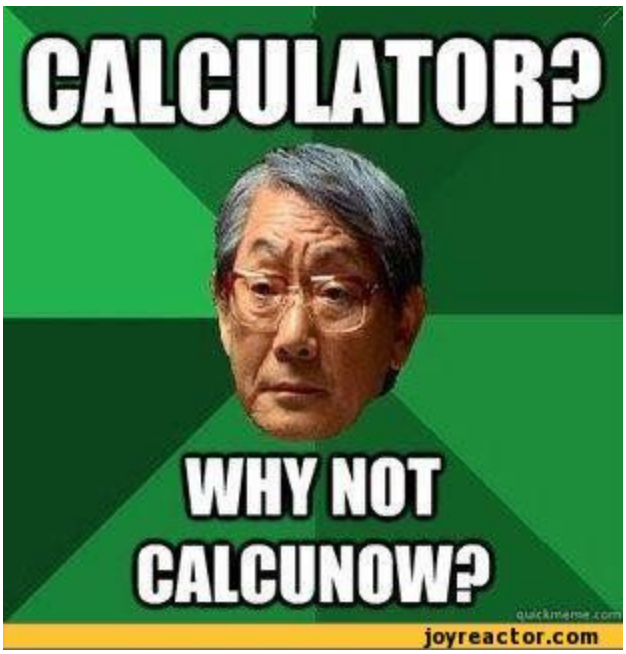
The A-Team

Clovis Community College
CSCI 41 10:00 AM



CalcuNow

May 03, 2017

Overview

This project involves creating a calculator capable of addition, subtraction, multiplication, division, exponential and modulus. The user will be able to assign values to variables and use those variables in these equations (ignoring order of operations). The twist is that after each variable is used it must increment by one. So if x=10 then x+x is 21. x was ten the first time it was used and was then incremented to eleven. After this calculation x was left as twelve.

Requirements

1. **Store up to 26 variables:** The user can assign an 8 bit value (0-255) to any lowercase letter (a-z).
2. **Retrieve the value of the variable in O1:** The project specifies that the retrieval of the 8 bit integer be done in Order One time.
3. **Throw all errors to main:** All error handling must be done in main.
4. **Use a class to store the values:** A class must be used to store the values, increment them and check if they have gone over 255.
5. **Use a data structure for variables:** Must use a data structure to assign each value to a letter.
6. **Document the code:** The code must be well commented, including pre and post conditions for every variable and documented class invariants for each function.

Major Concepts

We will use a hash table (in this case a basic array) to store the the values of each possible variable. Index 0 of the array will be ‘a’ , 1 is ‘b’, 2 is ‘c’, … 25 is ‘z’. It is possible to convert a char to these values by simply subtracting ‘a’ from the char. The limit of 26 vars given in point 1 is a blessing that makes this really cheap to implement. This will result in an O1 lookup so satisfies point 2 and is a data structure so satisfies point 5.

The class that stores the actual value will automatically increment the value every time you look up the value so that it is ready for the next time.

Point 3 is tricky and I think he will be lecturing on it soon. There is a powerpoint on canvas about this subject. I will be familiarizing myself with it soon. For now just make all errors cout “BAD INPUT” and move on.

Document all your code heavily! Before every function write down what you require it to take in and what you are promising to give out. Each time you use a variable write down what you expect it to be before it is used and what it will become after (i.e. //x must be initialized. it will be 5 greater than it was.) Sorry guys it’s a project requirement.

Pseudo Code

Create an 8bit class

- Declare an 8bit int
- Creat a constructor that takes in a value
- Write a check if over 255 function
- Write a get function
  - Store the initial value
  - Increment
  - Check if over 255
  - Return initial value

Declare an array called vars of 8bit classes that is 26 long (This will act as a hash table)

Make a function called setVar that takes in a stringstream (must be in order x = 10)

- Pass the three parts of the stringstream to a char, char and int;
- Confirm that the first char is a valid alpha the second is '=' and int is vaild
- Subtract 'a' from the char
- Make a pointer to a new 8bit with int value

Put the pointer in char slot of the vars array

Make a function called to_lower to convert a string to lowercase

For each char in a string

if it is of type alpha

Lowercase it

Make a function called do_math that takes in a known value and a stringstream & outputs an int

Enter a while loop that end when the stringstream is empty

Get the next entry of the stringstream (must always be a +,-,*,/,^,%)

Get the next entry after that (either a char or an int)

If a char

Use the char value to look in the array for a pointer to the class

Get the value from the class

Perform the given math function on the known value and the new value

Store the result as the known value

Return the known value

Main

Enter a While Loop that ends with EOF

Get user input as string

Pass string through the to_lower function

Convert string to stringstream

Get first entry of stringstream

If it is “let”

Pass stringstream to setVar

If it is “quit”

Quit

Otherwise

Pass the first entry and the stringstream to doMath

Milestones

1. Thursday May 4th be with you

   Assign a section of the code to each person. Hear back their comments/questions on their section.

2. Sunday May 7th

   Receive at least one draft of the sections from each person.

3. Monday May 8th

   Meet in person, either in class or at some other time/place to run the combined code, receive feedback and make plans for finalizing the project.

4. Wednesday May 10th

   Receive final code from all members. Combine the code. Win!

Extra Credit

The documentation does mention an extra credit possibility by adding a command called loop that will run an expression a given number of times. I believe this will be easy to add at the end given the way the code is structured. So if we meet all the above milestones there will be plenty of time for adding that.