Тинькофф

Pattern matching в C# 8.0

Полетаев Кирилл
Full Stack разработчик
mr.dead.toast@gmail.com

tinkoff.ru

О себе

- Full Stack разработчик в Tinkoff bank
- Любимые языки: C#, TS, PureData, F#
- Романтизирую веб технологии и мировую паутину
- Веду трансляции о веб разработке на Twitch.tv

А что такое паттерн матчинг?

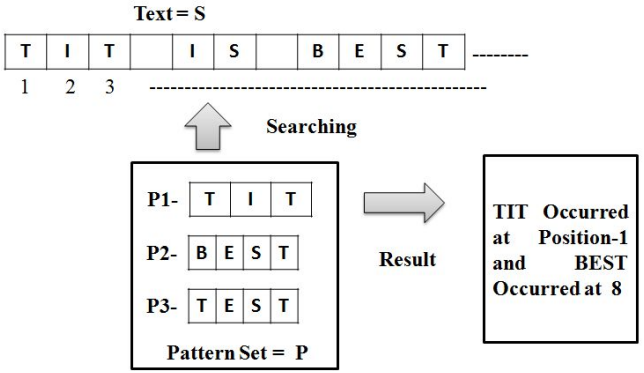
Что такое паттерн матчинг?

Pattern matching - Это языковая конструкция, представляющая проверку последовательности значений, или структуры со значениями на соответствие определенной форме(паттерну) для выполнения дальнейших инструкций

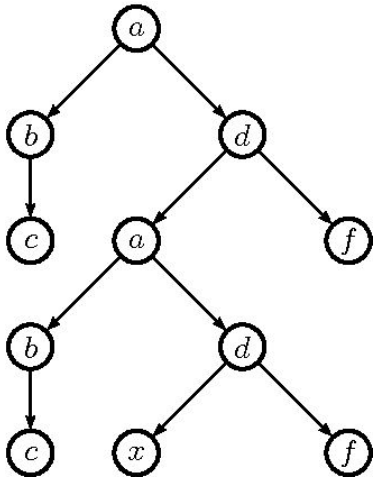
Виды паттерн матчинга

Паттерн матчинг разделяется на два типа:

String matching



Tree based



Откуда растут ноги?

- String pattern matching
  - Regular Language - Стивен Кол (1950)
  - SNOBOL - Дэвид Фарбер, Ральф Грисволд (1962 по 1967)
  - QED Editor - Кен Томпсон (1965-66)
  - SPITBOL - Робер Деуар, Энтони Маккан (1970)

- Tree pattern matching
  - LISP - Джон Маккарти, Стив Расселл (1962)
  - REFAL - Валентин Турчин (1968)
  - Prolog - Роберт Ковальски, Алайн Колмерауер (1972)
  - SASL - Дэвид Тюрнер (1972)
  - Scheme - Гай Л Стил Геральд Сусман (1975)

Как приготовить паттерн матчинг в C# 8?

Готовим паттерн матчинг: базовый синтаксис

```
// 1. Напрямую
var result = employee switch {
    { Id: 1 } => "Admin",
};

// 2. Как делегат
public static string GetType(ManagerEmployee employee) =>
    employee switch {
        {Id: 1} => "Admin"
    };
```

Готовим паттерн матчинг: Null и Default

```
public static string GetType(ManagerEmployee employee) =>
        null => throw new ArgumentNullException(),
        {} => throw new ArgumentException()
    };
```

Готовим паттерн матчинг: type matching

```
public static IEnumerable<Claim> AssignDefaultClaims(User u) =>
    u switch
    {
        AdminUser admin // 1. TYPE
            => new[] { new Claim("AdminTools", "true")},

        OperatorUser @operator // 2. Next TYPE
            => new[] { new Claim("OperatorTools", "true") },

        ConsumerUser consumer
            => new[] { new Claim("Id", $"{consumer.Id}") },

          // 3. Default section
        null => throw new ArgumentNullException(),
        _ => throw new ArgumentException()
    };
```

Готовим паттерн матчинг: value destructing

```
public static decimal SalaryBonusResolver(Employee e) =>
e switch
{
    CallCenterEmployee // 1
        {Rank: var rank, PerformanceRating: var rating, Income: var income } // 2
            => SalaryCalculator.CallCenterEmployeeBonus(income, rank, rating),

    ManagerEmployee // 1
        { Income: var income, Region: var region, Role: var role} // 2
            => SalaryCalculator.ManagerEmployeeBonus(income, region, role),

    null => throw new ArgumentNullException(),
    _ => throw new ArgumentException(),
};
```

Готовим паттерн матчинг: prop matching

```
public static Task SubscriptionModelResolver (SubscribeForm form) =>
form switch
{
    { UserId: 0 } => throw new ArgumentException(), // 1 Default type

    { PaidSubscription: false, UserId: var id } // 2. Destruct + Match
        => SubscribeService.SubscribeAsTrial(id),

    { SubscriptionType: SubscriptionType.Yearly, UserId: var id}
        => SubscribeService.SubscribeAsYearly(id),

    { SubscriptionType: SubscriptionType.Monthly, UserId: var id }
        => SubscribeService.SubscribeAsMonthly(id),

    null => throw new ArgumentNullException(), // 3. Default
    _ => throw new ArgumentException(),
};
```

Готовим паттерн матчинг: when query

```
public static Task CreatePost(PostModel model) =>
model switch
{
    { AuthorId: 0 } // 1. Validation
        => throw new ArgumentNullException(),

    { Text: var text } // 2. Destructing + Linq
        when string.IsNullOrWhiteSpace(text) // 3. when linq
        => throw new ArgumentException(),

    { AuthorId: var id, Media: var media, Text: var text }
        when media.Count != 0 // 4. When after destructing + match
            => PostService.CreatePostWithContent(id, media, text),

    { AuthorId: var id, Text: var text }
        => PostService.CreatePost(id, text),

    null => throw new ArgumentNullException()
};
```

Готовим паттерн матчинг: всё вместе

```
SubscribeForm // 1
    {PaidSubscription: true, UserId: var id, AdditionalModules: var adm} // 2
        when adm.Count != 0 // 3
             => SubscribeService.SubscribeWithModules(id, adm),// 4
```

Как делать не надо!

Как делать не надо: последовательность типов

```
public static string GetType(Employee employee) =>
    employee switch
    {
        Employee e => "value",
        CallCenterEmployee p => "value",
    };
```

Как делать не надо: паттерн матчинг без default

```
public static string GetType(Employee employee) =>
    employee switch
    {
        CallCenterEmployee p => "value",
    };
```

Как делать не надо: открываем бракеты

```
public static string GetType(Employee employee) =>
    employee switch
    {
        CallCenterEmployee p => { return "value" },
        Employee e => "value",
    };
```

Что почитать на досуге?

Ссылка на материалы

Полезные материалы:

https://gist.github.com/Keroosha/642e3be7e9703b48ad25285f37517bc7

Спасибо за внимание,
ваши вопросы

Telegram: @Keroosha
Twitter: @mrdeadtoast
Email: mr.dead.toast@gmail.com
Twitch.tv:
https://www.twitch.tv/mrdeadtoast