Задание 1.

Написать в ООП-стиле код программы, позволяющей работать с арифметическими выражениями разного вида, оперирующими вещественными числами: вычислять результат выражения, выводить запись выражения на консоль и в файл лога. Например, для вычисления выражений вида (10+4+2+3+7+1) и (1+2.5) будет использоваться класс Summator, выражений вида (2\*3\*7\*1) – класс Multiplier, и т.д.

В коде необходимо отразить следующее:

 Создать интерфейс ILoggable с 2 методами (функционал логирования):

Запись лога всего выражения на консоль:

void logToScreen()

Добавление записи лога всего выражения в файл лога:

void logToFile(const std::string& filename).

 Создать абстрактный класс ExpressionEvaluator, реализующий интерфейс ILoggable и предоставляющий чисто виртуальный метод double calculate() для вычисления результата произвольного выражения. Количество операндов должно храниться в отдельном члене класса. Сами операнды х1,х2,х3 и т.д. должны храниться в члене данного класса – массиве, в куче (динамической памяти).

 Класс ExpressionEvaluator должен предоставлять два конструктора и виртуальный деструктор. В конструкторе без параметров выделять память под 20 операндов и инициализировать их нулями, в конструкторе с параметром n – выделять память под n элементов и инициализировать нулями. Также необходимо реализовать 2 метода, позволяющие присвоить операндам конкретные значения:

Присвоить значение value одному операнду на позиции pos:

void setOperand(size\_t pos, double value)

Заполнить сразу группу из n операндов массивом значений ops:

void setOperands(double ops[], size\_t n)

 В деструкторе должна освобождаться память, выделенная в конструкторе.

 Создать два подкласса класса ExpressionEvaluator, работающих со стандартными выражениями, в соответствии с вариантом, из четырех возможных:

Summator – сумма всех операндов (х1 + х2 + х3 + х4 + ...)

Subtractor – разность всех операндов (х1 – х2 – х3 – х4 – ...)

Multiplier – произведение всех операндов (х1 \* х2 \* х3 \* х4 \* ...)

Divisor – частное всех операндов (х1/х2/х3/х4/...), но если хоть один

операнд равен 0, то результату выражения присвоить также 0.

 Создать подкласс CustomExpressionEvaluator, работающий со специфическими выражениями, вид которых приведен в варианте.

 Подклассы ExpressionEvaluator, для которых порядок следования операндов важен, должны также реализовывать интерфейс IShuffle. Данный интерфейс объявляет 2 перегруженных метода (функционал перемешивания операндов):

Произвольно перемешать операнды:

void shuffle()

Перемешать операнды, находящиеся на позициях i и j:

void shuffle(size\_t i, size\_t j)

В функции main() необходимо продемонстрировать работу созданных классов:

 Создать массив из трех указателей на класс обработки арифметических выражений.

 В соответствии с вариантом, создать в куче три объекта конкретных подклассов обработки арифметических выражений и установить на них указатели; присвоить их операндам значения двумя способами (поэлементным и групповым).

 Продемонстрировать полиморфизм: организовать проход в цикле по указателям и вывести лог выражения на консоль и в файл (в консоли отобразить еще и сам результат выражения).

 Организовать цикл по указателям, в теле которого средствами С++ проверить, реализует ли текущий объект интерфейс IShuffle. Если да, то вызвать один из методов shuffle() этого объекта и отобразить на экране запись выражения после перемешивания операндов, а также вычислить и отобразить результат нового выражения.

Ниже приведена желательная структура проекта (например, для случая, когда нужно использовать классы Summator и Multiplier):



Примечание. Отрицательные операнды при выводе записи выражения на экран и в файл должны автоматически браться в скобки (подробнее см. приложение В).

Вариант 1

Вид выражения CustomExpression: result = x1 + x2/2 + x3/3 + x4/4 + ...

Порядок создания и инициализации объектов подклассов:

CustomExpressionEvaluator: 6 операндов, присвоить группой 5, 16, -3, 10, 12, 2.

Subtractor: 4 операнда, присвоить группой 5.6,-2.1, 3.2, 1.5.

Multiplier: 3 операнда, присвоить поэлементно 1.5, -8, 2.5.

Метод shuffle() – поменять местами первый и последний операнды и изменить их знак. Метод shuffle(size\_t i, size\_t j) – поменять местами i-ый и j-ый операнды и изменить их знак.

Формат вывода:

Operands: 4

5.6-(-2.1)-3.2-1.5

Result = 3