УДК 004(891+42)

РАЗРАБОТКА ПРИЛОЖЕНИЯ ДЛЯ РЕШЕНИЯ ЗАДАЧ, СВЯЗАННЫХ С ЭКСПЕРТНЫМ ОЦЕНИВАНИЕМ

Д.Р. Зайчук
Брестский государственный технический университет, г. Брест
Научные руководители: Л.В. Лизун, Т.Г. Хомицкая

Введение. Экспертное оценивание (expert judgement) — процесс получения оценки проблемы на основе мнения специалистов в исследуемой области (экспертов) с целью последующего принятия решения или выбора. Существует много методов ранжирования альтернатив, основываясь на мнениях экспертов, полученных в виде оценки предпочтений. Некоторые из них учитывают компетентность экспертов в той или иной области (например, метод взвешивания), некоторые позволяют выбрать наилучший, опуская порядок предпочтения остальных целей (принцип Кондорсе) [1]. Однако, этапы экспертного оценивания одинаковы для всех методов:

1. Постановка цели исследования.
2. Выбор формы исследования, определение бюджета проекта.
3. Подготовка информационных материалов, бланков анкет, модератора процедуры.
4. Подбор экспертов.
5. Проведение экспертизы.
6. Анализ результатов (обработка экспертных оценок).
7. Подготовка отчета с результатами экспертного оценивания.

С учетом трудоемкости обработки экспертных оценок при большом количестве альтернатив и экспертов, разработка программного обеспечения, облегчающего процесс сбора и анализа информации, имеет смысл.

Постановка задачи. Необходимо разработать и протестировать приложение, осуществляющее расчеты, основанные на методах экспертных оценок. Необходимый функционал: выбор метода решения, ручной ввод данных, решение задачи, основываясь на выбранном методе, вывод решения на экран, вывод решения в файл. Реализуемые методы экспертных оценок: метод парных сравнений, метод последовательных сравнений, метод взвешивания, метод предпочтения, принцип Кондорсе, метод Кемени-Снелла.

Описание алгоритмов и структуры программы. Приложение будет состоять из трех модулей: модуль ввода, модуль вычисления и модуль вывода. Модуль ввода должен обеспечивать выбор метода, предоставлять возможность ввода исходных данных. Модуль вычисления должен обрабатывать данные и выдавать результат в подходящем для отрисовки и вывода в файл формате данных. Модуль вывода данных отвечает за отрисовку результата вычислений на мониторе пользователя и сохранение результата в файл.

Так как в модуле вычислений отличается только алгоритм обработки входных данных, в рамках данной статьи рассмотрим только один метод – принцип Кондорсе.

Рассмотрим принцип Кондорсе, базируясь на результатах частного ранжирования альтернатив a_1, a_2, a_3, a_4, a_5.

1. Эксперты осуществляют ранжирование альтернатив:

$Э_1 = (a_1, a_3, a_2, a_5, a_4)$;

$Э_2 = (a_1, a_2, a_4, a_3, a_5)$;

$Э_3 = (a_1, a_2, a_5, a_3, a_4)$;

$Э_4 = (a_2, a_3, a_1, a_5, a_4)$;

$Э_5 = (a_2, a_4, a_3, a_1, a_5)$.

2. Находятся оценки m_{ik}, характеризующие предпочтение альтернатив в парных сравнениях

n_{ik}	1	2	3	4	5
a_1					
a_2					
a_3					
a_4					
a_5					

В частности, три эксперта $Э_1$, $Э_2$, $Э_3$ указали что альтернатива a_1 является более предпочтительной чем альтернатива a_2, поэтому $m_{12} = 3$.

3. Выполняются проверки согласно принципу Кондорсе: наилучшей является альтернатива a_i, если $m_{ik} \geq m_{ki}$ для всех $k \neq i$. В нашем случае при i=1 имеем:

$m_{12} \geq m_{21}$ $(3 \geq 2)$; $m_{13} \geq m_{31}$ $(3 \geq 2)$; $m_{14} \geq m_{41}$ $(4 \geq 1)$; $m_{15} \geq m_{51}$ $(5 \geq 0)$,

т.е. альтернатива a_1 удовлетворяет правилу Кондорсе.

4. Выбирается альтернатива Кондорсе. В данном случае это a_1.

Разработка алгоритмов. В качестве разработки алгоритмов в «черновом» варианте будем использовать язык программирования C++, который далее будет переписан на JavaScript. Из подключаемых библиотек нам понадобятся string и iostream. Так как далее методы будут выноситься в отдельные модули, будем превентивно выносить алгоритмы в отдельные функции. Для простоты разработки опустим ввод данных из консоли.

В качестве примера приведен фрагмент кода (рис. 1) на языке программирования C++, осуществляющий обработку данных с использованием принципа Кондорсе.

Процесс реализации приложения. Наше программное обеспечение будем разрабатывать на фреймворке Nuxt.JS (https://nuxtjs.org/) для языка JavaScript. В качестве основного языка для разработки алгоритмов обработки данных будем использовать C++. Редактором кода будет выступать Visual Studio Code с расширениями для JS, HTML, CSS, Nuxt.JS и Git. Для оптимизации процесса разработки и предотвращения утери наработок нам понадобится репозиторий на GitHub.

```
void kondorse() {
    int inputArray[5][5] = {
        { 1, 3, 2, 5, 4},
        { 1, 2, 4, 3, 5},
        { 1, 2, 4, 5, 3},
        { 3, 1, 2, 5, 4},
        { 4, 1, 3, 2, 5}
    };
    int modMatrix[5][5];
    for (int i = 0; i < 5; i++) {
        for (int j = 0; j < 5; j++) {
            modMatrix[i][j] = 0;
        }
    }
    for (int i = 0; i < 5; i++) {
        for (int j = 0; j < 5; j++) {
            for (int k = j+1; k < 5; k++) {
                if (inputArray[i][j] < inputArray[i][k])
                    modMatrix[j][k]++;
                else
                    modMatrix[k][j]++;
            }
        }
    }
    for (int i = 0; i < 5; i++) {
        int sum = 0;
        for (int j = 0; j < 5; j++) {
            sum += modMatrix[i][j];
        }
        modMatrix[i][i] = sum;
    }
    for (int i = 0; i < 5; i++) {
        for (int j = 0; j < 5; j++) {
            cout << modMatrix[i][j] << "\t";
        }
        cout << endl;
    }
    for (int i = 0; i < 5; i++) {
        int res = 0;
        for (int j = 0; j < 5; j++) {
            if (modMatrix[i][j] > modMatrix[j][i])
                res++;
        }
        if (res == 4)
            cout << "best is: " << i + 1;
    }
    cout << endl;
}
```

Рисунок 1

Немного о принципах разработки с использованием Nuxt.JS.

Один из основных – atomic design, суть которого заключается в разбивании интерфейсов на фундаментальные блоки и комбинировании их, постепенно наращивая сложность. Следует понимать, что разделение элементов на типы носит относительный характер. Атомы и молекулы нам предоставляет ant-design, организмы, шаблоны и страницы будем создавать мы.

Таким образом, определена структура интерфейса приложения (рис. 2).

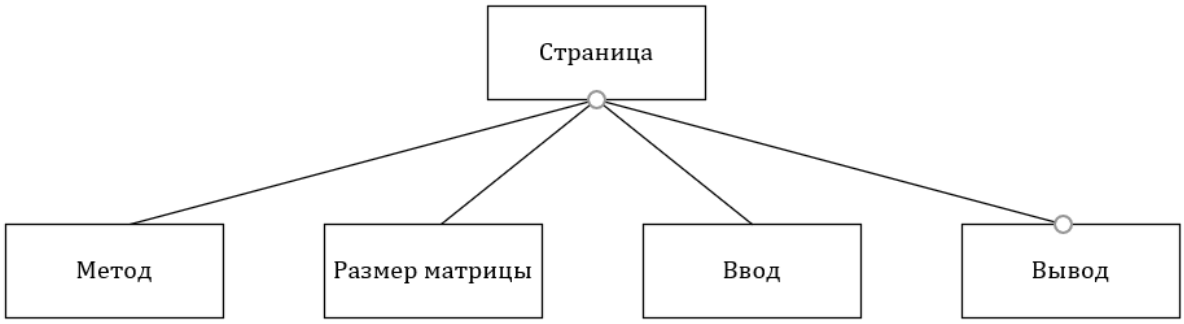

Рисунок 2

Второй принцип – разделить логику и отрисовку. Чтобы это реализовать, вся логика выносится во vuex, а в компонентах осуществляется только ввод-вывод данных. Алгоритмы обработки входных данных также выносятся в отдельный файл. Схема потока данных представлена на рис. 3.

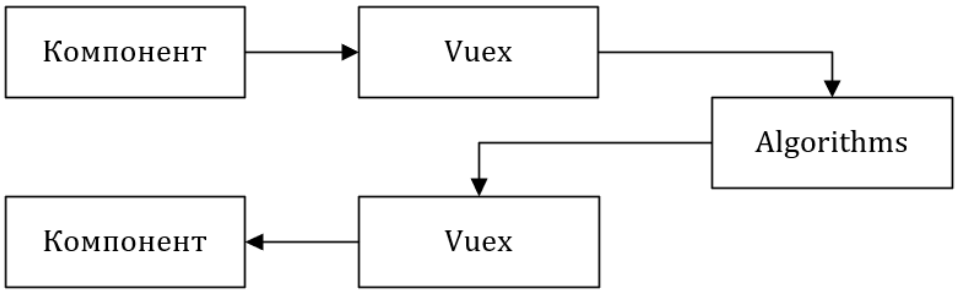

Рисунок 3

Практическое применение. Приложение способно решать задачи, связанные с экспертными оценками, используя шесть методов. При этом результат выводится на экран, а также может быть записан в файл. Формат вывода максимально приближен к формату оформления лабораторной работы, что может быть крайне полезно при проверке работ.

Список используемых источников.

1. Колбин, В. В. Математические методы коллективного принятия решений : учебное пособие / В. В. Колбин. — Санкт-Петербург : Лань, 2015. — 256 с.

УДК 378.147:51

КОЛЕБАНИЯ В ЦЕПИ ЕМКОСТИ С ИСКРОВЫМ ПРОМЕЖУТКОМ

А. В. Зарецкий, Н. Н. Сендер
Брестский государственный университет имени А. С. Пушкина, Брест,

Типичная схема использования конденсатора показана на рисунке 1. В цепь включен источник напряжения с ЭДС ε и сопротивлением R (роль R может играть внутреннее сопротивление источника напряжения). Ниже находится искровой промежуток; при разности потенциалов меньше определенного значения φ_1 искровой промежуток является изолятором. При $\varphi = \varphi_1$ проскакивает искра, между проводами воздух накаляется до высокой температуры и становится хорошим проводником. Суммарное сопротивление подводящих проводов и