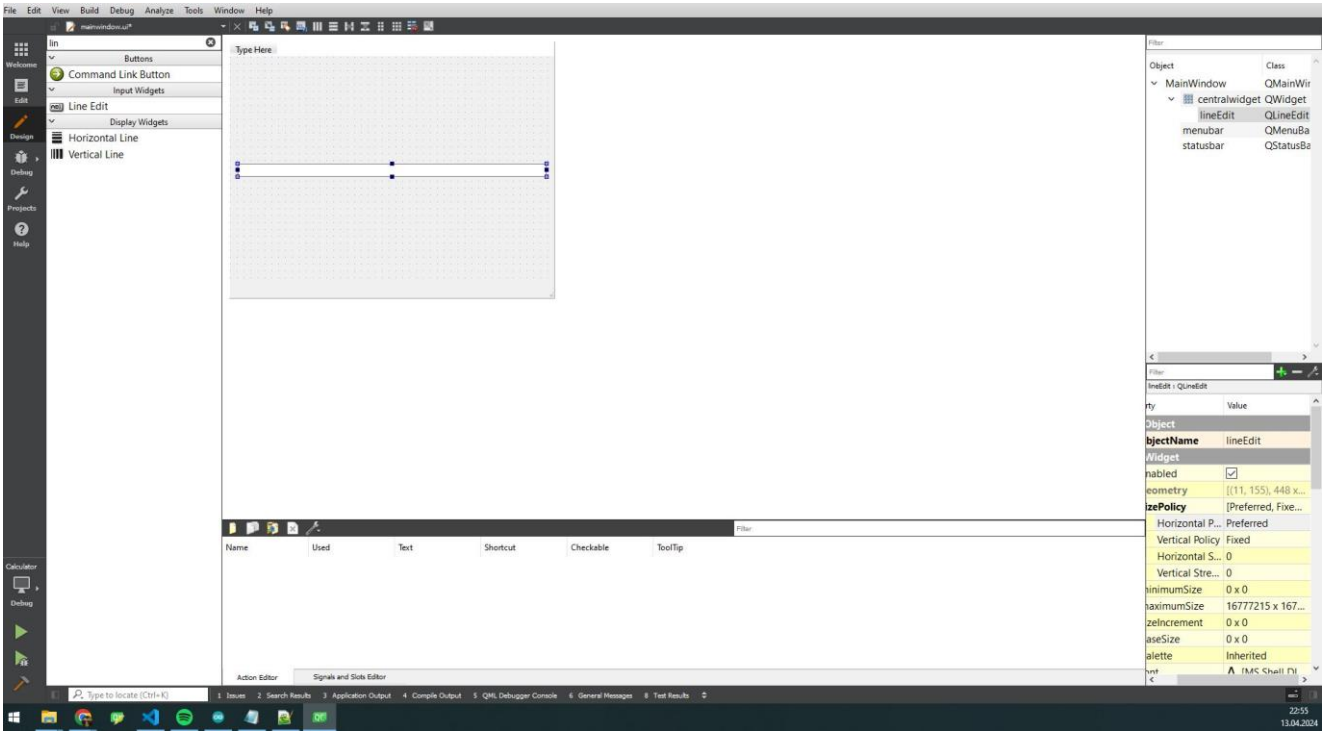
CALCULATOR

Amaç: Bu PDF dosyası, QT Creator kullanarak C++ ile temel bir hesap makinesi yapımını anlatmaktadır.

Kodları içeren GitHub deposunun bağlantısı şu şekildedir:
[https://github.com/Evrenotur/QT]

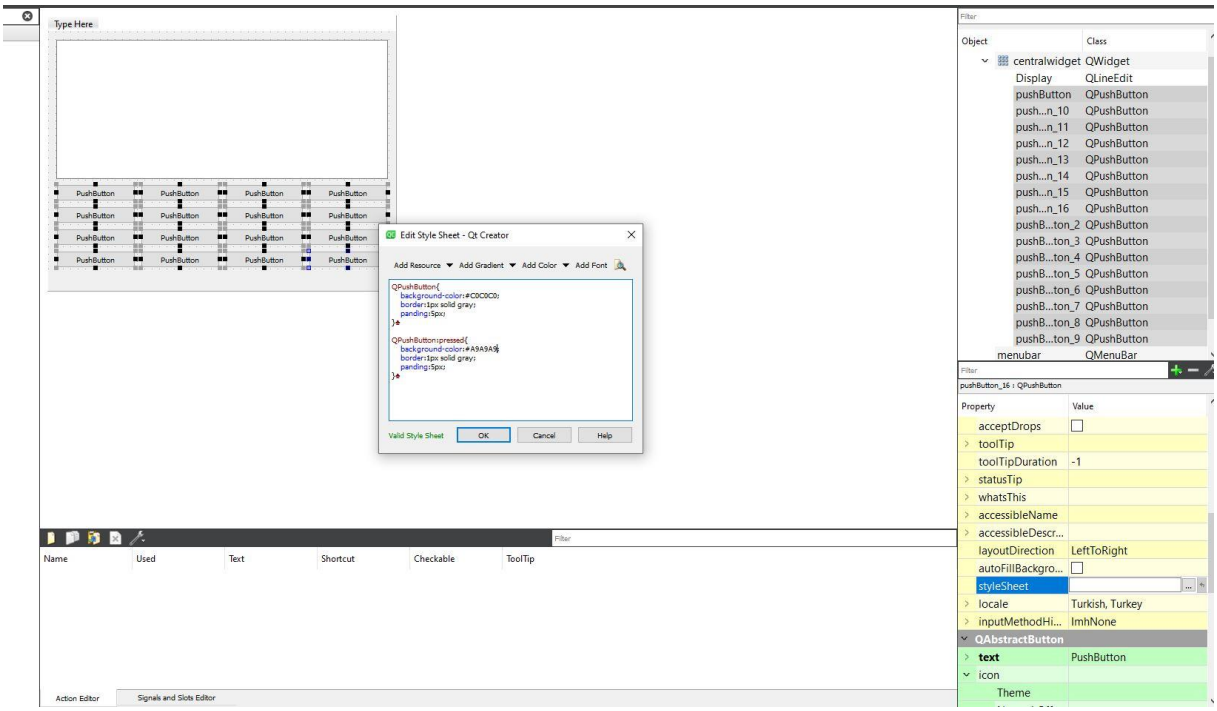
Şekil 1'de gösterildiği gibi, toolbox'dan lineedit aracını alıp pencereye sürükleyin.



Şekil 1

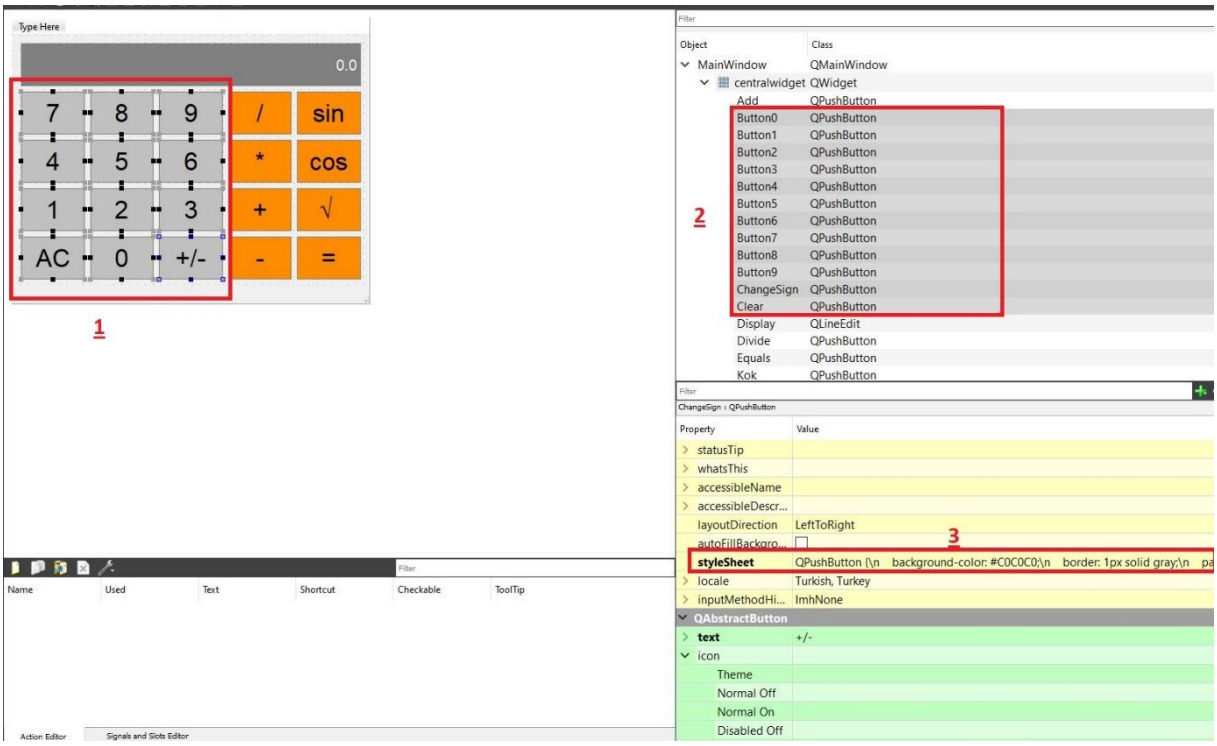
Şekil 2'de gösterildiği gibi, toolbox'dan pushbutton aracını alıp pencereye sürükleyin.

Not:Toplam 20 adet pencerede pushbutton olması gerekmektedir.



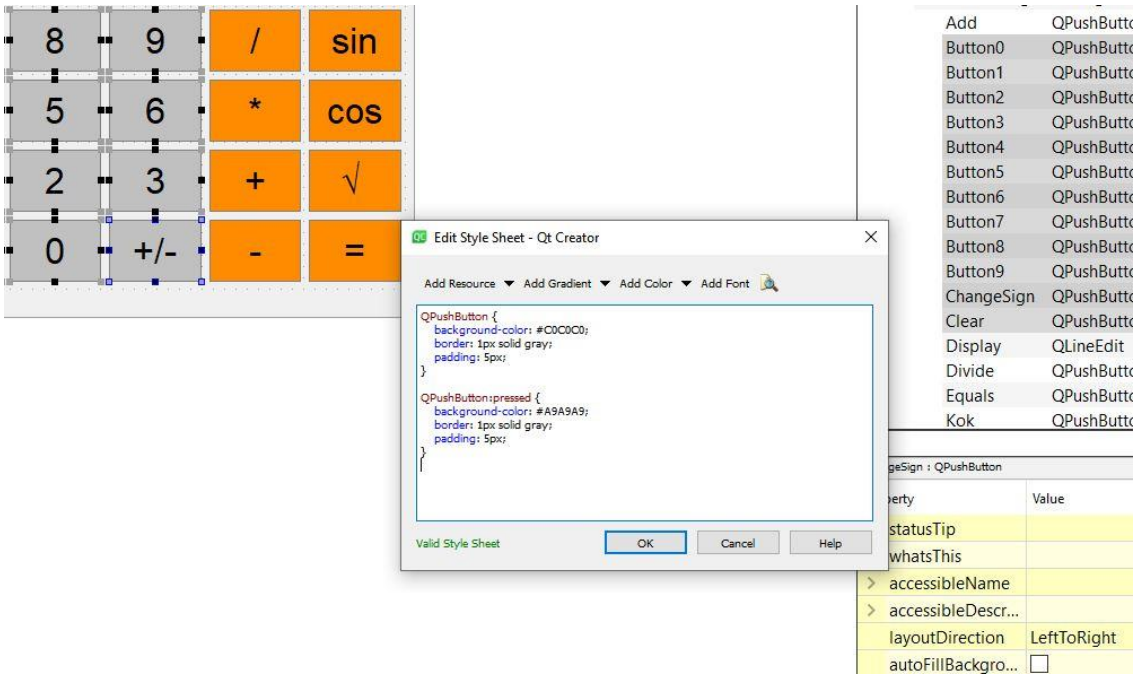
Şekil 2

Şekil 3'te belirtilen 1 numaralı kırmızı kutu içindeki gibi butonları Ctrl tuşuna basarak seçin. Ardından seçtiğiniz butonların isimlerini 2 numaralı kırmızı kutucuktaki gibi değiştirin. Son olarak, 3 numaralı kırmızı kutucukla seçili olan "styleSheet" parametresine çift tıklayın.



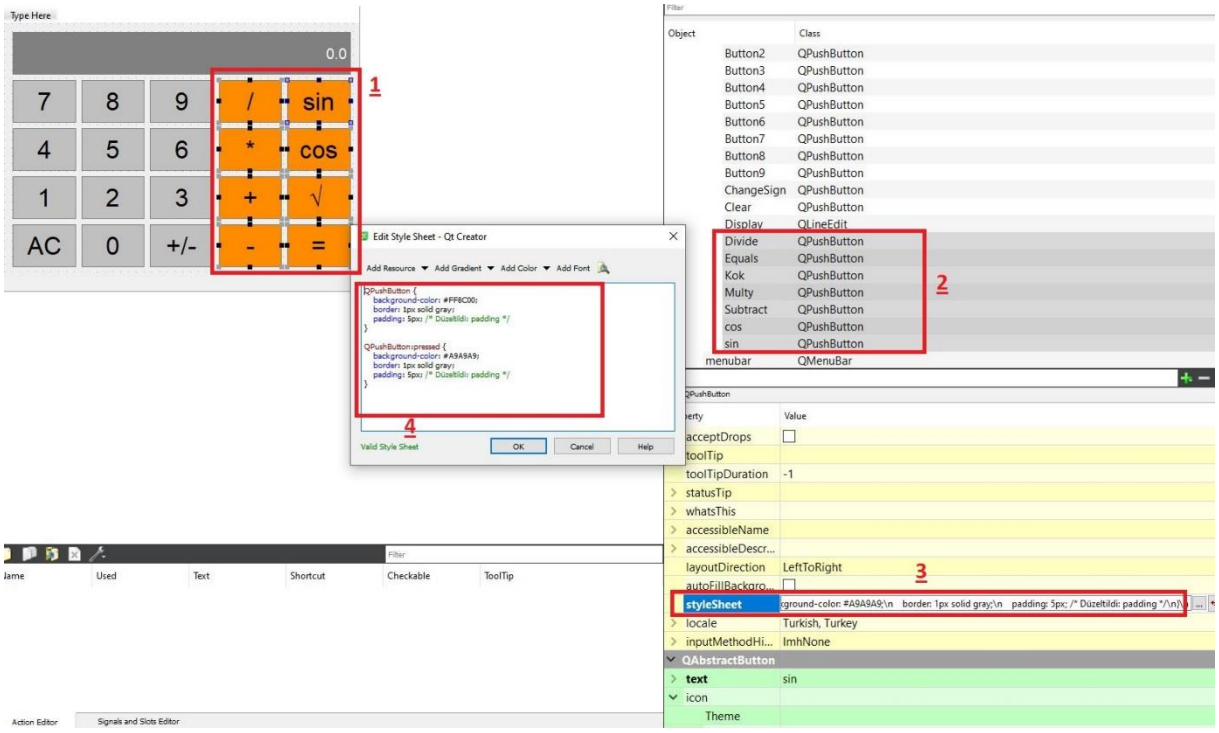
Şekil 3

"styleSheet" parametresine çift tıkladıktan sonra Şekil 4'te belirtilen pencere açılacaktır. Bu pencereye, butonların renk aralık özelliklerini içeren kodu aşağıdaki gibi ekleyin.



Şekil 4

Şekil 5'te 1 numaralı kırmızı kutuda belirtildiği gibi Ctrl tuşuna basarak pushbuttonları seçin. İkinci adım olarak, 2 numaralı kırmızı kutu içerisinde belirtilen butonların isimlerini şekildeki gibi değiştirin. Üçüncü adım olarak, "styleSheet" parametresine çift tıklayarak açılan Edit Style Sheet ekranında, 4 numaralı kırmızı kutu içerisindeki butonların renk ve görünüş özelliklerini içeren kodları ekleyin.



Şekil 5

Eğer matematiksel işlemleri gerçekleştiren kodları mainwindow.cpp dosyasına yazacaksak, öncelikle mainwindow.h dosyasında gerekli fonksiyon prototiplerini tanımlamamız gerekmektedir. Daha sonra bu fonksiyonların gerçek kodlarını mainwindow.cpp dosyasına ekleyeceğiz.

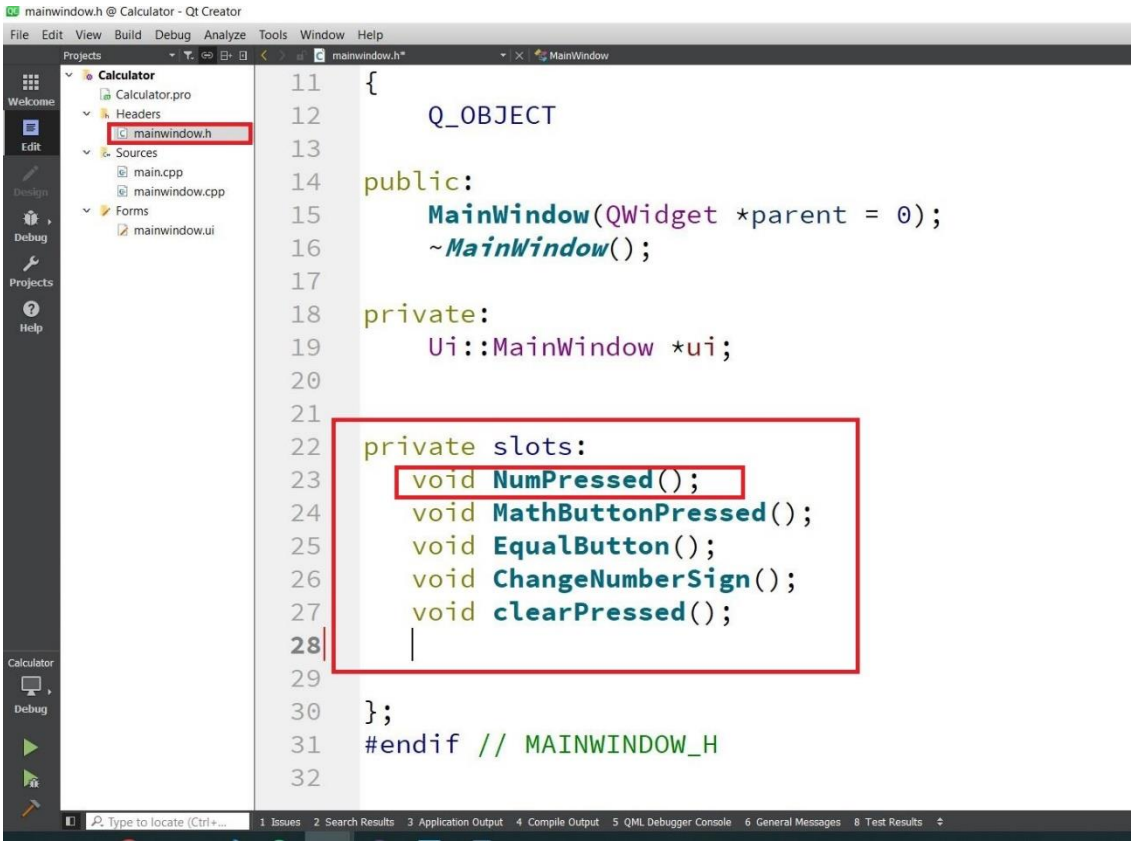
Örneğin, eğer "MathButtonPressed()" adlı bir fonksiyonumuz varsa ve bu fonksiyon butonlardan biri tıklandığında çağrılacaksa, mainwindow.h dosyasına bu fonksiyonun prototipini ekleriz:

```
public slots:
    void MathButtonPressed();
```

Daha sonra, mainwindow.cpp dosyasına bu fonksiyonun gerçek kodlarını ekleriz:

```
void MainWindow::MathButtonPressed() {
    // Matematiksel işlemleri gerçekleştirme kodları buraya gelecek
}
```

Şekil 6'da belirtildiği gibi, Headers dosyası içerisindeki mainwindow.h dosyası içerisine kırmızı kutu içerisinde beklenen fonksiyonları tanımlayın. Bu fonksiyonlar, gerekli matematiksel işlemleri yapacaktır.



Şekil 6

Şekil 7'de belirtildiği gibi, Sources dosyası içerisindeki mainwindow.cpp dosyası içerisinde, kırmızı kutu içerisinde tanımlanan fonksiyonun içerisine gerekli kodlar yazılmıştır.

Şekil 7

```
connect(ui->Add, SIGNAL(released()), this, SLOT(MathButtonPressed()));
```

Bu kod, bir sinyal-slot bağlantısı kurar. Qt framework'ünde sinyal-slot mekanizması, nesneler arasında iletişim kurmanın bir yoludur.

Bu kodun anlamı şu şekildedir:

- **`ui->Add`**: Arayüzdeki "Add" adlı bir nesneye işaret eder. Bu genellikle bir buton gibi bir grafik öğesidir.
- **`SIGNAL(released())`**: "released" adlı bir sinyal gönderir. Bu sinyal, kullanıcı butonu bıraktığında tetiklenir.
- **`this`**: Bu, mevcut sınıfın bir işaretçisidir. Bu durumda, sinyal-slot bağlantısının yapıldığı sınıfı ifade eder.
- **`SLOT(MathButtonPressed())`**: "MathButtonPressed()" adlı bir slotu çağırır. Bir slot, bir sinyal alındığında çağrılan bir işlevdir. Bu durumda, "MathButtonPressed()" adlı bir fonksiyonun çağrılmasını bekler.

Yani, bu kod, "Add" butonunun serbest bırakılması (released) durumunda "MathButtonPressed()" adlı bir fonksiyonu çağırır.

Şekil 8'de, tasarım penceresindeki butonların fonksiyonlara bağlama işlemi yapılmıştır.

```
/*Kütüpahaler Tanımlanır*/
#include "mainwindow.h"
#include "ui_mainwindow.h"
#include <cmath>
/*Değişkenler Tanımlanır*/
double calcVal = 0.0;
bool divTrigger = false;
bool multTrigger = false;
bool addTrigger = false;
bool subTrigger = false;
bool sinTrigger = false;
bool cosTrigger = false;
bool sqrtTrigger = false;
MainWindow::MainWindow(QWidget *parent)
    : QMainWindow(parent)
    , ui(new Ui::MainWindow)
{
    ui->setupUi(this);

    //Değişkenin değeri "calcVal" olarak ayarlanıyor ve bu değer metin kutusuna ("Display") yazılıyor.
    ui->Display->setText(QString::number(calcVal));

    // 10 adet QPushButton işaretçisi tutacak bir dizi tanımlanıyor.
    QPushButton *numButtons[10];

    for(int i = 0; i < 10; ++i)
    {
        QString butName = "Button" + QString::number(i); // Her bir düğme için bir isim oluşturuluyor, örneğin "Button 0", "Button 1" gibi.
        numButtons[i] = MainWindow::findChild<QPushButton*>(butName); // Her bir düğmeyi bulmak için MainWindow sınıfının findChild fonksiyonu kullanılıyor
        connect(numButtons[i], SIGNAL(released()), this, SLOT(NumPressed())); // Her bir düğmenin released sinyali NumPressed fonksiyonu ile bağlanıyor.
    }
     connect(ui->Add, SIGNAL(released()), this, SLOT(MathButtonPressed()));
     connect(ui->Subtract, SIGNAL(released()), this, SLOT(MathButtonPressed()));
     connect(ui->Multy, SIGNAL(released()), this, SLOT(MathButtonPressed()));
     connect(ui->Divide, SIGNAL(released()), this, SLOT(MathButtonPressed()));
      connect(ui->sin, SIGNAL(released()), this, SLOT(MathButtonPressed()));
      connect(ui->cos, SIGNAL(released()), this, SLOT(MathButtonPressed()));
      connect(ui->Kok, SIGNAL(released()), this, SLOT(MathButtonPressed()));
     connect(ui->Equals, SIGNAL(released()), this, SLOT(EqualButton()));
     connect(ui->ChangeSign, SIGNAL(released()), this, SLOT(ChangeNumberSign()));
       connect(ui->Clear, SIGNAL(released()), this, SLOT(clearPressed()));
}
```

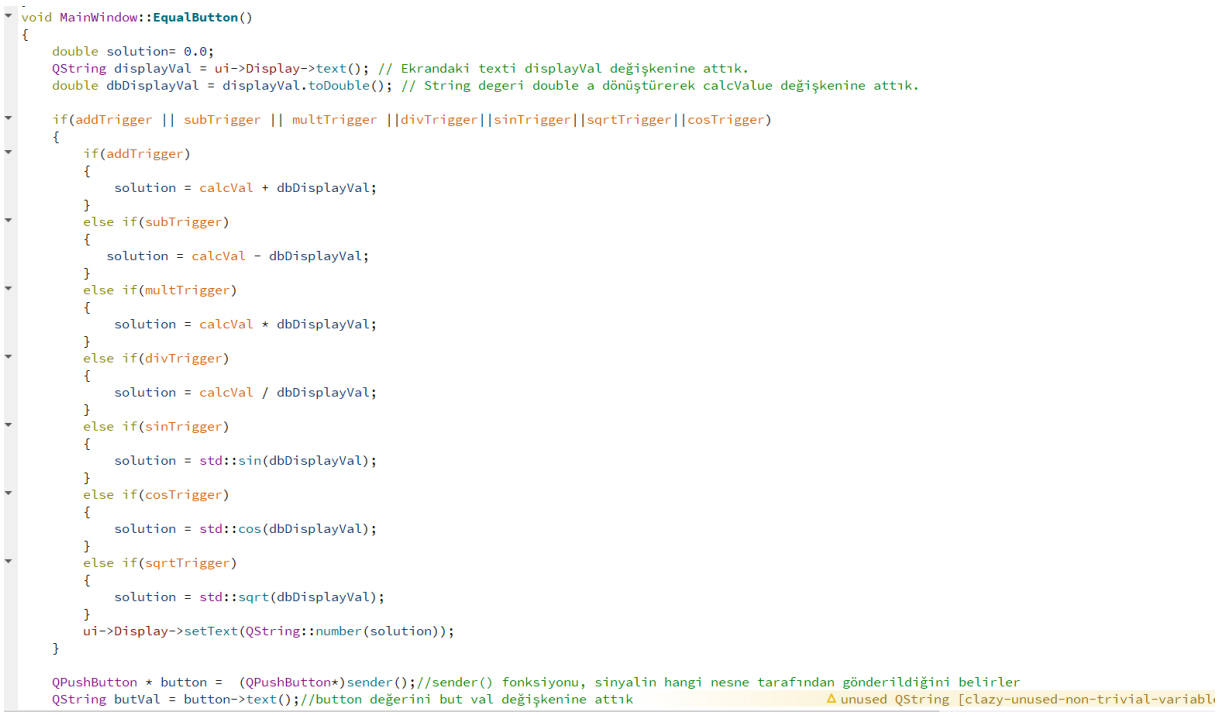
Şekil 8

Şekil 9'da, MathButtonPressed() fonksiyonu içerisinde kullanıcının hangi matematiksel işlemi yaptığı algılanmaktadır.

```
}
void MainWindow::MathButtonPressed()
{
divTrigger=false;
multTrigger=false;
addTrigger = false;
subTrigger = false;
sinTrigger = false;
cosTrigger = false;
sqrtTrigger = false;
QString displayVal = ui->Display->text(); // Ekrandaki texti displayVal değişkenine attık.
calcVal = displayVal.toDouble(); // String degeri double a dönüştürerek calcValue değişkenine attık.
QPushButton * button =  (QPushButton*)sender();//sender() fonksiyonu, sinyalin hangi nesne tarafından gönderildiğini belirler
QString butVal = button->text();//button değerini but val değişkenine attık
if(QString::compare(butVal,"/",Qt::CaseInsensitive) == 0)
{
 // Eğer butVal, "/" karakter dizisine eşitse (büyük/küçük harf duyarlılığı gözetmeksizin)...
    divTrigger = true;
}
else if(QString::compare(butVal,"*",Qt::CaseInsensitive) == 0)
{
 // Eğer butVal, "*" karakter dizisine eşitse (büyük/küçük harf duyarlılığı gözetmeksizin)...
   multTrigger = true;
}
else if(QString::compare(butVal,"+",Qt::CaseInsensitive) == 0)
{
 // Eğer butVal, "+" karakter dizisine eşitse (büyük/küçük harf duyarlılığı gözetmeksizin)...
   addTrigger = true;
}
else if(QString::compare(butVal,"-",Qt::CaseInsensitive) == 0)
{
 // Eğer butVal, "-" karakter dizisine eşitse (büyük/küçük harf duyarlılığı gözetmeksizin)...
  subTrigger = true;
}
else if(QString::compare(butVal,"sin",Qt::CaseInsensitive) == 0)
{
 // Eğer butVal, "-" karakter dizisine eşitse (büyük/küçük harf duyarlılığı gözetmeksizin)...
  sinTrigger = true;
}
else if(QString::compare(butVal,"cos",Qt::CaseInsensitive) == 0)
{
```

Şekil 9

Şekilde 10'da, EqualButton() basıldığında, kullanıcının yani eşittir işaretine bastığında hangi sonucun nasıl belirleneceği işlemleri belirtilmektedir.



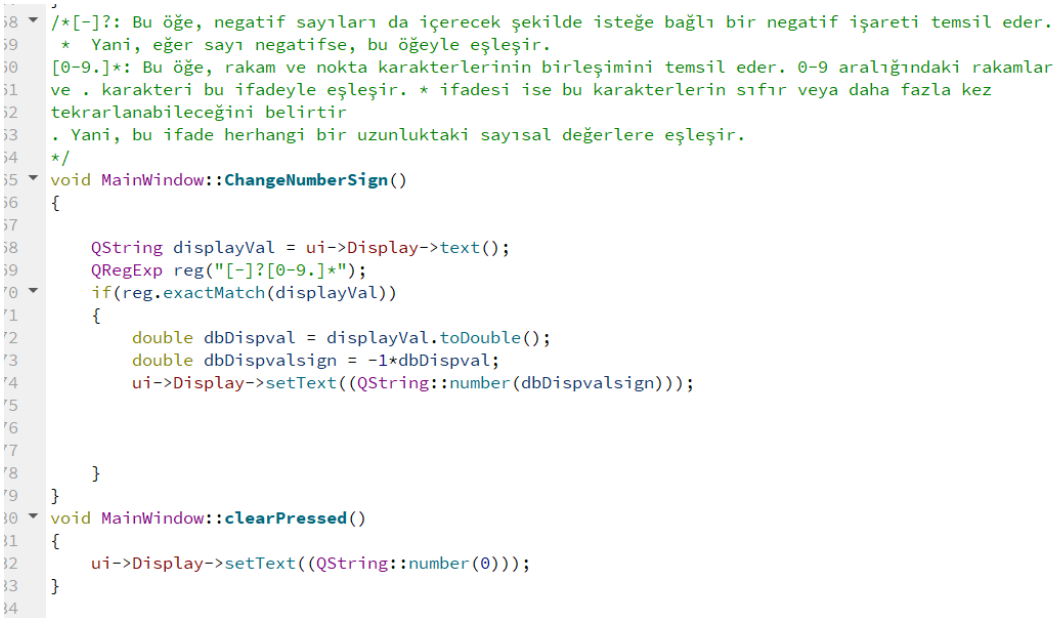
```
void MainWindow::EqualButton()
{
    double solution= 0.0;
    QString displayVal = ui->Display->text(); // Ekrandaki texti displayVal değişkenine attık.
    double dbDisplayVal = displayVal.toDouble(); // String degeri double a dönüştürerek calcValue değişkenine attık.

    if(addTrigger || subTrigger || multTrigger ||divTrigger||sinTrigger||sqrtTrigger||cosTrigger)
    {
        if(addTrigger)
        {
            solution = calcVal + dbDisplayVal;
        }
        else if(subTrigger)
        {
           solution = calcVal - dbDisplayVal;
        }
        else if(multTrigger)
        {
            solution = calcVal * dbDisplayVal;
        }
        else if(divTrigger)
        {
            solution = calcVal / dbDisplayVal;
        }
        else if(sinTrigger)
        {
            solution = std::sin(dbDisplayVal);
        }
        else if(cosTrigger)
        {
            solution = std::cos(dbDisplayVal);
        }
        else if(sqrtTrigger)
        {
            solution = std::sqrt(dbDisplayVal);
        }
        ui->Display->setText(QString::number(solution));
    }

    QPushButton * button =  (QPushButton*)sender();//sender() fonksiyonu, sinyalin hangi nesne tarafından gönderildiğini belirler
    QString butVal = button->text();//button değerini but val değişkenine attık                    △ unused QString [clazy-unused-non-trivial-variabl
```

Şekil 10

Şekil 11'de, ChangeNumberSign() fonksiyonu, işaretin nasıl değiştirildiğini belirtmektedir. Ayrıca, clearPressed() fonksiyonunun ekranı nasıl temizlediği içeren kod belirtilmektedir.



```
/*[-]?: Bu öğe, negatif sayıları da içerecek şekilde isteğe bağlı bir negatif işareti temsil eder.
 *  Yani, eğer sayı negatifse, bu öğeyle eşleşir.
[0-9.]*: Bu öğe, rakam ve nokta karakterlerinin birleşimini temsil eder. 0-9 aralığındaki rakamlar
ve . karakteri bu ifadeyle eşleşir. * ifadesi ise bu karakterlerin sıfır veya daha fazla kez
tekrarlanabileceğini belirtir
. Yani, bu ifade herhangi bir uzunluktaki sayısal değerlere eşleşir.
*/
void MainWindow::ChangeNumberSign()
{

    QString displayVal = ui->Display->text();
    QRegExp reg("[-]?[0-9.]*");
    if(reg.exactMatch(displayVal))
    {
        double dbDispval = displayVal.toDouble();
        double dbDispvalsign = -1*dbDispval;
        ui->Display->setText((QString::number(dbDispvalsign)));



    }
}
void MainWindow::clearPressed()
{
    ui->Display->setText((QString::number(0)));
}
```

Şekil 11