

Рабочий лист №1

Дата "1" февраля 2025 г.
 (заполняется оргкомитетом)

Шифр ПН-33
 (заполняется оргкомитетом)

Оценка работы

(таблица заполняется по итогам проверки работы членами жюри олимпиады)

№ задания	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	Итого (итоговый балл, подпись председателя жюри)
Балл																78
№ задания	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	
Балл																AB

Магистриум

(название олимпиады, заполняется участником)

Профиль ПРИКЛАДНАЯ ИНФОРМАТИКА

(профиль олимпиады, заполняется участником)

3) а) Так как нужно извлекать критическую персонифицированную информацию, то можно предположить, что числа представлены в виде набора цифр, а не набора слов. То есть, у нас есть строка из слов и чисел, идущая через пробел. Так же можно предположить, что нет двух идущих подряд пробелов. Еще можно предположить то, что числа идут только после упоминания расшифровки ватной аббревиатуры. Например: "СНИЛС". Так же нам заранее должен быть известен массив ватной аббревиатур.

б) Далее предполагается по полученной строке телеграфного разговора и массива аббревиатур вернуть словарь соответствия аббревиатур к значению этой аббревиатуры. Так же следует учесть то, что между аббревиатурой и числом не должно быть других аббревиатур.

2 лист

сумма 3 листа

1) В первых, нам надо хранить информацию в базе данных. Для этого будем использовать Postgres. В самом простом варианте нам хватит номер, текстовая расшифровка звонка и итоговая оценочная вероятность. Саму текстовую расшифровку можно и не хранить, если мы до этого достигли вероятности. Но, можно предположить, что звонок по телефону и тому же номеру может быть много. И вероятность множественности у одного звонка тоже может быть разной. Оставим пока последние знания безумно по очень правильно. Поэтому предлагаю сделать две таблицы: в первой будут храниться все данные, а во второй только итоговое, которое каждому номеру соответствуют лишь одну вероятность. Сама перекомпиляция можно сделать раз в две минуты. Итоговая таблица:

AllCalls:

id	- int
Time Stamp	- Time Stamp
telephone Number	- string
P	- double

ResultCalls:

id	telephoneNumber
P	

следует отметить, что в AllCalls telephoneNumber может повторяться, а в ResultCall он должен быть уникальным.

б) Сама архитектура будет основана на ASP.NET.

Предполагается, что данные будут поступать исключительно от внешних источников. Запись в базу данных тоже будет происходить постоянно. Для приема внешних вызовов можно использовать web-адреса сервера.

За собой генерацию и создание прецедентов (файлов) будем отвечать внешние сервисы. Нужно также добавить взаимодействие с ограниченным доступом (чтобы не каждый мог отправлять
3 март

Дополнительный рабочий лист
(без рабочего листа №1 недействителен)

Дата "1" февраля 2025 г.
(заполняется участником)

Шифр ПИ - 33
(заполняется участником)

3) Когда будем писать на языке C#. Ф-ция будет принимать массив (List) аббревиатур и цел. строку телефонного разговора. Для простоты ввода и значения будем тип string.

```
public Dictionary<string, string> GetPersonalInfo(string  
callString, List<string> abbreviations) {  
    Dictionary<string, string> result = new Dictionary<string, string>();  
    foreach (string abrv in abbreviations) {  
        string pattern = $"@_{abrv}\\D+(\\d+)";  
        Match match = Regex.Match(callString, pattern);  
        if (match.Success) {  
            string number = match.Groups[1].Value;  
            result[abrv] = number;  
        }  
    }  
    return result;  
}
```

Следует учесть, что эта ф-ция находит результаты не для всех аббревиатур.

2) В данном задании можно использовать ф-цию из 3 задания для составления аббревиатур. Мы запускаем этот ~~задание~~ код и запускаем составление. Далее этот словарь нужно отфильтровать. Кодом из первого задания (массив, который оставляет только числами правильное значение аббревиатур. Например, CVC код должен состоять только из 3 символов и так далее). После запуска "проверки" аббревиатур вероятность можно посчитать как разность от найденного количества аббревиатур от общего количества.

Код:

```
List<string> abbs = new List<string>{...}
string callString = "...";
```

← данные
← кодовая-часть строки телефонного
разбора

```
Dictionary<string, string> findAbbs = GetPersonalInfo(callString,
abbs);
```

```
Dictionary<string, string> filteredAbbs = FilteredAbbs(abbs);
```

```
double p = (double)filteredAbbs.Count / abbs.Count;
```

мы можем тоже использовать другие метрики и другие распределения вероятностей, например, когда у нас найдено 5 аббревиатур, а их всего 10, вероятность бонуса не 0.5, а выше. И мы можем считать ошибку, что эта вероятность полностью совершенства.

Дополнительный рабочий лист
(без рабочего листа №1 недействителен)

Дата " 1 " февраля 20 25 г.
(заполняется участником)

Шифр ПИ - 33
(заполняется участником)

3) 6) для этого храним подгружен ASP.NET Identity. В нем уже есть базовые вещи для построения авторизации и аутентификации.

6) Далее нужны будут наши алгоритмы и сервисы обработки данных. Разобьем их на классы C#:

PersonalInfoExtractorService (внутри есть ф-ция из 3 задания) - нужен для извлечения вашей информации.

ProbabilitySolvingService (внутри код из 2 задания) -

- нужен для определения вероятности именованных объектов по некоторым признакам.

ResultProbabilityService - класс для воссоединения именованной вероятности для каждого имени.

DbRepository - класс для добавления данных в БД. В нем должно быть метод добавления единичных данных в AllCalls от внешних сервисов (с воссоединением вероятности), а также метод для обновления всех данных из ResultCalls. Сам класс repository будет работать с Entity Framework (используя базу данных).

Благодаря EF не нужно писать запрос на языке SQL. Остаточное использовать много классов C#.

2) Все классы нужно использовать через DI (Dependency Injection). Это нужно для поддержания чистоты кода.

1) г) Таким же путем можно составить пример для возврата вероятности при отсутствии интереса. Другой и самую интересную можно составить через веб-сайт