**Experiment Report - 15 – test1\_codehistory**

1. **Summary Table of Errors Found**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Error ID | Line Number | Error Type | CSCR? | Self-Detected? | LLM? |
| E01 | line 18 | Syntax | √ |  | √ |
| E02 | line 30 | Syntax | √ |  | × |
| E03 | line 37 | Logic | √ |  | × |
| E04 | line 23 (46) | Semantic |  | √ | √ |
| E05 | line 42 (65) | Logic |  | × | √ |
| E06 | line 50 (73) | Semantic |  | √ | √ |

Additional Errors Found by Self: 0

CSCR Rate: 100%

Self-Review Detection Rate: 67%

LLM Rate: 67%

1. **Source Code**
2. #include "codehistory.h"
3. #include "historylinenumberarea.h"
4. #include <QPainter>
5. #include <QTextBlock>
6. CodeHistory::CodeHistory(QWidget \*parent)
7. : QPlainTextEdit(parent),
8. lineNumberArea(new HistoryLineNumberArea(this)),  *// 初始化行号区域*
9. highlighter(new SyntaxHighlighter(this->document()))  *// 初始化语法高亮*
10. {
11. *// 设置编辑器的初始宽度和高亮当前行的功能*
12. connect(this, &QPlainTextEdit::blockCountChanged, this, &CodeHistory::updateLineNumberAreaWidth);
13. connect(this, &QPlainTextEdit::updateRequest, this, &CodeHistory::updateLineNumberArea);
14. connect(this, &QPlainTextEdit::cursorPositionChanged, this, &CodeHistory::highlightCurrentLine);
15. updateLineNumberAreaWidth(0);
16. highlightCurrentLine();
17. setEditorFont("Courier");  *// 默认字体设置*
18. }
19. void CodeHistory::setEditorFont(const QString &fontFamily, int fontSize)
20. {
21. QFont font(fontFamily, fontSize);
22. this->setFont(font);
23. }
24. void CodeHistory::paintEvent(QPaintEvent \*event)
25. {
26. QPlainTextEdit::paintEvent(event);
27. lineNumberAreaPaintEvent();
28. }
29. int CodeHistory::lineNumberAreaWidth()
30. {
31. int digits = 1;
32. int max = qMax(1, blockCount());
33. while (max > 10) {
34. max /= 10;
35. ++digits;
36. }
37. int space = 3 + fontMetrics().horizontalAdvance(QLatin1Char('9')) \* digits;
38. return space;
39. }
40. void CodeHistory::lineNumberAreaPaintEvent(QPaintEvent \*event)
41. {
42. QPainter painter(lineNumber);
43. painter.fillRect(event->rect(), QColor("#f0f0f0"));
44. QTextBlock block = firstVisibleBlock();
45. int blockNumber = block.blockNumber();
46. int top = (int) blockBoundingGeometry(block).translated(contentOffset()).top();
47. int bottom = top + (int) blockBoundingRect(block).height();
48. while (block.isValid() && top <= event->rect().bottom()) {
49. if (block.isVisible() && bottom >= event->rect().top()) {
50. QString number = QString::number(blockNumber + 1);
51. painter.setPen(Qt::black);
52. painter.drawText(0, top, lineNumberArea->width(), fontMetrics().height(),
53. Qt::AlignRight, number);
54. }
55. block = block.next();
56. top = bottom;
57. bottom = top + (int) blockBoundingRect(block).height();
58. --blockNumber;
59. }
60. }
61. void CodeHistory::resizeEvent(QResizeEvent \*event)
62. {
63. QPlainTextEdit::resizeEvent(event);
64. QRect cr = contentsRect();
65. lineNumberArea->setGeometry(QRect(cr.left(), cr.top(), lineNumberAreaWidth(), cr.width()));
66. }