Отчет

# Описание работы программы

При запуске программы необходимо ввести один из флагов: «**--forward**» или «**--reverse**». После запуска необходимо ввести выражение в инфиксной или префиксной форме соответственно и нажать сочетание клавиш **Ctrl+D**, после чего программа закончит выполнение.

Если необходимо чтение из файла, то при запуске программы следует ввести сначала флаг «**--file**», затем имя файла без кавычек с расширением и флаг «**--forward**» или «**--reverse**». При чтении данных в префиксной форме из файла необходимо разделять числа нижним подчеркиванием (**“\_”**). Результат программы будет выведен в консоль.

# Описаниеалгоритма

## Функция reverse()

Изначально подразумевается, что будут введены как минимум первые два числа и оператор действия. Ход функции выглядит следующим образом:

1. Объявляем переменные **a** и **b** типа **integer** и **op** типа **char**.
2. Считываем целое число **a**.
3. Пока можно считать число **b**.
4. Считываем целое число **b** и оператор действия **op**.
5. Выполняем математическое действие и результат записываем в переменную **a**.
6. Выводим результат, записанный в переменную **a**.

## Функция forward()

Изначально подразумевается, что будут введены как минимум два числа и оператор действия. Ход функции выглядит следующим образом:

1. Создаем два массива: **symbols** типа **char** и **numbers** тип **integer**. Они будут использоваться в качестве стеков. Также создаем указатели на индекс верхнего элемента каждого стека, **top\_s** и **top\_n** соответственно, переменную **token** типа char, в которую будем записывать каждый считанный символ, и переменные **a**, **b** типа **integer** и **op** типа **char**, для выполнения математических операций.
2. Пока можем считать **token**.
3. Если **token** – целое число, то преобразовываем его в тип **integer** и помещаем на верхушку стека **numbers**.
4. Если **token** – оператор действия.
5. Пока стек не пуст и символ математического действия на верхушке стека **symbols** имеет больший или равный приоритет относительно считанного символа, вытаскиваем **op** из стека **symbols** и числа **a** и **b** из стека **numbers**, выполняем математическую операцию, результат кладем в стек **numbers**.
6. Кладем считанный оператор в стек **symbols**.
7. Если **token** – открывающаяся скобка, то кладем ее в стек **symbols**.
8. Если **token** – закрывающаяся скобка.
9. Пока символ на стеке **symbols** не равен открывающейся скобке, вытаскиваем **op** из стека **symbols** и числа **a** и **b** из стека **numbers**, выполняем математическую операцию, результат кладем в стек **numbers**.
10. Вытаскиваем открывающуюся скобку из стека **symbols**.
11. Пока стек **symbols** не пуст, вытаскиваем **op** из стека **symbols** и числа **a** и **b** из стека **numbers**, выполняем математическую операцию, результат кладем в стек **numbers**.
12. Выводим результат, лежащий в стеке **numbers** (он является единственным числом в данном стеке)

## Функция main()

Ход функции выглядит следующим образом:

1. Если никакие флаги не были введены пользователем, то выводится ошибка «**Nothing was read**» и программа прекращает выполнение.
2. Идет поочередный перебор введенных флагов.
3. Если введен только один флаг, то выполняется **reverse()** или **forward()** соответственно.
4. Если первым флагом введен «**--file**», то второй аргумент идет в качестве названия файла, а третий определяет, выполняется **forward()** или **reverse()**.
5. В ином случае выводится ошибка «**Wrong flags**» и программа прекращает выполнение.
6. Программа прекращает выполнение.

# Вывод

В результате выполнения лабораторной работы я научился использовать текстовые файлы в качестве потока ввода и вывода, заголовочные файлы. Попрактиковался в создании make файлов. Научился обрабатывать параметры запуска программы.