Индивидуальные задания

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 `x1,x2,x3` 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 (`x1 + x2 + x3 + x4 + ...`)
  `Subtractor` – разность всех операндов (`x1 - x2 - x3 - x4 - ...`)
  `Multiplier` – произведение всех операндов (`x1 * x2 * x3 * x4 * ...`)
  `Divisor` – частное всех операндов (`x1/x2/x3/x4/...`), 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 C++ 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.

Приложение А

Задание 1

Вариант 9
Вид выражения `CustomExpression`: `result = x1 / x2 + x3 + x4 + ...`
Порядок создания и инициализации объектов подклассов:
`Summator`: 7 операндов, присвоить группой `15, -3.5, 10.5, –2.1, 3.3, 4, 6.3.`
`CustomExpressionEvaluator`: 5 операндов, присвоить группой `15, 10, -3, 12, -6.5.`
`Subtractor`: 3 операнда, присвоить поэлементно `1.5, 4, -2.5`.
Метод `shuffle()` – переставить операнды так, чтобы сначала шли все отрицательные, а затем положительные. Метод `shuffle(size_t i, size_t j)` – если i-ый операнд отрицателен, а j-ый – нет, то поменять их местами, иначе – оставить без изменений.
Формат вывода:

```
[ -7- operands]
15 plus (-3.5) plus 10.5 plus (-2.1) plus 3.3 plus 4 plus 6.3
[ -RESULT- 33.5 ]
```

2. Дополнить код задания 3 лабораторной работы № 2, написав еще два класса по предметной области, в соответствии с вариантом. Продумать и корректно реализовать схему наследования классов. В главной функции продемонстрировать применение интерфейса, полиморфизм и RTTI на примере 3-4 объектов классов, по аналогии с заданием 1.

Приложение Б

Задание 2

Вариант 9
Класс САМОЛЕТ (ПАССАЖИРСКИЙ) + классы ГРУЗОВОЙ САМОЛЕТ, САМОЛЕТ.
Реализовать схему наследования классов и корректно распределить по классам данные: модель, авиалинии, год выпуска, вместимость, количество пассажиров, размеры и вес, грузовместимость. Интерфейс загрузки транспортного средства ILoadable с методом void load(int kilograms) – поместить груз на самолет. В грузовой самолет можно помещать груз вплоть до максимальной грузовместимости, иначе – выдавать сообщение об ошибке; в пассажирский – обратно пропорционально фактическому количеству пассажиров на борту (коэффициент зависимости можете задать произвольный); в случае превышения лимита – также выдавать сообщение об ошибке. В main() создать 2 грузовых и 2 пассажирских самолета, продемонстрировать полиморфизм load().