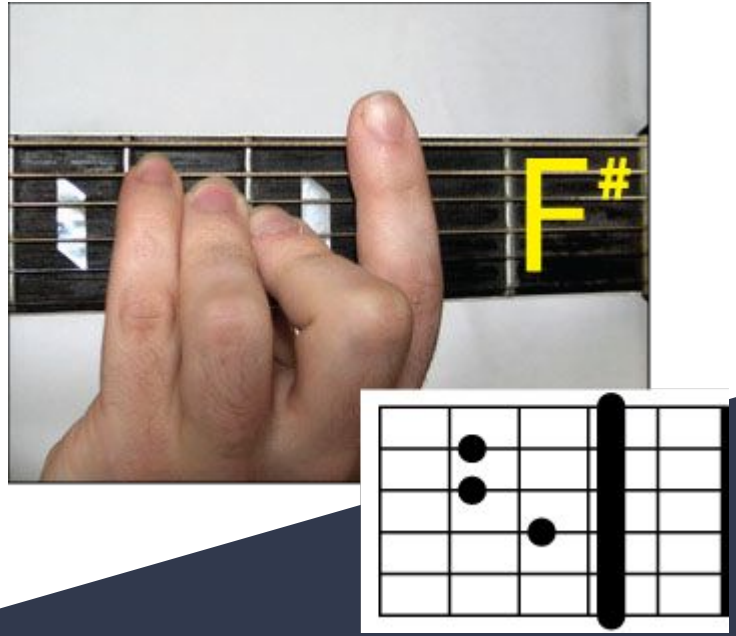
ПОЛИТЕХ
Санкт-Петербургский политехнический университет Петра Великого

F#
Введение

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



Введение в F#

Основы и ключевые особенности

Что такое F#?

- F# — это функциональный язык программирования, работающий в экосистеме .NET.
- Поддерживает гибридный стиль: функциональное, объектно-ориентированное и императивное программирование.
- Популярен в задачах с высокими требованиями к производительности и математической точности.

Основные особенности:

- Лаконичность кода.
- Поддержка функциональных и асинхронных вычислений.
- Высокая типобезопасность.

История F#

Этапы развития языка

История развития:

- Разработан в *Microsoft Research* под руководством Дона Сайма.
- Основан на *OCaml*, стремился объединить преимущества функционального программирования с мощью .NET.
- Первая версия вышла в 2005 году, официальная поддержка в *Visual Studio* — с 2010.
- Постепенно язык стал развиваться как независимый инструмент, используемый за пределами *Microsoft.*

Важные этапы:

- *2005*: Первый релиз F#.
- *2010*: F# включён в *Visual Studio.*
- *2013*: Появление F# *Software Foundation* для поддержки сообщества.

Области применения F#

Реальные сценарии использования

Сферы и области:

- Научные вычисления и моделирование.
- Обработка данных и машинное обучение.
- Веб-разработка и серверные приложения.
- Функциональное программирование для разработки надёжных и масштабируемых систем.

Примеры использования:

- Финансовые компании для моделирования рисков.
- Научные исследования и работа с большими данными.
- Обработка сигналов и работа с потоками данных в реальном времени.

Установка

Установка .NET SDK

F# является частью .NET платформы, поэтому для работы с F# вам нужно установить .NET SDK. Для этого:

1. Перейдите на [страницу загрузки .NET SDK](https://dotnet.microsoft.com/download).
2. Выберите нужную версию .NET SDK (обычно, последнюю стабильную версию).
3. Скачайте установочный файл для вашей операционной системы и следуйте инструкциям по установке.

Установка

Установка редактора кода

Для удобной работы с кодом F# рекомендуется использовать редактор, поддерживающий F#. Один из популярных вариантов - [Visual Studio Code](https://code.visualstudio.com/) с расширением Ionide.

Установка Visual Studio Code:

1. Перейдите на [страницу загрузки Visual Studio Code](https://code.visualstudio.com/).
2. Скачайте и установите редактор для вашей операционной системы.

Установка расширения Ionide:

1. Откройте Visual Studio Code.
2. Перейдите в раздел расширений (можно нажать Ctrl+Shift+X).
3. В строке поиска введите "Ionide".
4. Установите расширение Ionide-fsharp.

Создание первого проекта

Создание проекта F#

Для создания нового проекта F# используйте команду `dotnet new`. Откройте командную строку или терминал и выполните следующие шаги:

1. Создайте директорию для проекта и перейдите в неё:

```sh
mkdir MyFSharpApp
cd MyFSharpApp
```

2. Создайте новый проект F#:

```sh
dotnet new console -lang "F#"
```

Эта команда создаст новый проект консольного приложения на языке F#.

Создание первого проекта

Запуск первого проекта

1. В каталоге проекта вы найдете файл `Program.fs`. Откройте его в вашем редакторе кода. Этот файл содержит основной код приложения.

2. В файле `Program.fs` должен быть следующий код:

```fsharp
// Program.fs
[<EntryPoint>]
let main argv =
    printfn "Hello, world!"
    0 // Return an integer exit code
```

3. Сохраните изменения и запустите проект командой:

```sh
dotnet run
```

Вы должны увидеть вывод "Hello, world!" в командной строке.

Помещение проекта в Docker

Создание Dockerfile

1. В корне проекта создайте файл с именем `Dockerfile` и добавьте в него следующий код:

```
# Используем официальный образ SDK для сборки приложения
FROM mcr.microsoft.com/dotnet/sdk:5.0 AS build
WORKDIR /app

# Копируем файлы проекта и устанавливаем зависимости
COPY *.fsproj ./
RUN dotnet restore

# Копируем остальные файлы и собираем проект
COPY . ./
RUN dotnet publish -c Release -o out

# Используем официальный образ Runtime для запуска приложения
FROM mcr.microsoft.com/dotnet/aspnet:5.0 AS runtime
WORKDIR /app
COPY --from=build /app/out .

# Указываем команду для запуска приложения
ENTRYPOINT ["dotnet", "FirstIonideProject~.dll"]
```

Убедитесь, что имя DLL в `ENTRYPOINT` совпадает с именем скомпилированного файла вашего приложения.

Создание первого проекта

Создание и запуск Docker образа

1. Постройте Docker образ:

```sh
docker build -t proglangs .
```

2. Запустите контейнер из созданного образа:

```sh
docker run --rm proglangs
```

Вы должны увидеть вывод "Hello, world!" в консоли.

Оператор let.

Наиболее важным оператором, без которого сложно обойтись, является оператор let.

Общая форма определения значения:

let название_значения = данные/действия

Важно, что по умолчанию значения - immutable, для изменения mutable объектов используем оператор <-.

Основные типы данных.

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

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

Символьный и строковый типы: char, string

Комплексные типы данных:
- Tuples, Arrays, List
- Discriminated Unions.
- Optionals.

Тип unit.

Функции.

Общее объявление.
let имя_функции параметры = действия функции

Объявление функции с параметрами.
let название_функции параметр1 параметр2 ... параметрN = действия_функции

Типизация параметров при объявлении.
let имя_функции (параметр1: тип) (параметр2: тип) (параметрN: тип) = действия_функции

Вызов функции (в случае без типизации параметров).
название функции значение для параметра1 значение_для_параметра2 ... значение_для_параметраN

Типизация результата функции.
let имя_функции параметр1 параметр2 ... параметрN : тип_результата = действия_фукции

EntryPoint

Основные правила.

- [<EntryPoint>] указывает точку входа программы, аналогично функции main в других языках.
- Функция, помеченная EntryPoint, должна быть единственной в программе и находиться в одном из модулей. И быть последней в файле.
- Тип возвращаемого значения функции — целое число (int), которое используется как код завершения программы.
- Функция должна принимать один параметр — массив строк (string[]), представляющий аргументы командной строки.

Источники

Основной образовательные источники:

https://metanit.com/f/tutorial
https://www.javatpoint.com/f-sharp-tutorial/

Документация языка:

https://learn.microsoft.com/en-us/dotnet/fsharp/