# Calculations

Carrying on with preparing our Calculator we need to update our command to store the information that we need and to take these pieces of information into account

1. Update **operatorPressed** to store the chosen operator, store the current total and start a new number

2. Update **clearPressed** to clear the display, update the clear mode (AC or C) and start a new number

3. Update **numberPressed** - we make decisions of what to do when a number is pressed, depending on what is pressed and how many numbers are currently being displayed

4. Add a command **equalsPressed** to do the calculation

# Updating operatorPressed

When an operator is pressed we need to store the chosen operator, store the current total and start a new number:

1. We want to leave the code we currently have as we still want the correct border to show

2. If an operator button is currently selected when a new button is pressed, call the command **equalsPressed** so that any calculations that need to be done are done before we update the current operators

3. Store the selected operator in the script local **sOperator**

4. Store the contents of field **display** in the script local **sCurrentTotal**

5. After a user has pressed an operator we start a new number - so **sNewNumber** should be true

```
command operatorPressed pOperator
  set the showBorder of button "divide" to false
1 set the showBorder of button "multiply" to false
  set the showBorder of button "minus" to false
  set the showBorder of button "plus" to false

2 if sOperator is not empty then
    equalsPressed
  end if

1 if pOperator is not empty then
    set the showBorder of button pOperator to true
  end if

3 put pOperator into sOperator

4 put field "display" into sCurrentTotal

5 put true into sNewNumber
end operatorPressed
```

# Updating clearPressed

We need to update the **clearPressed** command to do the following:

1. Clear the display
2. Switch between two modes: **AC** and **C**
3. If we are in **AC** mode then we clear the current number, the running total and the chosen operator
4. If we are in **C** mode, we only want to clear the current display (which we did in step 1) and we switch to **AC** mode
5. Tell LiveCode that the next number typed in is a new number using the script local

```
command clearPressed
1   put 0 into field "display"

2   if the label of button "clear" is "AC" then
      if sOperator is not empty then
3       set the showBorder of button sOperator to false
      end if
      put 0 into sCurrentTotal
      put empty into sOperator
    else
4     set the label of button "clear" to "AC"
    end if

5   put true into sNewNumber
end clearPressed
```

# Updating numberPressed

Hopefully you will be familiar with how calculator's typically work! One of the things that calculator's do is to restrict the number of digits that can be entered and displayed.

When updating the command **numberPressed** in the card script, the first thing we want to do is stop the user from entering too many numbers:

1. Check if the there are 9 or more characters currently in the field **display**

2. If there are, then we want to exit the command straight away so that no further code is executed

```
command numberPressed pNumber

1  if the length of field "display" >= 9 then

     exit numberPressed

   end if

2
```

… code continued on next slide

# Updating numberPressed

If the user is entering a new number and they start by pressing 0 then nothing should happen.

1. We need to check if the user is entering a new number by checking whether **sNewNumber** is true. We ALSO need to check if the user is trying to enter 0.

2. If both of the above conditions are met, then we don't want to do anything - so once again we exit the command at this stage so that no further code is executed.

```
1   if sNewNumber is true and pNumber is 0 then
2       exit numberPressed
    end if
```

… code continued on next slide

# Updating numberPressed

You should also be aware that a number cannot have more than one decimal point. If you have a calculator to hand (there should be one on your desktop), try entering more than one decimal point. It cannot be done!

We need to make sure that our Calculator App does not allow users to enter multiple decimal points either:

1. First check if the user has chosen the decimal point AND if there is already a decimal point in the display

2. If the user is attempting to enter multiple decimal points then we exit the command now so that no more code is executed

```
1  if pNumber is "." and the text of field "display" contains "." then
2      exit numberPressed
   end if
```

… code continued on next slide

# Updating numberPressed

A further calculator feature that we must not forget to include in our own Calculator App is the difference between **AC** and **C**.

When there are numbers in the calculator, the label of the **clear** button is “C”, otherwise it is “AC”.

If we have got to this stage in the execution of the handler, then we will be entering a number into our display:

1. Check if the label of the **clear** button is currently “AC”

2. If it is, then change it to “C”

```
1  if the label of button "clear" is "AC" then
2      set the label of button "clear" to "C"
   end if
```

… code continued on next slide

# Updating numberPressed

Finally, we have got to the stage where we display the entered number!

1. If this is a continuing number or a decimal point then add the number to the end of the field, otherwise it is a new number and we replace the contents of the display with the new number

2. We are now no longer entering a new number, so update the **sNewNumber** script local variable to reflect this

Phew! That was a lot of code to add! Switch to run mode and see what happens when you enter numbers now.

```
if sNewNumber is false or pNumber is "." then
    put pNumber after field "display"
1
else
    put pNumber into field "display"
end if
2   put false into sNewNumber
end numberPressed
```

# Adding Command equalsPressed

Finally, we need to add a command **equalsPressed** to the card script so that the user can actually do the calculation. The **equalsPressed** command will

1. check an operator has been chosen
2. store the current value
3. do the calculation using
   a. current total
   b. current operator
   c. current value
4. update the display field
5. clear the operator(as it has been used)
6. start a new number

# Command equalsPressed

In the card script, add a command **equalsPressed** that:

1. check an operator has been chosen - if the current operator is empty, then there is nothing to calculate, so exit the command without doing anything

2. Store the value in field **display** in a temporary variable **tCurrentValue**

```
command equalsPressed
  local tCurrentValue

  if sOperator is empty then
    exit equalsPressed
  end if

  put field "display" into tCurrentValue
```

1

2

… code continued on next slide

# Command equalsPressed

Next the **equalsPressed** command will take the 3 pieces of information

1. **sCurrentTotal**
2. **sOperator**
3. **tCurrentValue**

Use an **if** statement to perform the correct calculation and update the **sCurrentTotal** variable and the “display” field.

```
if sOperator is "divide" then
   put sCurrentTotal / tCurrentValue into sCurrentTotal
else if sOperator is "multiply" then
   put sCurrentTotal * tCurrentValue into sCurrentTotal
else if sOperator is "minus" then
   put sCurrentTotal - tCurrentValue into sCurrentTotal
else if sOperator is "plus" then
   put sCurrentTotal + tCurrentValue into sCurrentTotal
end if

put sCurrentTotal into field "display"
```

… code continued on next slide

# Command equalsPressed

Finally the **equalsPressed** command will

1. Reset the operator border
2. Reset the **sOperator** variable because the operator has been used in the calculation
3. Put true into **sNewNumber**

Once a calculation has been done the next time we press a number we don't want it to be appended to the result displayed in the "display" field, so we tell the app to start a new number.

```
1  set the showBorder of button sOperator to false

2  put empty into sOperator

3  put true into sNewNumber

   end equalsPressed
```

# Testing the Calculator

You should now be able to test out your calculator. Go into **Run** mode and try it.

Don’t forget to save your work!