Group 12

Calculator

User's Manual

Version 1.3

Revision History

Date	Version	Description	Author
07/Dec/2024	1.1	filled in basic information	Noah Haskins
09/Dec/2024	1.2	Edited install instructions	Maren Proplesch
10/Dec/2024	1.3	Finalized Document	Noah Haskins

Table of Contents

Test Case

1.   Purpose

The overall purpose of this project was to create a basic calculator that followed PEMDAS (see glossary for definition). taking in there equations with the basic operations (+,-,/,*,**,%,()) with more details in Getting Started. The program lets the user input in their equation and gets the solution but it also tells them when they do something wrong i.e. if they divide by zero it tells them they can not divide by zero. This all can be accessed on our GetHub to run on cycle servers run "make cycle_run".

2.   Introduction

The Calculator software takes in an equation and returns a solution following PEMDAS. The purpose is to complete a given equation promptly. The calculator's features include checking that the equation is valid. Dividing by zero will raise an error. It also allows the user to get their solution with decimals included. To install the calculator one just needs to download the repo from our GitHub and then run the make file from source. Once the program is running just enter in your equation when prompted and hit enter and it will give you the solution. make defaults to using clang. To run on cycle servers run "make cycle_run". This builds and runs the calculator. "cd src; make cycle_run"

3.   Getting started

After running the program from the make file from the source the user is asked to enter an equation. The user needs to follow some basic rules when entering their problem. If the user is using addition they will press "**+**", subtraction is "**-**", multiplication "***"***, *division is "/", exponents are "******" or "**^**", the remainder is "**%**" and if they want to include parentheses they will use "**(**" and "**)**". See example #4.

4.   Advanced features

Although this is a simple program we made sure to make it clear what the user is doing wrong for example errors tell the user what they are doing wrong in their error code instead of just saying an invalid equation. as well as it allows the user to put in a + before their number to indicate that it is a positive number i.e. +8 - 2 = 10.

5.   Troubleshooting

Some common problems are if the user divides by zero or if they enter an invaded character i.e. "A". If the user divides by zero they will be told that they have entered an invalid integer letting them know that their equation needs to be changed for the equation to be completed. If they enter in a character then they will receive a prompt that tells them that they entered an invalid syntax which lets them know that they need to fix that mistake.

6.   Examples

Some examples for each operation would be addition is 1 + 1 = 2 then subtraction 1 - 1 = 0 for multiplication 2 * 2 = 4, division 2 / 2 = 1, modular 2 % 1 = 1, exponent 4 * 2 = 16 and finally parentheses 2 * (1 + 2) = 2 * (3) = 6 for a more complex problem "4/2+(3*4**2)%2-2". the program will follow PEMDAS so looking at the parentheses first the program will see "3*4**2" it will do the exponent first to get 4 to the power of 2 which is 16 then it will do 4 * 16 = 64 the program then looks at the new equation "4/2+(64)%2-2" the next step in PEMDAS is a division of 4 / 2 = 2 then modular of 64 % 2 = 0 now the looking back at the main equation "2+0-2" and following the last 2 steps of PEMDAS the software will do 2 + 0 = 2 and then 2 - 2 = 0.

7.   Glossary of terms

The glossary is documented on our github.

8.   FAQ

Q1: Does this program work on a Mac?

A1: Yes, compile with G++ or Clang with C++ version 17.

Q2: Who can I contact for support?

A2: This program comes as is. If you discover an issue, please create an issue on the github page.

Q3: Why does it start with parenthesis?

A3: The calculator follows the laws of PEMDAS where Parenthesis is the highest priority of operations