# Отчёт: Лабораторная работа №2.

Задача:

Написать простое консольное выражение, которое при вводе флага –forward считает математическое приложение введённое пользователем через std::cin (или из файла при наличии флага --file) напрямую. Иначе, если введён флаг --reverse, то, соответственно, необходимо подсчитать выражение, записанное с помощью ОПН – обратной польской нотации.

# Описание программы и алгоритмов:

Вначале представлено описание func.cpp. Далее main.cpp.

**Func.cpp**

Ниже представлено описание каждой функции отдельно.

**PrintVector**.

Функция просто выводит вектор типа std::string. Использовалась для отладки.

**FileEnter**.

Функция считывает математическое приложение из файла fileForTest.txt. Примечание: поддерживается только 1 математическое приложение, программа не предусматривает наличие двух разных математических выражений никак между собой несвязанных. Возвращает содержимое файла.

**ConsoleEnter**.

Функция осуществляет ввод из консоли. Возвращает введенное выражение.

**ConvertToStrArray**.

Функция разделяет входную строку на массив типа std::string. Используется именно std::string для чисел любой размерности ( Например, “1”,”1243423” и т.д.). Создается поток std::stringstream входной строки с разделением по пробелам. Элементы записываются в выходной массив. Возвращает массив типа std::string.

**IsNumber**.

Функция проверяет является ли элемент числом или нет.

**IsOperator**.

Функция проверяет является ли элемент одним из поддерживаемых операторов ({"+","-","\*","/","(",")"}).

**Calc**.

Вдальнейшем будет использоваться для подсчета ОПР. Получает на вход стэк , два элемента типа std::string , текущий оператор. Далее выполняется соотв. операция. Так как элементы типа std::string преобразовываем их в double с помощью функциии stod (расш. String to double) и считаем, затем преобразываем полученное число в первоначальный тип с помощью функции std::to\_string и кладем полученный элемент в стек.

**checkPriority**.

На вход поступает оператор. Функция проверяет приоритет того или иного оператора. Используется в дальнейшем для преобразования из обычной записи в ОПР.

**convertToPolishNotation**.

Функция преобразует обычную запись в ОПР. На вход дается массив полученный с помощью **ConvertToStrArray**. А также массив, куда будет записано преобразованное выражение. Алгоритм : 1. Если встречается элемент – число добавляем его в выходной массив. 2. Если элемент – оператор, то 2.1 Если оператор “(“ то кладем его в стек. Переходим к следующей итерации. 2.2 Если стек пустой то просто добавляем оператор в него, идем дальше. 2.3 Проверяем приоритет рассматриваемого оператора ( \* и / высший приоритет, + и – средний приоритет, ( и ) низший приоритет. 2.4 Проверяем приоритет последнего элемента стека. 2.5 Если оператор “)” то достаем из стека все операторы пока не встретим открывающую скобку и поочередно кладем их в выходной массив. 2.6 Если приоритет рассматриваемого оператора выше, чем у последнего элемента стека, то добавляем оператор в стек. 2.6.1 В противном случае достаем элементы и добавляем их в выходной массив поочереди, пока приоритет рассматриваемого оператора не станет выше, чем у последнего элемента или стек станет пустым. 2.7 Выводим все оставшиеся операторы в выходной массив. Получаем ОПН.

**calcPolishNotation**.

На вход дается ОПН. Далее используя стек последовательно подсчитывается выражение. Последовательно добавляем числа в stack. Так как +,-,\*,/ - операции бинарные, то есть для каждого оператора рассматриваем только последние два элемента, сразу же считаем их с помощью **calc**. Последний оставшийся элемент – ответ. Выводим его.

**Main.cpp**

Проверяем наличие флагов –file , --reverse. Если флаг –file отсутствует, то считываем выражение из консоли (функция **consoleEnter**). В противном случае вызываем функцию **FileEnter.** Затем выполняем **convertToStrArray**. Если присутствует флаг –reverse выполняем **calcPolishNotation.** В противном случае преобразуем выражение из обычной формы записи в ОПН с помощью **convertToPolishNotation**. Затем считаем выражение с помощью **calcPolishNotation**.

# Результаты выполнения программы на тестах.

Программа была протестирована на выражениях представленных, в тексте лабораторной работы, прикрепленном в google classroom. Так же программа тестировалась с помощью сложных выражений. Одно из таких выражений находится в файле fileForTest.txt. Были получены правильные ответы, которые проверялись с помощью калькулятора.

# Вывод

Программа работает и выдает правильные, соответствующие заданию ответы.