Calculator

1.0.0

制作者 Frank Chu

Chapter 1

README

1.1 实验二 基于 **QT GUI** 计算器程序设计

1.1.1 实验目的

1. 熟悉 Windows 应用程序开发的基本过程；
2. 学习 Qt 对话框应用程序开发过程；
3. 学习标准控件的使用

实验环境：Qt 6.0/Qt Creator8.0 或以上

1.1.2 实验主要原理说明

1.1.2.1 **Qt**常用部件

（提示：列出Qt Designer工具箱内的部件分类，及其中的常用的部件）

- Layouts
- Spacers
- Buttons
  - Push Button
  - Radio Button
- Item Views (Model-Based)
- Item Widgets (Item-Based)
- Containers
  - Group Box
- Input Widgets
  - Line Edit
- Display Widgets
  - Label
  - Text Browser

1.1.2.2 对话框访问部件的方法

1. 双击 dialog.ui，进入对话框编辑器
2. 直接按“运行”按钮，进行构建并运行，得到一空对话框
3. 结束运行，回到设计视图，设定对话框标题，最小及最大高宽度，并在左边的 Widgets 工具箱中把 Label, pushButton 两种 widgets 拖入对话框放好
4. 点运行，将会得到一个大小固定的对话框
5. 回到“设计”视图，点中 PushButton，并在属性页改变其按钮文本
6. 右击按钮，选择“转到槽”菜单项，然后选择信号“clicked()”，点击“确定”
7. 在该“槽”方法中，键入代码

1.1.2.3 基于事件驱动的消息机制简要说明

基于消息的事件驱动机制 - Tencent Cloud

基于消息的事件驱动机制是一个通用模型，广泛应用于桌面软件开发、网络应用程序开发、前端开发等技术方向中。可以理解为外部操作事件，被转化为消息存放于队列中；而每种类型的消息都有对应的处理；通过消息循环，完成读消息、调用消息处理这个过程。这个过程，只要应用不退出，会一直进行下去。下图的模型从Windows应用程序而来，但是具有一定的通用性。

1.1.2.4 Qt信号与槽通信机制简要说明

Qt 信号与槽机制详解 信号与槽（Signal & Slot） 是 Qt 编程的基础，也是 Qt 的一大创新。 因为有了信号与槽的编程机制，在 Qt 中处理界面各个组件的交互操作时变得更加直观和简单。 信号（Signal） 就是在特定情况下被发射的事件，例如 PushButton 最常见的信号就是鼠标单击时发射的 clicked () 信号，一个 ComboBox 最常见的信号是选择的列表项变化时发射的 CurrentIndexChanged() 信号。 GUI 程序设计的主要内容就是对界面上各组件的信号的响应，只需要知道什么情况下发射哪些信号，合理地去响应和处理这些信号就可以了。 槽（Slot）就是对信号响应的函数。

1.1.3 实验内容及实现

1.1.3.1 简易计算器

A、界面设计

（给出界面即可）

B、主要程序说明

（给出关键槽、成员变量等的处理代码，并加关键注释说明）

1. 简易计算器类 EasyCalculator
2. 计算按钮按下槽 EasyCalculator::on_calculateButton_clicked()
3. 清楚按钮按下槽 EasyCalculator::on_clearButton_clicked()
4. 私有的计算符号 EasyCalculator::op
5. 枚举变量 EasyCalculator::Operation

1.1.3.2 进阶计算器

A、界面设计

（给出界面，并简要说明实现方法）

- Buttons -> PushButton
- InputWidgets -> LineEdit
- Grid Layout
- Style Sheet

```
// Style Sheet
* {
border:none;
background-color:  rgb(238, 236, 236);
}
QPushButton {
background-color:  rgb(243, 243, 243);
}
QPushButton:hover {
border:1px solid rgb(193, 193, 193);
background-color:  rbg(221, 223, 221);
}
QPushButton#btn_numSign:hover, #btn_numDot:hover, #btn_num0:hover, #btn_num1:hover, #btn_num2:hover,
      #btn_num3:hover, #btn_num4:hover, #btn_num5:hover, #btn_num6:hover, #btn_num7:hover, #btn_num8:hover,
      #btn_num9:hover {
background-color:  rgb(221, 223, 221);
}
QPushButton#btn_numSign, #btn_numDot, #btn_num0, #btn_num1, #btn_num2, #btn_num3, #btn_num4, #btn_num5,
      #btn_num6, #btn_num7, #btn_num8, #btn_num9 {
background-color:  rgb(252, 252, 252);
}
```

B、主要程序说明

（给出关键槽、成员变量等的处理代码，并加关键注释说明）

1. 按钮组按下方法 Dialog::onButtonGroupClicked()
2. UI 初始化时将数字按钮组成组，关联按钮组的点击信号 Dialog::iniUI()
3. 前一个按钮输入的符号 OR 数字 Dialog::prevBtn
4. 储存左数右数和符号，进行运算 Dialog::vectorOfNumbersAndSigns

1.1.4 程序测试

给出程序测试结果，并简要说明测试过程

```
// 在简单计算器程序进行加减乘除
// 在复杂计算器进行加减乘除
```

1.1.5 讨论及心得

比如可以给出如下内容：

1、实验过程中遇到的问题与解决方法

实验中对于 QT 的控件使用有部分问题，通过网络查询，能够了解到具体的使用方法。

2、目前尚未解决的问题

目前进阶计算器在数据结构上可以进一步优化。

Chapter 2

命名空间索引

2.1 命名空间列表

这里列出了所有命名空间定义,附带简要说明:

Chapter 3

继承关系索引

3.1 类继承关系

此继承关系列表按字典顺序粗略的排序:

Chapter 4

类索引

4.1 类列表

这里列出了所有类、结构、联合以及接口定义等，并附带简要说明:

Chapter 5

文件索引

5.1 文件列表

这里列出了所有文件，并附带简要说明:

Chapter 6

命名空间文档

6.1 Ui 命名空间参考

Chapter 7

类说明

7.1 Dialog类 参考

The Dialog class UI 界面逻辑类

`#include <dialog.h>`

Public 槽

- void onButtonGroupClicked (QAbstractButton *btn)

  onButtonGroupClicked 当数字按钮组按下的的时候进行的反馈函数，在 *iniUI()* 中使用

Public 成员函数

- Dialog (QWidget *parent=nullptr)

  Dialog 构造函数
- ~Dialog ()

  Dialog 析构函数
- void iniUI ()

  iniUI UI 初始化函数，*1.* 将数字按钮组成组 *2.* 关联按钮组的点击信号

Private 属性

- Ui::Dialog * ui

  ui ui 界面变量
- QVector< QVariant > vectorOfNumbersAndSigns

  vectorOfNumbersAndSigns 储存左数右数和符号，进行运算。
- QString prevBtn

  prevBtn 前一个按钮输入的符号 *OR* 数字

7.1.1 详细描述

The Dialog class UI 界面逻辑类

在文件 dialog.h 第 26 行定义.

7.1.2 构造及析构函数说明

7.1.2.1 Dialog()

```
Dialog::Dialog (
            QWidget * parent = nullptr )
```

Dialog 构造函数

参数

parent	

在文件 dialog.cpp 第 4 行定义.

7.1.2.2 ~Dialog()

```
Dialog::~Dialog ( )
```

Dialog 析构函数

在文件 dialog.cpp 第 12 行定义.

7.1.3 成员函数说明

7.1.3.1 iniUI()

```
void Dialog::iniUI ( )
```

iniUI UI 初始化函数，1. 将数字按钮组成组 2. 关联按钮组的点击信号

在文件 dialog.cpp 第 17 行定义.

7.1.3.2 onButtonGroupClicked

```
void Dialog::onButtonGroupClicked (
            QAbstractButton * btn )  [slot]
```

onButtonGroupClicked 当数字按钮组按下的的时候进行的反馈函数，在 iniUI() 中使用

参数

btn	按下的按钮

name 计算器输入的数据

在文件 dialog.cpp 第 31 行定义.

7.1.4 类成员变量说明

7.1.4.1 prevBtn

```
QString Dialog::prevBtn  [private]
```

prevBtn 前一个按钮输入的符号 OR 数字

在文件 dialog.h 第 68 行定义.

7.1.4.2 ui

```
Ui::Dialog* Dialog::ui  [private]
```

ui ui 界面变量

在文件 dialog.h 第 58 行定义.

7.1.4.3 vectorOfNumbersAndSigns

```
QVector<QVariant> Dialog::vectorOfNumbersAndSigns  [private]
```

vectorOfNumbersAndSigns 储存左数右数和符号，进行运算。

在文件 dialog.h 第 63 行定义.

该类的文档由以下文件生成:

- Calculator/dialog.h
- Calculator/dialog.cpp

7.2 EasyCalculator类 参考

The EasyCalculator class 简单计算器类

`#include <EasyCalculator.h>`

Public 类型

- enum class Operation { Addition , Subtraction , Multiplication , Division }
  The Operation enum

Public 成员函数

- EasyCalculator (QWidget *parent=nullptr)
  但凡*Qobject*子类，都必须在类的第一行放置该宏
- ~EasyCalculator ()
  Destruction Function

Private 槽

- void on_btn_plus_clicked ()
  on_btn_plus_clicked 槽函数，用来响应以下预定义的系统信号，函数名由系统维护，不得随意更改
- void on_btn_minus_clicked ()
  on_btn_minus_clicked 当 *on_btn_minus_clicked* 被点击时调用
- void on_btn_multiply_clicked ()
  on_btn_multiply_clicked
- void on_btn_divide_clicked ()
  on_btn_divide_clicked
- void on_calculateButton_clicked ()
  on_pushButton_clicked 计算按钮按下时的逻辑
- void on_clearButton_clicked ()
  on_clearButton_clicked 清除按钮按下时的逻辑

Private 属性

- Ui::EasyCalculator * ui
- Operation op = Operation::Addition
  op enum, default value is addition 系统界面对象指针，我们可以通过本指针访问所有放置在界面上的小部件*(Widgets)*

7.2.1 详细描述

The EasyCalculator class 简单计算器类

注解

【C语言/C++QT实现win10系统简易计算器！手把手教你开发～】 https://www.bilibili.com/video/BV1Fe4y1n7ng

在文件 EasyCalculator.h 第 14 行定义.

7.2.2 成员枚举类型说明

7.2.2.1 Operation

```
enum class EasyCalculator::Operation  [strong]
```

The Operation enum

枚举值

Addition	加法
Subtraction	减法
Multiplication	乘法
Division	除法

在文件 EasyCalculator.h 第 37 行定义.

7.2.3 构造及析构函数说明

7.2.3.1 EasyCalculator()

```
EasyCalculator::EasyCalculator (
            QWidget * parent = nullptr )
```

但凡Qobject子类，都必须在类的第一行放置该宏

EasyCalculator constructor 本对话框的构造函数

参数

parent	

在文件 EasyCalculator.cpp 第 4 行定义.

7.2.3.2 ~EasyCalculator()

```
EasyCalculator::~EasyCalculator ( )
```

Destruction Function

在文件 EasyCalculator.cpp 第 11 行定义.

7.2.4 成员函数说明

7.2.4.1 on_btn_divide_clicked

```
void EasyCalculator::on_btn_divide_clicked ( )  [private], [slot]
```

on_btn_divide_clicked

在文件 EasyCalculator.cpp 第 34 行定义.

7.2.4.2 on_btn_minus_clicked

```
void EasyCalculator::on_btn_minus_clicked ( )  [private], [slot]
```

on_btn_minus_clicked 当 on_btn_minus_clicked 被点击时调用

在文件 EasyCalculator.cpp 第 22 行定义.

7.2.4.3 on_btn_multiply_clicked

```
void EasyCalculator::on_btn_multiply_clicked ( )  [private], [slot]
```

on_btn_multiply_clicked

在文件 EasyCalculator.cpp 第 28 行定义.

7.2.4.4 on_btn_plus_clicked

```
void EasyCalculator::on_btn_plus_clicked ( )  [private], [slot]
```

on_btn_plus_clicked 槽函数，用来响应以下预定义的系统信号，函数名由系统维护，不得随意更改

在文件 EasyCalculator.cpp 第 17 行定义.

7.2.4.5 on_calculateButton_clicked

```
void EasyCalculator::on_calculateButton_clicked ( )  [private], [slot]
```

on_pushButton_clicked 计算按钮按下时的逻辑

注解

LineEdit 的使用 在本例中，我们使用三个 LineEdit 小部件供用户输入两个运算数， 并将计算结果显示在 第三个只读的 Line Edit widget 中。 Line Edit 是单行文本框，其中文本的数据类型是 QString。使用以下语句读取 LineEdit 中的 QString 对象并将其转化为浮点数。

```
lhs = this->ui->lhsLineEdit->text().toFloat();
rhs = this->ui->rhsLineEdit->text().toFloat();
this->ui->resultLineEdit->setText(QString::asprintf("%f", result));
```

setText 方法可以将参数 QString 字串设置到行文本框内显示。 本方法可以按格式字串（与 c 语言 printf 格式串相同）将 list_of_objects 转化为 QString 对象。 例如以下函数可以将整型变量 x 转化为 QString 类型对象返回。

```
QString::asprintf(\format string",list_of_objects);
QString::asprintf(\%d",x);
```

在文件 EasyCalculator.cpp 第 60 行定义.

7.2.4.6 on_clearButton_clicked

```
void EasyCalculator::on_clearButton_clicked ( )  [private], [slot]
```

on_clearButton_clicked 清除按钮按下时的逻辑

在文件 EasyCalculator.cpp 第 93 行定义.

7.2.5 类成员变量说明

7.2.5.1 op

```
Operation EasyCalculator::op = Operation::Addition  [private]
```

op enum, default value is addition 系统界面对象指针，我们可以通过本指针访问所有放置在界面上的小部件(Widgets)

在文件 EasyCalculator.h 第 83 行定义.

7.2.5.2 ui

```
Ui::EasyCalculator* EasyCalculator::ui  [private]
```

在文件 EasyCalculator.h 第 78 行定义.

该类的文档由以下文件生成:

- EasyCalculator/EasyCalculator.h
- EasyCalculator/EasyCalculator.cpp

Chapter 8

文件说明

8.1 Calculator/dialog.cpp 文件参考

```
#include "dialog.h"
#include "ui_dialog.h"
#include <QButtonGroup>
```

8.2 dialog.cpp

浏览该文件的文档.

```
00001 #include "dialog.h"
00002 #include "ui_dialog.h"
00003 #include <QButtonGroup>
00004 Dialog::Dialog(QWidget *parent)
00005     :  QDialog(parent)
00006     , ui(new Ui::Dialog)
00007 {
00008     ui->setupUi(this);
00009     iniUI();
00010 }
00011
00012 Dialog::~Dialog()
00013 {
00014     delete ui;
00015 }
00016
00017 void Dialog::iniUI() {
00018     // 1.  Put Button into a group
00019     auto buttonGroup = new QButtonGroup(this);
00020     auto btnList = findChildren<QPushButton*>();
00021     for (auto btn:  btnList) {
00022         buttonGroup->addButton(btn);
00023     }
00024
00025     // 2.  关联按钮组的点击信号
00026     connect(buttonGroup, &QButtonGroup::buttonClicked, this, &Dialog::onButtonGroupClicked);
00027
00028     this->vectorOfNumbersAndSigns.resize(5);
00029 }
00030
00031 void Dialog::onButtonGroupClicked(QAbstractButton* btn) {
00032     // Print in terminal
00033     qInfo() << btn->text();
00034
00035     float val = this->ui->lineEdit->text().toFloat();
00036
00037     // if the number pressed, print
00038
00042     QString name = btn->text();
00043
```

```
00044     // 根据按钮的点击，处理不同的逻辑
00045     // 如果是数字，直接显示
00046     if ((name >= "0" && name <= "9") || name == ".") {
00047         // 一开始显示零，输入一个数
00048         if(ui->lineEdit->text() == "0" && name != ".") {
00049             ui->lineEdit->clear();
00050         }
00051 
00052         // 如果点击数字键，输入框有数据，而且表达式框只有两个数据，重置输入的
00053         if(this->prevBtn == "+" || this->prevBtn == "-" || this->prevBtn == "*" || this->prevBtn ==
    "/" || this->prevBtn == "=") {
00054             this->ui->lineEdit->clear();
00055         }
00056         this->ui->lineEdit->insert(name);
00057     }
00058     else if (name == "+") {
00059         if(this->vectorOfNumbersAndSigns[2].isNull()) {
00060             // 把 lhs 数字和操作符存起来
00061             this->vectorOfNumbersAndSigns[0] = val;
00062             this->vectorOfNumbersAndSigns[1] = "+";
00063         }
00064     }
00065     else if (name == "-") {
00066         if(this->vectorOfNumbersAndSigns[2].isNull()) {
00067             // 把 lhs 数字和操作符存起来
00068             this->vectorOfNumbersAndSigns[0] = val;
00069             this->vectorOfNumbersAndSigns[1] = "-";
00070         }
00071     }
00072     else if (name == "*") {
00073         if(this->vectorOfNumbersAndSigns[2].isNull()) {
00074             // 把 lhs 数字和操作符存起来
00075             this->vectorOfNumbersAndSigns[0] = val;
00076             this->vectorOfNumbersAndSigns[1] = "*";
00077         }
00078     }
00079     else if (name == "/") {
00080         if(this->vectorOfNumbersAndSigns[2].isNull()) {
00081             // 把 lhs 数字和操作符存起来
00082             this->vectorOfNumbersAndSigns[0] = val;
00083             this->vectorOfNumbersAndSigns[1] = "/";
00084         }
00085     }
00086     else if (name == "=") {
00087         this->vectorOfNumbersAndSigns[2] = val;
00088         this->vectorOfNumbersAndSigns[3] = "=";
00089         if(this->vectorOfNumbersAndSigns[1] == "+") {
00090            this->vectorOfNumbersAndSigns[4] = this->vectorOfNumbersAndSigns[0].toFloat() +
    this->vectorOfNumbersAndSigns[2].toFloat();
00091         }
00092         else if(this->vectorOfNumbersAndSigns[1] == "-") {
00093             this->vectorOfNumbersAndSigns[4] = this->vectorOfNumbersAndSigns[0].toFloat() -
    this->vectorOfNumbersAndSigns[2].toFloat();
00094         }
00095         else if(this->vectorOfNumbersAndSigns[1] == "*") {
00096             this->vectorOfNumbersAndSigns[4] = this->vectorOfNumbersAndSigns[0].toFloat() *
    this->vectorOfNumbersAndSigns[2].toFloat();
00097         }
00098         else if(this->vectorOfNumbersAndSigns[1] == "/") {
00099             this->vectorOfNumbersAndSigns[4] = this->vectorOfNumbersAndSigns[0].toFloat() /
    this->vectorOfNumbersAndSigns[2].toFloat();
00100         }
00101         this->ui->lineEdit->setText(this->vectorOfNumbersAndSigns[4].toString());
00102         this->vectorOfNumbersAndSigns.clear();
00103     }
00104     else if (name == "C") {
00105         this->ui->lineEdit->clear();
00106         this->ui->expLineEdit->clear();
00107         this->vectorOfNumbersAndSigns.clear();
00108     }
00109     else if (name == "CE") {
00110         this->ui->lineEdit->clear();
00111         this->vectorOfNumbersAndSigns.pop_back();
00112     }
00113     else if (name == "DEL") {
00114         this->ui->lineEdit->setCursorPosition(ui->lineEdit->cursorPosition() - 1);
00115         this->ui->lineEdit->del();
00116     }
00117 
00118     // Display Sign and Number
00119     ui->expLineEdit->clear();
00120     for (auto var:  this->vectorOfNumbersAndSigns) {
00121         this->ui->expLineEdit->insert(var.toString());
00122     }
00123 
00124     this->prevBtn = name;
00125 }
```

```
00126
00127
00128
00129
00130
00131
```

8.3 Calculator/dialog.h 文件参考

```
#include "QtWidgets/qabstractbutton.h"
#include <QDialog>
#include <QtWidgets/QWidget>
#include <QVector>
```

类

- class Dialog

  The Dialog class UI 界面逻辑类

命名空间

- namespace Ui

8.4 dialog.h

浏览该文件的文档.

```
00001 /*
00002 * @Author:  Frank Chu
00003 * @Date:  2022-12-01 13:38:03
00004 * @LastEditors:  Frank Chu
00005 * @LastEditTime:  2022-12-01 14:27:04
00006 * @FilePath:  /Cpp/lab/Cpp-lab02-week12/Calculator/dialog.h
00007 * @Description:
00008 *
00009 * Copyright (c) 2022 by Frank Chu, All Rights Reserved.
00010 */
00011 #ifndef DIALOG_H
00012 #define DIALOG_H
00013
00014 #include "QtWidgets/qabstractbutton.h"
00015 #include <QDialog>
00016 #include <QtWidgets/QWidget>
00017 #include <QVector>
00018
00019 QT_BEGIN_NAMESPACE
00020 namespace Ui { class Dialog; }
00021 QT_END_NAMESPACE
00022
00026 class Dialog :  public QDialog
00027 {
00028     Q_OBJECT
00029
00030 public:
00035     Dialog(QWidget *parent = nullptr);
00036
00040     ~Dialog();
00041
00045     void iniUI();
00046 public slots:
00047
00052     void onButtonGroupClicked(QAbstractButton *btn);
00053
00054 private:
00058     Ui::Dialog *ui;
00059
00063     QVector<QVariant> vectorOfNumbersAndSigns;
00064
00068     QString prevBtn;
00069 };
00070 #endif // DIALOG_H
```

8.5 Calculator/main.cpp 文件参考

```
#include "dialog.h"
#include <QApplication>
```

函数

- int main (int argc, char ∗argv[ ])

8.5.1 函数说明

8.5.1.1 main()

```
int main (
            int argc,
            char * argv[] )
```

在文件 main.cpp 第 5 行定义.

8.6 main.cpp

浏览该文件的文档.

```
00001 #include "dialog.h"
00002
00003 #include <QApplication>
00004
00005 int main(int argc, char *argv[])
00006 {
00007     QApplication a(argc, argv);
00008     Dialog w;
00009     w.show();
00010     return a.exec();
00011 }
```

8.7 EasyCalculator/main.cpp 文件参考

```
#include "EasyCalculator.h"
#include <QApplication>
```

函数

- int main (int argc, char ∗argv[ ])

8.7.1 函数说明

8.7.1.1 main()

```
int main (
              int argc,
              char * argv[] )
```

在文件 main.cpp 第 5 行定义.

8.8 main.cpp

浏览该文件的文档.

```
00001 #include "EasyCalculator.h"
00002
00003 #include <QApplication>
00004
00005 int main(int argc, char *argv[])
00006 {
00007     QApplication a(argc, argv);
00008     EasyCalculator w;
00009     w.show();
00010     return a.exec();
00011 }
```

8.9 EasyCalculator/EasyCalculator.cpp 文件参考

```
#include "EasyCalculator.h"
#include "ui_EasyCalculator.h"
```

8.10 EasyCalculator.cpp

浏览该文件的文档.

```
00001 #include "EasyCalculator.h"
00002 #include "ui_EasyCalculator.h"
00003
00004 EasyCalculator::EasyCalculator(QWidget *parent)
00005     :  QDialog(parent)
00006     , ui(new Ui::EasyCalculator)
00007 {
00008     ui->setupUi(this);
00009 }
00010
00011 EasyCalculator::~EasyCalculator()
00012 {
00013     delete ui;
00014 }
00015
00016
00017 void EasyCalculator::on_btn_plus_clicked()
00018 {
00019     this->op = Operation::Addition;
00020 }
00021
00022 void EasyCalculator::on_btn_minus_clicked()
00023 {
00024     this->op = Operation::Subtraction;
```

```
00025 }
00026
00027
00028 void EasyCalculator::on_btn_multiply_clicked()
00029 {
00030     this->op = Operation::Multiplication;
00031 }
00032
00033
00034 void EasyCalculator::on_btn_divide_clicked()
00035 {
00036     this->op = Operation::Division;
00037 }
00038
00060 void EasyCalculator::on_calculateButton_clicked()
00061 {
00062     float lhs, rhs, result;
00063     lhs = this->ui->lhsLineEdit->text().toFloat();
00064     rhs = this->ui->rhsLineEdit->text().toFloat();
00065 //    if(this->op == Operation::Addition) {
00066 //        result = lhs + rhs;
00067 //        this->ui->resultLineEdit->setText(QString::asprintf("%f", result));
00068 //    }
00069 //    else if(this->op == Operation::Subtraction) {
00070 //        result = lhs - rhs;
00071 //    }
00072
00073     switch (this->op) {
00074     case Operation::Addition:
00075         result = lhs + rhs;
00076         break;
00077     case Operation::Subtraction:
00078         result = lhs - rhs;
00079         break;
00080     case Operation::Multiplication:
00081         result = lhs * rhs;
00082         break;
00083     case Operation::Division:
00084         result = lhs / rhs;
00085         break;
00086     default:
00087         break;
00088     }
00089     this->ui->resultLineEdit->setText(QString::asprintf("%f", result));
00090 }
00091
00092
00093 void EasyCalculator::on_clearButton_clicked()
00094 {
00095     this->ui->lhsLineEdit->clear();
00096     this->ui->rhsLineEdit->clear();
00097     this->ui->resultLineEdit->clear();
00098 }
00099
```

8.11 EasyCalculator/EasyCalculator.h 文件参考

```
#include <QDialog>
```

类

- class EasyCalculator

  The EasyCalculator class 简单计算器类

命名空间

- namespace Ui

8.12 EasyCalculator.h

浏览该文件的文档.

```
00001 #ifndef EASYCALCULATOR_H
00002 #define EASYCALCULATOR_H
00003
00004 #include <QDialog>
00005
00006 QT_BEGIN_NAMESPACE
00007 namespace Ui { class EasyCalculator; }
00008 QT_END_NAMESPACE
00009
00014 class EasyCalculator :  public QDialog
00015 {
00019     Q_OBJECT
00020
00021 public:
00022
00027     EasyCalculator(QWidget *parent = nullptr);
00028
00032     ~EasyCalculator();
00033
00037     enum class Operation {
00038         Addition,
00039         Subtraction,
00041         Multiplication,
00043         Division
00044     };
00045
00046 private slots:
00050     void on_btn_plus_clicked();
00051
00055     void on_btn_minus_clicked();
00056
00060     void on_btn_multiply_clicked();
00061
00065     void on_btn_divide_clicked();
00066
00070     void on_calculateButton_clicked();
00071
00075     void on_clearButton_clicked();
00076
00077 private:
00078     Ui::EasyCalculator *ui;
00079
00083     Operation op = Operation::Addition;
00084 };
00085
00086 #endif // EASYCALCULATOR_H
```

8.13 README.md 文件参考

Index