Kansas Instruments

<KI-69>
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.]

Revision History

Date	Version	Description	Author
<03/12/23>	<1.0>	<Updated and Completed Parts 5-8 of the User Manual>	<Ginny Ke>
<03/12/23>	<1.0>	<Updated and Completed Parts 1-4 of the User Manual>	<Dylan Sailors>

Table of Contents

Test Case

1. Purpose

The purpose of our software product is to develop a 'user-friendly' calculator in C++ capable of handling a wide range of mathematical expressions given by the user. The calculator aims to provide users with a convenient way for evaluating arithmetic expressions, ranging from simple calculations to more complex equations involving various operators and functions.

2. Introduction

This application is designed to aid you with your mathematical calculations by providing a platform for evaluating arithmetic expressions. Whether you need to perform basic calculations or compute more complex equations, our calculator is here to assist you.

Features:

- Capable of handling various mathematical operators (+, -, *, /, etc.).
- Supports parentheses for grouping expressions.
- Provides basic mathematical functions like addition, subtraction, multiplication, and division.
- Offers additional functions such as power functions, square root (if given as a power function) and parenthesis handling.
- Designed for easy installation and usage.

Installation:

Simply download the necessary calculator files from our GitHub, navigate to the folder in your terminal, use the 'make build' target to compile and run the calculator, 'make clean' target to delete program, and 'make run' target to run the calculator if it has already been compiled. All you have to do next is start inputting your mathematical expressions!

3. Getting started

Here's a quick guide on how to use the code for yourself.

- Use the 'make build' target to compile and run the calculator.
- Use the 'make clean' target to delete program.
- Use the 'make run' target to run calculator if it has already been compiled.
- Once the program is running, it will prompt the user to enter an expression to evaluate.
- Examples of expressions to enter can include:
    - '4 + 2'
    - '(4*2) - 3'
    - '-3^2'
    - '((4*2)-3'
    - 'hello'
- If the expression is valid, then the program will output "Result: {result}".
- If the expression is invalid, then the program will output "Error, Invalid Expression".
- To exit the program, enter '0'.
- Otherwise, the program will continually ask the user to enter an expression until they enter '0'.

4. Advanced features

The code features the ability to go back to previously inputted expressions, can loop, and can handle complex expressions given.

5. Troubleshooting

All tests were run within the cycle servers and Linux machines provided at the University of Kansas.

Compiler Issues:
Make sure that you are running a g++ compiler and using the make build lines of code. To troubleshoot first check that you have the most updated version of g++ compiler with the following command
(g++ --version).
Update Compile Version:
Run the command prompt (cmd) followed by (mingw-get update) to update the package list, and then run (mingw-get upgrade).
Handling Large Expression:
In the case that the program encounters an issue running a large expression check that your machine has enough memory. Additionally, to free up memory or space you can use the make clean command (rm –f my_calculator) this will remove your output file. Please note that this will permanently delete the output file and any prior expressions that you may have stored.

6. Examples

Here are examples demonstrating how to use the software to evaluate different arithmetic expressions:
Valid Expressions include those using operators: s +, -, *, /, %, and ^. Additionally, valid expressions have matched parenthesis and grouping symbols. Additional spaces within expressions are ignored and any invalid expressions will return with the correlating error message.
Compilation Step Examples:
Make Build: compiles main.cpp then executes the output in a file called my_calculator.

```
g++ -std=c++11 main.cpp -o my_calculator
./my_calculator
```

Make Clean: use with caution as it permanently deletes the file referred to. The command will delete all filse ending with the ~ symbol and current directory without asking for confirmation.

```
rm -f my_calculator
rm -f *~
```

Make Run: executes program called my_calculator.

```
./my_calculator
```

The following outputs are results of valid expression after the steps of compilation –refer to test cases document for an in-depth view.

```
Enter expression to evaluate: 3+4
Result: 7
Enter expression to evaluate: 10*2/5
Result:4
```

Invalid expression examples:
Unmatched Parenthesis:

```
Enter Expression to evaluate: 2*(4+3-1
*error* program terminates
```

Operators without Operands:

```
Enter expression to evaluate: *5+2
Result: nan
```

7. Glossary of terms

Make Build (build): compiles and runs the calculator. Command is given in the examples and the Makefile file.

Make Clean (clean): deletes the current program Command is given in the examples and the Makefile file.
Make Run (run): runs the calculator after compilation Command is given in the examples and the Makefile file.
User: individual that uses the calculator (actor in use case).
Operation: user chooses an operator to be performed.

Operand: object that the operation is being performed.
Nan: error message signifying an invalid expression

8. FAQ

Q: What happens if I try to divide by zero?
A: Division by zero is mathematically undefined and will prompt an error message then terminate the program. Upon termination of the program, you will need to compile and run the g++ command again to be able to reenter expressions.
Q: Is there a limit to the length of the expression that the program can handle?
A: Although there is no limit to the length of the expression its best to avoid entering long expressions to avoid entering and invalid expression. Additionally, it could increase the time that you would receive an output.
Q: Do I need to worry about whitespaces?
A: The program can handle whitespaces so you may add as many spaces as possible in between operands or values.
Q: Can the program handle decimal values?
A: Yes, the program can handle decimal values and will output values in decimal format as well.