ПОЛИТЕХ
Санкт-Петербургский политехнический университет Петра Великого

F#

#2 Данные и переменные

Подготовили ст.гр. 5030102/10201
Дмитриев Михаил
Романчук Евгений
Соломатов Александр

План презентации:

01. Типы данных

02. Операции над типами

03. Преобразование/ приведение типов

04. Типизация и вывод типов

05. Сравнение типов

06. Области видимости

Типы данных.

01.

Основные примитивные типы.

```
// such int types
let a: int = 10  // целое число
let b: int64 = 100L  // 64-битное целое число
let c: byte = 255uy  // беззнаковый байт
// such float types
let pi: float = 3.14159  // 64-битное вещественное число
let e: float32 = 2.71828f  // 32-битное вещественное число
// types for boolean
let bool: bool = true  // логическое значение
// types for text
let sym: char = 'a'  // символ
let str: string = "Hello"  // строка
// types for other
let t: unit = ()  // пустое значение
```

Целочисленные типы: int, int16 , int64, uint16, byte, sbyte.

Числа с плавающей точкой: float, float32, double.
Булевы: bool

Символьный и строковый типы: char, string

Тип unit.

01.01

Пользовательские типы

01.02

```
// Tuples:
let tuple = (1, "Hello", true) // Кортеж из разных типов
//Arrays:
let numbers: int[] = [| 1; 2; 3; 4; 5 |]
// List:
let list = [1; 2; 3; 4] // Список целых чисел

// Records:
type Person = { Name: string; Age: int }

let person1 = { Name = "Alice"; Age = 30 }
printfn "%s is %d years old." person1.Name person1.Age

// Discriminated Unions
type Shape =
    | Circle of radius: float
    | Rectangle of width: float * height: float
```

Комплексные типы данных:

Tuples, Arrays, List
Discriminated Unions.
Optionals, Records.
FunctionType.

Пользовательские типы. Продолжение

```
// Classes
type Counter(initialValue: int) =
    let mutable count = initialValue

    member this.Increment() =
        count <- count + 1

    member this.GetCount() = count

// Way for Interfaces
type IDrawable =
    abstract member Draw: unit -> unit

type Circle(radius: float) =
    interface IDrawable with
        member this.Draw() =
            printfn "Drawing a circle with radius %f" radius
```

бъявление классов и интерфейсов.

02. Операции над типами

Обобщенные типы

Обобщенные типы

Обобщенные типы позволяют писать функции и структуры, которые работают с любыми типами данных, предоставляя универсальные решения

03.1 Преобразование типов

Явное приведение типов

Явное приведение типов

```
let a = 10
let b = float a  // Преобразование int в float

let strValue = "123"
let intValue = int strValue  // Преобразование строки в
int
```

F# требует явного указания на преобразование типов для повышения безопасности.
Это предотвращает случайные ошибки в коде, когда данные могут быть неправильно интерпретированы.

03.2 Преобразование типов

Преобразование сложных типов

Преобразование сложных типов

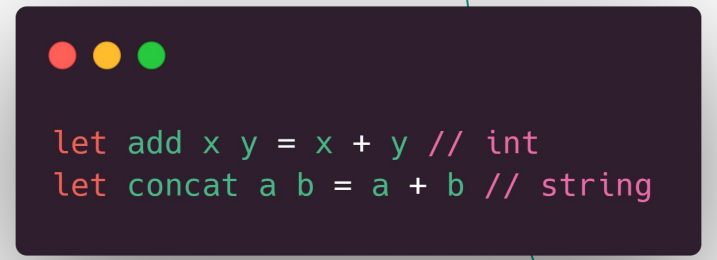
```
// Преобразование кортежей
let tuple1 = (1, "text")
let tuple2 = (fst tuple1, snd tuple1 + "
addition")
// Преобразование записей
type OldPerson = { Name: string; Age: int }
type NewPerson = { FullName: string; Age: int }

let convertToNewPerson (p: OldPerson) =
    { FullName = p.Name; Age = p.Age }
```

Для более сложных типов, таких как записи или кортежи, F# позволяет использовать специальные функции или паттерн-матчинг для преобразования.

04. Типизация и вывод типов

Строгая статическая типизация

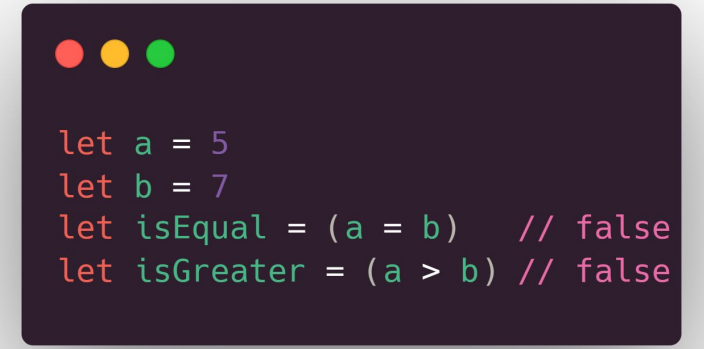
Строгая статическая типизация



F# является статически типизированным языком, что означает, что все типы переменных известны на этапе компиляции. Это делает код более предсказуемым и безопасным.

05. Сравнение типов

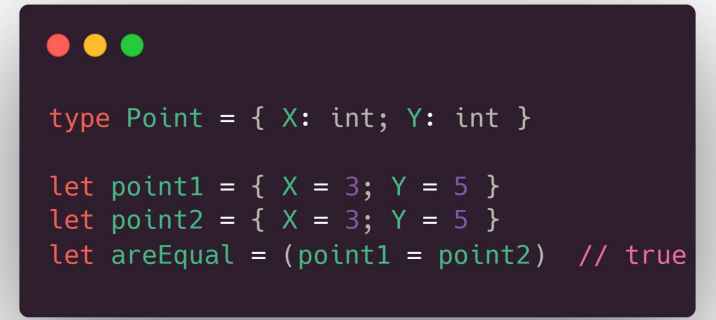
Введение



```
let a = 5
let b = 7
let isEqual = (a = b)   // false
let isGreater = (a > b) // false
```

- Сравнение типов в F# — это проверка равенства и порядка между значениями.
- Основные операторы: `=`, `<>`, `<`, `>`, `<=`, `>=`.

Равенство и неравенство



- Операторы равенства: = и <>.
- Эти операторы работают для всех примитивных типов и некоторых пользовательских типов.

Сравнение величин

```
let x = 5
let y = 10
let isLess = (x < y)  // true
```

- Операторы <, >, <=, >= используются для числовых значений.
- Нельзя сравнивать значения разных типов без явного приведения.

Проблема разных типов

```
let x = 5   // int
let y = 5.0 // float
// Ошибка компиляции: невозможно сравнить int и float
```

- Сравнение `int` и `float` вызывает ошибку компиляции.
- F# поддерживает строгое разделение типов для безопасности.

06. Области видимости

Введение

```
let exampleFunction () =
    let localVariable = 10
    printfn "%d" localVariable
//localVariable доступна только внутри функции.
```

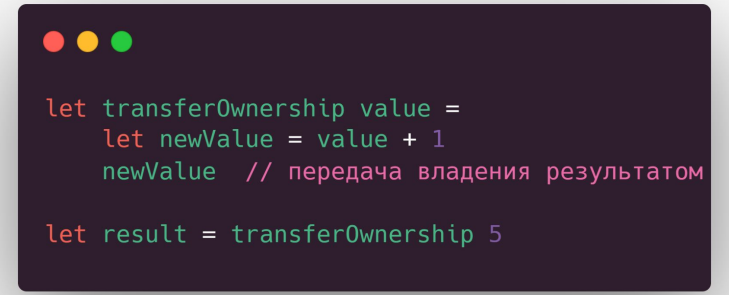
- Область видимости определяет, где переменные и функции могут быть использованы.
- F# поддерживает локальные и модульные области видимости.

Модульная область видимости


```
module MyModule =
    let moduleVariable = 42
//moduleVariable доступна только внутри MyModule.
```

- **Модули позволяют организовать код и ограничивать доступ к переменным и функциям.**

Передача владения



```
let transferOwnership value =
    let newValue = value + 1
    newValue  // передача владения результатом

let result = transferOwnership 5
```


- **Владение переменной может быть передано через параметры функции или через возвращаемое значение.**

Список литературы

Основные образовательные ресурсы:

- Metanit: F# Tutorial
- JavaTpoint: F# Tutorial

Официальная документация:

- Документация F# на Microsoft Learn

Наш Git-репозиторий с примерами:

- F# Tutorial на GitHub