Whatever Co.

Whatever Calculator

User’s Manual

Version <1.0>

Revision History

Date	Version	Description	Author
10/12/2024	1.0	Created User Manual to explain functions and user input to the user	All Team Members

Table of Contents

1. Purpose 4
2. Introduction 4
3. Getting started 4
4. Advanced features 4
5. Troubleshooting 4
6. Example of uses 4
7. Glossary 5
8. FAQ 5

User's Manual

1. Purpose

The User Manual for the Whatever Calculator aims to serve as a comprehensive guide for users to understand and effectively use the arithmetic evaluator. It will provide clear instructions, troubleshoot common issues, and offer examples for a seamless user experience.

2. Introduction

This is the User Manual for the *Whatever Calculator* in C++. The design for this software is to evaluate arithmetic expressions with operators, +, -, *, /, % and ^, and supports parentheses and the precedence is in PEMDAS. Please note that while it does support parentheses, it does not support implicit multiplication. Example, the program supports *(1) * 2* but it is not able to interpret *(1)(2)*. Further clarification will be found in *Sections 3.3 and 5.*

3. Getting started

3.1 Prerequisites

- Browser (e,g, Google Chrome, Safari, etc.)
- Compiler that is able to handle C++ (e.g, Visual Studio Code, Clang, etc.)

3.2 Installation:

- Download source code from the following Github Repository.
- Compile the program using a valid Compiler.
- Execute the binary file that is now compiled.

3.3 Usage:

3.3.1 Running the Whatever Calculator:

- After executing the binary file, the user will now be prompted with, ***"Enter an arithmetic expression (or 'quit' to quit)***", this is where the user will type in the arithmetic expression following the proper format.

3.3.2 Operators and Order of Operations

- The arithmetic expression should be typed in infix form <*operand*><*operator*><*operand*>,(e.g, 2+2) not prefix <*operator*><*operand*><*operand*>, (e.g, +22), or in postfix form <*operand*><*operand*><*operator*>, (e.g, 22+).
- The precedence for the operators follows (PEMDAS) for which it in the table below, gives the precedence. Please note, that although it follows PEMDAS, it does not support expressions that are defined in parentheses (e.g, (2+3)/2).

The Whatever Calculator supports the following operators:

Operator	*Function*	*Precedence*
( )	*Parenthesis*	*5*
**	*Exponentiation*	*4*
%	*Modulo*	*3*
*/, **	*Division, Multiplication*	*2*
+, -	*Addition, Subtraction*	*1*

4. Advanced features

The *Whatever Calculator* is able to handle any number of spaces as well as any float values. For example operations done on the number 1.1 will still execute and run in the code and have the correct outputs.

5. Troubleshooting

The following issues are very common issues for which users can very easily encounter and easily run into on accident. The user should use this section as a guide for troubleshooting should any of these errors occur during the use of the program.

Error (Invalid Arithmetic Expression):

- **This error happens whenever the user puts in too many operators (e.g 3 ++ 4). The solution to this error is to please be wary of the amount of operators the user inputs or please understand that this calculator uses infix expression (See section 7 for more clarity)**

Error (Invalid Character):

- **This error happens whenever the user tries to input a character that is invalid (e.g any symbols or characters that are not in the table in section 3.3.2). Also having too many operands (e.g fx * 2) will also result in this error. The solution to this is for the user to be careful of the input and inputting the correct symbols for the intended functions.**

Error (Modulus by zero):

- **This error happens whenever the user tries to modulo (e.g 5 %0). The solution to this error is to please not modulo any number(s) by 0 as no matter what values the number modulo by 0 is always undefined.**

Error (Cannot pop from empty stack):

- **This error happens whenever the user inputs nothing and simply just presses enter when the code is executed. The solution to this error is for the user to please input values into the code**

Error (Unmatched Parentheses):

- **This error happens whenever the user has parentheses that does not close up and does not match with any other parentheses, for example, 2 * (4 + 3 - 1. The solution is to make sure that the user always closes parentheses.)**

Error (Mismatched Parentheses):

- **This error is very similar to the parentheses above, *"Unmatched Parentheses"*, this error happens when the user has a parentheses that does not match up with another parentheses for example, ((3+4) - 2) + (1). As you can see, the user is missing one parentheses at the end.**

Error (Operator without Operands):

- **This error happens whenever the program sees an operand, but it does not have a matching operator (e.g, * 5 + 2). The solution to this is to please make sure you are using the calculator in infix expression (See section 7 for more clarity). Please note that although * 5 +2 is invalid, having the expression -5 + 2 is valid. (See section 3.3 for more clarity).**
-

Error (Missing Operator):

- This error happens whenever the user tries implicit multiplication (e.g, 5 (2+3)) The solution to this is to please understand that this program does not take implicit multiplication (See section 3 for more clarity)

Error (Modulus by Zero):

- This error happens whenever the user tries to take the modulus of a number by zero. The solution to this is to understand that it is impossible to take the modulus of a number by zero.

The most common problem users will encounter is inputting invalid expressions. A simple solution to this is to look over section 3.3 in the users manual or section 6 for examples of arithmetic expressions.

6. Examples

The following examples down below are sample inputs that the user is able to provide after executing the binary file. While all the provided inputs are in the form of whole numbers, please note that the inputs can also be in the form of decimals. Furthermore, the following inputs can also be negative and have valid outputs. (See section 4 for more Clarity).

Addition:
Input: 2 + 2
Output: 4
Subtraction:
Input 2 - 2
Output: 0
Multiplication:
Input: 2 * 2
Output: 4
Division:
Input: 2 / 2
Output:1
Exponentials:
Input: 2 ** 2
Output: 4

Modulus:
Input: 2 % 2
Output: 0
Parentheses:
Input: (2) / 2
Result: 1

7. Glossary of terms

Whatever Calculator: Program that is able to evaluate arithmetic expressions, handling operators and numeric values from input.

Github Repository: Online platform where you can store code and update code, allowing you to see revision history and see past versions of your code

Compiler: Program that translates source code in a high-level programming language into machine code that is able to be executed by a valid computer with a supported OS.

Binary File: The file that is generated from the source code after the compiler is run, it is a simple single file for which in Windows can be represented by an .exe file and in other OS systems it might not have an extension after it.

Operand: A value or variable that is used to perform an operation.

Operator: A set of symbols that allows the user to manipulate values or variables.

Precedence: The order for which a set of items goes in a certain order based on a defined order.

Prefix Expression: Another form to write an expression for which all of the numeric operators come before the actual operand or variable.

Infix Expression: The most common way to write an expression for which it follows PEMDAS precedence and is the typical form that is mostly used for which the operand or variable always has either one operator before or after the expression.

Postfix Expression: Another way to write the expression for which the expression follows the order for which the operator(s) always comes after the operand or variable.

8. FAQ

Check this section out if you have any questions!

- **Why am I getting a syntax error?**
  - This occurs if the entered expression is incomplete or invalid. Check for mismatched parentheses, unsupported symbols, or incorrect input.
- **How can I calculate percentages?**
  - Multiply your number by the percentage in its decimal form. For example: 50 * 0.2 for 20% of 50.
- **What happens if I forget to close a parenthesis in my expression?**
  - The program will detect unmatched parentheses and display an error message. Make sure to balance opening and closing parentheses in your expression.

- **How do I handle complex expressions with nested parentheses?**
    - Our program is designed to automatically recognize and evaluate expressions within parentheses based on the order of operations (PEMDAS). This should cause no issue.
- **Are unary negatives supported?**
    - Yes, a valid arithmetic expression may be "---1" which should output -1.
- **What operators does the *Whatever Calculator* support?**
    - Parentheses (( )), Exponentiation (**), Multiplication (*), Division (/), Addition (+), Subtraction (-).
- **What happens if I enter an invalid character?**
    - Invalid characters, such as symbols or letters, will prompt an error message. Double check that your expression only includes valid operators and numeric constants. (See section 6 for more clarity).