Технічні науки

UDC 004.021

Mychka Sviatoslav
Student of the
Kharkiv National University of Radio Electronics
Мичка Святослав Олегович
студент
Харківського національного університету радіоелектроніки
Мычка Святослав Олегович
студент
Харьковского национального университета радиоэлектроники

Supervisor:
Holian Nataliia
Associate Professor of the department of Software Engineering
Kharkiv National University of Radio Electronics

AN ALGORITHM FOR FINDING THE WEIGHTED COST OF LIVING IN RENTED HOUSING IN THE SOFTWARE SYSTEM FOR AUTOMATING THE CALCULATIONS OF RENTED HOUSING LIVING COST

АЛГОРИТМ ЗНАХОДЖЕННЯ ЗВАЖЕНОЇ ВАРТОСТІ ПРОЖИВАННЯ В ОРЕНДОВАНОМУ ЖИТЛІ В ПРОГРАМНІЙ СИСТЕМІ ДЛЯ АВТОМАТИЗАЦІЇ ПІДРАХУНКУ ВАРТОСТІ ОРЕНДОВАНОГО ЖИТЛА

АЛГОРИТМ НАХОЖДЕНИЯ ВЗВЕШЕННОЙ СТОИМОСТИ ПРОЖИВАНИЯ В АРЕНДОВАННОМ ЖИЛЬЕ В ПРОГРАММНОЙ СИСТЕМЕ ДЛЯ АВТОМАТИЗАЦИИ ПОДСЧЁТА СТОИМОСТИ АРЕНДОВАННОГО ЖИЛЬЯ

Summary. *Theoretical issues related to the development of an algorithm for finding the cost of living in rented housing, depending on specific characteristics.*

Key words: *algorithm, weighted average, calculation, cost, housing, rent.*

Анотація. *Досліджено теоретичні питання щодо розробки алгоритму для знаходження вартості проживання в орендованому житлі, залежно від конкретних характеристик.*

Ключові слова: *алгоритм, зважене середнє, підрахунок, вартість, житло, оренда.*

Аннотация. *Исследованы теоретические вопросы разработки алгоритма для нахождения стоимости проживания в арендованном жилье, в зависимости от конкретных характеристик.*

Ключевые слова: *алгоритм, взвешенное среднее, подсчет, стоимость, жилье, аренда.*

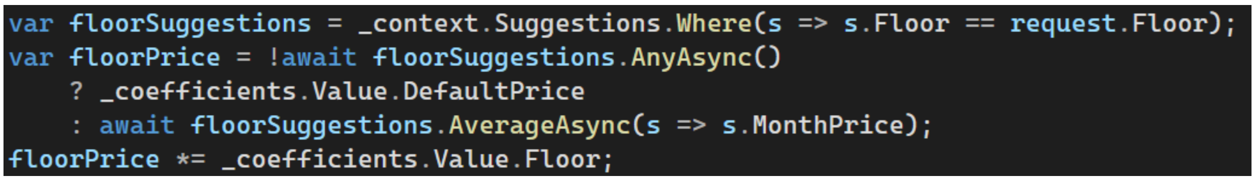
Rented housing prices are growing nowadays. Therefore, there is a need for a solution that will take control over the rapidly rising prices, offering transparent criteria that will indicate whether one rented housing should be more expensive to live in than the other one. Those criteria may be district of the city where the housing is located, distance from public transport stops or even the floor.

The best solution for this problem is to create a software that will automatize the calculations needed for fair prices distribution. The system will offer a website where housing owners will have an ability to advertise the houses for rent and find suiting prices.

The system to be developed is relevant because it is designed to solve the stated problem. With its help, housing prices will be formed using previous suggestions by such weighting criteria as the city area, proximity to public transport and the floor. The system will set a range of acceptable prices for accommodation this way, without allowing tenants to pay too much for housing and without allowing the owners to set unreasonably low prices.

The essence of the algorithm is to apply certain weights to the arithmetic mean of the cost of living in a house that meets a certain criterion and the formation of the resulting value. The algorithm has 4 steps and is developed using C# programming language [1], and the figures depict syntax of this language, too.

After receiving a POST request via HTTP [2], on the first 2 steps separate prices are formed according to criteria. There is an example of code that performs these operations shown on figure 1. This code performs a search operation on the same floor and finds the arithmetic mean of their prices. If no such suggestions are found, some default value is set.

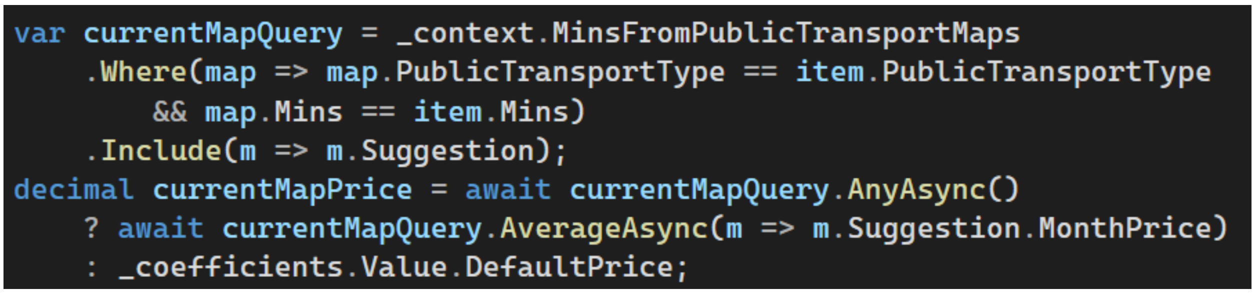


```
var floorSuggestions = _context.Suggestions.Where(s => s.Floor == request.Floor);
var floorPrice = !await floorSuggestions.AnyAsync()
    ? _coefficients.Value.DefaultPrice
    : await floorSuggestions.AverageAsync(s => s.MonthPrice);
floorPrice *= _coefficients.Value.Floor;
```

Fig. 1. Formation of separate criteria prices

Similar methods are used to search for proposals by district, only the coefficients and models change.

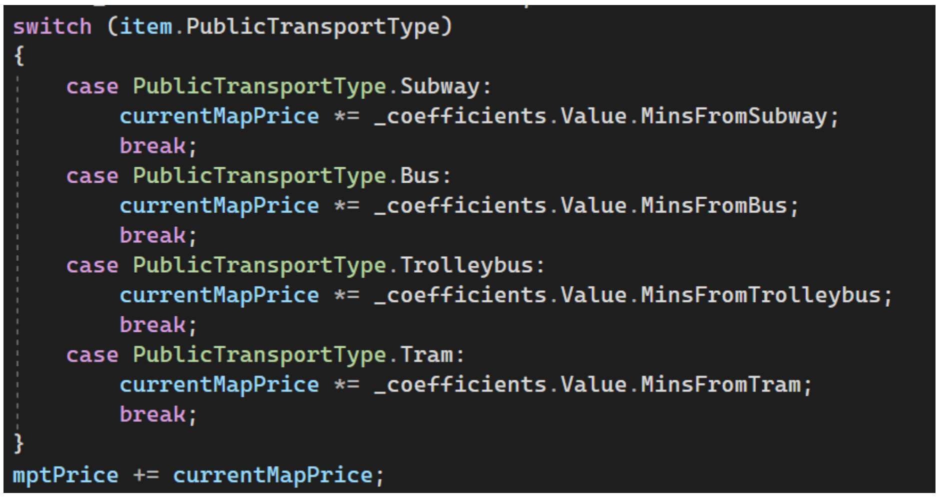
After finding the weighted prices by floor and area, the algorithm needs to find the weighted coefficients for each of the elements of the dictionary that contains proximity to public transport, which comes in the request from the client part. This is performed for each part of the dictionary. Figure 2 shows calculating currentMapPrice, which contains raw price on each iteration.



```
var currentMapQuery = _context.MinsFromPublicTransportMaps
    .Where(map => map.PublicTransportType == item.PublicTransportType
        && map.Mins == item.Mins)
    .Include(m => m.Suggestion);
decimal currentMapPrice = await currentMapQuery.AnyAsync()
    ? await currentMapQuery.AverageAsync(m => m.Suggestion.MonthPrice)
    : _coefficients.Value.DefaultPrice;
```

Fig. 2. Calculating raw average prices for every public transport type

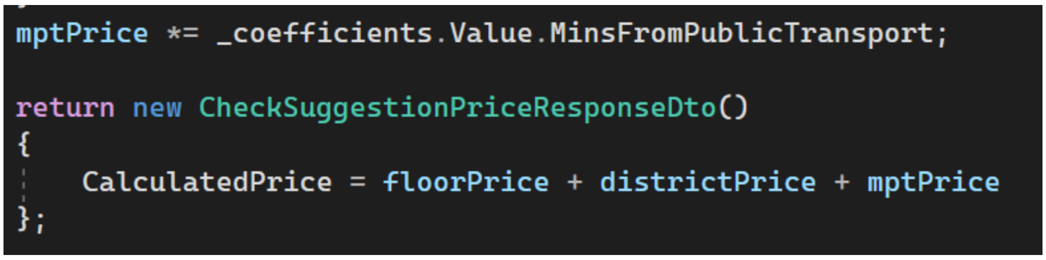
Every value that has been calculated by the algorithm on this step needs to be multiplied by the weight coefficient of its type (fig. 3). Then these values are added to a general variable that stores overall price for housing that have the same proximity to public transport.



```
switch (item.PublicTransportType)
{
    case PublicTransportType.Subway:
        currentMapPrice *= _coefficients.Value.MinsFromSubway;
        break;
    case PublicTransportType.Bus:
        currentMapPrice *= _coefficients.Value.MinsFromBus;
        break;
    case PublicTransportType.Trolleybus:
        currentMapPrice *= _coefficients.Value.MinsFromTrolleybus;
        break;
    case PublicTransportType.Tram:
        currentMapPrice *= _coefficients.Value.MinsFromTram;
        break;
}
mptPrice += currentMapPrice;
```

Fig. 3. Calculating overall price for the 3rd step

On the 4th step the algorithm multiplies the overall price for the 3rd step by an overall proximity to public transport weight coefficient and calculates the resulting price (fig. 4).



```
mptPrice *= _coefficients.Value.MinsFromPublicTransport;

return new CheckSuggestionPriceResponseDto()
{
    CalculatedPrice = floorPrice + districtPrice + mptPrice
};
```

Fig. 4. Calculating overall price

Therefore, the algorithm calculations may be represented using formulas:

$$p_x = \frac{\sum_{i=1}^{n} p_{xi}}{n},$$

$$result = \sum p_x * c_x,$$

where:

- p_x – the price for xth criterion, like floor, district etc,
- n – count of suggestions with the same *x* value, for example, located on the same floor or in the same district,

- p_{xi} – the price of i^{th} suggestion from n found,
- c_x – weight coefficient for x^{th} criterion, for example, for floor, district, public transport proximity etc.

The algorithm was designed to be the main part of the system for automating the calculations of rented housing living cost. This is used for approximating and allowing to set fair prices.

Literature

1. C# 8.0 and .NET Core 3.0 – Modern Cross-Platform Development: Build applications with C#, .NET Core, Entity Framework Core, ASP.NET Core, and ML.NET using Visual Studio Code, 4th Edition // Packt Publishing, 2019. 818 p.
2. Adam Freeman. Pro ASP.NET Core 3 // Publisher: Apress, 2020. 1109 p.