ЛАБОРАТОРНАЯ РАБОТА №3

Курс «Объектно-ориентированное программирование»

Тема: Наследование и полиморфизм. Абстрактные классы и интерфейсы. RTTI.

Цель: Научиться реализовывать на C++ наследование классов, программировать абстрактные классы и интерфейсы, виртуальные методы, а также динамически определять тип объекта во время выполнения программы.

Темы для предварительной проработки [УСТНО]:

- Наследование.
- Виртуальные функции. Полиморфизм.
- Чисто виртуальные функции, абстрактные классы и интерфейсы.
- Динамическое определение типа в C++ (RTTI).

Индивидуальные задания [КОД] :

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(int pos, double value)
     ```
     Заполнить сразу группу из n операндов массивом значений ops:
     ```
     void setOperands(double ops[], int 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.

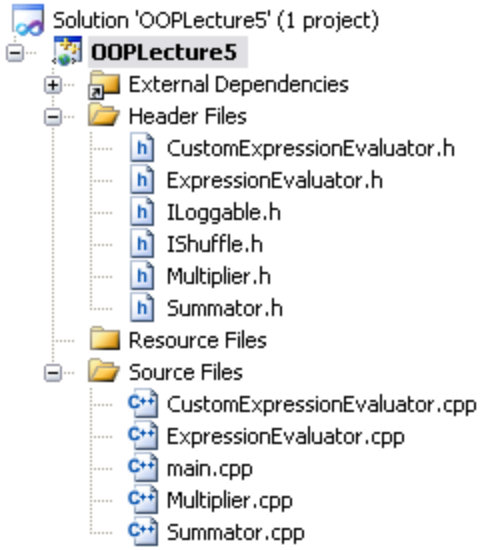
- Создать подкласс `CustomExpressionEvaluator`, работающий со специфическими выражениями, вид которых приведен в варианте.
- Подклассы `ExpressionEvaluator`, для которых порядок следования операндов важен, должны также реализовывать интерфейс `IShuffle`. Данный интерфейс объявляет 2 перегруженных метода (функционал перемешивания операндов):

  Произвольно перемешать операнды:
  ```
  void shuffle()
  ```
  Перемешать операнды, находящиеся на позициях i и j:
  ```
  void shuffle(int i, int 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.

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).

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.

   Примечание. Данное задание творческое. Вы можете свободно вводить различные свойства и метода классов, не указанные явно в задании.

Контрольные вопросы [ОТЧЕТ]**:**

1. Перечислите механизмы повторного использования кода в ООП.
2. Что такое наследование? Приведите три примера схем наследования классов.
3. Типы и виды наследования в C++.
4. В чем заключается разница между ранним и поздним связыванием?
5. Что такое полиморфизм? Виртуальные методы.
6. Что такое абстрактный класс? Чисто виртуальные методы.
7. Что из себя представляет класс-интерфейс? Для чего он используется?
8. Порядок создания и удаления подклассов. Зачем нужен виртуальный деструктор?
9. Что означает RTTI? Какие есть возможности RTTI в C++?

Рекомендуемые источники:

[1] Страуструп Б. Язык программирования C++ / Б. Страуструп. – СПб.; М.: Невский диалект; Бином, 2015. – 1136 с.

[2] Лафоре Р. Объектно-ориентированное программирование в C++ / Р. Лафоре. – СПб.: Питер, 2013. – 928 с.

[3] Прата С. Язык программирования C++ / С. Прата; пер. с англ. Ю. Н. Артеменко. – Лекции и упражнения, 6-е изд.: Пер. с англ. – М.: Вильямс, 2012. – 1248 с.

[4] Дейтел Х. М. Как программировать на C++ / Х. М. Дейтел, П. Дж. Дейтел. – М.: Бином, 2000. – 1024 с.

[5] Мейерс С. Эффективное использование STL / С. Мейерс. – СПб: Питер, 2002. – 224 с.

[6] Мейерс С. Эффективный и современный C++: 42 рекомендации по использованию C++11 и C++14 / С. Мейерс. – М.: Вильямс, 2016. – 304 с.

[7] Александреску А. Современное проектирование на C++: обобщенное программирование и прикладные шаблоны проектирования / А. Александреску. – М.: Вильямс, 2008. – 336 с.

Приложение А. Варианты индивидуальных заданий.

Задание 1.

Вариант 0 (введен для примера, фрагменты кода можно найти в приложении В)

Вид выражения `CustomExpression`: `result = x1 * x2 + x3 + x4 + ...`

Порядок создания и инициализации объектов подклассов:
`CustomExpressionEvaluator`: 3 операнда, присвоить поэлементно `5, 4, -2`.
`Summator`: 10 операндов, присвоить группой `5,-2.5, 3.5, 8, 1.5, -9.5, 3, -4, 6.5,-2.5`.
`Multiplier`: 3 операнда, присвоить поэлементно `0.25, 8, 3.5`.

Метод `shuffle()` – поменять местами первый и последний операнды.
Метод `shuffle(int i, int j)` – поменять местами i-ый и j-ый операнды.

Формат вывода:
```
[3]
5 * 4 + (-2)
18
```

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(int i, int 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
```

Вариант 2

Вид выражения `CustomExpression`: `result = x1 - x2 + x3 - x4 + ...`

Порядок создания и инициализации объектов подклассов:
`Summator`: 7 операндов, присвоить группой `5, 12.5, 9, -1.5, -9.5, 0, 11`.
`Divisor`: 4 операнда, присвоить поэлементно `100, -4, 2.5, -4`.
`CustomExpressionEvaluator`: 5 операндов, присвоить группой `5, 4, -2, 9, 3`.

Метод `shuffle()` – переставить операнды так, чтобы сначала шли все отрицательные, а затем положительные. Метод `shuffle(int i, int j)` – если i-ый операнд отрицателен, а j-ый – нет, то поменять их местами, иначе – оставить без изменений.

Формат вывода:
```
Expression [4] : 100 / (-4) / 2.5 / (-4)
= 2.5
```

Вариант 3

Вид выражения `CustomExpression`: `result = (x1 + x2) * x3 * x4 * ...`

Порядок создания и инициализации объектов подклассов:
`Subtractor`: 2 операнда, присвоить поэлементно `23.65, -12.15`.
`CustomExpressionEvaluator`: 5 операндов, присвоить группой `2.5, -5, -3, 2, 10`.
`Multiplier`: 4 операнда, присвоить поэлементно `2.5, -3, 4, -1`.

Метод `shuffle()` – поменять местами первый и последний операнды, имеющие дробную часть. Метод `shuffle(int i, int j)` – если хотя бы один из i-ого и j-ого операндов имеет дробную часть, поменять их местами, иначе – не менять.

Формат вывода:
```
Total : 4
( 2.5 + (-5) ) * (-3) * 2 * 10
Equals to 150
```

Вариант 4

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

Порядок создания и инициализации объектов подклассов:
`CustomExpressionEvaluator`: 4 операнда, присвоить поэлементно `50, 40, -10, -2`.
`Multiplier`: 4 операнда, присвоить группой `-0.5, -8, 1.5, 16`.
`Summator`: 6 операндов, присвоить группой `2.6,-8.1, 13.2, 1.5, 3.4, -4`.

Метод `shuffle()` – отсортировать все операнды в порядке убывания.
Метод `shuffle(int i, int j)` – отсортировать все операнды между i-ым и j-ым.

Формат вывода:
```
4 operands :
50 + 2*40 + 3*(-10) + (-2)
-> 98
```

Вариант 5

Вид выражения `CustomExpression`: `result = x1 + x2 * x3 + x4 * x5 + ...`

Порядок создания и инициализации объектов подклассов:
`Divisor`: 4 операнда, присвоить поэлементно `150, -3, 10, -2.5`.
`CustomExpressionEvaluator`: 5 операндов, присвоить группой `5, 16, -3, 10, 12`.
`Multiplier`: 5 операндов, присвоить группой `1.5, 4, -2.5 -8, -15`.

Метод `shuffle()` – отсортировать все операнды в порядке возрастания.
Метод `shuffle(int i, int j)` – если хотя бы один из i-ого и j-ого операндов имеет дробную часть, поменять их местами, иначе – не менять.

Формат вывода:
```
5 + 16*(-3) + 10*12 < Total 5 >
< Result 77 >
```

Вариант 6

Вид выражения `CustomExpression`: `result = (x1 + x2)/2 + x3 + x4 + ...`

Порядок создания и инициализации объектов подклассов:
`Subtractor`: 4 операнда, присвоить группой `10.5, 2.5, -3, 1.5`.
`CustomExpressionEvaluator`: 6 операндов, присвоить группой `5, 15, -8, 1, 2, 3`.
`Summator`: 3 операнда, присвоить поэлементно `1.5, -4, 2.5`.

Метод `shuffle()` – отсортировать все положительные операнды в порядке убывания.
Метод `shuffle(int i, int j)` – поменять местами i-ый и j-ый операнды.

Формат вывода:
```
Operand1, Operand2, Operand3, Operand4
10.5 minus 2.5 minus (-3) minus 1.5
Result = 3.5
```

Вариант 7

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

Порядок создания и инициализации объектов подклассов:
`Summator`: 2 операнда, присвоить поэлементно `39.1, -12.7`.
`Multiplier`: 4 операнда, присвоить группой `-4.5, 2, 3, -10`.
`CustomExpressionEvaluator`: 6 операндов, присвоить группой `5, 16, -3, 10, 12`.

Метод `shuffle()` – отсортировать все отрицательные операнды в порядке убывания.
Метод `shuffle(int i, int j)` – отсортировать все операнды между i-ым и j-ым.

Формат вывода:
```
Op1, Op2, Op3, Op4 : (-4.5) x 2 x 3 x (-10)
-> 270
```

Вариант 8

Вид выражения `CustomExpression`: `result = x1 * x2 / x3 * x4 / x5 * ...`

Порядок создания и инициализации объектов подклассов:
`CustomExpressionEvaluator`: 5 операндов, присвоить группой `5, 10, -2.5, -40, -2`.
`Subtractor`: 9 операндов, присвоить группой `120,-12, 83.2, -1.5, 5, 7, 2, 18.5, 76`.
`Multiplier`: 2 операнда, присвоить поэлементно `-1.5, 80`.

Метод `shuffle()` – поменять местами первый и последний операнды, имеющие дробную часть. Метод `shuffle(int i, int j)` – если хотя бы один из i-ого и j-ого операндов имеет дробную часть, поменять их местами, иначе – не менять.

Формат вывода:
```
<4>
5 times 10 divided by (-2.5) times (-40) divided by (-2)
<Result: -400>
```

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(int i, int 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 ]
```

Вариант 10

Вид выражения `CustomExpression`: `result = x1 - x2 / x3 - x4 / x5 - ...`

Порядок создания и инициализации объектов подклассов:
`CustomExpressionEvaluator`: 5 операндов, присвоить группой `1, -5, 2.5, 10, 8.`
`Summator`: 6 операндов, присвоить группой `15,-7.5, 3.2, 8.7, -1.5, -9.5.`
`Multiplier`: 3 операнда, присвоить поэлементно `-8, 7, -0.5.`

Метод `shuffle()` – отсортировать все операнды в порядке возрастания по модулю.
Метод `shuffle(int i, int j)` – отсортировать все операнды между i-ым и j-ым.

Формат вывода:
```
Expression: [5]
1 - (-5)/2.5 - 10/8
Result: [1.75]
```

Задание 2.

Вариант 1

Класс СТУДЕНТ + классы ПРЕПОДАВАТЕЛЬ, ЛИЧНОСТЬ.
Реализовать схему наследования классов и корректно распределить по классам данные: ФИО, пол, год рождения, год поступления, номер зачетки, средний балл, стаж, год начала работы в университете, должность, ученая степень, ученое звание.
Интерфейс начислителя суммы оплаты труда `ISalaryCalculation` с методом `double calculate()` – рассчитать сумму за месяц. Реализация метода в классе преподавателей: сумма равна 5000 + 700 за научную степень кандидата наук (или + 1200 за степень доктора наук) + 2200 за звание доцента (или + 3500 за звание профессора) + 700 за каждые 5 лет стажа. Реализация метода в классе студентов – если средний балл выше 4.5, то начислить 1000, иначе – 700. В `main()` создать 2 преподавателей и 2 студентов, продемонстрировать полиморфизм `calculate()`.

Вариант 2

Класс МАГАЗИН + классы АПТЕКА, БУТИК.
Реализовать схему наследования классов и корректно распределить по классам данные: название, адрес, год основания, номер, суммарная прибыль, доход, время работы, количество клиентов со скидкой, тип (круглосуточно или нет).
Интерфейс налогоплательщика `ITaxPayment` с методом `void payTax()`. Для бутика налог рассчитывается с учетом земельного налога, для аптеки – без (коэффициенты задайте произвольно). Сумма налогов должна быть вычтена из суммарной прибыли. В `main()` создать 2 аптеки и 1 бутик, продемонстрировать полиморфизм `payTax()`.

Вариант 3

Класс МУЗЫКАНТ + классы ЛИЧНОСТЬ, ЗРИТЕЛЬ.
Реализовать схему наследования классов и корректно распределить по классам данные: имя, фамилия, пол, год рождения, инструмент, рейтинг, место в зале, стаж (количество концертов).
Интерфейс посетителя концерта `IVisitor` с методом `void visit()`. Для музыканта посещение концерта означает наращивание количества концертов на 1, а также изменение рейтинга на некоторую величину (по результатам концерта). Для зрителя посещение означает назначение ему первого свободного места (список свободных мест хранится в файле; при резервировании места оно удаляется из списка в файле). В `main()` создать 1 музыканта и 3 зрителей, продемонстрировать полиморфизм `visit()`.

Вариант 4

Класс ТЕЛЕФОН + классы МОБИЛЬНОЕ УСТРОЙСТВО, ПЛАНШЕТ.
Реализовать схему наследования классов и корректно распределить по классам данные: фирма-производитель, модель, номер телефона, последний набранный номер, остаток на счету, вес, цвет, цена, уровень заряда.
Интерфейс возможности звонка `ICallable` с методом `void call(const std::string& recepient)` – позвонить по номеру `recipient`. Реализация метода в классе телефона:

проверка корректности номера телефона (содержит только цифры), снятие суммы со счета и расходование 3% заряда. Реализация метода в классе планшета: израсходовать 10% заряда (звонок идет по скайпу). В `main()` создать 2 телефона и 2 планшета, продемонстрировать полиморфизм `call()`.

Вариант 5

Класс КОНДИЦИОНЕР + классы БЫТОВОЕ УСТРОЙСТВО, ОБОГРЕВАТЕЛЬ.
Реализовать схему наследования классов и корректно распределить по классам данные: фирма, модель, вес, температура, режим, год выпуска, мощность.
Интерфейс возможности управления / регулировки устройства `IControllable` с методом `void control(int temperature)` – отрегулировать устройство в зависимости от установленной в параметре температуры. Реализация метода в классе кондиционера: если температура задана меньше 10 градусов, то выдать сообщение и выключиться, иначе присвоить текущему режиму разный номер в зависимости от температуры (т.е. выставить режим). Реализация метода в классе обогревателя: если задана температура выше 45 градусов, то выдать сообщение и выключиться, иначе присвоить текущему режиму разный номер в зависимости от температуры. В `main()` создать 2 кондиционера и 1 обогреватель, продемонстрировать полиморфизм `control()`.

Вариант 6

Класс АВТОМОБИЛЬ + классы ЛЕГКОВОЙ АВТОМОБИЛЬ, МИКРОАВТОБУС.
Реализовать схему наследования классов и корректно распределить по классам данные: фирма, модель, номер, цена, год выпуска, пробег.
Интерфейс расходователя топлива `IFuelConsumer` с методом `double consume(double distance_passed)` – рассчитать расход топлива по пройденной дистанции. Реализация метода в классе легкового автомобиля и микроавтобуса отличается коэффициентами (ввести произвольно). В `main()` создать 2 легковых автомобиля и 2 микроавтобуса, продемонстрировать полиморфизм `consume()`.

Вариант 7

Класс СТАДИОН + классы КОРТ, СПОРТИВНОЕ СООРУЖЕНИЕ.
Реализовать схему наследования классов и корректно распределить по классам данные: адрес, футбольный клуб, количество секторов, вместимость, посещаемость, площадь, тип покрытия, количество абонементов.
Интерфейс предоставителя абонементов `ISeasonTicketProvider` с методом `void ticket()` – сгенерировать новый абонемент с выдачей сообщения на экран номера абонемента. Реализация метода в классе стадиона: при выдаче абонемента (все свободные места для абонементов хранятся в файле) уменьшается количество свободных абонементов; максимум – 100 билетов. Реализация метода в классе корта: выдать абонемент можно только на летние месяцы (если текущий месяц – не летний, то не выдать билет); максимум – 15 билетов. В `main()` создать 1 корт и 2 стадиона, продемонстрировать полиморфизм `ticket()`.

Вариант 8

Класс ФОТОГРАФ + классы ДИЗАЙНЕР, СОТРУДНИК.
Реализовать схему наследования классов и корректно распределить по классам данные: имя, фамилия, пол, год рождения, год начала деятельности, рейтинг, телефон, количество фотографий в портфолио.
Интерфейс обработки компьютерных изображений `IPictureProcessor` с методом `void process(const std::string& photo_path)`. Реализация метода в классе фотографа: вывод на экран сообщения "Photo was processed by /имя_фотографа/", а также наращивание количества фотографий в портфолио. Реализация метода в классе дизайнера: вывод на экран сообщения "Image was produced by /имя_дизайнера/", а также изменение рейтинга на некоторую величину. В `main()` создать 2 фотографов и 2 дизайнеров, продемонстрировать полиморфизм `process()`.

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()`.

Вариант 10

Класс РЕСТОРАН + классы ЗАВЕДЕНИЕ ОБЩЕПИТА, КАФЕ.
Реализовать схему наследования классов и корректно распределить по классам данные: название, адрес, год основания, рейтинг, количество посетителей за месяц, количество мест, количество блюд в меню, суммарная прибыль.
Интерфейс заведения с возможностью резервирования стола `IReservable` с методом `void reserve()`. Для ресторана резервирование производится путем уменьшения количества свободных мест на 1; при этом происходит прирост прибыли на 100 грн, для кафе – также путем уменьшения количества свободных мест на 1 и наращивание рейтинга на величину 0.1. В `main()` создать 2 ресторана и 2 кафе, продемонстрировать полиморфизм `reserve()`.

Приложение В. Фрагменты кода для варианта 0.

Задание 1.

Необходимый минимум функционала, который должен быть отражен в функции `main()`:

```
void main()
{
	// Создать массив указателей на абстрактный класс обработки арифметических выражений.
	// Например, так:
	ExpressionEvaluator* evaluators[3];

	// создать объект первого типа (CustomExpressionEvaluator) ( x1*x2 + x3 + x4 + ... )
	// и заполнить его данными первым способом (установить отдельно каждый операнд)

	evaluators[0] = new CustomExpressionEvaluator(3);
	evaluators[0]->setOperand(0, 5);
	evaluators[0]->setOperand(1, 4);
	evaluators[0]->setOperand(2, -2);	// должно вычисляться 5*4 + (-2) = 18

	// создать объект второго типа (Summator) (x1 + x2 + x3 + x4 + ...)
	// и заполнить его данными вторым способом (массивом операндов)

	evaluators[1] = new Summator(10);
	double sum_operands[] = { 5, -2.5, 3.5, 8, 1.5, -9.5, 3, -4, 6.5, -2.5 };
	evaluators[1]->setOperands(sum_operands, 10);

	// должно вычисляться 5 + (-2.5) + 3.5 + 8 + 1.5 + (-9.5) + 3 + (-4) + 6.5 + (-2.5) = 9

	// создание объекта третьего типа (Multiplier) (x1 * x2 * x3 * x4 * ...)
	// и заполнить его данными вторым способом

	evaluators[2] = new Multiplier(3);
	double operands[] = { 0.25, 8, 3.5 };
	evaluators[2]->setOperands(operands, 3);	// должно вычисляться 0.25 * 8 * 3.5 = 7

	// проход в цикле по указателям evaluators
	// и вывод на консоль и в файл лога выражения (в консоли еще сам результат выражения)

	for (size_t i = 0; i < 3; ++i)			// демонстрация полиморфизма
	{
		evaluators[i]->logToFile("Lab3.log");
		evaluators[i]->logToScreen();
		std::cout << evaluators[i]->calculate() << std::endl;
	}

	// ... здесь организовать еще цикл по указателям evaluators, в теле которого
	// ... проверить тип текущего объекта, и если он реализует интерфейс IShuffle,
	// ... то вызвать метод shuffle() этого объекта, после чего метод calculate()
	// ... и затем отобразить на экране результат перемешивания и вычисления выражения


	// ... освобождение памяти
	// ...
}
```

Результат работы программы (последние три строки отражают результат перемешивания операндов выражения `CustomExpression`: первый и последний поменялись местами):

```
[3]
5 * 4 + (-2)
18

[10]
5 + (-2.5) + 3.5 + 8 + 1.5 + (-9.5) + 3 + (-4) + 6.5 + (-2.5)
9

[3]
0.25 * 8 * 3.5
7

[3]
(-2) * 4 + 5
-3
```

Приложение C. Режим «дозаписи» в файл ios::app в fstream.

```
#include <ios>
#include <fstream>

void main()
{
    std::ofstream log("logfile.txt", std::ios_base::app | std::ios_base::out);

    log << "Log Record\n";       // при повторном запуске строка допишется в файл

    return 0;
}
```