Step 1: calculator.ui

At first I created the calculator design from QT creator. I created the Screen object using the QLineEdit class and displayed the entered values as a QString here as the keys are pressed. I set the color of the screen with HTML color codes using styleSheet property from QWidget.

Then I created buttons for hexadecimal numbers, + - = operators and clr using the QPushButton class. Then I created buttons for hexadecimal numbers, + - = operators and clr using the QPushButton class. I also edited the color and the text that will be written on them from the QWidget section. Drag and drop operations in Designer mode created the .ui file.

Step 2: main.cpp

In the main.cpp file defined in the project file I opened, there is the QAplication class that manages the GUI applications, the w object defined in the Calculator class, which is the main object, and the function that shows its widgets.

Step 3: calculator.h

In this file, we have declared the functions that signal when the keys are pressed.

Step 4: calculator.cpp

In this file, we first defined which function will signal when the button is pressed. Then we sequentially determined the content of the functions.

ClearPressed(): If you press the clr key on the calculator, it should reset the screen, so we print 0 as a string to the Screen object with the setText function.

NumberPressed(): In this function, which works when one of the hexadecimal numbers is pressed, the button object is defined first. Then we get the value written in this button with buttonVal. With screenVal, we get the value written on the screen at that moment. If there is a 0 on the screen, we print the currently printed number on the screen. If it is not zero, we add the number to the continuation of the current number.

OprButtonPressed(): In this function, we have defined the operations to be performed when + - = keys are pressed. If there is 0 on the screen and the incoming operation is =, we do not make any changes. but if + or - operations come it means positive or negative input so we put one of these operation symbols instead of 0.

If anything other than 0 is written on the screen, we get the value that is written at that moment. If that string value does not contain + or -, it means that there is a number, so we add the operator to the end of that number if operator = is not, but if operator = we do not need to do anything.

If the current value contains + or -, we first check if this operator is at the end, if it is the last, the new operator will be valid and we will update the value on the screen.

If the current value contains + or -, there may be either an operation or a signed input at the beginning, so if the first operator is -, we create a new positive string without taking it. We define a bool named negNum in order to be able to handle this negative number correctly later. if this positive string contains + then there is an addition operation inside. In order to obtain the numbers separately, we separate the string from + with the split function and convert it to integer numbers in base 16 with the built-in toInt() function. Then, if the bool we hold is true, we multiply the first number by -1 and do the addition. We convert the result of the addition operation to hexadecimal and print it as a string. We do the same for the - operator.

If positive value(string) does not contain + or - and the = operator comes, we print the current value to the screen.

if the positive value does not contain + or - and something other than the = operator came, there is a signed input at the beginning and the + or - operator has arrived, so we add the new operator to the end of the current value.