Simple Statistic Calculator

21-NOV-2016

Report to accompany my simple statistic calculator program

Niall Hunt

CS1021
Intro to Computing 1

Assignment #1

Contents

Stage 1: Console Input

Solution

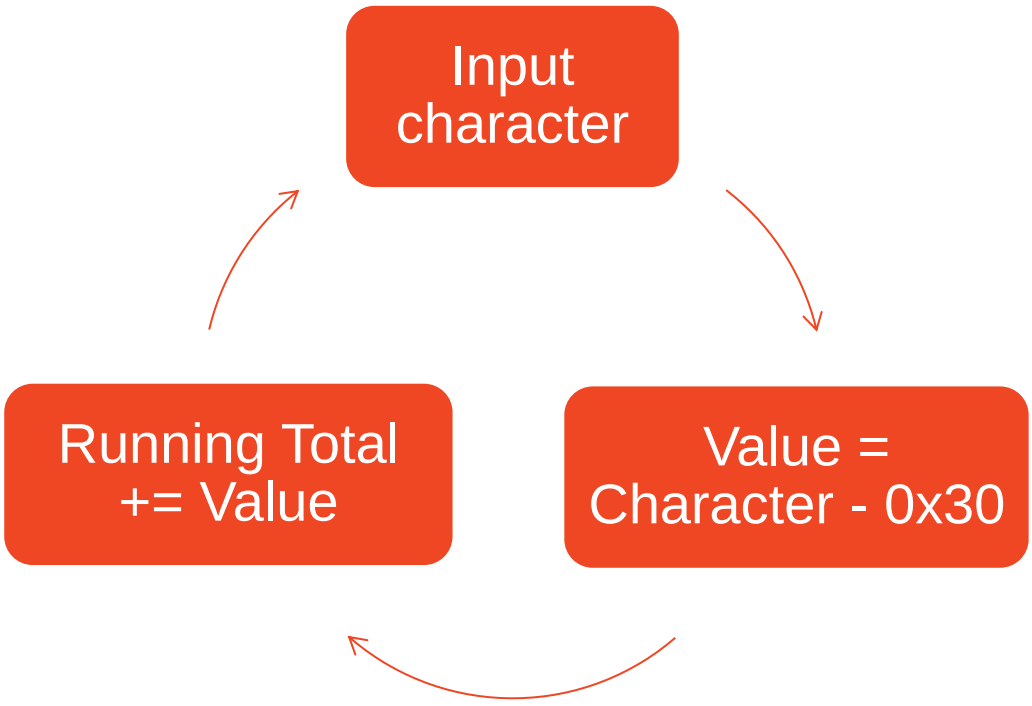
Stage 1

In stage 1 of this assignment I created a program to read the input of a multi-digit number from the user.

To do this I use a loop to take in the input (using BL getkey) and calculate its true value. The user inputs a character which has an ASCII code. 0x30 is subtracted from the character to get its true value.

The running total is multiplied by 10 and the value is added. This continues until the user inputs an enter (CR).

After the user inputs CR, the value of the multi-digit number is stored in R4.



Stage 1: Console Input

Extra Mile

As an extra precaution, I check to see if the character pressed is actually a digit between '0' and '9'. If the character is not a digit the program loops back to the start to wait for another input. If the character is CR the program terminates.

```
if((ch >= '0') && (ch <= '9'))
{

        print(ch);
        value = ch – 0x30;
        total = total * 10;
        total = total + value;

}
```

Stage 2: Statistics Evaluation

Solution

Stage 2

I used my program from stage 1 to read values entered by the user. However this time the user inputs a list of numbers separated by spaces. To end the program CR must be pressed.

I also allow the user to input negative numbers. If the character '-' is entered I use a register as a boolean (isNegative). Later when the user enters a space, an if statement checks to see if it is negative. If so, it subtracts the number from 0 to get its true value.

```
if(isNegative)
{
        value = 0 – value;
}
```

To calculate the statistics I have a loop that runs every time space is entered. In the loop the count, sum, min and max are all calculated. They are all stored in their own register.

Count	R6
Sum	R7
Min	R8
Max	R9

Stage 2: Statistics Evaluation

Calculations

Count: The count is calculated by adding one every time a space is entered.

Sum: The sum is calculated by adding the total of the inputted number to the previous sum.

Min and Max: When count is equal to zero both the min and max are set to the first inputted number. After that each new number is compared to the min and max. If the number is lower than the min it replaces the min and if it is higher than the max it replaces the max.

Mean: The mean is only calculated once. At the end of the loop it checks to see if the character is equal to CR. If the character is equal to CR the loop terminates and the mean is calculated.

The mean is calculated by dividing the sum by the count. This is done by repeated subtraction (as done in a previous exercise) and returns a quotient but no decimals. I leave the result as it is and do not round the mean up or down.

If the sum is zero this calculation will not work. To get around this I use another boolean (isNegative) and take the absolute value of the sum. After the calculation I use an if statement to see if the sum was negative. If so, the mean is taken away from zero to get its actual value.

The mean is stored in R10.

Stage 2: Statistics Evaluation

Extra Mile

Here I tried to make my program as robust as possible. I tested a lot of different combinations of inputs to figure out what errors could happen. I dealt with as many errors as I could.

The program will only allow:

- The user to input numbers, negative signs, spaces or enter
- One space to be inputted at a time
- One negative sign to be inputted at a time
- A negative sign to be inputted at the start of a number

The program will not add one to the count if the first character entered is a space and/or the last character entered is a space.

I also allow the user to input negative numbers.

Stage 3: Displaying Result

Stage 3

I built on the previous program to display the mean. The mean is stored in R0 after the calculations.

To display the result, the individual digits of the mean must be printed from left to right. To do this we must divide the mean by a power of ten that has as many digits as the mean. This will give us a quotient equal to the first digit and a remainder of the other digits. If you repeat this with the remainder you will be able to display the number.

Example

Mean = 1234
1234 / 1000 = 1 Remainder 234
Print 1
234 / 100 = 2 Remainder 34
Print 2
.
.
.
4 / 10 = 0 Remainder 4
Print 4

I did this by having a loop to calculate consecutive powers of ten until the power of ten is bigger than the mean. I then use the previous power of ten to divide into the mean to get my first digit. I then run the loop again to find the new power of ten to divide by.

Solution

Stage 3: Displaying Result

Dealing with Zeros

I store the previous two iterations of the power of ten. I use the smallest value to check if it is bigger than the mean. If it is I print a ‘0’.

If the greatest power of ten is equal to 10 I print the mean directly.

Example

101 / 100 = 1 Remainder 01

Print 1

As 1 < the power of ten two iterations ago

Print 0

Greatest power of ten = 10

Print 1 (mean)

Extra Mile

If the mean is negative I print a negative sign and then, using the absolute value of the mean, print the mean as I would with a positive number.

Testing

Input	Reason	Result	Correct?	Fixed?
0	To see if my program could handle a sum of zero	Count Sum Min Max Mean all equaled 0	Correct	N/A
1	To see what my program does with only one input	All equaled 1	Correct	N/A
Entered characters other than numbers, spaces, negative signs or enter	To see if my program allowed incorrect inputs	The characters were not displayed or stored	Correct	N/A
Space at start of sequence	To see if the count was disrupted	Count was 1 over what it should be	Incorrect	Fixed so this will no longer Happen
Space at end of sequence	To see if the count was disrupted	Count was 1 over what it should be	Incorrect	Fixed so this will no longer Happen
A negative sign on its own	To see if the count was disrupted	Count was 1 over what it should be	Incorrect	Fixed so this will no longer Happen
First character entered is a space	To see if the count was disrupted	Count was 1 over what it should be	Incorrect	Fixed so this will no longer Happen
100 200 300	To test the printing of zeros	Mean printed was 2	Incorrect	Fixed so this will no longer Happen
-100 -200 -300	To test the printing of negative means	Mean printed was 200	Incorrect	Fixed so this will no longer Happen