щоб дізнатися більше, зверніться до документації з додатка фрагментів Kate.

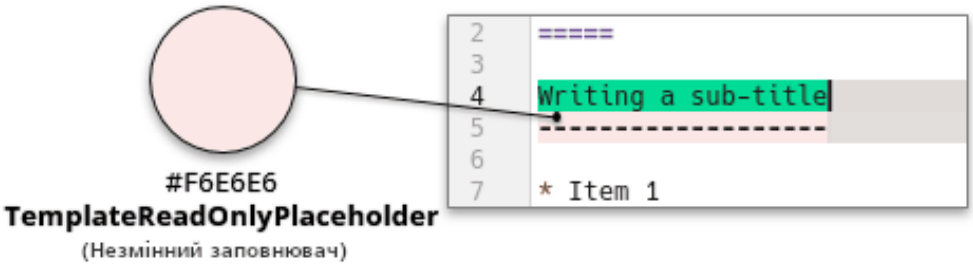
`TemplateFocusedPlaceholder` **(Фокусований змінний заповнювач)**

Цей колір використовується додатком фрагментів Kate для позначення замінника, який ви редагуєте. Щоб дізнатися більше, зверніться до документації з додатка фрагментів Kate.



`TemplateReadOnlyPlaceholder` **(Незмінний заповнювач)**

Цей колір використовується додатком фрагментів Kate для позначення замінника, який не можна змінити вручну, наприклад замінника, який заповнюється автоматично. Щоб дізнатися більше, зверніться до документації з додатка фрагментів Kate.



6.3.4.2 Стилі звичайного тексту

Стилі звичайного тексту успадковують свої властивості від стилів підсвіченого тексту, що надає редактору можливість показувати текст без великої різниці у шрифтах, наприклад, для тексту коментарів використовується той же стиль у майже всіх форматах тексту, які може підсвічувати KSyntaxHighlighting.

> **Примітка**
> На ці стилі тексту можна посилатися з `типових стилів`, які використовують у файлах XML визначення підсвічування синтаксису. Наприклад, атрибут «Normal» є еквівалентним до атрибута «dsNormal» у файлах XML, а «DataType» є еквівалентним до «dsDataType». Див. розділ Розділ 6.2.3.3 у документації щодо підсвічування синтаксису.

> **Підказка**
> Слід вибирати придатні до читання кольори із доброю контрастністю, особливо у поєднанні із `кольорами редактора`. Див. Розділ 6.3.6.1.

У файлі JSON відповідний ключ `text-styles` має значення *об'єкта*, кожен ключ якого відповідає назві *типового стилю тексту*. Ключі є еквівалентами ключів, які використовують у визначеннях підсвічування синтаксису. Тут `усі доступні ключі стилю тексту є обов'язковими`. Список ключів наведено нижче.

```
"text-styles": {
    "Normal" : {
        "text-color" : "#1f1c1b",
        "selected-text-color" : "#ffffff",
        "bold" : false,
        "italic" : false,
        "underline" : false,
        "strike-through" : false
    },
    "Keyword" : {
        "text-color" : "#1f1c1b",
        "selected-text-color" : "#ffffff",
        "bold" : true
    },
    "Function" : {
        "text-color" : "#644a9b",
        "selected-text-color" : "#452886"
    },
    Інші ключі стилю тексту...
}
```

Значенням кожного ключа *типового стилю тексту* є об'єкт JSON, у якому задано такі значення, як *color*, *bold*, *italic*. Ось ці ключі:

`text-color` — *рядкове* значення із шістнадцятковим кодом кольору. Ця пара ключ-значення є обов'язковою.

`selected-text-color` — колір тексту, коли його позначено. Загалом, те саме значення, що і «text-color». Якщо текст позначено, тло визначається значенням TextSelection у кольорах редактора, тому вам слід забезпечити добру контрастність та можливість читання на тлі. Значенням є *рядок* із шістнадцятковим кодом кольору. Ця пара ключ-значення є обов'язковою.

`bold` — *булеве* значення, яке визначає, чи є шрифт тексту напівжирним. Цей ключ є необов'язковим. Типовим значенням є `false` (ні).

`italic` — *булеве* значення, яке визначає, чи є шрифт тексту курсивним. Цей ключ є необов'язковим. Типовим значенням є `false` (ні).

`underline` — *булеве* значення, яке визначає, чи є текст підкресленим. Цей ключ є необов'язковим. Типовим значенням є `false` (ні).

`strike-through` — *булеве* значення, яке визначає, чи є текст перекресленим. Цей ключ є необов'язковим. Типовим значенням є `false` (ні).

`background-color` — визначає тло тексту. Використовують, наприклад, у коментарях. Значенням є *рядок* із шістнадцятковим кодом кольору. Цей ключ є необов'язковим. Типовим є текст без будь-якого тла.

`selected-background-color` — визначає тло позначеного тексту. Значенням є *рядок* із шістнадцятковим кодом кольору. Цей ключ є необов'язковим. Типовим є текст без будь-якого тла.

У графічному інтерфейсі для керування темами кольорів KTextEditor ці атрибути можна змінити на вкладці `Стилі звичайного тексту`. Для назви у списку стилів буде використано стиль для відповідного запису. Це надасть вам змогу одразу бачити результат застосування стилю. У кожному стилі ви зможете вибрати загальні атрибути, а також кольори тексту і тла. Щоб скинути колір тла, клацніть правою кнопкою миші і скористайтеся контекстним меню.

Нижче наведено доступні ключі стилю тексту. Список ключів, які використано у файлі JSON, наведено *напівжирним* шрифтом. У дужках подано назви, які використано у графічному інтерфейсі, якщо вони є різними.

Звичайний текст і початковий код

`Normal` — типовий стиль звичайного тексту і початкового коду без спеціального підсвічування.

`Keyword` — стиль тексту для вбудованих ключових слів мови.

`Function` — стиль тексту для визначень і викликів функцій.

`Variable` — стиль тексту для змінних, якщо такий застосовний. Наприклад, назви змінних у PHP/Perl типово починаються з символу `$`, тому усі ідентифікатори, які відповідають шаблону `$foo`, буде підсвічено як змінні.

`ControlFlow` (**Потік керування**) — стиль тексту для ключових слів потоку керування, зокрема *if*, *then*, *else*, *return*, *switch*, *break*, *yield*, *continue* тощо.

`Operator` — стиль тексту для операторів, зокрема `+`, `-`, `*`, `/` , `%`, тощо.

`BuiltIn` (**Вбудований блок**) — стиль тексту для вбудованих класів, функцій та об'єктів мови.

`Extension` (**Розширення**) — стиль тесту для відомих розширень, зокрема класів Qt™, функцій/макросів у C++ та Python або boost.

`Preprocessor` (**Попередня обробка**) — стиль тексту для інструкцій препроцесора та визначень макросів.

`Attribute` (**Атрибут**) — стиль тексту для анотацій або атрибутів функцій чи об'єктів, наприклад `@override` у Java або `__declspec(...)` і `__attribute__((...))` у C++.

Числа, типи і сталі

`DataType` (**Тип даних**) — стиль тексту для вбудованих типів даних, зокрема *int*, *char*, *float*, *void*, *u64* тощо.

`DecVal` (**Десяткове/Значення**) — стиль тексту для десяткових значень.

`BaseN` (**Ціле у системі числення з основою N**) — стиль тексту для чисел в записі із основою числення, відмінною від 10.

`Float` (**З рухомою крапкою**) — стиль тексту для чисел із рухомою крапкою.

`Constant` (**Стала**) — стиль тексту для сталих мови програмування або визначених користувачем сталих, наприклад *True*, *False*, *None* у Python або *nullptr* у C/C++; або математичних стали, зокрема *PI*.

Рядки і символи

`Char` (**Символ**) — стиль тексту для окремих символів, зокрема `'x'`.

`SpecialChar` (**Спеціальний символ**) — стиль тексту для екранованих символів у рядках, наприклад «`hello\n`», та інших символів зі спеціальним значенням у рядках, зокрема символів-замінників та операторів формальних виразів.

`String` — стиль тексту для рядків, подібних до «`привіт, світе`».

`VerbatimString` (**Рядок буквально**) — стиль тексту для буквальних або необроблених рядків, зокрема `'raw \backlash'` у Perl, CoffeeScript та командних оболонках, а також `r'\raw'` у Python або рядків, подібних до документації HERE.

`SpecialString` (**Спеціальний рядок**) — стиль тексту для спеціальних рядків, зокрема формальних виразів у ECMAScript, коду у рівняннях LaTeX, коду SQL тощо.

`Import` (**Імпортування, модулі, включення**) — стиль тексту для включень, імпортувань, модулів та пакунків LaTeX.

Коментарі і документація

`Comment` — стиль тексту для звичайних коментарів.

`Documentation` (**Документація**) — стиль тексту для коментарів, які відповідають за документацію з програмного інтерфейсу, зокрема `/** коментарів doxygen */` або `˝˝˝рядків_документації˝˝˝`.

`Annotation` (**Анотація**) — стиль тексту для анотацій у коментарях або командах документування, зокрема `@param` у Doxygen або JavaDoc.

`CommentVar` (**Змінна коментаря**) — стиль тексту, який відповідає назвам змінних у наведених вище командах у коментарі, зокрема `foobar` у «`@param foobar`» для коду Doxygen або JavaDoc.

`RegionMarker` (**Позначка ділянки**) — стиль тексту для позначок ділянки, яка типово визначається командами `/ /BEGIN` і `/ /END` у коментарях.

`Information` (**Інформація**) — стиль тексту для відомостей, нотаток і підказок, зокрема ключового слова `@note` у Doxygen.

`Warning` — стиль тексту для попереджень, зокрема ключового слова `@warning` у Doxygen.

`Alert` — стиль тексту для спеціальних слів у коментарях, зокрема `TODO`, `FIXME`, `XXXX` та `WARNING`.

Різне

`Error` — стиль тексту для позначення підсвічування помилок та помилкових синтаксичних конструкцій.

`Others` — стиль тексту для атрибутів, які не можна віднести до жодного із інших типових стилів.

6.3.4.3 Стилі тексту нетипового підсвічування

Тут ви можете встановити стилі тексту для специфічного визначення синтаксису, перевизначивши `стиль звичайного тексту`, який описано у попередньому розділі.

У файл JSON теми цей ключ відповідає ключу `custom-styles`, чиїм значенням є *об'єкт*, кожен ключ підлеглої схеми якого відповідає `назві визначення підсвічування синтаксису`. Його значенням *об'єкт*, кожен ключ якого відповідає `назві атрибутів стилю`, яку визначено елементах `itemData` файла XML підсвічування синтаксису, і відповідне значення є підлеглим об'єктом із ключами *text-color*, *selected-text-color*, *bold*, *italic*, *underline*, *strike-through*, *background-color* і *selected-background-color*, який визначено у попередньому розділі. Усі ці значення є необов'язковими, оскільки, якщо їх не вказано, буде використано стиль, який визначено у `text-styles`.

Наприклад, у цьому фрагменті коду визначення підсвічування синтаксису «ISO C++» містить спеціальний стиль тексту для атрибутів «Модифікатори типу» і «Стандартні класи». У відповідному файлі XML, «isocpp.xml», визначений атрибут «Стандартні класи» використовує типовий стиль `BuiltIn` (або dsBuiltIn). У цьому атрибуті буде перезаписано лише одне значення — `text-color` — новим кольором «#6431b3».

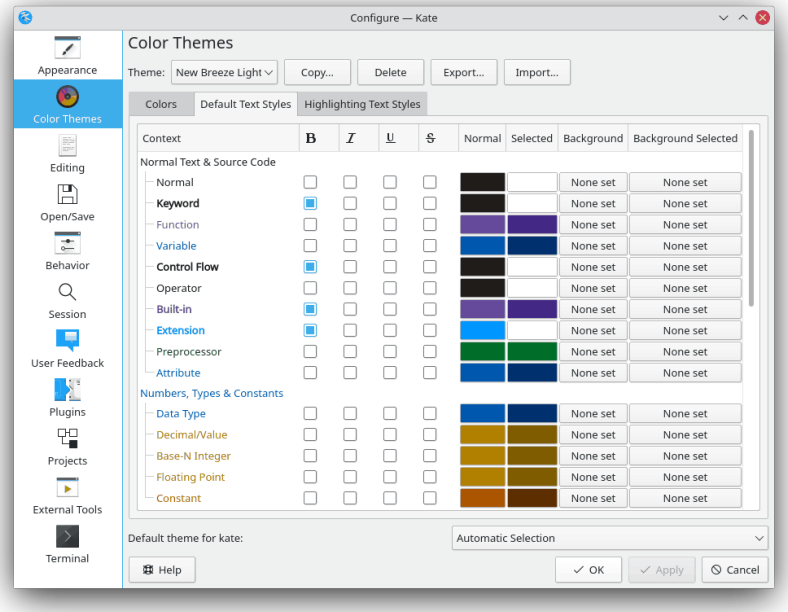
```
"custom-styles": {
    "ISO C++": {
        "Standard Classes": {
            "text-color": "#6431b3"
        },
        "Type Modifiers": {
            "bold": true,
            "selected-text-color": "#009183",
            "text-color": "#00b5cf"
        }
    }
}
```

> **Примітка**
>
> - Вам слід зробити так, щоб ці стилі тексту було пов'язано із назвами атрибутів у файлах XML підсвічування синтаксису. Якщо буде оновлено файл XML, а деякі атрибути буде перейменовано або вилучено, нетиповий стиль, який визначено у темі, стане незастосовним.
> - Визначення підсвічування синтаксичних конструкцій часто включають інші визначення. Наприклад, визначення підсвічування «QML» включає визначення підсвічування «JavaScript», оскільки вони містять спільні функціональні можливості підсвічування.

У графічному інтерфейсі для керування темами KTextEditor ці атрибути можна змінити на вкладці `Стилі підсвіченого тексту`. Типово, редактор попередньо вибирає підсвічування для поточного документа. Ви зауважите, що багато визначень підсвічувань місять інші визначення підсвічувань, що позначається групами у списку стилів. Наприклад, у більшості визначень підсвічувань імпортують підсвічування «Alert», а багато визначень підсвічувань програмного коду імпортують підсвічування «Doxygen».

6.3.5 Графічний інтерфейс тем кольорів

Найпростішим способом створення і редагування тем кольорів є використання графічного інтерфейсу — діалогового вікна «Налаштувати», яке надається KTextEditor. Щоб отримати до нього доступ, скористайтеся пунктом меню **Параметри** → **Налаштувати** *назва_програми*... у вікні вашого текстового редактора. У відповідь буде відкрито діалогове вікно **Налаштувати**. У цьому діалоговому вікні вам слід вибрати сторінку **Теми кольорів** на бічній панелі.



Діалогове вікно параметрів Kate для керування темою кольорів.

За допомогою цього діалогового вікна ви можете налаштувати усі кольори будь-якої теми, а також створити або скопіювати тему, вилучити тему, експортувати тему до файла `.theme` у форматі JSON або імпортувати тему із зовнішніх файлів `.theme`. У кожній темі передбачено параметри для кольорів і стилів тексту.

Типово, вносити зміни до вбудованих тем не можна. Для внесення змін вам слід скопіювати типову тему і змінити її назву.

Щоб встановити остаточну тему у вашому текстовому редакторі, вам слід вибрати тему за допомогою розташованого у нижній частині вікна спадного списку із міткою **Типова тема для** *назва_програми* і натиснути кнопку **Застосувати** або **Гаразд**. Типово, активним є варіант `Автоматичний вибір`. Використання цього варіанта призводить до вибору теми кольорів, яка найкраще пасує до *схеми кольорів Плазми KDE* для редагування тексту. Зазвичай, вибір здійснюється між «Світлою Breeze» і «Темною Breeze». Вибір залежить від того, світлою чи темною є схема кольорів середовища.

> Підказка
> Ви можете змінити загальну схему кольорів KDE за допомогою модуля Кольори програми «Системні параметри». Ви також можете змінити її у деяких програмах, зокрема Kate та KDevelop, окремо. Для цього скористатися пунктом меню **Параметри** → **Схема кольорів**.

6.3.5.1 Створити тему

Щоб створити тему, спочатку слід скопіювати наявну тему. Виберіть наявну тему, якою ви хочете скористатися як основою, наприклад «Світу Breeze» або «Темну Breeze», і натисніть кнопку **Копіювати**. Потім впишіть у показане поле назви назву нової теми.

Якщо ви хочете внести зміни до вбудованої теми або теми, призначеної лише для читання, вам слід спочатку скопіювати тему, вказавши для копії іншу назву.

6.3.5.2 Імпортування або експортування файлів JSON тем

Ви можете експортувати вибрану тему (включно із вбудованими) до файла JSON із суфіксом назви `.theme` за допомогою кнопки **Експортувати**. У відповідь на натискання кнопки програма відкриє діалогове вікно для збереження файла. Щоб додати тему кольорів із зовнішнього файла JSON, просто натисніть кнопку **Імпортувати** і виберіть за допомогою діалогового вікна навігації файловою системою файл `.theme`.

> Підказка
>
> - Як ми вже згадували вище, налаштовані користувачами теми зберігаються у каталозі `org.kde.syntax-highlighting/themes/` . Коли ви копіюєте або створюєте тему, вона автоматично з'являється у цьому каталозі. Крім того, імпортування або додавання теми є еквівалентним до копіювання зовнішнього файла `.theme` до цього каталогу. KSyntaxHighlighting автоматично вибирає файли тем кольорів з вказаного каталогу.
> - Якщо ви хочете оприлюднити створену вами тему, важливо перевірити коректність об'єкта metadata у файлі JSON, додавши до нього відповідні умови ліцензування та вказавши відповідний номер модифікації.

6.3.5.3 Редагування тем кольорів

6.3.5.3.1 Кольори

Тут можна коригувати кольори області редагування тексту. Докладніший опис параметрів наведено у розділі Розділ 6.3.4.1.

6.3.5.3.2 Стилі звичайного тексту

Стилі звичайного тексту успадковують свої властивості від стилів підсвіченого тексту, що надає редактору можливість показувати текст без великої різниці у шрифтах, наприклад, для тексту коментарів використовується той же стиль у майже всіх форматах тексту, які може підсвічувати KSyntaxHighlighting.

Під час підготовки до показу назви стилю у списку стилів буде використано стиль, назву якого ви бачитимете, отже під час налаштування стилю вам легше буде знайти потрібний пункт у списку.

Для кожного стилю ви можете обрати загальні атрибути, а також колір тексту і тла. Щоб повернути колір тла до початкового значення, наведіть на цього вказівник миші і клацніть правою кнопкою миші, щоб викликати відповідне контекстне меню.

Докладний опис атрибутів цієї області наведено у розділі Розділ 6.3.4.2.

6.3.5.3.3 Стилі підсвіченого тексту

Тут ви можете змінити стилі тексту, які використовуватимуться певним типом підсвічування. У редакторі буде попередньо обрано колір, який використовується у вашому поточному документі. Щоб працювати з іншим стилем підсвічування, виберіть якого зі спадного списку **Підсвічування**, розташованому над списком стилів.

Під час підготовки до показу назви стилю у списку стилів буде використано стиль, назву якого ви бачитимете, отже під час налаштування стилю вам легше буде знайти потрібний пункт у списку.

У кожному стилі ви можете обрати загальні атрибути, а також кольори тексту та тла. Щоб повернути колір тла до початкового значення, наведіть на цього вказівник миші і клацніть правою кнопкою миші, щоб викликати відповідне контекстне меню. Крім того, ви зможете порівнювати стиль з типовим стилем, що використовується для елемента, і встановлювати його або не робити цього.

Ви побачите, що багато стилів підсвічування містять інші стилі підсвічування, які у списку стилів зібрано у групи. Наприклад, більшість стилів підсвічування імпортують стиль підсвічування «Увага», а багато з форматів підсвічування коду — підсвічування «Doxygen». Зміна кольорів у таких групах, впливатиме на стилі, лише якщо її було використано у форматі підсвічування, який ви редагуєте.

6.3.6 Підказки та поради

6.3.6.1 Контрастність кольорів тексту

Важливим аспектом роботи із темами кольорів є вибір контрастного кольору тексту, який зробить простішим читання, особливо у поєднанні із кольором тла.

Для пошук контрастних кольорів можна скористатися програмою `Kontrast`. Ця програма повідомить вам, чи є комбінації кольорів тексту і тла придатними до читання і доступними для користувачів. Тому ця програма є чудовим інструментом для створення тем кольорів.

Ви можете встановити `Kontrast` з сайта програм KDE або з пакунка Flatpak на Flathub (лише у GNU/Linux).

Подібні функціональні можливості реалізовано у програмі GNOME `Contrast`. Встановити цю програму можна за допомогою пакунка Flatpak з Flathub (лише у GNU/Linux).

За допомогою скрипту мовою python у сховищі підсвічувань синтаксичних конструкцій ви можете візуалізувати усі кольори у темі, а також переглянути контрастність з різними налаштованими кольорами тла.

6.3.6.2 Пропозиції щодо однорідності підсвічування синтаксису

До KSyntaxHighlighting включено понад 300 визначень підсвічування синтаксичних конструкцій, тому було б дуже добре, аби ви переконалися, що нова тема виглядає добре для усіх визначень підсвічувань синтаксичних конструкцій. У вбудованих темах кольорів використано наведені нижче принципи, якими ми рекомендуємо (не обов'язково) користуватися для досягнення належного показу для усіх визначень підсвічувань синтаксичних конструкцій:

- Користуйтеся напівжирним шрифтом для стилів тексту «Keyword» і «ControlFlow».
- Не використовуйте кольору тла у стилі тексту, окрім «Alert» і «RegionMarker».

Більшість визначень підсвічувань синтаксичних конструкцій добре виглядають у типових темах «Світла Breeze» та «Темна Breeze». Тому іншим способом забезпечення сумісності із визначеннями є використання подібних кольорів у стилях тексту, зокрема *зеленого* для «Preprocessor» та «Others», *синього* для «DataType» і «Attribute» або *пурпурового* для «Function».

Зауважте, що ці рекомендації не є обов'язковими для створення і оприлюднення теми.

6.4 Створення скриптів мовою JavaScript

Можливості компонента редактора KatePart можна дуже просто розширити за допомогою написання скриптів. Для написання скриптів слід використовувати мову ECMAScript (широко відому як JavaScript). У KatePart передбачено підтримку двох типів скриптів: скрипти встановлення відступів та скрипти командного рядка. Функціональні можливості, подібні до скриптів командного рядка, також є частиною додатка фрагментів.

6.4.1 Скрипти додавання відступів

Скрипти встановлення відступів, які також будемо називати інструментами відступів, автоматично встановлюють відступи у тексті під час його введення. Наприклад, після натискання клавіші **Enter** програма зазвичай збільшує відступ у наступному рядку.

У наступних розділах наведено покрокові настанови щодо створення основи простого інструменту відступів. На першому кроці вам слід створити файл `*.js` з назвою, наприклад, `javascript.js` у локальній домашній теці `$XDG_DATA_HOME /katepart5/script/indentation`. Тут змінна середовища `XDG_DATA_HOME` типово має значення `~/.local` або `~/.local/share`.

У Windows® ці файли зберігаються у `%USERPROFILE%\AppData\Local\katepart5\script\indentation`. *`%USERPROFILE%`* зазвичай є скороченням для `C:\Users\користувач`.

6.4.1.1 Заголовок скрипту додавання відступів

Заголовок файла `javascript.js` подається у межах закоментованого блоку і має таку форму

```
var katescript = {
    "name": "JavaScript",
    "author": "Example Name <example.name@some.address.org>",
    "license": "BSD License",
    "revision": 1,
    "kate-version": "5.1",
    "required-syntax-style": "javascript",
    "indent-languages": ["javascript"],
    "priority": 0,
}; // kate-script-header, must be at the start of the file without  ↩
    comments
```

Нижче ми зупинимося докладніше на кожному з записів заголовка.

- `name` [обов'язковий запис]: це назва інструменту відступів, яку буде показано у меню **Інструменти** → **Відступ** і діалогових вікнах налаштування.
- `author` [необов'язковий запис]: ім'я автора та дані щодо способу встановлення з ним зв'язку.
- `license` [необов'язковий запис]: скорочена форма умов ліцензування, зокрема BSD або LGPLv3.
- `revision` [обов'язковий запис]: версія скрипту. Внесення змін до коду скрипту має призводити до збільшення його версії.

- `kate-version` [обов'язковий запис]: мінімальне значення версії KatePart, у якій працюватиме скрипт.
- `required-syntax-style` [необов'язковий запис]: потрібний вам стиль синтаксису, який відповідає вказаному значенням `style` у файлах визначення підсвічування синтаксичних конструкцій. Цей запис є важливим для інструментів відступів, які працюють на основі певних даних щодо підсвічування у документі. Якщо буде вказано стиль синтаксичних конструкцій, інструментом відступів можна буде скористатися, лише якщо буде задіяно відповідний інструмент підсвічування тексту. Таким чином можна запобігти «невизначеній поведінці», спричиненій використанням інструменту відступів без потрібної для його роботи схеми підсвічування. Наприклад, у такий спосіб налаштовано інструмент відступів Ruby у файлах `ruby.js` і `ruby.xml`.
- `indent-languages` [необов'язковий запис]: масив JSON стилів синтаксичних конструкцій, які може обробляти інструмент відступів, наприклад `[˝c++˝, ˝java˝`.
- `priority` [необов'язковий запис]: якщо якомусь файлу з визначеним підсвічуванням синтаксичних конструкцій відповідає декілька інструментів відступів, за допомогою цього запису буде встановлено пріоритет застосування інструменту відступів.

6.4.1.2 Код інструменту відступів

Тепер, озброєні знаннями про формат заголовка, ви можете перейти до вивчення того, як же працює сам скрипт встановлення відступів. Основа подібного скрипту виглядає так:

```
// вимагаємо бібліотек JS katepart, наприклад range.js, якщо використову ↩
    ється Range
require ("range.js");

triggerCharacters = "{}/:;";
function indent(line, indentWidth, ch)
{
    //скрипт буде викликано під час обробки символу нового рядка (ch ==  ↩
        '\n') і всіх символів, вказаних у
    // загальній змінній triggerCharacters. Під час використання пункту  ↩
        меню ІнструментиФорматований відступ
    // змінна ch матиме порожнє значення, тобто ch == ''.
    //
    // див. також розділ «Інтерфейс (API) роботи зі скриптами»
    return -2;
}
```

У функції `indent()` передбачено три параметри:

- `line`: рядок, у якому слід встановити відступ
- `indentWidth`: ширина відступу у пробілах
- `ch`: символ нового рядка (`ch == '\n'`), символ перемикання, вказаний за допомогою `triggerCharacters` або порожній рядок, якщо користувачем було обрано пункт меню **Інструменти → Форматований відступ**.

Значення, повернуте функцією `indent()`, визначатиме спосіб встановлення відступу у рядку. Якщо буде повернуто ціле число, його обробку буде виконано за такими варіантами:

- повернуто значення `-2`: нічого не робити
- повернуто значення `-1`: зберегти відступ (його буде визначено на основі попереднього непорожнього рядка)
- повернуто значення `0`: числа $>= 0$ визначають глибину відступу у пробілах

Крім того, може бути повернуто масив з двох елементів:

- `повернуто [ відступ, вирівнювання ];`

Першим елементом такого масиву є глибина відступу, подібна на значення, про яке ми говорили раніше. Інший же елемент є абсолютним значенням, що відповідає стовпчику «вирівнювання». Якщо це значення буде більшим за значення відступу, до відступу після додавання відступу, визначеного першим параметром, буде додано різницю між цими значенням. Якщо ж значення другого параметра буде меншим за відступ, його буде проігноровано. Одночасне використання для встановлення відступів пробілів і символів табуляції часто називають «мішаним режимом».

Розглянемо такий приклад: для встановлення відступів використовується символ табуляції, а ширину відступу визначено як 4 пробіли. У нашому прикладі, <таб> — це символ табуляції, а «.» — пробіл:

```
1: <таб><таб>foobar("привіт",
2: <таб><таб>......."світе");
```

Під час встановлення відступу для рядка 2 функція `indent()` повертає значення [8, 15]. У результаті буде додано два символи табуляції для встановлення відступу до стовпчика 8, а потім буде додано 7 пробілів для вирівнювання за другим параметром, більшим за перший, отже рядок залишатиметься вирівняним під час перегляду файла за будь-якої встановленої ширити відступу табуляції.

У типовому пакунку KDE для KatePart передбачено декілька інструментів відступів. Код цих інструментів мовою JavaScript зберігається у каталозі `$XDG_DATA_DIRS /katepart5/script/indentation`.

У Windows® ці файли зберігаються у `%USERPROFILE%\AppData\Local\katepart5\script\indentation`. *`%USERPROFILE%`* зазвичай є скороченням для `C:\\Users\\користувач`.

Під час розробки скрипту встановлення відступів корисним буває перезавантаження скриптів для перевірки коректності поведінки під час встановлення відступів. Замість перезапуску програми достатньо просто перейти до командного рядка і виконати команду **reload-scripts**.

Якщо вами було створено якийсь корисний скрипт, вам варто зробити його вашим власним внеском до проєкту розробки KatePart на сторінці нашого проєкту на GitLab.

6.4.2 Скрипти командного рядка

> **Примітка**
> Функціональні можливості, подібні до скриптів командного рядка, також є частиною додатка фрагментів. Скрипти додатка можуть бути початковою позицією для розробки, особливо невеликих нетипових скриптів.

Оскільки всі користувачі мають різні потреби, у KatePart передбачено підтримку невеличких допоміжних інструментів для пришвидшення роботи з фрагментами тексту за допомогою вбудованого командного рядка. Наприклад, команду **sort** (впорядкувати) реалізовано саме за допомогою такого інструменту або скрипту. У цьому розділі ви знайдете пояснення щодо способу створення файлів `*.js`, які допоможуть вам розширити можливості KatePart додаванням допоміжних скриптів.

Скрипти командного рядка зберігаються у одній теці зі скриптами встановлення відступів. Отже на першому кроці вам слід створити файл `*.js` з назвою `myutils.js` у локальній домашній теці `$XDG_DATA_HOME /katepart5/script/commands`. Тут змінна середовища `XDG_DATA_HOME` типово має значення `~/.local` або `~/.local/share`.

У Windows® ці файли зберігаються у `%USERPROFILE%\AppData\Local\katepart5\script\commands`. *`%USERPROFILE%`* зазвичай є скороченням для `C:\\Users\\користувач`.

6.4.2.1 Заголовок скрипту командного рядка

Заголовок кожного скрипту командної оболонки має бути вбудовано до JSON на початку скрипту так:

```
var katescript = {
    "author": "Example Name <example.name@some.address.org>",
    "license": "LGPLv2+",
    "revision": 1,
    "kate-version": "5.1",
    "functions": ["sort", "moveLinesDown"],
    "actions": [
        {   "function": "sort",
            "name": "Sort Selected Text",
            "category": "Editing",
            "interactive": "false"
        },
        {   "function": "moveLinesDown",
            "name": "Move Lines Down",
            "category": "Editing",
            "shortcut": "Ctrl+Shift+Down",
            "interactive": "false"
        }
    ]
}; // kate-script-header, must be at the start of the file without ↩
    comments
```

Тепер про кожен із записів докладніше:

- `author` [необов'язковий запис]: ім'я автора та дані щодо способу встановлення з ним зв'язку.
- `license` [необов'язковий запис]: скорочена форма умов ліцензування, зокрема BSD або LGPLv2.
- `revision` [обов'язковий запис]: версія скрипту. Внесення змін до коду скрипту має призводити до збільшення його версії.
- `kate-version` [обов'язковий запис]: мінімальне значення версії KatePart, у якій працюватиме скрипт.
- `functions` [обов'язковий запис]: масив JSON команд скрипту.
- `actions` [необов'язковий запис]: масив JSON об'єктів JSON, який визначає пункти дій, які буде показано у меню програми. Докладніший опис наведено у розділі щодо призначення клавіатурних скорочень.

Оскільки значення `functions` є масивом JSON, окремий скрипт може містити довільну кількість команд для командного рядка. Доступ до кожної із функцій можна оримати за допомогою вбудованого командного рядка KatePart.

6.4.2.2 Код скрипту командного рядка

Всі оголошені у заголовку функції має бути реалізовано у тілі скрипту. Наприклад, у файлі скрипту з наведеного вище прикладу слід реалізувати дві функції, **sort** і **moveLinesDown**. Всі функції має бути записано відповідно до таких синтаксичних правил:

```
// вимагаємо бібліотек JS katepart, наприклад range.js, якщо використову ↩
    ється Range
require ("range.js");
```

```
function <назва>(параметр1, параметр2, ...)
{
    // ...реалізація, див. також: Інтерфейс (API) роботи зі скриптами
}
```

Параметри у командному рядку передаються функції як *параметр1*, *параметр2* тощо. Щоб надати користувачам можливість ознайомлюватися з документацією щодо команди, просто реалізуйте функцію 'help' у такий спосіб:

```
function help(cmd)
{
    if (cmd == "sort") {
        return i18n("Sort the selected text.");
    } else if (cmd == "...") {
        // ...
    }
}
```

Після цього виконання команди **help sort** у командному рядку призведе до виклику відповідної функції довідки (help) з параметром *cmd* встановленим у назву вказаної команди, тобто, у нашому прикладі, *cmd == "sort"*. У відповідь на команду KatePart покаже користувачеві визначений вами текст довідки.

Під час розробки скрипту командного рядка корисним буває перезавантаження скриптів для перевірки коректності поведінки під час встановлення відступів. Замість перезапуску програми достатньо просто перейти до командного рядка і виконати команду **reload-scripts**.

6.4.2.2.1 Призначення клавіатурних скорочень

Для створення скриптів, доступ до яких можна буде отримувати з меню програми та за допомогою клавіатурних скорочень, скрипт повинен мати надавати відповідний заголовок. У наведеному вище прикладі, обидві функції `sort` та `moveLinesDown` показують відповідні пункти меню через таку частину заголовка скрипту:

```
var katescript = {
    ...
    "actions": [
        {   "function": "sort",
            "name": "Sort Selected Text",
            "icon": "",
            "category": "Editing",
            "interactive": "false"
        },
        {   "function": "moveLinesDown",
            "name": "Move Lines Down",
            "icon": "",
            "category": "Editing",
            "shortcut": "Ctrl+Shift+Down",
            "interactive": "false"
        }
    ]
};
```

Поля для однієї дії є такими:

- `function` [обов'язкове поле]: пункт функції, який має бути показано у меню **Інструменти** → **Скрипти**.
- `name` [необов'язковий запис]: текст пункту, який буде показано у меню скриптів.

- `icon` [необов'язковий запис]: піктограма, яку буде показано поряд з текстом пункту меню. Можна використовувати назву будь-якої з піктограм KDE.
- `category` [необов'язковий запис]: якщо буде вказано категорію, пункт скрипту буде додано у підменю.
- `shortcut` [необов'язковий запис]: клавіатурне скорочення, вказане за допомогою цього значення буде типовим. Приклад: `Ctrl+Alt+t`. Докладніше про можливі скорочення можна дізнатися з документації до Qt.
- `interactive` [необов'язковий]: якщо для роботи скрипту потрібні дані, введені користувачем, встановіть значення `true`.

Якщо вами було створено якийсь корисний скрипт, вам варто зробити його вашим власним внеском до проєкту розробки KatePart на сторінці нашого проєкту на GitLab.

6.4.3 Інтерфейс (API) роботи зі скриптами

Програмним інтерфейсом роботи зі скриптами, основи якого викладено тут, можна користуватися у будь-яких скриптах, зокрема скриптах роботи з відступами та командному рядку програми. Класи `Cursor` і `Range` визначаються бібліотечними файлами у `$XDG_DATA_DIRS /katepart5/libraries`. Якщо вам потрібно ними скористатися у певному скрипті для використання якихось із функцій `Document` або `View`, будь ласка, включіть до скрипту потрібну вам бібліотеку такою командою:

```
// вимагаємо бібліотек JS katepart, наприклад range.js, якщо використову ↩
    ється Range
require ("range.js");
```

Щоб розширити стандартний програмний інтерфейс (API) для роботи зі скриптами власними функціями і прототипами просто створіть файл у локальній теці файлів налаштування KDE, `$XDG_DATA_HOME /katepart5/libraries`, і включіть його до вашого скрипту за допомогою такого коду:

```
require ("vashscript.js");
```

У Windows® ці файли зберігаються у `%USERPROFILE%\AppData\Local\katepart5\libraries`. *`%USERPROFILE%`* зазвичай є скороченням для `C:\\Users\\користувач`.

Рекомендованим способом розширення можливостей прототипів, зокрема `Cursor` або `Range`, *не* є внесення змін до загальних файлів `*.js`. Для цього вам краще внести зміи до прототипу `Cursor` у JavaScript після команди включення `cursor.js` до вашого скрипту за допомогою `require`.

6.4.3.1 Курсори і діапазони

Оскільки KatePart є текстовим редактором, весь програмний інтерфейс (API) за можливості засновано на курсорах та діапазонах тексту. Об'єкт Cursor (курсор) є простим кортежем `( line, column)` (рядок, стовпчик), що визначає позицію у тексті документа. Об'єкт Range (діапазон) це фрагмент тексту від початкової позиції курсора до кінцевої позиції курсора. Докладніше про програмний інтерфейс (API) ми поговоримо у наступних розділах.

6.4.3.1.1 Прототип Cursor (курсор)

`Cursor();`

Конструктор. Повертає об'єкт Cursor (курсор) у позиції `(0, 0)`.

Приклад: `var cursor = new Cursor();`

`Cursor(int рядок, int стовпчик);`

Конструктор. Повертає об'єкт Cursor (курсор) у позиції (рядок, стовпчик).

Приклад: `var cursor = new Cursor(3, 42);`

`Cursor(Cursor інший);`

Конструктор копіювання. Повертає копію курсора *інший*.

Приклад: `var copy = new Cursor(other);`

`Cursor Cursor.clone();`

Повертає клон курсора.

Приклад: `var clone = cursor.clone();`

`Cursor.setPosition(int рядок, int позиція);`

Встановлює курсор у місце вказане параметрами *рядок* і *позиція*.

Актуальна версія: KDE 4.11

`bool Cursor.isValid();`

Перевіряє, чи є курсор коректним. Курсор вважається некоректним, якщо значення його позицій у рядку і/або стовпчику дорівнюють `-1`.

Приклад: `var valid = cursor.isValid();`

`Cursor Cursor.invalid();`

Повертає новий некоректний курсор, розташований у позиції `(-1, -1)`.

Приклад: `var invalidCursor = cursor.invalid();`

`int Cursor.compareTo(Cursor інший);`

Порівнює поточний курсор з курсором *інший*. Повертає

- `-1`, якщо поточний курсор розташовано перед курсором *інший*,
- `0`, якщо курсори у однакових позиціях і
- `+1`, якщо поточний курсор розташовано після курсора *інший*.

`bool Cursor.equals(Cursor інший);`

Повертає `true`, якщо цей курсор і курсор *інший* є однаковими. Якщо це не так, повертає `false`.

`String Cursor.toString();`

Повертає об'єкт курсора як рядок у формі «`Cursor(line, column)`».

6.4.3.1.2 Прототип Range (діапазон)

`Range();`

Конструктор. Виклик `new Range()` повертає Range (діапазон) у (0, 0) - (0, 0).

`Range(Cursor початок, Cursor кінець);`

Конструктор. Виклик функції `new Range(початок, кінець)` повертає Range (діапазон)(*початок*, *кінець*).

`Range(int початковийРядок, int початковийСтовпчик, int кінцевийРядок, int кінцевийСтовпчик);`

Конструктор. Виклик функції `new Range(початковийРядок, початковийСтовпчик, кінцевийРядок, кінцевийСтовпчик)` повертає Range (діапазон) від позиції (*початковийРядок*, *початковийСтовпчик*) до позиції (*кінцевийРядок*, *кінцевийСтовпчик*).

`Range(Range інший);`

Конструктор копіювання. Повертає копію Range *інший*.

`Range Range.clone();`

Повертає клон діапазону.

Приклад: `var clone = range.clone();`

`bool Range.isEmpty();`

Повертає `true`, якщо початкова і кінцева позиції курсора є рівними.

Приклад: `var empty = range.isEmpty();`

Актуальна версія: KDE 4.11

`bool Range.isValid();`

Повертає `true`, якщо початкова і кінцева позиції курсора є коректними. Якщо це не так, повертає `false`.

Приклад: `var valid = range.isValid();`

`Range Range.invalid();`

Повертає Range (діапазон) від (-1, -1) до (-1, -1).

`bool Range.contains(`*`Cursor`* `курсор);`

Повертає `true`, якщо позиція курсора міститься у діапазоні, у іншому випадку повертає `false`.

`bool Range.contains(`*`Range`* `інший);`

Повертає `true`, якщо поточний діапазон містить діапазон *`інший`*. Якщо це не так, повертає `false`.

`bool Range.containsColumn(`*`int`* `стовпчик);`

Повертає `true`, якщо *`стовпчик`* належить напіввідкритому інтервалу `[start.column, end.column)`. Якщо це не так, повертає `false`.

`bool Range.containsLine(`*`int`* `рядок);`

Повертає `true`, якщо *`рядок`* належить до напіввідкритого інтервалу `[start.line, end.line)`. Якщо це не так, повертає `false`.

`bool Range.overlaps(`*`Range`* `інший);`

Повертає `true`, якщо поточний діапазон і діапазон *`інший`* мають ненульовий перетин. Якщо це не так, повертає `false`.

`bool Range.overlapsLine(`*`int`* `рядок);`

Повертає `true`, якщо *`рядок`* належить до інтервалу `[start.line, end.line]`. Якщо це не так, повертає `false`.

`bool Range.overlapsColumn(`*`int`* `стовпчик);`

Повертає `true`, якщо *`стовпчик`* належить інтервалу `[start.column, end.column]`. Якщо це не так, повертає `false`.

`bool Range.onSingleLine();`

Повертає `true`, якщо діапазон починається і завершується на тому самому рядку, тобто якщо *`Range.start.line == Range.end.line`*.

Актуальна версія: KDE 4.9

`bool Range.equals(`*`Range`* `інший);`

Повертає `true`, якщо поточний діапазон і діапазон *`інший`* тотожні. Якщо це не так, повертає `false`.

`String Range.toString();`

Повертає діапазон у форматі «`Range(Cursor(line, column), Cursor(line, column))`».

6.4.3.2 Загальні функції (Global Functions)

У цьому розділі наведено всі загальні функції.

6.4.3.2.1 Читання і включення файлів

`String read(String назва файла);`

Виконає пошук файла з назвою *назва файла* у каталозі `katepart5/script/files` і поверне його вміст як рядок.

`void require(String назва файла);`

Виконає пошук файла з назвою *назва файла* у каталозі `katepart5/script/libraries` і обробить його код. У `require` передбачено вбудований захист від повторного включення одного і того самого файла.

Актуальна версія: KDE 4.10

6.4.3.2.2 Налагоджування (Debugging)

`void debug(String текст);`

Виводить *текст* до `stdout` у консоль, з якої запущено програму.

6.4.3.2.3 Переклад (Translation)

Повноцінна локалізація стане можливою лише за використання декількох функцій, призначених для перекладу рядків у скриптах, а саме `i18n`, `i18nc`, `i18np` і `i18ncp`. Робота цих функцій подібна до роботи функцій перекладу рядків у KDE.

За допомогою функцій перекладу і системи перекладу KDE вбудовані у скрипт рядки повідомлень може бути перекладено мовою інтерфейсу програми. Рядки у скриптах, які є частиною офіційної збірки KatePart буде автоматично видобуто і подано для перекладу командами перекладачів KDE. Іншими словами, якщо ви є розробником основної гілки KatePart, ви не маєте перейматися видобуванням повідомлень і їхнім перекладом. Втім, слід зауважити, що переклад працюватиме лише у межах інфраструктури KDE, тобто переклад нових рядків скриптів, розроблених поза межами KDE, неможливий. Тому вам варто надіслати вашу роботу до основної гілки розробки Kate, щоб уможливити належний переклад.

`void i18n(String текст, параметр1, ...);`

Перекладає *текст* мовою, використаною у інтерфейсі програми. Параметри *параметр1*, ... є необов'язковими. Вони є замінниками рядків `%1`, `%2` тощо.

`void i18nc(String контекст, String текст, параметр1, ...);`

Перекладає *текст* мовою, використаною у інтерфейсі програми. Крім того, перекладачі зможуть побачити рядок *контекст*. За допомогою контексту перекладачам буде простіше правильно перекласти рядок. Параметри *параметр1*, ... є необов'язковими. Вони є замінниками рядків `%1`, `%2` тощо.

`void i18np(String однина, String множина, int кількість, параметр1, ...);`

Перекладає *однину* або *множину* повідомлення мовою, використаною у інтерфейсі програми залежно від вказаного значення *кількість*. Параметри *параметр1*, ... є необов'язковими. Вони є замінниками рядків `%1`, `%2` тощо.

`void i18ncp(String контекст, String однина, String множина, int кількість, параметр1, ...);`

Перекладає *однину* або *множину* повідомлення мовою, використаною у інтерфейсі програми залежно від вказаного значення *кількість*. Крім того, перекладачі зможуть побачити рядок *контекст*. За допомогою контексту перекладачам буде простіше правильно перекласти рядок. Параметри *параметр1*, ... є необов'язковими. Вони є замінниками рядків `%1`, `%2` тощо.

6.4.3.3 Програмний інтерфейс `View`

Яким би чином не було запущено скрипт, він завжди користуватиметься загальною змінною «`view`», що відповідає поточній активній панелі перегляду. Нижче наведено список всіх типових функцій об'єкта View.

`void view.copy()`

Копіювати позначений фрагмент, якщо щось позначено. Якщо нічого не позначено, копіювати поточний рядок, якщо позначено пункт налаштувань `[ ] Копіювати/Вирізати поточний рядок, якщо нічого не позначено`.

Актуальна версія: KDE Frameworks 5.79

`void view.cut()`

Вирізати позначений фрагмент, якщо щось позначено. Якщо нічого не позначено, вирізати поточний рядок, якщо позначено пункт налаштувань `[ ] Копіювати/Вирізати поточний рядок, якщо нічого не позначено`.

Актуальна версія: KDE Frameworks 5.79

`void view.paste()`

Вставити вміст буфера обміну даними.

Актуальна версія: KDE Frameworks 5.79

`Cursor view.cursorPosition()`

Повертає поточну позицію курсора у області перегляду.

`void view.setCursorPosition(`*`int`* `рядок, `*`int`* `стовпчик); void view.setCursorPosition(`*`Cursor`* `курсор);`

Встановлює для поточного курсора позицію, вказану напряму (рядок, стовпчик), або позицію вказаного курсора.

`Cursor view.virtualCursorPosition();`

Повертає позицію віртуального курсора. Всі символи табуляцій буде враховано за допомогою відповідної кількості пробілів, яка залежатиме від поточної ширини табуляції.

`void view.setVirtualCursorPosition(`*`int`* `рядок, `*`int`* `стовпчик); void view.setVirtualCursorPosition(`*`Cursor`* `курсор);`

Встановлює для віртуального курсора позицію, вказану напряму (рядок, стовпчик), або позицію вказаного курсора.

`String view.selectedText();`

Повертає позначений фрагмент тексту. Якщо жодного фрагменту не позначено, повертає порожній рядок.

`bool view.hasSelection();`

Повертає `true`, якщо у області перегляду позначено фрагмент тексту, інакше повертає `false`.

`Range view.selection();`

Повертає діапазон позначеного фрагменту тексту. Якщо жодного фрагменту не позначено, буде повернуто некоректний діапазон.

`void view.setSelection(`*`Range`* `діапазон);`

Встановлює позначення тексту за вказаним діапазоном.

`void view.removeSelectedText();`

Вилучає позначений текст. Якщо у області перегляду не було позначено жодного тексту, ніяких дій виконано не буде.

`void view.selectAll();`

Позначає весь текст у документі.

`void view.clearSelection();`

Знімає позначення фрагменту тексту, не вилучаючи сам текст.

`void view.setBlockSelection(bool on);`

Вмикає або вимикає режим позначення блоку.

`bool view.blockSelection();`

Повертає `true`, якщо увімкнено режим позначення блоку, інакше повертає `false`.

`void view.align(Range діапазон);`

Належним чином повторно встановити відступи рядків у вказаному діапазоні *діапазон* за поточними параметрами відступів.

`void view.alignOn(Range діапазон, String взірець = "");`

Вирівнює рядки у діапазоні *діапазон* у позиції, яку задано формальним виразом *взірець*. Якщо буде вказано порожній *взірець*, вирівнювання типово відбудеться за першим непорожнім символом. Якщо у взірці буде вказано блок захоплення, вирівнювання відбудеться за захопленим блоком.

Приклади:

`view.alignOn(document.documentRange(), '-');` вставить пробіли до першого - у кожному з рядків для вирівнювання усіх дефісів за вказаною позицією.

`view.alignOn(document.documentRange(), ':\\s+(.)');` вставить пробіли до першого непорожнього символу, який стоїть після двокрапки, для вирівнювання усіх таких символів за однаковою позицією.

`object view.executeCommand(String команда, String аргументи, Range діапазон);`

Виконує команду *команда* із додатковими аргументами *аргументи* і необов'язковим діапазоном *діапазон*. Повернутий об'єкт *object* є булевою властивістю *object.ok*, яка вказує на те, чи було успішним виконання команди *команда*. Якщо сталася помилка, у рядку *object.status* міститиметься повідомлення про помилку.

Актуальна версія: KDE Frameworks 5.50

`Range view.searchText(Range діапазон, String взірець, bool назад = false);`

Шукати перше включення *взірця* у *діапазоні* і повернути знайдений діапазон. Пошук буде виконано у зворотному напрямку, якщо для додаткового булевого параметра *назад* буде встановлено значення `true`.

Повернутий діапазон буде некоректним (див. Range.isValid()), якщо *взірець* не буде знайдено у *діапазоні*.

Актуальна версія: KDE Frameworks 5.97

6.4.3.4 Програмний інтерфейс (API) `Document`

Яким би чином не було запущено скрипт, він завжди користуватиметься загальною змінною «`document`», що відповідає поточній активній панелі перегляду. Нижче наведено список всіх типових функцій об'єкта Document.

`String document.fileName();`

Повертає назву файла документа або порожній рядок для незбережених буферів з текстом.

`String document.url();`

Повертає адресу URL документа повністю або порожній рядок для незбережених буферів з текстом.

`String document.mimeType();`

Повертає тип MIME документа або тип MIME `application/octet-stream`, якщо відповідного типу MIME встановити не вдасться.

`String document.encoding();`

Повертає поточне кодування символів, яке буде використано для збереження даних файла.

`String document.highlightingMode();`

Повертає загальний режим підсвічування, використаний у всьому документі.

`String document.highlightingModeAt(`*`Cursor`* `позиція);`

Повертає режим підсвічування, використаний за вказаною позицією у тексті.

`Array document.embeddedHighlightingModes();`

Повертає масив режимів підсвічування, вбудованих до поточного документа.

`bool document.isModified();`

Повертає `true`, якщо у документі є незбережені зміни, інакше повертає `false`.

`String document.text();`

Повертає увесь вміст документа у формі єдиного рядка тексту. Розриви рядків буде позначено символом розриву рядка «`\n`».

`String document.text(`*`int`* `зРядка,` *`int`* `зіСтовпчика,` *`int`* `доРядка,` *`int`* `доСтовпчика);`
`String document.text(`*`Cursor`* `з,` *`Cursor`* `до);` `String document.text(`*`Range`* `діапазон);`

Повертає фрагмент тексту у вказаному діапазоні. Для того, щоб код було легше читати, ми рекомендуємо вам використовувати засновану на об'єктах Cursor і Range версію функції.

`String document.line(`*`int`* `рядок);`

Повертає рядок за його номером у тексті. Якщо вказаний номер не належатиме до діапазону номерів рядків документа, буде повернуто порожній рядок.

`String document.wordAt(`*`int`* `рядок,` *`int`* `стовпчик);` `String document.wordAt(`*`Cursor`* `курсор);`

Повертає слово за вказаною позицією курсора.

`Range document.wordRangeAt(`*`int`* `рядок,` *`int`* `стовпчик);` `Range document.wordRangeAt(`*`Cursor`* `курсор);`

Повертає діапазон слова за вказаним розташуванням курсора. Повернуте значення діапазону буде некоректним (див. Range.isValid()), якщо текст розташовано за кінцем рядка. Якщо за вказаним розташуванням курсора не буде слова, функцією буде повернуто порожній діапазон.

Актуальна версія: KDE 4.9

`String document.charAt(`*`int`* `рядок,` *`int`* `стовпчик);` `String document.charAt(`*`Cursor`* `курсор);`

Повертає символ за вказаною позицією курсора.

`String document.firstChar(`*`int`* `рядок);`

Повертає перший символ вказаного рядка *`рядок`*, який не є пробілом. Першим символом рядка вважається символ у стовпчику 0. Якщо рядок є порожнім або складається лише з пробілів, функція повертає порожній рядок.

`String document.lastChar(`*`int`* `рядок);`

Повертає останній символ вказаного рядка *`рядок`*, який не є пробілом. Якщо рядок є порожнім або складається лише з пробілів, функція повертає порожній рядок.

`bool document.isSpace(`*`int`* `рядок,` *`int`* `стовпчик);` `bool document.isSpace(`*`Cursor`* `курсор);`

Повертає `true`, якщо символ у вказаній позиції курсора є пробілом. Якщо це не так, повертає `false`.

`bool document.matchesAt(int рядок, int стовпчик, String текст); bool document.matchesAt(Cursor курсор, String текст);`

Повертає `true`, якщо вказаний *текст* розташовано за відповідною позицією курсора. Якщо це не так, повертає `false`.

`bool document.startsWith(int рядок, String текст, bool пропускатиПробіли);`

Повертає `true`, якщо рядок з вказаним номером починається з фрагмента тексту *текст*. Якщо це не так, повертає `false`. За допомогою параметра *пропускатиПробіли* можна вказати програмі, чи слід ігнорувати пробіли.

`bool document.endsWith(int рядок, String текст, bool пропускатиПробіли);`

Повертає `true`, якщо рядок з вказаним номером завершується фрагментом тексту *текст*. Якщо це не так, повертає `false`. За допомогою параметра *пропускатиПробіли* можна вказати програмі, чи слід ігнорувати пробіли.

`bool document.setText(String текст);`

Замінює весь вміст документа на *текст*.

`bool document.clear();`

Вилучає з документа весь текст.

`bool document.truncate(int рядок, int стовпчик); bool document.truncate(Cursor курсор);`

Обрізає рядок з вказаним номером на вказаному стовпчику або позиції курсора. Повертає `true` у разі успіху або `false`, якщо рядка з вказаним номером у документі немає.

`bool document.insertText(int рядок, int стовпчик, String текст); bool document.insertText(Cursor курсор, String текст);`

Вставляє фрагмент тексту *текст* у вказану позицію курсора. Повертає `true` у разі успіху або `false`, якщо документ відкрито у режимі лише для читання.

`bool document.removeText(int зРядка, int зіСтовпчика, int доРядка, int доСтовпчика); bool document.removeText(Cursor з, Cursor до); bool document.removeText(Range діапазон);`

Вилучає текст у вказаному діапазоні. Повертає `true` у разі успіху або `false`, якщо документ відкрито у режимі лише для читання.

`bool document.insertLine(int рядок, String текст);`

Вставляє вказаний фрагмент тексту до рядка з вказаним номером. Повертає `true` у разі успіху або `false`, якщо документ відкрито у режимі лише для читання або рядка з вказаним номером у документі немає.

`bool document.removeLine(int рядок);`

Вилучає рядок з вказаним номером. Повертає `true` у разі успіху або `false`, якщо документ відкрито у режимі лише для читання або рядка з вказаним номером у документі немає.

`bool document.wrapLine(int рядок, int стовпчик); bool document.wrapLine(Cursor курсор);`

Обрізає рядок з вказаним номером за вказаним розташуванням курсора. Повертає `true` у разі успіху. У інших випадках повертає `false` (наприклад, якщо номер рядка < 0).

Актуальна версія: KDE 4.9

`void document.joinLines(int початковийРядок, int кінцевийРядок);`

Об'єднує рядки у діапазоні від *початковийРядок* до *кінцевийРядок*. Послідовні рядки тексту завжди відокремлюються одинарним пробілом.

`int document.lines();`

Повертає кількість рядків у документі.

`bool document.isLineModified(int рядок);`

Повертає `true`, якщо у рядку *рядок* зараз містяться незбережені дані.

Актуальна версія: KDE 5.0

`bool document.isLineSaved(int рядок);`

Повертає `true`, якщо *рядок* було змінено, після чого документ було збережено. Інакше кажучи, у рядку вже не міститься незбережених даних.

Актуальна версія: KDE 5.0

`bool document.isLineTouched(int рядок);`

Повертає `true`, якщо у рядку *рядок* зараз містяться незбережені дані або його вміст було раніше змінено.

Актуальна версія: KDE 5.0

`bool document.findTouchedLine(int початковийРядок, bool напрямок);`

Шукати наступний змінений рядок, починаючи з рядка *початковийРядок*. Пошук виконується у напрямку кінця або початку документа залежно від напрямку пошуку, визначеного параметром *напрям*.

Актуальна версія: KDE 5.0

`int document.length();`

Повертає кількість символів у документі.

`int document.lineLength(int рядок);`

Повертає довжину рядка з номером *рядок*.

`void document.editBegin();`

Започатковує групу редагування для впорядкування операцій зі скасування або повторення дій. Не забувайте, що викликати `editEnd()` слід саме стільки разів, скільки разів було викликано `editBegin()`. Виклики `editBegin()` використовують вбудований лічильник, отже їх може бути вкладено один у оден.

`void document.editEnd();`

Завершує групу редагування. Останній виклик `editEnd()` (тобто відповідник першого виклику `editBegin()`) завершує крок з редагування.

`int document.firstColumn(int рядок);`

Повертає перший відмінний від пробілу стовпчик у вказаному за допомогою параметра номера, *рядок*, рядку. Якщо у рядку будуть лише пробіли, функція поверне значення `-1`.

`int document.lastColumn(int рядок);`

Повертає останній відмінний від пробілу стовпчик у вказаному за допомогою параметра номера, *рядок*, рядку. Якщо у рядку будуть лише пробіли, функція поверне значення `-1`.

`int document.prevNonSpaceColumn(int рядок, int стовпчик); int document.prevNonSpaceColumn(Cursor курсор);`

Повертає номер стовпчика з символом, відмінним від пробілу. Пошук буде виконано у напрямку початку документа, починаючи з вказаної позиції курсора.

`int document.nextNonSpaceColumn(int рядок, int стовпчик); int document.nextNonSpaceColumn(Cursor курсор);`

Повертає номер стовпчика з символом, відмінним від пробілу. Пошук буде виконано у напрямку кінця документа, починаючи з вказаної позиції курсора.

`int document.prevNonEmptyLine(int рядок);`

Повертає наступний непорожній рядок, що містить символи, відмінні від пробілів. Пошук відбуватиметься у напрямку початку документа.

`int document.nextNonEmptyLine(int рядок);`

Повертає наступний непорожній рядок, що містить символи, відмінні від пробілів. Пошук відбуватиметься у напрямку кінця документа.

`bool document.isInWord(String символ, int атрибут);`

Повертає `true`, якщо вказаний *символ* з вказаним параметром *атрибут* може бути частиною слова. Якщо це не так, повертає `false`.

`bool document.canBreakAt(String символ, int атрибут);`

Повертає `true`, якщо вказаний *символ* з вказаним параметром *атрибут* може бути використано для розбиття рядка. Якщо це не так, повертає `false`.

`bool document.canComment(int атрибутПочатку, int атрибутКінця);`

Повертає `true`, якщо діапазон тексту, початок і кінець якого визначаються на основі вказаних атрибутів, можна закоментувати. Якщо це не так, повертає `false`.

`String document.commentMarker(int атрибут);`

Повертає позначку коментаря для окремого рядка коментарів з вказаним параметром *атрибут*.

`String document.commentStart(int атрибут);`

Повертає позначку, якою має розпочинатися багаторядковий коментар для вказаного значення параметра *атрибут*.

`String document.commentEnd(int атрибут);`

Повертає позначку, якою має розпочинатися багаторядковий коментар для вказаного значення параметра *атрибут*.

`Range document.documentRange();`

Повертає діапазон всього документа.

`Cursor documentEnd();`

Повертає курсор, розташований у останній позиції останнього рядка у документі.

`bool isValidTextPosition(int рядок, int позиція); bool isValidTextPosition(Cursor курсор);`

Повертає `true`, якщо вказану позицію курсора розташовано у коректній позиції у тексті. Позиція у тексті є коректною, лише якщо її розташовано на початку, у середині або у кінці коректного рядка тексту. Крім того, текстова позиція є некоректною, якщо її розташовано посередині замінника символу Unicode.

Актуальна версія: KDE 5.0

`int document.attribute(int рядок, int стовпчик); int document.attribute(Cursor курсор);`

Повертає атрибут за вказаною позицією курсора.

`bool document.isAttribute(int рядок, int стовпчик, int атрибут); bool document.isAttribute(Cursor курсор, int атрибут);`

Повертає `true`, якщо атрибут за вказаною позицією курсора дорівнює значенню параметра *атрибут*. Якщо це не так, повертає `false`.

`String document.attributeName(int рядок, int стовпчик); String document.attributeName(Cursor курсор);`

Повертає назву атрибута придатну до читання. Ця назва відповідає назві `itemData` у файлах підсвічування синтаксичних конструкцій.

`bool document.isAttributeName(int рядок, int стовпчик, String назва); bool document.isAttributeName(Cursor курсор, String назва);`

Повертає `true`, якщо назва атрибута у певній позиції курсора відповідає вказаному значенню параметра *назва*. Якщо це не так, повертає `false`.

`String document.variable(String ключ);`

Повертає значення вказаної змінної документа *ключ*. Якщо змінної з вказаною назвою у документі не існує, повертає порожній рядок.

`void document.setVariable(String ключ, String значення);`

Встановлює значення відповідної змінної документа *ключ*.

Див. також змінні документа Kate

Актуальна версія: KDE 4.8

`int document.firstVirtualColumn(int рядок);`

Повертає віртуальний стовпчик першого відмінного від пробілу символу у вказаному рядку або `-1`, якщо рядок є порожнім або містить лише символи пробілів.

`int document.lastVirtualColumn(int рядок);`

Повертає віртуальний стовпчик останнього відмінного від пробілу символу у вказаному рядку або `-1`, якщо рядок є порожнім або містить лише символи пробілів.

`int document.toVirtualColumn(int рядок, int стовпчик); int document.toVirtualColumn(Cursor курсор); Cursor document.toVirtualCursor(Cursor курсор);`

Перетворює вказану «реальну» позицію курсора на віртуальну позицію курсора, повертає або ціле значення (int), або об'єкт Cursor.

`int document.fromVirtualColumn(int line, int virtualColumn); int document.fromVirtualColumn(Cursor virtualCursor); Cursor document.fromVirtualCursor(Cursor віртуальнийКурсор);`

Перетворює вказану віртуальну позицію курсора на «реальну» позицію курсора, повертає або ціле значення (int), або об'єкт Cursor.

`Cursor document.anchor(int рядок, int стовпчик, Char символ); Cursor document.anchor(Cursor курсор, Char символ);`

Виконує пошук у напрямку початку документа, починаючи від вказаної позиції курсора, вказаного символу. Наприклад, якщо функції буде передано символ «(», функція поверне позицію початкової дужки «(». Відповідність дужок не враховуватиметься, тобто інші «(...)» буде проігноровано.

`Cursor document.rfind(int рядок, int стовпчик, String текст, int атрибут = -1); Cursor document.rfind(Cursor курсор, String текст, int атрибут = -1);`

Виконує пошук у напрямку початку документа вказаного фрагмента тексту з відповідним значенням параметра *атрибут*. Параметр *атрибут* буде проігноровано, якщо він матиме значення `-1`. Функція поверне значення некоректного курсора, якщо фрагмент тексту знайти не вдасться.

`int document.defStyleNum(int рядок, int стовпчик); int document.defStyleNum(Cursor курсор);`

Повертає типовий стиль курсора, використаний за вказаною позицією у тексті.

`bool document.isCode(int рядок, int стовпчик); bool document.isCode(Cursor курсор);`

Повертає `true`, якщо атрибут за вказаною позицією курсора не дорівнює жодному зі значень стилів: `dsComment`, `dsString`, `dsRegionMarker`, `dsChar`, `dsOthers`.

`bool document.isComment(int рядок, int стовпчик); bool document.isComment(Cursor курсор);`

Повертає `true`, якщо значенням атрибута символу у позиції курсора є `dsComment`. Якщо це не так, повертає `false`.

`bool document.isString(int рядок, int стовпчик); bool document.isString(Cursor курсор);`

Повертає `true`, якщо значенням атрибута символу у позиції курсора є `dsString`. Якщо це не так, повертає `false`.

`bool document.isRegionMarker(int рядок, int стовпчик); bool document.isRegionMarker(Cursor курсор);`

Повертає `true`, якщо значенням атрибута символу у позиції курсора є `dsRegionMarker`. Якщо це не так, повертає `false`.

`bool document.isChar(int рядок, int стовпчик); bool document.isChar(Cursor курсор);`

Повертає `true`, якщо значенням атрибута символу у позиції курсора є `dsChar`. Якщо це не так, повертає `false`.

`bool document.isOthers(int рядок, int стовпчик); bool document.isOthers(Cursor курсор);`

Повертає `true`, якщо значенням атрибута символу у позиції курсора є `dsOthers`. Якщо це не так, повертає `false`.

`void document.indent(Range діапазон, int зміна);`

Встановити відступи у всіх рядка у діапазоні *діапазон* на *зміна* табуляцій або *зміна* разів на `tabSize` пробілів, залежно від визначених користувачем параметрів. Значення *зміна* може бути від'ємним.

6.4.3.5 Програмний інтерфейс редактора

Окрім програмного інтерфейсу документа та області перегляду, існує і загальний програмний інтерфейс редактора, який надає доступ до загальних функцій керування редактором за допомогою скриптів.

`String editor.clipboardText();`

Повертає текст, який зберігається у загальному буфері обміну даними.

Актуальна версія: KDE Frameworks 5.50

`String editor.clipboardHistory();`

Редактор зберігає журнал буфера обміну даними, який містить до 10 записів. Ця функція повертає усі записи, які зараз перебувають у журналі буфера обміну даними.

Актуальна версія: KDE Frameworks 5.50

`void editor.setClipboardText(String текст);`

Встановлює для вмісту буфера обміну даними значення *текст*. Запис *текст* буде додано до журналу буфера обміну даними.

Актуальна версія: KDE Frameworks 5.50

Розділ 7

Налаштування KatePart

Вибір пункту меню **Параметри** → **Налаштувати** *Назва програми*... відкриває діалогове вікно **Налаштування**. За допомогою цього діалогового вікна можна змінити різні параметри програми. Параметри, які користувач може змінити, впорядковано за категоріями, категорію можна обрати за допомогою вертикального списку у лівій частині діалогового вікна. За допомогою трьох кнопок, розташованих вздовж нижньої частини діалогового вікна, користувач може керувати процесом.

Ви можете викликати **Довідку**, застосувати поточні параметри і закрити діалогове вікно за допомогою кнопки **Гаразд**, або **Скасувати** зміни. Докладний опис категорій **Вигляд**, **Шрифти та кольори**, **Редагування**, **Відкрити/зберегти** та **Додатки** ви знайдете далі за текстом.

7.1 Налаштування компонента редактора

У цій групі містяться всі сторінки, що стосуються компонента редактора KatePart Більшість параметрів на цих сторінках є типовими, їх може бути перевизначено визначенням типу файлів, змінними документами, або змінити параметри для окремого документа під час сеансу редагування.

7.1.1 Вигляд

7.1.1.1 Загальні

Шрифт редактора

Тут ви можете обрати шрифт для тексту у редакторі. Ви можете обрати будь-який зі шрифтів доступних у системі і встановити типовий розмір. У нижній частині діалогового вікна буде показано текст прикладу, отже, ви зможете спостерігати за результатом вашого вибору.

Докладніший опис того, як вибрати шрифт, можна знайти у розділі Вибір шрифтів підручника з основ роботи у KDE.

Показувати помітки пробілів

Ніколи
Редактор ніколи не показуватиме крапки для позначення пробілів.

Наприкінці рядка
Редактор показуватиме крапки для позначення додаткових пробілів наприкінці рядка.

Завжди
Редактор завжди показуватиме крапки для позначення пробілів.

Розмір помітки пробілу
Скористайтеся повзунком для зміни розміру видимого маркера.

Показувати помітки табуляцій
Якщо позначено, редактор буде показувати символ », що позначає табуляцію у тексті.

Пошук і підсвічування парних дужок

Підсвічувати діапазон між вибраними дужками
Якщо увімкнено, діапазон між вибраними парами дужок буде підсвічено.

Показувати попередній перегляд відповідної відкритої дужки
Якщо позначено, редактор показуватиме підказку із відповідною до відкритої дужки.

Блимати відповідною дужкою, коли курсор потрапляє на парну до неї дужку
Якщо позначено, пересування дужок ({, [,], }, (або)) призводитиме до швидкого показу блимання на відповідній пересунутій дужці.

Показувати рядки з відступами
Якщо позначено цей пункт, у редакторі буде показано вертикальні лінії, які допоможуть вам визначати рядки з відступами.

Лічильники

Показувати кількість слів
Показує кількість слів і символів у документі та поточному позначеному фрагменті на смужці стану. Відповідним пунктом можна також скористатися у контекстному меню смужки стану.

Показувати кількість рядків
Показує загальну кількість рядків у документів на смужці стану. Відповідним пунктом можна також скористатися у контекстному меню смужки стану.

Згортати перший рядок
Якщо позначено, перший рядок згортатиметься, якщо це можливо. Це корисно, якщо файл починається з рядка коментаря, зокрема повідомлення щодо авторських прав.

Динамічне перенесення рядків
Якщо буде позначено цей пункт, рядки тексту буде перенесено на межі області перегляду на екрані.

Динамічно переносити на позначці статичного перенесення
Якщо позначено, редактор динамічно переноситиме рядки тексту на позиції статичного перенесення рядків.

Не брати до уваги межі слів при динамічному перенесенні рядків
Якщо позначено, редактор не братиме до уваги межі слів при перенесенні рядків тексту.

Помітки динамічного перенесення слів
За допомогою цього пункту ви зможете визначити умову, за якої має бути показано позначки динамічного перенесення рядків. Можливі варіанти: **Вимкнено**, **Разом з номерами** або **Завжди ввімкнено**.

Відступ для перенесених рядків
Крім того, за допомогою цього параметра ви можете встановити максимальну ширину у відсотках до ширини екрана, за досягнення якої динамічно перенесені рядки більше не будуть вертикально вирівнюватися. Наприклад, за значення у 50%, рядки, рівень відступу яких буде більшим за 50% ширини екрана, не буде вертикально вирівняно з наступними перенесеними рядками.

Множник висоти рядка
Це значення буде помножено на типову висоту рядка шрифту. Значення 1.0 означає, що буде використано типову висоту.

7.1.1.2 Поля

Згортання блоків коду

Показувати стрілки для згортання блоків коду

Якщо позначено цей пункт, у поточній області перегляду буде показано позначки згортання коду, якщо увімкнено згортання коду.

Попередній перегляд згорнутих блоків при наведенні

Якщо позначено цей пункт, у відповідь на наведення вказівника на згорнутий фрагмент документа програма показуватиме панель зі згорнутим фрагментом тексту з документа.

Видимість стрілок згортання

Перемкнути стрілочки згортання між **Показувати при наведенні** і **Показувати завжди**.

Лівий бік

Показувати позначки

Якщо позначено цей пункт, у лівій частині піктограми ви бачитимете рамку піктограми. Рамка піктограми може, наприклад, показувати позначки закладок.

Показати номери рядків

Якщо позначено цей пункт, ліворуч від тексту буде показано номери рядків.

Підсвічувати змінені і незбережені рядки

Якщо буде позначено цей пункт, програма показуватиме маркери змін у рядках. Щоб дізнатися більше, зверніться до розділу Розділ 3.9.

Смужки гортання

Показувати позначки

Якщо позначено цей пункт, у поточному перегляді на вертикальній смужці прокрутки буде показано позначки. Цими позначками буде відмічено, наприклад, закладки.

Показувати панель перегляду при наведенні на смужку гортання

Якщо позначено цей пункт, наведення вказівника миші на смужку гортання призводитиме до показу невеличкої панелі попереднього перегляду тексту із декількома рядками документа, які розташовано поруч із місцем у документі, якому відповідає розташування вказівника. Таким чином можна пришвидшити перехід до потрібної частини документа.

Мінікарта

Показувати мінікарту

Якщо буде позначено цей пункт, для кожної панелі перегляду на вертикальній панелі гортання буде показано загальну мінікарту документа.

Докладніший опис мінікарти на смузі гортання можна знайти у розділі Розділ 3.10.

Ширина мінікарти

Регулює ширину мінікарти на смужці гортання, визначається у пікселях.

Видимість смужок гортання

Перемкнути панель гортання у стан «увімкнути», «вимкнути» або «показувати, лише якщо потрібно». Наведіть вказівник миші на синій прямокутник і клацніть лівою кнопкою миші, щоб переглянути діапазон номерів рядків, показаний на екрані. Перетягніть вказівник миші, не відпускаючи лівої кнопки, щоб автоматично гортати вміст документа.

Впорядкувати меню закладок

За датою створення

Кожна закладка буде додаватися знизу, незалежно від місця в документі на яке вона вказує.

За номерами рядків

Закладки будуть впорядковані по номерах рядків на які вони вказують.

7.1.2 Теми кольорів

За допомогою цього розділу діалогового вікна ви можете налаштувати всі кольори у всіх темах кольорів, які у вас є, а також створити нові схеми, вилучити наявні або просто **Використати схему кольорів системи**. У кожній зі схем є параметри для кольорів і стилі для звичайного і підсвіченого тексту.

KatePart попередньо вибере поточну активну тему. Якщо ви бажаєте працювати над якоюсь іншою темою, почніть з вибору цієї теми зі спадного списку **Виберіть тему**. За допомогою кнопок **Копіювати** і **Вилучити** ви можете створити нову тему (скопіювавши одну з наявних) або вилучити наявну.

Докладний опис можна знайти у розділі Розділ 6.3.5.

7.1.3 Редагування

7.1.3.1 Загальні

Перенесення слів

Перенесення слів — це функція, яка змушує редактор автоматично починати новий рядок тексту і переносити курсор на початок цього нового рядка. KatePart буде автоматично розпочинати новий рядок тексту, коли поточний рядок досягне довжини, визначеної параметром Переносити слова на:.

Переносити слова на фіксованій позиції

Вмикає або вимикає статичне перенесення слів.

Малювати вертикальну лінію у позиції перенесення рядків

Якщо позначено цей пункт, у стовпчику перенесення слів буде намальовано вертикальну лінію, її розташування визначатиметься параметрами, вказаними у вікні, яке відкривається пунктом **Параметри** → **Налаштувати редактор...**, на вкладці Редагування. Будь ласка, зауважте, що помітку статичного перенесення слів буде показано, лише якщо ви використовуєте моноширинний шрифт.

Переносити слова на:

Якщо позначено пункт Переносити слова на фіксованій позиції, значення у цьому пункті визначить довжину (у символах) відступу, з якого редактор буде автоматично починати новий рядок.

Типовий режим введення

Вибраний режим введення буде увімкнено після відкриття нового вікна редагування. Ви можете вмикати або вимикати режим вводу vi для окремого вікна редагування за допомогою меню **Зміни**.

Дужки

Якщо позначено пункт **Автоматично закривати дужки, якщо введено початкову дужку**, після введення користувачем лівої дужки ([, (, або {) KatePart автоматично вводитимете праву дужку (},), або]) праворуч від курсора.

Символи-обмежувачі

Ви можете вибрати символи-обмежувачі за допомогою відповідного спадного списку.

Якщо позначено фрагмент тексту, введення одного з цих символів призводитиме до згортання позначеного фрагмента тексту.

Копіювання і вставлення

Пересувати позначений текст при перетягуванні

За допомогою цього пункту можна увімкнути або вимкнути перетягування зі скиданням для фрагментів позначеного тексту у вікні редактора.

Копіювати/Вирізати поточний рядок, якщо викликано без позначеного тексту

Якщо ви позначите цей пункт і у тексті не буде позначених фрагментів, дія з копіювання або вирізання виконуватиметься над рядком тексту, у якому перебуватиме курсор.

Не пересувати текстовий курсор при вставлянні фрагментів за допомогою миші

Якщо позначено цей пункт і ви вставляєте якийсь фрагмент тексту у вікно редактора за допомогою клацання середньою кнопкою миші, KatePart не пересуватиме текстовий курсор у позицію клацання.

7.1.3.2 Навігація текстом

Пересування текстового курсора

Кмітливий перехід до початку та кінця рядка

Якщо позначено цей пункт, натискання клавіші Home призводитиме до того, що курсор пропускатиме пробіли і переходитиме до початку тексту у новому рядку.

PageUp/PageDown пересувають курсор

За допомогою цього пункту можна змінити поведінку курсора після натискання користувачем клавіш **Page Up** або **Page Down**. Якщо пункт не позначено, курсор зберігатиме свою відносну позицію у вікні KatePart, у результаті дії буде пересунуто лише текст, який ви бачитимете на екрані. Отже, якщо курсор знаходився посередині видимого тексту, після виконання дії він залишиться на тому ж місці (хіба що пересунеться у межах рядка, якщо в ньому недостатньо символів). Якщо ж цей пункт буде позначено, за натискання вказаних клавіш курсор буде пересунуто на верхній або нижній рядок видимого тексту нової сторінки, яку буде показано в результаті прогортання тексту на один екран вниз або вгору.

Увімкнути перестрибування до великих літер

За допомогою цього пункту можна змінити поведінку курсора у відповідь на натискання клавіатурних скорочень **Ctrl**-← та **Ctrl**-→. Якщо пункт не позначено, курсор перестрибуватиме через цілі слова. Якщо пункт позначено, курсор зупинятиметься на великих літерах у словах, які написано літерами різних регістрів.

Автоцентрування курсора:

Встановлює, за можливості, кількість видимих рядків над та під курсором.

Режим вибору тексту

Звичайний

Вибрану ділянку буде заміщено набраним текстом, і її буде втрачено при переміщенні курсора.

Стійкий

Вибір залишається навіть після пересування курсора та вводу тексту.

Дозволити гортання за кінець документа

За допомогою цього пункту можна уможливити гортання за кінець документа. Ним можна скористатися, якщо ви бажаєте розташувати по центру у вертикальному напрямку кінцеву частину документа, або пересунути її у верхню частину вікна редактора.

Клавіша Backspace вилучає основу символу разом із діакритичними знаками

Якщо позначено, замість основи складеного символу вилучатиметься увесь символ, разом із діакритичними позначками. Корисно для індійських локалей.

Модифікатор мультикурсора

За допомогою цього пункту ви можете налаштувати модифікатор, який буде використано для створення мультикурсора у відповідь на клацання лівою кнопкою миші. Вам слід буде натиснути модифікатор і клацнути лівою кнопкою миші, щоб створити курсор у бажаному місці. Див. створення декількох курсорів, щоб дізнатися про інші способи створення декількох курсорів.

7.1.3.3 Відступ

Типовий режим відступу:

Виберіть автоматичний режим відступу, який ви бажаєте використати як типовий. Наполегливо рекомендуємо вам скористатися режимом `Не вибрано` або `Звичайний`, а потім скористатися налаштуваннями типу файлів, щоб встановити інші режими відступу для форматів тексту на зразок коду C/C++ або XML.

Створювати відступ за допомогою

Табулятори

Якщо буде позначено цей пункт, редактор додаватиме символи табуляції у відповідь на натискання клавіші **Tab** або під час автоматичного встановлення відступів.

Пробіли

Якщо позначено цей пункт, редактор вставлятиме обчислену кількість пробілів, відповідно до поточної позиції у тексті та параметра `tab-width`, коли ви натискатимете клавішу **Tab** або під час автоматичного визначення відступів.

Табулятори і пробіли

Якщо буде позначено цей пункт, редактор додаватиме пробіли на початку рядка, як це описано вище, під час встановлення відступу або у відповідь на натискання **Tab**, але вставлятиме символи табуляції, якщо клавішу **Tab** натиснуто всередині або наприкінці рядка.

Ширина табуляції:

За допомогою цього пункту можна визначити кількість пробілів, які буде показано замість символу табуляції.

Ширина відступу:

Ширина відступу — це кількість пробілів, які буде використано для відступу у рядку. Якщо налаштовано встановлення відступів за допомогою символів табуляції, символ табуляції буде вставлено, якщо відступи поділяються за допомогою ширини табуляції.

Властивості відступу

Залишати додаткові пробіли

Якщо буде знято позначку з цього пункту, зміна рівня відступу вирівнюватиме рядок до відступу, довжина якого кратна до значення, вказаного у пункті **Ширина відступу**.

Коригувати відступ для тексту, вставленого з буфера обміну даними

Якщо позначено цей пункт, у тексті, який ви вставлятимете з буфера обміну інформацією, буде встановлено відступи. Виконання дії **Вернути** вилучить ці відступи.

Дії відступу

Клавіша Backspace зменшує рівень відступу, якщо перший символ — пробіл

Якщо позначено цей пункт, клавіша **Backspace** зменшуватиме рівень відступу, якщо курсор розташовано на початкових пробілах рядка.

Режим табуляції (якщо нічого не вибрано)

Якщо ви бажаєте, щоб клавіша **Tab** вирівнювала поточний рядок у поточному блоці коду так, як це робиться у emacs, зробіть **Tab** скороченням для дії **Форматований відступ**.

Завжди переходити на наступної позиції табуляції

Якщо позначено цей пункт, клавіша **Tab** завжди вставлятиме пробіл до наступної позиції табуляції. Якщо позначено пункт **Використовувати пробіли замість табуляції** на вкладці **Загальні** сторінки **Редагування**, буде вставлено пробіли, у іншому випадку буде вставлено один символ табуляції.

Завжди збільшувати рівень відступу

Якщо позначено цей пункт, клавіша **Tab** завжди робитиме відступ у поточному рядку на кількість позицій символів, вказану у параметрі **Ширина відступу**.

Збільшувати рівень відступу, якщо перший символ — пробіл

Якщо позначено цей пункт, клавіша **Tab** або робитиме відступ на поточному рядку, або переводитиме курсор на наступну позицію табуляції. Якщо позиція, у яку ви вставляєте символ, знаходиться на або перед першим символом рядка, який не є пробілом, або якщо вибрано ділянку тексту, у поточний рядок буде додано відступ на кількість позицій символів, визначену параметром **Ширина відступу**. Якщо позиція, у яку ви вставляєте символ, знаходитиметься за першим символом рядка, який не є пробілом, і нічого не виділено, буде вставлено пробіл до наступної позиції табуляції. Якщо позначено пункт **Використовувати пробіли замість табуляції** на вкладці **Загальні** сторінки **Редагування**, буде вставлено відповідну кількість пробілів, у іншому випадку буде вставлено один символ табуляції.

7.1.3.4 Автозавершення

Загальне

Увімкнути автозавершення

Якщо буде позначено цей пункт, програма показуватиме під час введення слів список можливих варіантів завершення слова.

Автоматично вибирати перший варіант доповнення

Якщо позначено, програма завжди вибиратиме перший із варіантів автоматичного доповнення. Ви зможете вставити цей варіант до тексту натисканням клавіші **Enter**. Якщо ви не хочете, щоб такий вибір здійснювався автоматично, наприклад, завжди хочете вставляти за допомогою **Enter** розрив рядка, зніміть позначку з цього пункту.

Мінімальна довжина слова для завершення

Під час введення тексту інструмент автоматичного завершення слів шукатиме слова у документі, початок яких збігається з вже введеною частиною нового слова. За допомогою цього пункту можна вказати мінімальну кількість символів, у разі введення яких буде задіяно автоматична завершення слів і відкрито панель зі списком варіантів.

Вилучати кінець під час доповнення

Вилучати кінець попереднього слова, якщо вибрано пункт автодоповнення зі списку.

Автозавершення ключових слів

Якщо позначено, вбудований засіб автоматичного доповнення використовуватиме ключові слова, визначені засобом підсвічування синтаксичних конструкцій.

7.1.3.5 Перевірка правопису

Ці параметри налаштування докладно описано у документації до модуля Системних параметрів з назвою Перевірка правопису.

7.1.3.6 Режим вводу VI

Загальне

Пріоритет команд Vi перед скороченнями Kate

Якщо позначено цей пункт, команди Vi перевизначатимуть вбудовані команди KatePart. Наприклад, **Ctrl**+**R** виконуватиме дію з повторення скасованої дії, а не стандартну дію (відкриття діалогового вікна пошуку з заміною).

Показувати відносні номери рядків

Якщо позначено цей пункт, номером поточного рядка завжди вважатиметься 0. Номери рядків нижче і вище обчислюватимуться відносно цього рядка.

Схема відповідності клавіш

Схеми відповідності клавіш використовуються для зміни значення введених після натискання клавіш символів. Таким чином, ви можете змінити відповідність між натисканням клавіш і командами або створити відповідність між натисканням клавіші і послідовністю команд.

Приклад:

F2 -> `I-- ` **Esc**

Таким чином, до рядка буде додано `I-- ` у відповідь на натискання **F2**.

7.1.4 Відкриття/Збереження

7.1.4.1 Загальні

Формат файла

Кодування

Визначає стандартне кодування для відкриття/збереження файлів, якщо кодування не буде змінено за допомогою діалогового вікна відкриття/збереження або параметра командного рядка.

Визначення кодування

Оберіть цей пункт зі спадного списку, щоб або вимкнути автовиявлення або скористатися **Універсальним** для вмикання автоматичного визначення всіх кодувань. Але алгоритм, ймовірно, зможе визначити лише кодування utf-8/utf-16, вибір діапазону кодувань покращить евристичні здатності програми і дасть кращі результати. Якщо ні кодування, визначене як стандартне вище, ні кодування вказане у діалоговому вікні відкриття, ні кодування вказане у командному рядку не відповідатиме даним, що зберігаються у файлі, буде виконано цю процедуру визначення кодування.

Резервне кодування

Визначає резервне кодування для спроб відкриття файлів, якщо кодування, визначене як стандартне вище, вказане за допомогою діалогового вікна відкриття/збереження або командного рядка, не відповідає вмісту файла. Перш ніж виконувати спробу зміни кодування модуль Kate спробує визначити кодування читанням можливої позначки порядку байтів на початку файла. Якщо буде виявлено таку позначку, програма обере належне кодування Unicode. Якщо позначки не буде виявлено і кодування не вдасться визначити автоматично, буде використано резервне кодування.

Кінець рядка

Тут можна обрати потрібний режим завершення рядків для вашого активного документа. Ви можете обрати один з варіантів: UNIX®, DOS/Windows® або Macintosh.

Автоматичне визначення кінця рядка

Позначте цей пункт, якщо ви бажаєте, щоб редактор автоматично визначав тип кінця рядка. Для всього файла буде використано перший зі знайдених типів кінця рядка.

Увімкнути позначку порядку байтів (ППБ)

Позначка порядку байтів (ППБ) — це символ Unicode, який додають на початку документів у кодуванні Unicode. За її допомогою програми для редагування визначають порядок байтів текстового файла. Докладніше про неї можна дізнатися зі статті Позначка порядку байтів.

Верхня межа довжини рядка

На жаль, через недоліки у Qt™ KatePart працює з дуже довгими рядками надзвичайно повільно. Через це у KatePart передбачено автоматичне розбиття рядків, кількість символів у яких перевищує вказане у цьому пункті значення. Щоб вимкнути розбиття, встановіть значення `0`.

Автоматичне очищення під час збереження

Вилучати кінцеві пропуски

Редактор може автоматично вилучати зайві пробіли наприкінці рядків під час збереження файла. Ви можете вибрати режим **ніколи**, щоб вимкнути цю можливість, режим **у змінених рядках**, щоб цю дію було виконано лише над рядками, до яких було внесено зміни з часу останнього збереження документа або режим **у всьому документі**, щоб зайві пробіли було безумовно вилучено у всіх рядках документа.

Додавати символ нового рядка наприкінці файла під час збереження

За позначення цього пункту редактор автоматично додаватиме символ нового рядка наприкінці файла, якщо його не було на час збереження файла.

Увімкнути автоматичне збереження (лише для локальних файлів)

Позначте цей пункт, якщо ви хочете, щоб редактор автоматично зберігав документи, поки ви над ними працюєте.

Автоматично зберігати документи, коли редактор втрачає фокусування

Редактор автоматично зберігатиме документи, коли ви перемикатиметеся на компоненти, наприклад панель термінала у Kate.

Інтервал автозбереження

Тут ви можете визначити інтервал автоматичного збереження у секундах. Якщо буде вказано інтервал 0, документ не буде зберігатися із регулярними інтервалами.

7.1.4.2 Додатково

Записувати файл резервної копії при зберіганні

Створення резервної копії під час збереження призведе до того, що KatePart копіюватиме дисковий файл (попередньо збереженій версії файл) у файл з назвою <префікс><назва файла><суфікс>, перш ніж зберігати файл зі старою назвою. Файл резервної копії може бути корисним для відновлення результатів роботи, якщо щось піде не так під час зберігання або ви захочете пізніше відновити попередню версію файла. Типовим суфіксом є `~`, типовий префікс — порожній.

Локальні файли

Позначте цей пункт, якщо ви бажаєте робити резервні копії локальних файлів під час збереження.

Віддалені файли

Позначте цей пункт, якщо ви бажаєте робити резервні копії файлів на віддалених комп'ютерах під час збереження.

Префікс для файлів резервних копій

Введіть префікс для назв файлів резервних копій.

Суфікс для файлів резервних копій

Введіть суфікс, який буде додано до файлів резервних копій.

Режим файла резервної пам'яті

Програма KatePart здатна відновлювати (більшу) частину незбереженої роботи у разі аварійного завершення або вимикання живлення. Під час редагування документа програма автоматично створює файл свопінгу (.<filename>.kate-swp). Якщо користувачем не буде збережено зміни, а KatePart аварійно завершить роботу, файл свопінгу залишиться на диску. Під час відкриття файла KatePart виконує пошук файла свопінгу документа. Якщо файл буде виявлено, програма спитає користувача про те, чи хоче він відновлювати втрачені дані. Користувач матиме можливість переглянути відмінності між початковим та відновленим файлом. Файл свопінгу вилучається після кожного збереження документа або штатного завершення роботи програми.

KatePart синхронізує файли свопінгу на диску кожні 15 секунд, але лише якщо до документа було внесено зміни з часу останньої синхронізації. Користувач може вимкнути синхронізацію файлів. Для цього достатньо вибрати пункт **Вимкнути**. Вимикання синхронізації може призвести до втрати даних.

Якщо увімкнено файл резервної пам'яті, можна перемикатися між двома режимами, а саме, **Увімкнено, зберігати у типовому каталозі** і **Увімкнено, зберігати у нетиповому каталозі**.

Зберігати файли резервної пам'яті у

Типово, файли резервної пам'яті зберігатимуться до тієї самої теки, де зберігається файл. Якщо вибрано **Увімкнено, зберігати у нетиповому каталозі** для режиму файлів резервної пам'яті, файли резервної пам'яті зберігатимуться у вказаній теці. Корисно для використання у мережевих файлових системах для запобігання зайвому обміну даними мережею.

Зберігати файли резервної пам'яті кожні

KatePart синхронізує файли свопінгу на диску кожні 15 секунд, але лише якщо до документа було внесено зміни з часу останньої синхронізації. Ви можете визначити інший проміжок між послідовними синхронізаціями даних.

7.1.4.3 Режими і типи файлів

За допомогою цієї сторінки ви можете змінити типове налаштування для документів вказаного типу MIME. Після завантаження документа у редактор програма намагатиметься встановити відповідність між даним файлом і визначеним шаблоном файлів або типом MIME зі списку, і якщо відповідність буде знайдено, застосує визначені змінні. Якщо буде знайдено декілька відповідностей з типами файлів, використовуватиметься тип файлів з найвищим пріоритетом.

Тип файла:

Типом файлів з найвищим пріоритетом є той тип, який знаходиться на верхівці у спадному списку. Якщо було виявлено належність до декількох типів, їх також буде показано у списку.

Створити

Ця кнопка використовується для створення нового типу файлів. Після натискання цієї кнопки розташовані нижче поля буде спорожнено, і ви зможете записати до них властивості нового типу файлів.

Вилучити

Щоб вилучити існуючий тип файлів, виберіть його зі спадного списку і натисніть кнопку «Вилучити».

Властивості *поточного типу файлів*

Типом файлів з найвищим пріоритетом є той тип, який знаходиться на верхівці у спадному списку. Якщо було виявлено належність до декількох типів, їх також буде показано у списку.

Назва:
Назва типу файлів буде текстом відповідного пункту меню. Цю назву буде показано у списку підменю **Інструменти** → **Типи файлів**

Розділ:
Назва розділу використовується для впорядкування типів файлів у меню. Його також буде використано у підменю **Інструменти** → **Типи файлів**.

Змінні:
За допомогою цього рядка ви можете налаштувати параметри KatePart для файлів, які належать до цього типу MIME за допомогою змінних KatePart. Ви можете встановити майже будь-який параметр налаштування, зокрема підсвічування, режим відступів тощо.

Натисніть кнопку **Змінити**, щоб переглянути список всіх доступних змінних та описи цих змінних. Позначте пункт, щоб увімкнути певну змінну. Значення змінної можна встановити праворуч. Для деяких змінних передбачено спадний список можливих значень, для інших вам доведеться вказати значення вручну.

Докладніше про ці змінні можна дізнатися з розділу Налаштування зі змінними документа.

Підсвічування:
Якщо ви створите новий тип файлів, за допомогою цього спадного списку ви зможете обрати для нього тип файлів для підсвічування.

Режим відступів:
За допомогою цього спадного списку можна визначити режим додавання відступів у нових документах.

Суфікси файлів:
За допомогою шаблонів ви можете обирати файли за назвою. Типовим шаблоном заміни є зірочка з суфіксом файла, наприклад `*.txt; *.text`. Окремі шаблони у рядку слід відокремлювати крапкою з комою.

Типи MIME:
Показує майстер, за допомогою якого можна дуже просто вибирати типи MIME.

Пріоритет:
Встановлює пріоритет файлів цього типу. Якщо файл може бути віднесено до декількох типів, буде використано тип з найвищим пріоритетом.

7.2 Налаштування змінних документа

Змінні KatePart реалізовано у KatePart як змінні документа, подібно до рядка режимів у emacs і vi. У katepart, рядки мають формат `kate: НАЗВА_ЗМІННОЇ ЗНАЧЕННЯ; [ НАЗВА _ЗМІННОЇ ЗНАЧЕННЯ; ... ]`. Рядки можуть, звичайно ж, бути коментарями, якщо файл має формат, у якому передбачено коментарі. Назви змінних мають бути цілими словами (без пробілів), наступна частина рядка, до наступної крапки з комою, є значенням змінної. Використання крапки з комою є обов'язковим.

Ось приклад рядка змінних, який вмикає примусове встановлення відступів для файлів C++, java або javascript:

```
// kate: replace-tabs on; indent-width 4; indent-mode cstyle;
```

> **Примітка**
> Програма шукатиме рядки змінних лише у перших і останніх 10 рядках документа.

Крім того, змінні документа можуть зберігатися у файлі з назвою `.kateconfig` у будь-якому каталозі. Значення параметрів з цього файла буде застосовано у разі введення рядків режиму у будь-якому з файлів у відповідному каталозі та його підкаталогах. Для змінних документа у `.kateconfig` слід використовувати ті самі синтаксичні правила, що і для рядків режиму, але із додатковими параметрами.

Існують змінні для визначення майже всіх параметрів налаштування KatePart, крім того, змінними можуть користуватися додатки, у останньому випадку ці змінні має бути описано у документації до додатка.

У KatePart передбачено підтримку читання налаштувань з файлів `.editorconfig`, якщо встановлено бібліотеку editorconfig. KatePart автоматично шукає `.editorconfig` під час відкриття файла. Втім, перевага надається файлам `.kateconfig`.

7.2.1 Як KatePart використовує змінні

Під час читання налаштування, katepart шукатиме змінні у таких місцях (у вказаному порядку):

- Загальне налаштування.
- Додаткові дані сеансу.
- Налаштування типу файлів.
- Змінні документа у файлі `.kateconfig`.
- Змінні документа у самому документі.
- Параметри, вказані під час редагування за допомогою меню або командного рядка.

Як ви вже зрозуміли, змінні документа мають наступний за найвищим пріоритет. Кожного разу, коли документ зберігатиметься, змінні документа буде повторно прочитано, вони перевизначать зміни, внесені за допомогою пунктів меню або командного рядка.

Будь-яка змінна, якої немає у наведеному нижче списку, і яка зберігається в документі, запит на яку може бути надіслано іншими об'єктами, такими як додатки, які можуть використовувати ці додатки з власною метою. Наприклад, у режимі відступів зі змінними змінні документа буде використано для налаштування режиму.

Список змінних, наведених тут, стосується версії 5.38 KatePart. У майбутніх версіях може бути додано нові змінні. Існує 3 можливих типи значень змінних, які приймають наступні значення:

- BOOL — on|off|true|false|1|0
- INTEGER — будь-яке ціле число
- STRING — всі інші змінні

7.2.2 Можливі змінні

`auto-brackets` **[BOOL]**

Увімкнути автоматичне вставлення дужок.

`auto-center-lines` **[INT]**

Встановлює кількість автоматично вирівняних по центру рядків.

`background-color` **[STRING]**

Встановлює колір тла документа. Значенням має бути рядок, який можна інтерпретувати яка колір, наприклад, `#ff0000`.

`backspace-indents` **[BOOL]**

Увімкнути або вимкнути вилучення відступів у відповідь на натискання **Backspace**.

`block-selection` **[BOOL]**

Вмикає або вимикає режим прямокутного виділення.

`bom | byte-order-mark | byte-order-marker` **[BOOL]**

Увімкнути або вимкнути позначення порядку байтів (ППБ) під час збереження файлів у форматі Unicode (utf8, utf16, utf32).

Актуальна версія: Kate 3.4 (KDE 4.4)

`bracket-highlight-color` **[STRING]**

Встановлює колір підсвічування дужок. Значенням має бути рядок, який можна інтерпретувати як коректний колір, наприклад `#ff0000`.

`current-line-color` **[STRING]**

Встановлює колір поточного рядка. Значенням має бути рядок, який можна інтерпретувати як коректний колір, наприклад `#ff0000`.

`default-dictionary` **[STRING]**

Визначає типовий словник, який буде використано для перевірки правопису.

Актуальна версія: Kate 3.4 (KDE 4.4)

`dynamic-word-wrap` **[BOOL]**

Вмикає або вимикає динамічне перенесення рядків.

`eol | end-of-line` **[STRING]**

Визначає режим використання символу кінця рядка. Можливі значення: `unix`, `mac` і `dos`

`folding-markers` **[BOOL]**

Вмикає або вимикає показ позначок згортання.

`folding-preview` **[BOOL]**

Увімкнути перегляд згорнутого на межі поля редагування.

`font-size` **[INT]**

Встановлює розмір у точках шрифту документа.

`font` **[STRING]**

Встановлює гарнітуру шрифту поточного документу. Значенням має бути чинна назва шрифту, наприклад `courier`.

`hl | syntax` **[STRING]**

Встановити підсвічування синтаксису. Коректними рядками є всі пункти відповідних меню. Наприклад, для C++ достатньо вказати `C++`.

`icon-bar-color` **[STRING]**

Встановлює колір смужки піктограми. Значенням має бути рядок, який можна інтерпретувати як коректний колір, наприклад, `#ff0000`.

`icon-border` **[BOOL]**

Вмикає або вимикає показ смужки піктограм.

`indent-mode` **[STRING]**

Встановлює режим автоматично встановлення відступу. Можна використовувати значення `none`, `normal`, `cstyle`, `haskell`, `lilypond`, `lisp`, `python`, `ruby` і `xml`. Докладніше про це можна дізнатися з розділу Розділ 3.8.

`indent-pasted-text` **[BOOL]**

Увімкнути або вимкнути коригування відступу для тексту, вставленого з буфера обміну даними.

Актуальна версія: Kate 3.11 (KDE 4.11)

`indent-width` **[INT]**

Встановлює ширину відступу.

`keep-extra-spaces` **[BOOL]**

Визначає, чи буде враховано існуючі пробіли під час обчислення ширини відступу.

`line-numbers` **[BOOL]**

Вмикає або вимикає показ номерів рядків.

`newline-at-eof` **[BOOL]**

Додати порожній рядок наприкінці файла (EOF) під час збереження документа.

Актуальна версія: Kate 3.9 (KDE 4.9)

`overwrite-mode` **[BOOL]**

Вмикає або вимикає режим перезапису (заміни).

`persistent-selection` **[BOOL]**

Вмикає або вимикає режим стійкого вибору.

`replace-tabs-save` **[BOOL]**

Вмикає або вимикає перетворення табуляції у пробіли під час збереження документа.

`replace-tabs` **[BOOL]**

Вмикає або вимикає динамічне перетворення табуляції у пробіли.

`remove-trailing-spaces` **[РЯДОК]**

Вилучити кінцеві пробіли під час збереження документа. Можливі аргументи:

- `не вилучати`, `-` or `0`: не вилучати пробіли наприкінці рядків.
- `змінені`, `mod`, `+` або `1`: вилучити лише пробіли наприкінці змінених рядків. Змінені рядки буде визначено за допомогою системи визначення змінених рядків, вбудованої до програми.
- `всі`, `*` або `2`: вилучити пробіли наприкінці рядків у всьому документі.

`scrollbar-minimap` **[BOOL]**

Мінікарта на смужці гортання.

`scrollbar-preview` **[BOOL]**

Перегляд на смужці гортання.

`scheme` **[STRING]**

Встановлює схему кольорів. Щоб змінна запрацювала, її значенням має бути рядок назви схеми кольорів, яка існує у ваших налаштуваннях.

`selection-color` **[STRING]**

Встановлює колір виділеного тексту. Значенням має бути рядок, який можна інтерпретувати як коректний колір, наприклад, `#ff0000`.

`show-tabs` **[BOOL]**

Вмикає або вимикає візуалізацію символів табуляції.

`smart-home` **[BOOL]**

Вмикає або вимикає кмітливу навігацію документом за допомогою клавіші Home.

`tab-indents` **[BOOL]**

Вмикає або вимикає створення відступів клавішею **Tab**.

`tab-width` **[INT]**

Визначає показану ширину відступу табуляції.

`undo-steps` **[INT]**

Встановлює кількість дій, які записуватимуться до журналу скасування дій.

Зауваження: у Kate 3 (KDE 4) вважається застарілою. Цю змінну буде проігноровано. Максимальну кількість дій зі скасування нічим не обмежено.

`word-wrap-column` **[INT]**

Встановлює ширину статичного перенесення рядків.

`word-wrap-marker-color` **[STRING]**

Встановлює колір позначки перенесення рядків. Значенням має бути рядок, який можна інтерпретувати як коректний колір, наприклад `#ff0000`.

`word-wrap` **[BOOL]**

Вмикає або вимикає статичне перенесення рядків.

7.2.3 Додаткові параметри у файлах `.kateconfig`

KatePart завжди шукає файл `.kateconfig` для локальних файлів (не файлів, які зберігаються на віддалених ресурсах). Крім того, можна встановити параметри на основі замінників (для суфіксів назв файлів), ось так:

```
kate: tab-width 4; indent-width 4; replace-tabs on;
kate-wildcard(*.xml): indent-width 2;
kate-wildcard(Makefile): replace-tabs off;
```

У цьому прикладі в усіх файлах використовуватиметься табуляція у 4 пробіли, ширина абзацного відступу у 4 пробіли, а табуляції буде замінено на відповідну кількість пробілів. Втім для усіх файлів `*.xml` буде встановлено ширину абзацного відступу у 2 пробіли. У файлах Makefile використовуватимуться табуляції, тобто їх не буде замінено на пробіли.

Замінники слід відокремлювати один від одного крапкою з комою. Інакше кажучи, декілька суфіксів файлів можна вказати ось так:

```
kate-wildcard(*.json;*.xml): indent-width 2;
```

Крім того, ви можете скористатися типами MIME для визначення певних категорій файлів. Наприклад, щоб встановити у всіх файлах коду C++ відступ у 4 пробіли, можна скористатися таким записом:

```
kate-mimetype(text/x-c++src): indent-width 4;
```

> **Примітка**
> Окрім підтримки визначення у файлах `.kateconfig`, замінники і залежні від типу MIME змінні документів можна визначати і у самих файлах коду у коментарях.

Розділ 8

Подяки і ліцензування

Код програми засновано на початковому коді KWrite, авторські права на який належали Jochen Wilhelmy digisnap@cs.tu-berlin.de, ©2000

Учасники розробки:

- Christoph Cullmann cullmann@kde.org
- Michael Bartl michael.bartl1@chello.at
- Phlip phlip_cpp@my-deja.com
- Anders Lund anders@alweb.dk
- Matt Newell newellm@proaxis.com
- Joseph Wenninger kde@jowenn.at
- Jochen Wilhelmy digisnap@cs.tu-berlin.de
- Michael Koch koch@kde.org
- Christian Gebauer gebauer@kde.org
- Simon Hausmann hausmann@kde.org
- Glen Parker glenebob@nwlink.com
- Scott Manson sdmanson@altel.net
- John Firebaugh jfirebaugh@kde.org
- Nibaldo González nibgonz@gmail.com

Документацію до KatePart засновано на початковій версії документації до KWrite. Зміни вносилися лише там, де це потрібно для опису усіх можливих варіантів застосування KatePart.

Автором початкової версії документації до KWrite є Thad McGinnis ctmcginnis@compuserve.com. Значні зміни було внесено Christian Tibirna tibirna@kde.org. Перетворення документації у формат DocBook та виправлення помилок виконано Lauri Watts lauri@kde.org. Документацію було оновлено Anne-Marie Mahfouf annma@kde.org та Anders Lund anders@alweb.dk

Супровід поточної версії документації до KatePart здійснює T.C. Hollingsworth tchollingsworth@gmail.com. Щоб створити запит щодо змін, надішліть повідомлення щодо вади за допомогою системи стеження за вадами KDE або надайте латку на сторінці нашого проєкту у GitLab.

Переклад українською: Юрій Чорноіван yurchor@ukr.net

Розділ 9

Режим введення VI

Erlend Hamberg
Переклад українською: Юрій Чорноіван

9.1 Режим введення VI

Метою використання режиму VI є не повна заміна Vim і підтримка всіх можливостей Vim. Його метою є використання способу редагування текстів Vim, — та вивчених прийомів користування Vim, — у програмах, які використовують текстовий редактор KatePart для вбудованих режимів редагування.

Режим VI чудово інтегрується у сторонні програми і відхиляється від поведінки Vim там, де це має сенс. Наприклад, команда `:w` у режимі VI KatePart відкриває діалогове вікно збереження файла.

Увімкнути режим VI для всіх нових панелей редагування можна за допомогою пункту меню **Параметри** → **Налаштувати KatePart...**+**Редагування** → **Режим вводу VI**. На цій вкладці ви зможете встановити параметри роботи режиму введення VI, ви значити або змінити прив'язки клавіш у цьому режимі. Увімкнути або вимкнути режим введення VI можна також за допомогою пункту **Режим вводу VI** меню **Зміни**. (Типовим клавіатурним скороченням є **Meta**+**Ctrl**+**V**, де **Meta** зазвичай відповідає клавіша **Windows**).

> Примітка
> На відміну від більшості клавіатурних скорочень KDE, багато клавіатурних команд режиму Vi є залежними від регістру символів. Це означає, що команди `y` і `Y` мають зовсім різне призначення. Щоб ввести команду `y` (копіювати), переконайтеся, що режим **Caps Lock** (літер верхнього регістру) вимкнено і натисніть клавішу **Y**. Щоб ввести команду `Y` (копіювати до кінця рядка), скористайтеся комбінацією клавіш **Shift**+**Y**.
> Висловлене вище зауваження не стосується команд, у яких використовується клавіша. Ці команди можна ввести у будь-якому з режимів **Caps Lock** без натискання **Shift**, але у частині команд використовується комбінація з **Ctrl**, після якої слід натиснути іншу клавішу, регістр якої слід брати до уваги. Наприклад, щоб ввести команду «**Ctrl**+**W**, `h`» (перемкнутися на праву панель у розділеному перегляді), переконайтеся що режим літер верхнього регістру вимкнено, натисніть комбінацію клавіш **Ctrl**+**W**, відпустіть клавіш, а потім натисніть клавішу **H**.

9.1.1 Несумісності з Vim

З Vim несумісні лише декілька можливостей режиму VI KatePart, якщо не брати до уваги багатьох речей, які просто не реалізовано. Нижче наведено список цих можливостей з поясненнями причин несумісності.

- KatePart: `U` і **Ctrl**+**R** відповідають повторному виконанню скасованої команди.

  Vim: **Ctrl**+**R** звичайне повторення дії, **U** призначено для скасування всіх останніх змін у одному рядку.

  Причиною того, що **U** прив'язано до повторення дії у режимі VI KatePart, є те, що клавіатурне скорочення **Ctrl**+**R** типово зайнято у KatePart заміною (пошук з заміною). Типово режим VI не змінює клавіатурні скорочення KatePart (ви можете зробити це вручну за допомогою сторінки **Параметри** → **Налаштувати KatePart...**+**Редагування** → **Режим вводу Vi**), тому дія з повернення редагування має бути доступна у «звичайному» режимі натискання комбінацій клавіш. Окрім того, дія команди `U` у Vim не дуже добре збігається з внутрішньою системою скасування KatePart, отже її підтримка є доволі складним завданням.

- KatePart: `print` відкриває діалогове вікно **Друк**.

  Vim: `print` друкує рядки вказаного діапазону, подібно до свого дідуся, ed.

  Команди на зразок `:print` доступні не лише у режимі VI, але і користувачам «звичайних» режимів KatePart. Тому `:print` відкриває діалогове вікно друку відповідно до принципів однорідності, замість імітації поведінки Vim.

- KatePart: `Y` копіює дані до кінця рядка.

  Vim: `Y` копіює весь рядок, подібно до команди `yy`.

  Поведінка VI для команди `Y` насправді є вадою. Для обох команд, зміни і вилучення, команди `cc`/ `dd` виконують дію на поточному рядку, а `C`/`D` виконує дію над даними від позиції курсора до кінця рядка. Але обидві команди, `yy` і `Y`, копіюють поточний рядок. У режимі VI KatePart `Y` копіює дані до кінця рядка. Така поведінка описана як «логічніша» у документації до Vim.

- KatePart: `O` і `o` відкривають [*кількість*] нових рядків і переводять програму у режим вставлення.

  Vim: `O` і `o` відкривають нових рядок і вставляють текст [*кількість*] разів з виходом з режиму вставлення.

  Так зроблено в результаті узагальнення досвіду багатьох людей, які висловлювали здивування поведінкою Vim на каналі vim у IRC (#vim на Libera Chat).

9.1.2 Перемикання режимів

- *Звичайний режим* надає вам змогу вводити команди навігації або редагування документа і є типовим. Повернутися до цього режиму з будь-якого іншого режиму можна натисканням клавіші **Esc**.
- *Візуальний режим* надає вам змогу позначати текст у документі. У цьому режимі можна користуватися більшістю команд звичайного режиму. Ви можете перевести команду у цей режим введенням літери `v` для позначення символів або `V` для позначення рядків.
- *Режим вставлення* надає вам змогу редагувати документ безпосередньо. Ви можете перевести програму у цей режим натисканням клавіші `i` або за допомогою одної з декількох інших команд, наведених нижче.
- *Режим команд* відкриває панель командного рядка KatePart. За допомогою цієї панелі ви зможете виконувати значну частину команд, доступних у реалізаціях Vi, а також деякі специфічні для KatePart команди. Докладніше про ці команди можна дізнатися з розділу Розділ 5.2. Користуватися режимом команд просто: натисніть клавішу **:**, вкажіть команду і натисніть клавішу **Enter**.

9.1.3 Інтеграція з командами Kate

- Програма автоматично перемикатиметься на візуальний режим, якщо фрагмент тексту буде позначено за допомогою миші. Також перемикання відбуватиметься у разі використання функцій Kate, які позначають фрагменти тексту, зокрема «Вибрати все» (вибір або за допомогою меню, або за допомогою комбінації клавіш **Ctrl**+**A**.)

- Передбачено підтримку позначок Vi та закладок Kate. У разі створення позначки у режимі Vi буде також створено відповідну закладку Kate, яку буде показано у меню **Закладки**. І навпаки, якщо буде створено закладку Kate, програма також створить відповідну позначку Vi у нульовій позиції рядка.

9.1.4 Підтримувані команди звичайного/візуального режимів

`a`	Перейти у режим вставлення, додавати символи після курсора
`A`	Перейти у режим вставлення, додавати символи після рядка
`i`	Перейти у режим вставлення, додавати символи перед курсором
Ins	Перейти у режим вставлення, додавати символи перед курсором
`I`	Перейти у режим вставлення, вставляти до першого непорожнього символу у рядку
`gi`	Перейти у режим вставлення, вставити перед місцем, у якому було здійснено останній вихід з режиму вставлення
`v`	Перейти у візуальний режим; позначати символи
`V`	Перейти у візуальний режим; позначати рядки
Ctrl+**v**	Перейти у візуальний режим; позначати блоки
`gb`	Перейти у візуальний режим; повторно позначити останній позначений фрагмент
`o`	Додати новий рядок під поточним
`O`	Додати новий рядок перед поточним
`J`	Об'єднати рядки
`c`	Змінити: з наступним визначенням позиції для вилучення і переходом у режим вставлення
`C`	Змінити до кінця рядка: вилучити текст до кінця рядка і увійти до режиму вставлення
`cc`	Змінити рядок: вилучити рядок і перейти у режим вставлення
`s`	Замінити символ
`S`	Замінити рядок
`dd`	Вилучити рядок
`d`	З наступним визначенням позиції для вилучення
`D`	Вилучити до кінця рядка
`x`	Вилучити символ праворуч від курсора
Del	Вилучити символ праворуч від курсора
`X`	Вилучити символі ліворуч від курсора
`gu`	З визначенням позиції для встановлення нижнього регістру
`guu`	Перевести всі символи поточного рядка у нижній регістр

gU	З визначенням позиції для встановлення верхнього регістру
gUU	Перетворити символи поточного рядка на символи у верхньому регістрі
y	З визначенням позиції для копіювання
yy	Копіювати рядок
Y	Копіювати рядок
p	Вставити після курсора
P	Вставити перед курсором
]p	Вставити після курсора з відступом
[p	Вставити перед курсором з відступом
r	З визначенням символу, який слід замінити символ за курсором
R	Увійти до режиму заміни
:	Увійти у командний режим
/	Шукати
u	Вернути
Ctrl+**R**	Повторити
U	Повторити
m	Встановити позначку (може бути використано для наступних пересувань текстом)
n	Знайти наступне
N	Знайти попереднє
>>	Збільшити відступ рядка
<<	Зменшити відступ рядка
>	Збільшити відступ рядків
<	Зменшити відступ рядків
Ctrl+**F**	Сторінка вниз
Ctrl+**B**	Сторінка вгору
ga	Вивести значення ASCII для символу
.	Повторити останню зміну
==	Команда вирівнювання рядка
=	Команда вирівнювання рядків
~	Змінити регістр поточного символу
Ctrl+**S**	Розділити перегляд горизонтально
Ctrl+**V**	Розділити перегляд вертикально
Ctrl+**W**, w	Циклічний перехід до наступної панелі поділу
Ctrl+**W**, h **CtrlW** ←	Перейти до лівої панелі поділу
Ctrl+**W**, l **CtrlW** →	Перейти до правої панелі поділу
Ctrl+**W**, k **CtrlW** ↑	Перейти до верхньої панелі поділу
Ctrl+**W**, j **CtrlW** ↓	Перейти до нижньої панелі поділу

9.1.5 Підтримувані пересування кодом

Цими командами можна скористатися для пересування документом у звичайному та візуальному режимах або у поєднанні з однією з описаних вище команд. Перед цими командами можна вказувати параметр, який визначає кількість рухів, які слід виконати.

h	Ліворуч
←	Ліворуч
Backspace	Ліворуч
j	Вниз
↓	Вниз
k	Вгору
↑	Вгору
l	Праворуч
→	Праворуч
Пробіл	Праворуч
$	У кінець рядка
End	У кінець рядка
0	До першого символу рядка (позиція 0)
Home	До першого символу рядка
^	До першого непорожнього символу рядка
f	З додаванням символу, до якого слід перейти праворуч від курсора
F	З додаванням символу, до якого слід перейти ліворуч від курсора
t	З додаванням символу, до якого слід перейти праворуч від курсора, з розташуванням курсора перед цим символом
T	З додаванням символу, до якого слід перейти ліворуч від курсора, з розташуванням курсора перед цим символом
gg	До першого рядка
G	До останнього рядка
w	До наступного слова
W	До наступного слова, відокремленого пробілом
b	До попереднього слова
B	До попереднього слова, відокремленого пробілом
e	До кінця слова
E	До кінця слова, відокремленого пробілом
ge	До кінця попереднього слова
gE	До кінця попереднього слова, відокремленого пробілом
|	З визначенням номера позиції у рядку для переходу
%	З визначенням елемента для переходу
‘	До позначки
‘	До першого непробільного символу рядка з позначкою
[[	До попередньої початкової дужки

]]	До наступної початкової дужки
[]	До попередньої завершальної дужки
][	До наступної завершальної дужки
Ctrl+**I**	Перейти до наступного місця
Ctrl+**O**	Повернутися до попереднього місця
H	Перейти до першого рядка на екрані
M	Перейти до рядка посередині екрана
L	Перейти до останнього рядка на екрані
%значення у відсотках	Перейти до вказаної у відсотках позиції документа
gk	Перейти на рядок вгору візуально (у разі використання динамічного перенесення рядків)
gj	Перейти на рядок вниз візуально (у разі використання динамічного перенесення рядків)
Ctrl+←	Перейти на слово ліворуч
Ctrl+→	Перейти на слово праворуч

9.1.6 Підтримувані текстові об'єкти

Цими командами можна скористатися для позначення частин тексту документа.

iw	Блок слова: слово з пробілами
aw	Слово: слово без пробілів
i˝	Від попередніх подвійних лапок (˝) до наступних подвійних лапок, разом з лапками
a''	Від попередніх подвійних лапок (˝) до наступних подвійних лапок, без лапок
i'	Від попередніх одинарних лапок (˝) до наступних одинарних лапок, разом з лапками
a'	Від попередніх одинарних лапок (˝) до наступних одинарних лапок, без лапок
i(	Від попередньої початкової круглої дужки [(] до наступної завершальної круглої дужки [)], разом з дужками
a(	Від попередньої початкової круглої дужки [(] до наступної завершальної круглої дужки [)], без дужок
i[	Від попередньої початкової квадратної дужки ([) до наступної завершальної квадратної дужки (]), разом з дужками
a[	Від попередньої початкової квадратної дужки ([) до наступної завершальної квадратної дужки (]), без дужок
i{	Від попередньої початкової фігурної дужки ({) до наступної завершальної фігурної дужки (}), разом з дужками

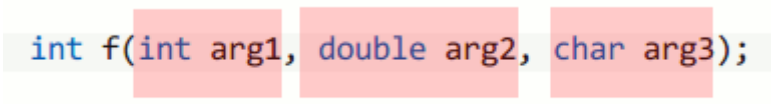
`a{`	Від попередньої початкової фігурної дужки (`{`) до наступної завершальної фігурної дужки (`}`), без дужок
`i<`	Від попередньої початкової кутової дужки (`<`) до наступної завершальної кутової дужки (`>`), разом з дужками
`a<`	Від попередньої початкової кутової дужки (`<`) до наступної завершальної кутової дужки (`>`), без дужок
`i‘`	Від попередніх зворотних лапок (‘) до наступних зворотних лапок, разом з лапками
`a‘`	Від попередніх зворотних лапок (‘) до наступних зворотних лапок, без лапок

9.1.7 Підтримувані команди режиму вставлення

Ctrl+**D**	Зменшити відступ
Ctrl+**T**	Збільшити відступ
Ctrl+**E**	Вставити знизу
Ctrl+**Y**	Вилучити слово
Ctrl+**W**	Вилучити слово
Ctrl+**U**	Вилучити рядок
Ctrl+**J**	Новий рядок
Ctrl+**H**	Вилучити символ у напрямку назад
Ctrl+**Home**	Перейти до першого символу документа
Ctrl+**R** `n`	Вставити вміст регістра `n`
Ctrl+**O**, *команда*	Увійти до звичайного режиму лише для однієї команди
Ctrl+**A**	Збільшити поточне позначене число
Ctrl+**X**	Зменшити поточне позначене число

9.1.8 Текстовий об’єкт, обмежений комами

Цього об’єкта немає у Vim. За допомогою текстового об’єкта, обмеженого комами, просто змінювати списки параметрів у C-подібних мовах та інших списках, поділених комами. Таким текстовим об’єктом є фрагмент тексту між двома комами або між комою і дужкою. На нашій ілюстрації показано три діапазони текстових об’єктів, позначені червоним тлом.



Відокремлені комами текстові об’єкти визначають діапазони даних. Якщо, наприклад, курсор перебуває у `arg2`*, натискання клавіш* `ci`*, («змінити між комами») вилучить* `double arg2` *і розташує курсор між двома комами у режимі вставлення. Це дуже зручний спосіб зміни параметрів функцій.*

9.1.9 Нереалізовані можливості

Як ми вже зазначали раніше, метою режиму VI KatePart не є 100% підтримка можливостей Vim.

Додаток A

Формальні вирази

У цьому додатку наведено короткий, але, як сподіваються автори, достатній і інформативний вступ до світу *формальних виразів*. Тут наведено документацію щодо формальних виразів у тому вигляді, у якому вони використовуються у KatePart, і який не є сумісним ні з формальними виразами у perl, ні з формальними виразами, наприклад, у grep.

A.1 Вступ

За допомогою *формальних виразів* можна описати можливий вміст рядка тексту так, щоб ваш опис був зрозумілим програмі і вона змогла визначати відповідність тексту шуканому рядку, а також, у випадку програм з додатковими можливостями, зберігати знайдені фрагменти для наступного використання.

Приклад: припустімо, вам потрібно знайти у тексті абзаци, які починаються зі слів «Іван» або «Франко», за якими слідує одна з форм дієслова «говорити».

Якщо б ви виконували пошук у звичайний спосіб, ви мали б почати з пошуку імені, «Іван», можливо у супроводі «говори», десь так: `Іван говори`, серед відповідників вам слід би було відкинути ті, які стоять не на початку абзацу, а також ті, у яких літери «говори» не є літерами слів «говорить», «говорив» тощо. Потім, звичайно ж, вам слід було б повторити пошук для прізвища...

За допомогою формальних виразів завдання з пошуку виконується в одну дію і з вищим рівнем точності.

Щоб досягти цього, за допомогою формальних виразів визначаються правила, за допомогою яких створюється узагальнена форма рядка пошуку. У нашому прикладі це правило можна висловити буквально так: «Рядок, який починається з «Іван» або «Франко» (перед цим словом може бути до 4 пробілів або символів табуляції), продовжується пробілом, за яким йде слово «говори», яке закінчується на «ть» або «в»». Все це визначає формальний вираз:

```
^[ \t]{0,4}(Іван|Франко) говори(ть|в)
```

У наведеному вище прикладі продемонстровано всі чотири компоненти сучасних формальних виразів, а саме:

- Шаблони
- Умовні вирази
- Лічильники
- Зворотні посилання

Символ каретки (`^`) на початку виразу є умовним виразом, який визначає, що наступний рядок має розпочинати рядок у тексті.

Рядки `[ \t]` і `(Іван|Франко) говори(ть|в)` є шаблонами. Перший з них визначає *клас символів*, до якого належить пробіл або символ горизонтальної табуляції; у другому шаблоні міститься спочатку підшаблон, який визначає рядок `Іван` *або* `Франко`, потім рядка `говори` і нарешті підшаблону, що визначає закінчення: `ть` *або* `ив`

Рядок `{0,4}` є лічильником, який повідомляє інструментові пошуку, що рядок знаходиться «будь-де, за від 0 до 4 рядками, вказаними раніше».

Програмне забезпечення, яке працює з формальними виразами, підтримує концепцію *зворотних посилань*, використання цієї концепції надає змогу зберегти повністю знайдену частину рядка або підрядки, позначені круглими дужками, а потім у певний спосіб використати ці посилання. Отже, ми можемо повністю позбутися ручної роботи з пошуку рядка (під час пошуку за допомогою формального виразу у текстовому документі, відкритому у редакторі, цей рядок буде позначено як виділений), пошуку імені або навіть фраз з різними закінченнями у дієслів.

Разом, формальний вираз визначає саме той рядок, який ми бажали знайти, і лише у потрібному місці тексту.

У наступних розділах буде докладно описано побудову і використання шаблонів, класів символів, умовних виразів, лічильників і зворотних посилань. Нарешті у останньому розділі ви знайдете декілька корисних прикладів.

A.2 Шаблони

Шаблони складаються з рядків і класів символів. У шаблонах можуть міститися підшаблони, тобто шаблони, взяті у круглі дужки.

A.2.1 Символи керування

У шаблонах, так само як і у класах символів, певні символи мають особливе призначення. Для пошуку цих символів не за значенням, а за візуальним представленням, ці символи слід позначити або *екранувати*, щоб повідомити рушієві пошуку за формальним виразом, що відповідні символи слід шукати за їх візуальним представленням.

Це завдання можна виконати за допомогою додавання перед символом зворотної навскісної риски (`\`).

Рушій пошуку за формальним виразом без додаткових повідомлень ігноруватиме екрановані символи, які не мають спеціального призначення, отже екранування, наприклад, символу «j» (`\j`) є безпечним. Якщо ви не пам'ятаєте, чи може символ мати спеціальне призначення, ви можете про всяк випадок його заекранувати.

Екранування, звичайно ж, можна застосовувати і до самого символу зворотної навскісної риски, щоб програма сприймала введену вами зворотну навскісну риску як зворотну навскісну риску, вам слід ввести `\\`.

A.2.2 Класи символів і абревіатури

Клас символів — це вираз, якому відповідає один символ з вказаного набору символів. У формальних виразах, класи символів визначаються зазначенням дозволених символів класу у квадратних дужках, `[]`, або за допомогою абревіатур класів, які описано нижче.

Прості класи символів містять один або декілька буквених символів, наприклад, `[abc]` (це означає одну з літер: «a», «b» або «c») або `[0123456789]` (цей клас визначає одну цифру).

Оскільки для літер і цифр характерний певний порядок (абетка або порядок цифр), ви можете скорочувати запис за допомогою діапазонів: `[a-c]` рівнозначний до `[abc]`, а `[0-9]` —

до `[0123456789]`. Можна також використовувати комбінації таких конструкцій, наприклад, `[a-fynot1-38]` (звичайно ж, остання комбінація відповідає одному з символів «a»,«b»,«c»,«d», «e»,«f»,«y»,«n»,«o»,«t», «1»,«2»,«3» або «8»).

Оскільки великі літери відрізняються від своїх малих еквівалентів, для створення класу символів, у якому б регістр не мав значення, з літер «a» і «b», вам слід записати `[aAbB]`.

Звичайно ж можна створити і «негативний клас», до якого б входили «всі літери окрім вказаних». Для запису такого класу вам слід скористатися символом каретки (`^`) на початку запису класу:

Набір `[^abc]` відповідає будь-якому символу, *окрім* «a», «b» і «c».

Окрім звичайних символів, передбачено використання деяких скорочень, які спрощують запис формальних виразів:

`\a`

Цей рядок відповідає символу дзвінка у ASCII (BEL, 0x07).

`\f`

Цей рядок відповідає символу зміни сторінки у ASCII (FF, 0x0C).

`\n`

Цей рядок відповідає символу зміни рядка у ASCII (LF, 0x0A, символ нового рядка у Unix).

`\r`

Цей рядок відповідає символу повернення каретки у ASCII (CR, 0x0D).

`\t`

Цей рядок відповідає символу горизонтальної табуляції у ASCII (HT, 0x09).

`\v`

Цей рядок відповідає символу вертикальної табуляції у ASCII (VT, 0x0B).

`\xhhhh`

Такий рядок відповідає символу Unicode з шістнадцятковим номером hhhh (у межах від 0x0000 до 0xFFFF). \0ooo (тобто \нульо ooo) відповідає символу з ASCII/Latin-1 з вісімковим номером ooo (у межах від 0 до 0377).

. (крапка)

Цей рядок відповідає будь-якому символу (зокрема символу нового рядка).

`\d`

Цей рядок відповідає будь-якій цифрі. Він рівнозначний до `[0-9]`

`\D`

Цей рядок відповідає будь-якому символу, окрім цифри. Він рівнозначний до `[^0-9]` або `[^\d]`

`\s`

Цей рядок відповідає символу проміжку між символами. З практичної точки зору, він рівнозначний до `[ \t\n\r]`

`\S`

Цей рядок відповідає символу, який не є пробілом. З практичної точки зору, цей рядок еквівалентний до `[^ \t\r\n]` або `[^\s]`

`\w`

Цей рядок відповідає будь-якому «символу слова» — у нашому випадку, літері, цифрі або символу підкреслювання. Цей рядок еквівалентний до рядка `[a-zA-Z0-9_]`

`\W`

Цей рядок відповідає будь-якому символу, який не є символом запису слів, тобто будь-якому символу, який не є літерою, цифрою або символом підкреслювання. Рядок еквівалентний до рядка `[^a-zA-Z0-9_]` або `[^\w]`

Передбачено також підтримку *позначення класів POSIX* — `[:<назва класу>:]`. Наприклад, `[:digit:]` є тим самим, що і `\d`, а `[:space:]` — тим самим, що і `\s`. Із повним списком класів символів POSIX можна ознайомитися тут.

Класи-скорочення можна використовувати в межах загальних класів, наприклад, щоб знайти літеру слова, символ пробілу або крапку, ви можете скористатися записом `[\w \.]`

A.2.2.1 Символи з особливим призначенням у класах символів

Перелічені далі символи мають особливе призначення у конструкціях класів символів «[]», отже для включення їх за візуальним значенням до класу ці символи слід екранувати:

`]`

Завершує визначення класу символів. Цей символ слід екранувати, якщо він не є найпершим символом у класі (можна використовувати одразу за неекранованим символом каретки).

`^` (знак вставки)

Змінює значення вказаного класу на протилежне, якщо є першим символом класу. Цей символ слід екранувати, якщо він є першим символом класу.

- (дефіс)

Визначає діапазон. Цей символ завжди слід екранувати, якщо він визначає символ у класі символів.

\ (зворотна навскісна риска)

Символ керівної послідовності. Цей символ завжди слід екранувати, якщо він потрібен вам як символ зворотної навскісної риски.

A.2.3 Варіанти: пошук «одного з»

Якщо вам потрібно знайти один з декількох варіантів шаблонів, ви можете відокремити ці шаблони символом `|` (вертикальною рискою).

Наприклад, щоб знайти один з рядків, «Іван» або «Юрій», вам слід скористатися виразом `Іван|Юрій`.

A.2.4 Підшаблони

Підшаблони — це шаблони взяті у круглі дужки, у світі формальних виразів існує декілька способів використання підшаблонів.

A.2.4.1 Визначення варіантів

Ви можете скористатися підшаблоном, щоб згрупувати набір варіантів у більший шаблон. Варіанти відокремлюються символом «|» (вертикальною рискою).

Наприклад, для того, щоб знайти одне зі слів «int», «float» або «double», ви можете скористатися шаблоном `int|float|double`. Якщо вам потрібно знайти одне з цих слів, і вам відомо, що за цим словом має йти пробіл, а потім якісь літери, згрупуйте варіанти у підшаблон: `(int|float|double)\s+\w+`.

A.2.4.2 Збереження знайденого тексту (зворотні посилання)

Якщо ви хочете скористатися зворотним посиланням, скористайтеся рядком `(ШАБЛОН)` для запам'ятовування бажаної частини рядка. Щоб запобігти запам'ятовуванню шаблона, скористайтеся групуванням без захоплення вмісту `(?:ШАБЛОН)`.

Наприклад, якщо вам потрібно знайти два однакових слова, відокремлені комою і, можливо, пробільними символами, ви можете скористатися формальним виразом `(\w+),\s*\1`. За допомогою підшаблону `\w+` буде знайдено послідовність символів слів, збіг з формальним виразом буде зареєстровано, якщо за цією послідовністю буде кома, 0 або більше пробільних символів і така сама послідовність символів слів. (Рядок `\1` відповідає *першому підшаблону, взятому у круглі дужки*)

> **Примітка**
> Щоб уникнути двозначностей, пов'язаних із використанням \1 з якимись цифрами за ним (наприклад \12 може означати 12 підшаблон або просто перший підшаблон із наступним 2) ми використовуємо синтаксичну конструкцію \{12} для багатоцифрових номерів підшаблонів.
> Приклади:
>
> - \{12}1 — це «використати підшаблон 12»
> - \123 — це «використати підшаблон 1, потім 23 як звичайний текст»

A.2.4.3 Умовні вирази з перевіркою

Умовний вираз з перевіркою — це підшаблон, що починається з символів `?=` або `?!`.

Наприклад, для того, щоб знайти рядок «Білл», за яким не буде рядка « Гейтс», можна скористатися таким виразом: `Bill(?! Gates)`. (Таким чином буде знайдено послідовності «Білл Клінтон» і «Біллі Кід», але проігноровано всі інші варіанти.)

Підшаблони, які було використано для перевірки умови, не будуть запам'ятовуватися рушієм пошуку.

Див. також Умовні вирази.

A.2.4.4 Умовні вирази з перевіркою і пошуком назад

Умовний вираз з перевіркою і пошуком назад — це підшаблон, що починається з символів `?<=` або `?<!`.

Пошук назад працює так само, як і пошук вперед, але пошук виконується у протилежному напрямку. Наприклад, знайти рядок «команда», але лише якщо перед ним немає рядка «Зе», ви можете скористатися таким виразом: `(?<!Зе)команда`.

Підшаблони, які було використано для перевірки умови, не будуть запам'ятовуватися рушієм пошуку.

Див. також Умовні вирази

A.2.5 Символи зі спеціальним значення у шаблонах

Перелічені нижче символи мають особливе призначення в межах шаблону, отже, якщо вам потрібно буде знайти ці символи за візуальним представленням, вам слід буде екранувати їх:

\ (зворотна навскісна риска)

Символ керівної послідовності.

`^` **(знак вставки)**
: Умовний початок рядка.

`$`
: Умовне завершення рядка.

`()` **(ліва і права круглі дужки)**
: Визначення підшаблону.

`{}` **(ліва і права фігурні дужки)**
: Визначення числових лічильників.

`[]` **(ліва і права квадратні дужки)**
: Визначення класів символів.

`|` **(вертикальна риска)**
: Логічне АБО. Відокремлює варіанти значення.

`+` **(знак «плюс»)**
: Лічильник, 1 або більше.

`*` **(зірочка)**
: Лічильник, 0 або більше.

`?` **(знак питання)**
: Необов'язковий символ. Може бути інтерпретовано яка лічильник, 0 або 1.

A.3 Лічильники

За допомогою *лічильників* можна виконувати пошук вказаної кількості або певного діапазону кількостей відповідників символу, класу символів або підшаблону.

Лічильники слід вказувати між фігурними дужками (`{` і `}`). У загальному випадку вони виглядають так: `{[мінімальна кількість][,[максимальна кількість]]}`

Використання найкраще пояснюється прикладом:

`{1}`
: Точно один збіг.

`{0,1}`
: Жодного або 1 збіг.

`{,1}`
: Те саме, але меншими зусиллями ;)

`{5,10}`
: Не менше 5, але не більше 10 збігів.

`{5,}`
: Не менше 5 збігів, без обмежень згори.

Крім того, існує декілька скорочень:

`*` **(зірочка)**
: те саме, що і `{0,}`, шукати без обмеження на кількість відповідників.

`+` **(знак «плюс»)**
: те саме, що і `{1,}`, принаймні один відповідник.

`?` **(знак питання)**
: те саме, що і `{0,1}`, жодного або один відповідник.

A.3.1 Жадібність

За використання лічильників без зазначення максимальної кількості, типово, буде виконано пошук якомога більшої кількості відповідників формального виразу, така поведінка називається *жадібною*.

У сучасних рушіях пошуку за формальними виразами передбачено можливість «вимикання жадібності», забезпечення доступу до цієї можливості є проблемою лише графічного інтерфейсу. Наприклад, у діалоговому вікні пошуку за формальним виразом може бути поле для позначки з міткою «Мінімальна кількість збігів», а також певним чином позначено, що жадібність є типовою поведінкою.

A.3.2 Приклади використання

Ось декілька прикладів використання лічильників.

`^\d{4,5}\s`

Буде знайдено цифри у «1234 поїхали» і «12345 давай», але не у «567 одинадцять» і у «223459 десь».

`\s+`

Буде знайдено один або декілька пробільних символів.

`(ля){1,}`

Буде знайдено «ляляля», а також «ля» у словах «шабля» або «пляшка».

`/?>`

Буде знайдено «/>» у «<closeditem/>», а також «>» у «<openitem>».

A.4 Умовні вирази

За допомогою *умовних виразів* відповідники формального виразу буде знайдено лише за виконання певних керованих умов.

За використання умовного виразу не буде виконуватися пошук певного символу, рушій скоріше вивчатиме оточення можливого відповідника формального виразу і визначатиме, чи той це вираз, який вам потрібен. Наприклад, за використання умови *межа слова* рушій пошуку не намагатиметься знайти символ, який не є символом слова, поряд зі знайденим словом, — рушій просто переконається, що цей символ не є символом слова. Це означає, що за використання умовного виразу, рядок буде знайдено і там, де пробільного символу немає, наприклад, наприкінці рядка.

У деяких умовних виразах можуть міститися шаблони, але за їх використання виконуватиметься лише перевірка на збіг рядка з вказаним шаблоном, — результат перевірки не буде включено до результатів пошуку.

У формальних виразах, описаних у цій документації, підтримуються такі умовні вирази:

`^` (каретка: початок рядка)

Пошук на початку рядка.

За виразом `^Петре` буде знайдено рядок «Петре» у рядку «Петре, агов!», але не у рядку «Агов, Петре!»

`$` (кінець рядка)

Пошук наприкінці рядка.

За виразом `ти\?$` буде знайдено останнє «ти» у рядку «Це зробив не ти?», але нічого не буде знайдено у рядку «Це ж не ти зробив?»

`\b` **(межа слова)**

Пошук рядка, з одного боку якого стоїть символ запису слів, а з іншого — символ, який не використовується для запису слів.

Цей вираз корисний для перевірки на те, чи завершується у певному місці слово, наприклад під час пошуку цілого слова. Вираз `\by\b` буде знайдено як окреме слово «у» у рядку «Він увійшов у вікно», але не як частину «у» слова «увійшов».

`\B` **(відсутність межі слова)**

Пошук всіх відповідників, які не буде пропущено за використання умовного виразу «\b».

Такий вираз можна використовувати, наприклад, для пошуку внутрішніх частин слів: вираз `\By\B` буде знайдено у слові «вухо», але не у слові «увійшов» або виразі «Бачив у кіно».

`(?=ШАБЛОН)` **(перевірка на збіг)**

За допомогою умовного виразу перевірки на збіг можна перевірити на відповідність частину рядка, яку розташовано безпосередньо за можливим відповідником. За використання перевірки на збіг частина рядка вважатиметься невідповідною, якщо за нею не йде вказаний *ШАБЛОН* умовного виразу, але текст цього шаблону не буде додано до результату пошуку.

За виразом `зручно(?=\w)` буде знайдено рядок «зручно» у слові «зручності», а у рядку «Це дуже зручно!» нічого не буде знайдено.

`(?!ШАБЛОН)` **(перевірка на відсутність збігу)**

За перевірки на відсутність збігу знайдений рядок не буде вважатися потрібним, якщо за ним йде рядок вказаний як *ШАБЛОН*.

Вираз `const \w+\b(?!\s*&)` відповідатиме частині «const char» у рядку «const char* foo», але не частині «const QString» у рядку «const QString& bar», оскільки символ «&» збігається з шаблоном перевірки на відсутність збігу.

`(?<=ШАБЛОН)` **(перевірка на збіг із пошуком назад)**

Пошук назад працює так само, як і пошук вперед, але пошук виконується у протилежному напрямку. За допомогою умовного виразу перевірки на збіг із пошуком назад можна перевірити на відповідність частину рядка, яку розташовано безпосередньо перед можливим відповідником. Частина рядка вважатиметься відповідною, лише якщо перед нею не йде вказаний *ШАБЛОН* умовного виразу, але текст цього шаблону не буде додано до результату пошуку.

За виразом `(?<=диво)сміх` буде знайдено «сміх», якщо перед ним є рядок «диво» (тобто буде знайдено «дивосміх», але не «вогнесміх» і не окреме слово «сміх»).

`(?<!ШАБЛОН)` **(перевірка на відсутність збігу із пошуком назад)**

За перевірки на відсутність збігу із пошуком назад знайдений рядок не буде вважатися потрібним, якщо перед ним йде рядок вказаний як *ШАБЛОН*.

За виразом `(?<![\w\.])[0-9]+` буде знайдено «123» у рядках «=123» і «-123», але не буде знайдено «123» у «.123» і «слово123».

`(ШАБЛОН)` **(Захоплення групи)**

Засіб обробки захопить і запам'ятає шаблон у дужках так, щоб ним можна було скористатися для зворотного посилання. Наприклад, вираз `(&quot;+)[^&quot;]*\1` відповідає одразу рядку `˝˝˝˝текст˝˝˝˝` і рядку `˝текст˝`.

Докладніший опис можна знайти у розділі Збереження знайденого тексту (зворотні посилання).

`(?:ШАБЛОН)` **(Групування без захоплення)**

Засіб обробки не захоплюватиме і не запам'ятовуватиме шаблон у дужках. Якщо ви не маєте наміру використовувати запам'ятовування груп, варто завжди користуватися саме цим варіантом запису груп.

Додаток Б

Предметний покажчик