Infix Expression Evaluator (Mid-term)

Author

- Name: Hùng Ngọc Phát
- Student ID: 19120615
- Class: 19CTT4, Ho Chi Minh University of Science.

Input/output

- Parameters:
  - No parameter: read and evaluate an **INFIX** expression from "input.txt".
  - If only one parameter is passed, it will be treated as an infix expression (`-i` is implicitly added).
  - `-i` : evaluate an infix expression.
  - `-s` or `-p` : evaluate a suffix/postfix expression
  - `-f` : read expression from a file. Must additionally specify `-i/-s/-p` .
  - Examples:

```
# These commands will evaluate an infix expression
$ ./eval "1+1"
$ ./eval -i "1+1"

# These commands will evaluate a suffix/postfix expression
$ ./eval -s "1+1"
$ ./eval -p "1+1"

# This command will evaluate an infix expression from "expression.inp"
$ ./eval -i -f "expression.inp"

# This command will evaluate an infix expression from "input.txt" (default input file)
$ ./eval
```

- `input.txt` (default input file): contains an expression of string type.
- `output.txt` : contains the result. Additionally includes suffix expression if converted from infix expression. Custom output filename is not supported.
- `errorlog.txt` : contains input expression, converted expression, parameter list, error desription if an exception occurs.

Source code structure

- `main.cpp` : contains the main function and the parameter handling function.
- `PostfixEval` : contains the function to evaluate a postfix expression (of string type). Returns a float if evaluated successfully. Else, throws a runtime_error.
- `InfixToPostfix` : defines the function to convert an infix expression (of string type) to a postfix expression (of string type). Note that more syntax error checking are done in this file than in `PostfixEval` because this is what an actual human uses in practice, not the computer-ish postfix expression.
- `StackChar` : contains definitions and methods for a stack of `char` type, which lies underneath InfixToPostfix.
- `StackFloat` : contains definitions and methods for a stack of `float` type, which lies underneath PostfixEval.
- `Utils` : contains inline functions that are shared between `InfixToPosfix` and `PostfixEval` .
- `FileRW` : contains functions to read and write to file
- `makefile` : make rule for GNU make. Built binary will be placed in BUILD folder.\n **Warning**: a suitable g++ executable path `$(G++)` must be re-specified in `makefile` .

Building

- This repository can be built by running GNU make in the same folder with `makefile`.

```
$ make
```

- Be sure to set the `g++` path (`$(G++)`) correctly in `makefile`. Default is `g++`.
  - On Windows, install `mingw-64` and add it to `PATH`, then set `$(G++) := g++`.
  - On Linux, such as Debian-based distros, install `build-essential` from `apt`, then set `$(G++) := g++`.
  - On macOS, install `gcc` from Homebrew, then set `$(G++)` accordingly to your installed gcc version. For example, `g++-9`. You can also use `clang++` without having to install `g++`. Simply set `$(G++) := clang++`