LINEDIT - Редактор на линии

Обща структура на проекта: Проектът „Редактор на линии” реализира прост текстов редактор, който работи с текстов файл и позволява редактиране на текуща линия, както и промяна на текуща линии чрез преместване. Редактирането се осъществява чрез добавяне и изтриване на редове на дадена позиция. Проектът следва структура, вдъхновена от MVC модела: EditorUI, който осъществява взаимодействието с потребителя; Editor, който осъществява връзката между интерфейса и структурите данни, поддържа състоянието (текущ ред) и основните функции от спецификицията; LineList с помощни класове Line и LineIterator, които имплементират вътрешното представяне – едносвързан списък от редове.

Структура на класовете, алгоритми и преодоляни проблеми:

Struct Line – играе ролята на възел за свързания списък, носи в себе си текст и указател към следващ ред. Предоставя и удобен конструктор с параметър по подразбиране за указател, така че да може да се използва и само със стойност.

Class LineIterator – итератор за свързания списък, позволява преместване напред с предефинирани опретори ++ (префиксен и постфиксен), както и дереферииране за получаване на стойност с предефиниран оператор *. Освен това позволява проверка за валидност чрез предефиниран оператор за cast към bool, оператори за сравнение (дали сочи към една и съща „кутия” от списъка), както и функция за преместване напред с целочислен брой стъпки moveForward(int). Конструира се чрез указател към Line, подаден на конструктора.

Class LineList – представлява имплементация на едносвързан списък от редове, ползвана вътрешно от класа за поддържане на желаните операции с текстов редактор. Във връзка с изискванията на проекта да има виртуална крайна позиция, която да принтира <<END>> и пред която да може да се вмъква този списък използва sentinel за края си. По този начин някой функционалности, например принтирането на <<END>>, се реализират без допълнителни случаи, а други функции се имплементират със случаи, но по-малко на брой. Поддържат се функции за вмъкване на поредица от линии, записани в динамичен масив, преди дадена позиция – **insertBefore** (LineIterator&, vector<char*>&), изтриване на дадена позиция – **removeAt** (LineIterator&), принтиране на текста от дадена позиция натам – **printFrom** (LineIterator&), както и придвижване до конкретна позиция, подадена като число (индекс) – findPosition(int), връщаша итератор към тази позиция. Също така може да достъпваме броя на редовете, пазещ се като променлива size, и имаме помощна функция **previous**(LineIterator&), която намира предишния на даден елемент в списъка (така се справяме с този проблем, предизвикан от едносвързаността на списъка) и директно връща указател към желаната линия.

Class Editor – осъществява реалната логика и връзката между потребителския интерфейс и вътрешното представяне чрез структура от данни. За целта той има член-данни име на текущ файл (filename) и индекс на текущата позиция (currentPosition). Помощни член данни са LineList обект, пазещ редовете, итератор към текущата позиция и поток out от тип ofstream, който се използва за съхраняване на промените. Функциите, които предоставя, са желаните от изискванията: за добавяне на редове преди текущия, което не променя сочената текуща позиция (за тази цел към нея се прибавя броя на прибавените редове, за да е коректна); премахване на текуща линия, при което сочим към следващата, принтиране на всички редове от текущия до края. Всичко това се осъществява с подходящи извиквания на методи на списъка и с помощта на итератора, сочещ текуща позиция. Освен това може да се придвиждваме напред – чрез метода на итератора, или назад – пресмятаме новата позиция и директно отиваме на нея с findPosition метода. Възможно е отваряне на нов файл и съхраняване на данни. Извършва се проверка на индекса, за да не се излезе от границите на списъка, и се реализира желаното

поведение придвижването с по-голямо число да се интерпретира като преместване на първи или последен ред съответно.

Class EditorUI – грижи се за взаимодействието с потребителя. За целта чете от поток (по подразбиране стандартния вход) команди и има един метод run(), в който приема команди и ги парсва. Командите са дефинираните в спецификацията с желаното от нея поведение плюс команда OP, следвана от име на файл, за отваряне на файла с това име като текущ обработван. Този клас има грижата да извика командата за съхраняване на промените преди изход.

Бъдещи подобрения:

Текущата функционалност може да се надгражда и да се правят промени в следните насоки:

- Потребителски интерфейс – конзолният да се замени с графичен
- Да може директно да се редактира даден ред, без да се налага да се изтрие и след това добави нов ред с желаното ново съдържание
- В допълнение към това да се добави курсор – указател към текуща позиция в текущия ред, която да се мести и чрез нея да осъществяваме промени като в традиционните текстови редактори
- Да се избере по-ефективно вътрешно представяне – поне двусвързан списък или може би rope за забързване на някои операции при големи по размер файлове
- Да се имплементира движение не само по редове, но чрез курсора и вътре в редове, т.е. до даден ред и дадена позиция
- Да се даде възможност за връщане назад при грешка, eg Undo на Delete или добавяне на редове