Петрозаводский государственный университет
Институт математики и информационных технологий
Кафедра информатики и математического обеспечения

Направление подготовки бакалавриата

09.03.02 — Информационные системы и технологии

Отчет по проекту

Разработка приложения «Калькулятор»

Выполнил:
студент 1 курса группы 22106

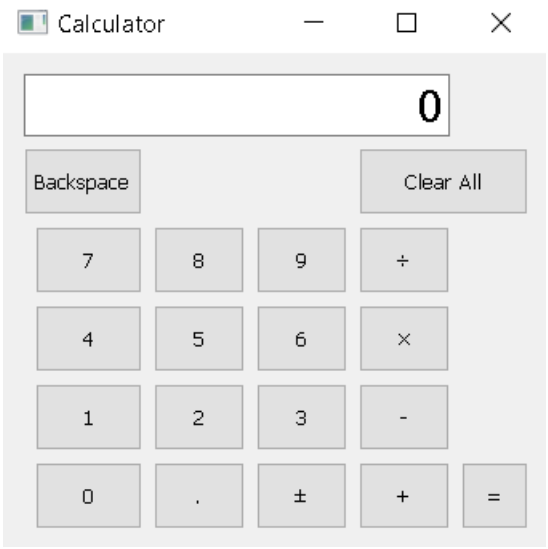
Андрей Моисеев _____
подпись

Петрозаводск — 2021

Введение

Цель проекта: Создать калькулятор и продемонстрировать его функционал

Интерфес приложения:



1 Описание приложения

Обычный калькулятор. Неотъемлемая часть любой вычислительной техники, работающей с числами - клавиши для ввода цифр. Поэтому кнопки от «0» до «9» в представлении не нуждаются. Операторы основных математических операций, такие как умножение(«*»), деление («/»), сложение («+»), вычитание («-») и сравнение («=») знакомы нам по урокам математики.

2 Реализация приложения

1) Определение класса калькулятора

```
class Calculator : public QWidget
{
    Q_OBJECT

public:
    Calculator(QWidget *parent = nullptr);

private slots:
    void digitClicked();
    void additiveOperatorClicked();
    void multiplicativeOperatorClicked();
    void equalClicked();
    void pointClicked();
    void changeSignClicked();
    void backspaceClicked();
    void clearAll();
```

2) Для каждой кнопки мы вызываем private createButton() функция с соответствующей текстовой меткой и слотом для подключения к кнопке.

```
    QGridLayout *mainLayout = new QGridLayout;

    mainLayout->setSizeConstraint(QLayout::SetFixedSize);
    mainLayout->addWidget(display, 0, 0, 1, 5);
    mainLayout->addWidget(backspaceButton, 1, 0, 1, 2);
    mainLayout->addWidget(clearAllButton, 1, 4, 1, 2);


    for (int i = 1; i < NumDigitButtons; ++i) {
        int row = ((9 - i) / 3) + 2;
        int column = ((i - 1) % 3) + 1;
        mainLayout->addWidget(digitButtons[i], row, column);
    }

    mainLayout->addWidget(digitButtons[0], 5, 1);
```

```
16      mainLayout->addWidget(pointButton, 5, 2);
17      mainLayout->addWidget(changeSignButton, 5, 3);
18      mainLayout->addWidget(divisionButton, 2, 4);
19      mainLayout->addWidget(timesButton, 3, 4);
20      mainLayout->addWidget(minusButton, 4, 4);
21      mainLayout->addWidget(plusButton, 5, 4);
22      mainLayout->addWidget(equalButton, 5, 5);
23      setLayout(mainLayout);
24
25      setWindowTitle(tr("Calculator"));
```

3) Реализация класса Button

```
1  Button::Button(const QString &text, QWidget *parent)
2      : QToolButton(parent)
3  {
4      setSizePolicy(QSizePolicy::Expanding, QSizePolicy::Preferred);
5      setText(text);
6  }
```

3 Приложение Калькулятор

```
1  #include "calculator.h"
2  #include "button.h"
3
4  #include <QGridLayout>
5  #include <QLineEdit>
6  #include <QtMath>
7
8  Calculator::Calculator(QWidget *parent)
9      : QWidget(parent), sumSoFar(0.0)
10     , factorSoFar(0.0), waitingForOperand(true)
11 {
12
13     display = new QLineEdit("0");
14
15     display->setReadOnly(true);
16     display->setAlignment(Qt::AlignRight);
17     display->setMaxLength(15);
```

```
18
19      QFont font = display->font();
20      font.setPointSize(font.pointSize() + 8);
21      display->setFont(font);
22
23      for (int i = 0; i < NumDigitButtons; ++i)
24          digitButtons[i] = createButton(QString::number(i), SLOT(digitClicked()));
25
26      Button *pointButton = createButton(tr("."), SLOT(pointClicked()));
27      Button *changeSignButton = createButton(tr("\302\261"), SLOT(changeSignClicked()));
28      Button *backspaceButton = createButton(tr("Backspace"), SLOT(backspaceClicked()));
29      Button *clearAllButton = createButton(tr("Clear All"), SLOT(clearAll()));
30      Button *divisionButton = createButton(tr("\303\267"), SLOT(multiplicativeOperatorClicked()));
31      Button *timesButton = createButton(tr("\303\227"), SLOT(multiplicativeOperatorClicked()));
32      Button *minusButton = createButton(tr("-"), SLOT(additiveOperatorClicked()));
33      Button *plusButton = createButton(tr("+"), SLOT(additiveOperatorClicked()));
34      Button *equalButton = createButton(tr("="), SLOT(equalClicked()));
35
36      QGridLayout *mainLayout = new QGridLayout;
37
38      mainLayout->setSizeConstraint(QLayout::SetFixedSize);
39      mainLayout->addWidget(display, 0, 0, 1, 5);
40      mainLayout->addWidget(backspaceButton, 1, 0, 1, 2);
41      mainLayout->addWidget(clearAllButton, 1, 4, 1, 2);
42
43      for (int i = 1; i < NumDigitButtons; ++i) {
44          int row = ((9 - i) / 3) + 2;
45          int column = ((i - 1) % 3) + 1;
46          mainLayout->addWidget(digitButtons[i], row, column);
47      }
48
49      mainLayout->addWidget(digitButtons[0], 5, 1);
50      mainLayout->addWidget(pointButton, 5, 2);
51      mainLayout->addWidget(changeSignButton, 5, 3);
52      mainLayout->addWidget(divisionButton, 2, 4);
53      mainLayout->addWidget(timesButton, 3, 4);
54      mainLayout->addWidget(minusButton, 4, 4);
55      mainLayout->addWidget(plusButton, 5, 4);
56      mainLayout->addWidget(equalButton, 5, 5);
57      setLayout(mainLayout);
```

```
58
59      setWindowTitle(tr("Calculator"));
60  }
61
62  void Calculator::digitClicked()
63  {
64      Button *clickedButton = qobject_cast<Button *>(sender());
65      int digitValue = clickedButton->text().toInt();
66      if (display->text() == "0" && digitValue == 0.0)
67          return;
68
69      if (waitingForOperand) {
70          display->clear();
71          waitingForOperand = false;
72      }
73      display->setText(display->text() + QString::number(digitValue));
74  }
75
76
77
78  void Calculator::additiveOperatorClicked()
79
80  {
81      Button *clickedButton = qobject_cast<Button *>(sender());
82      if (!clickedButton)
83        return;
84      QString clickedOperator = clickedButton->text();
85      double operand = display->text().toDouble();
86
87      if (!pendingMultiplicativeOperator.isEmpty()) {
88
89          if (!calculate(operand, pendingMultiplicativeOperator)) {
90              abortOperation();
91              return;
92          }
93          display->setText(QString::number(factorSoFar));
94          operand = factorSoFar;
95          factorSoFar = 0.0;
96          pendingMultiplicativeOperator.clear();
97      }
```

```
98
99
100     if (!pendingAdditiveOperator.isEmpty()) {
101
102         if (!calculate(operand, pendingAdditiveOperator)) {
103             abortOperation();
104             return;
105         }
106         display->setText(QString::number(sumSoFar));
107     } else {
108         sumSoFar = operand;
109     }
110
111     pendingAdditiveOperator = clickedOperator;
112
113     waitingForOperand = true;
114 }
115
116 void Calculator::multiplicativeOperatorClicked()
117 {
118     Button *clickedButton = qobject_cast<Button *>(sender());
119     if (!clickedButton)
120       return;
121     QString clickedOperator = clickedButton->text();
122     double operand = display->text().toDouble();
123
124     if (!pendingMultiplicativeOperator.isEmpty()) {
125         if (!calculate(operand, pendingMultiplicativeOperator)) {
126             abortOperation();
127             return;
128         }
129         display->setText(QString::number(factorSoFar));
130     } else {
131         factorSoFar = operand;
132     }
133
134     pendingMultiplicativeOperator = clickedOperator;
135     waitingForOperand = true;
136 }
137
```

```
138 void Calculator::equalClicked()
139 {
140     double operand = display->text().toDouble();
141 
142     if (!pendingMultiplicativeOperator.isEmpty()) {
143         if (!calculate(operand, pendingMultiplicativeOperator)) {
144             abortOperation();
145             return;
146         }
147         operand = factorSoFar;
148         factorSoFar = 0.0;
149         pendingMultiplicativeOperator.clear();
150     }
151     if (!pendingAdditiveOperator.isEmpty()) {
152         if (!calculate(operand, pendingAdditiveOperator)) {
153             abortOperation();
154             return;
155         }
156         pendingAdditiveOperator.clear();
157     } else {
158         sumSoFar = operand;
159     }
160 
161     display->setText(QString::number(sumSoFar));
162     sumSoFar = 0.0;
163     waitingForOperand = true;
164 }
165 
166 void Calculator::pointClicked()
167 {
168     if (waitingForOperand)
169         display->setText("0");
170     if (!display->text().contains('.'))
171         display->setText(display->text() + tr("."));
172     waitingForOperand = false;
173 }
174 
175 void Calculator::changeSignClicked()
176 {
177     QString text = display->text();
```

```
178     double value = text.toDouble();
179
180     if (value > 0.0) {
181         text.prepend(tr("-"));
182     } else if (value < 0.0) {
183         text.remove(0, 1);
184     }
185     display->setText(text);
186 }
187
188 void Calculator::backspaceClicked()
189 {
190     if (waitingForOperand)
191         return;
192
193     QString text = display->text();
194     text.chop(1);
195     if (text.isEmpty()) {
196         text = "0";
197         waitingForOperand = true;
198     }
199     display->setText(text);
200 }
201
202 void Calculator::clearAll()
203 {
204     sumSoFar = 0.0;
205     factorSoFar = 0.0;
206     pendingAdditiveOperator.clear();
207     pendingMultiplicativeOperator.clear();
208     display->setText("0");
209     waitingForOperand = true;
210 }
211
212 Button *Calculator::createButton(const QString &text, const char *member)
213 {
214     Button *button = new Button(text);
215     connect(button, SIGNAL(clicked()), this, member);
216     return button;
217 }
```

```
218
219 void Calculator::abortOperation()
220 {
221     clearAll();
222     display->setText(tr("[U+2639]"));
223 }
224
225 bool Calculator::calculate(double rightOperand, const QString &pendingOperator)
226 {
227     if (pendingOperator == tr("+")) {
228         sumSoFar += rightOperand;
229     } else if (pendingOperator == tr("-")) {
230         sumSoFar -= rightOperand;
231     } else if (pendingOperator == tr("\303\227")) {
232         factorSoFar *= rightOperand;
233     } else if (pendingOperator == tr("\303\267")) {
234         if (rightOperand == 0.0)
235             return false;
236         factorSoFar /= rightOperand;
237     }
238     return true;
239 }
```