MyCalc Application (Mid-term)

Author

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

Disclaimer

- This program is written almost without using C++ standard library goodies (except `std::string` and `std::exception`), so it is very prone to bugs.
- Major syntax errors will be checked in the conversion process from infix to postfix expression, because this is what people use in practice.
- Since the conversion algorithm is almost accurate, as well as for algorithm efficiency, postfix evaluation stage will not check for syntax errors as careful as it was in the conversion stage. Of course, it do check for errors in case the user required to evaluate a postfix expression in the first place, but still, not as careful as in the conversion process.

Input

- Parameters (3): MyCalc.exe `expression_type` `input_filename` `output_filename`
  - `expression_type`: can either be `-i` for infix or `-p` for postfix.
  - `input_filename`: the name tells it all.
  - `output_filename`: the name tells it all.
- Alternatives: `--help` for displaying help message.

Output

- Write the evaluated result to `stdout` and the specified output file.
- In case there is an exception: write the error message in the specified output file and `stdout`. If no output file was specified (e.g. missing parameters exception), the error message will be written to `errorlog.txt`.

Expression syntax rule

- **Supported operators:** `+` `-` `*` `/` `^`.
- **Supported number type:** `int` and `float`. Negative number is also supported.
- **Decimal seperator:** `. (dot)`. For example: `2.5`, not `2,5`.
- **Number constraint** matches C++ 14's `float` constraint.
- **In infix expressions:** spaces are allowed, and they will be stripped before conversion starts.

- **In postfix expressions:** tokens are seperated by spaces, not limited to quantity, because they will be ignored. However, negative sign must be placed **right before** its first digit, like this: `-5.2`, not like this: `- 5.2`.
  That tokens are seperated by only **ONE** space is **HIGHLY** recommended.
- Other common mathematics rules.

Examples

```
# The following one evaluates a postfix expression
# from input.txt and write the result to output.txt
$ MyCalc.exe -p input.txt output.txt

# Similar to the above with an infix expression
$ MyCalc.exe -i foo.txt bar.txt
```

Building

- Some prebuilt x86_64 binaries for 3 main OSes are already available in the Release folder.

OS Name	Executable filename	Tested on
Microsoft Windows	MyCalc.exe	Windows 10 2004
GNU/Linux	MyCalc-linux	Ubuntu 20.04 (WSL)
Apple macOS	MyCalc-darwin	macOS Mojave 10.14.6

- This repository can be built by running GNU make in the `Source` folder.

```
$ cd somewhere_in_your_pc
$ cd MyCalc/Source
$ make
```

- Be sure to set the `g++` path (`$(G++)`) correctly in `makefile`. The default path is `g++`.
  - On Windows, install `mingw-64` and add it to `PATH`.
  - On Linux, such as Debian-based distros, install `build-essential` with `apt`.
  - On Arch Linux, run `sudo pacman -Sy base-devel`.
  - On macOS, install `gcc` from Homebrew, then set `$(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++`.
- If you want to build the repository yourself, remember to enable `C++14` support by specifying `-std=c++14`.

Source code structure

- `main.cpp`: contains the `main` function. Handles commandline parameters; calls the conversion and/or the evaluation subroutines as well as handles exceptions.

- `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 `static inline` functions that are shared between `InfixToPosfix` and `PostfixEval`, including:
  - `isoperator(char)`: the name tells it all.
  - `priority(char)`: gets the priority (precedence) of an operator. For example, predecence of `^` > `*` = `/` > `+` = `-` > `non_operators`.
  - `error_string_gen(string, int, char, string)`: generates a string describing an error which happened during runtime. Will be thrown with `std::runtime_error`.
- `FileRW`: contains functions to read and write to files.
- `makefile`: make rule for GNU make. Built binary will be placed in `Release` folder.
  Warning: a proper g++ executable path `$(G++)` must be re-specified in `makefile`.