<Compiler Expression Parser>

User's Manual

Version <1.0>

[Note: The following template is provided for use with the Unified Process for EDUcation. Text enclosed in square brackets and displayed in blue italics (style=InfoBlue) is included to provide guidance to the author and should be deleted before publishing the document. A paragraph entered following this style will automatically be set to normal (style=Body Text).]

[To customize automatic fields (which display a gray background when selected), select File>Properties and replace the Title, Subject and Company fields with the appropriate information for this document. After closing the dialog, automatic fields may be updated throughout the document by selecting Edit>Select All (or Ctrl-A) and pressing F9, or simply click on the field and press F9. This must be done separately for Headers and Footers. Alt-F9 will toggle between displaying the field names and the field contents. See Word help for more information on working with fields.]

Compiler Expression Parser	Version: <1.0>
User's Manual	Date: <12/12/2024>
<document identifier>	

Revision History

Date	Version	Description	Author
12/12/2024	1.0	<User Manual for the Compiler Expression Parser>	<Nevan Snider>

Table of Contents

Test Case

1.   Purpose

The Compiler Expression Parser works similar to how normal calculators work, where the user is able to input a mathematical expression, and receive the answer to the equations.

2.   Introduction

The program has the ability to run a variety of mathematical operations, such as:
Addition
Subtraction
Multiplication
Division
Exponentiation

As well as the ability to use parentheses to manually prioritize order of operations.

To run the software, run the makefile by opening a console and typing "make". This will then create a binary file named "cli" which when run will launch the program.

3.   Getting started

Upon running the software, the user is greeted with an empty text box, where they are free to enter any mathematical expression.

The following operations can be achieved with their respective operators:
Addition: **+**
Subtraction: **-**
Multiplication: *****
Division: **/**
Exponentiation: ******

4.   Advanced features

The program automatically keeps a history of the users previously entered expressions. The user is able to go back in the history by 1 at any time by typing "BACK".

The user is able to end the program at any time by typing "END"

5.   Troubleshooting

If you are getting an error indicating that your input is invalid, make sure your expression doesn't contain any hanging operators (i.e. 5+ where the + doesn't have a number following it) and that it only contains strictly valid characters (all numbers, and the listed supported operators)

Compiler Expression Parser	Version: <1.0>
User's Manual	Date: <12/12/2024>
<document identifier>	

6.   Examples

Addition: 5+5
Subtraction: 5-5
Multiplication: 5*5
Division: 5/5
Exponentiation: 5**5
Multiple Operations with parentheses: (5+3)*12-(32+3)

7.   Glossary of terms

Operators: A symbol used to denote a specific mathematical action (i.e. + for addition)
Expression: A line of characters used to calculate a specific value (i.e. 3*10)

1.

8.   FAQ

How can I exit the program?
Type: "END"

How can I go back in the history?
Type: "BACK"

Why is my expression leading to an unintended answer?
Double check that your order of operations are correct by using parentheses when necessary