Programowanie Funkcyjne 2020/2021 – laboratorium – studia stacjonarne

Laboratorium 1

Zagadnienia:

- Środowisko pracy - Visual Studio
- Interaktywny i wsadowy tryb pracy
- Wykonywanie obliczeń – wyrażenia
- Wybrane typy w języku F#
- Definiowanie stałych
- Definiowanie funkcji
- Funkcje rekurencyjne

Środowisko pracy - Visual Studio

- Tworzymy nowy projekt (Visual F#-> Console Application)
- Wyniki pracy zapisujemy w plikach tekstowych z rozszerzeniem fs
- Kod uruchamiamy w konsoli (F# Interactive)
- Alt+' – uruchomienie w konsoli bieżącej linii kodu
- Alt+Enter – uruchomienie w konsoli zaznaczonego fragmentu kodu
- Po wpisaniu wyrażenia bezpośrednio do konsoli należy zakończyć je podwójnym średnikiem (;;), aby zostało zinterpretowane

Wykonywanie obliczeń – wyrażenia

- Standardowe operatory arytmetyczne (+, -, *, /, %)
- Nie można mieszać typów wyrażeń
- Standardowe operatory nie sprawdzają przekroczenia zakresu (unchecked)
- Operatory sprawdzające przekroczenie zakresu są dostępne w module Microsoft.FSharp.Core.Operators.Checked

Wybrane typy w języku F#

Typ	**Opis**	**Przykładowe literały**	**Nazwa .NET**
bool	True/false values	true, false	System.Boolean
byte	8-bit unsigned integers	0uy, 19uy, 0xFFuy	System.Byte
sbyte	8-bit signed integers	0y, 19y, 0xFFy	System.SByte
int16	16-bit signed integers	0s, 19s, 0x0800s	System.Int16
uint16	16-bit unsigned integers	0us, 19us, 0x0800us	System.UInt16
int, int32	32-bit signed integers	0, 19, 0x0800, 0b0001	System.Int32
uint32	32-bit unsigned integers	0u, 19u, 0x0800u	System.UInt32
int64	64-bit signed integers	0L, 19L, 0x0800L	System.Int64
uint64	64-bit unsigned integers	0UL, 19UL, 0x0800UL	System.UInt64
single, float32	32-bit IEEE floating-point	0.0f, 19.7f, 1.3e4f	System.Single
double, **float**	64-bit IEEE floating-point	0.0, 19.7, 1.3e4	System.Double
decimal	High-precision decimal values	0M, 19M, 19.03M	System.Decimal
bigint	Arbitrarily large integers	0I, 19I	Math.BigInt
bignum	Arbitrary-precision rationals	0N, 19N	Math.BigNum
unit	The type with only one value	()	Core.Unit
string	Sequence of Unicode UTF-16 chars.	"Mickey Mouse"	System.String

Definiowanie stałych i funkcji

- Za pomocą słowa **let**, np. `let x = 5`
- Funkcje tak samo, tylko z parametrami, np. `let funkcja x = 2*x`
- Wbudowany system wnioskowania o typach próbuje wydedukować typy argumentów i zwracanego wyniku
- Domyślnym typem jest `int`
- Można (a czasami nawet trzeba) podać jawne adnotacje o typach, np. `let funkcja (x:float) (y:float):float = x+y`
- Często wystarczy częściowa adnotacja, resztę wykryje system wnioskowania, np. `let funkcja (x:float) y = x+y`

Funkcje rekurencyjne w języku F# definiujemy za pomocą konstrukcji:
let rec *nazwa_funkcji argumenty = ciało funkcji;;*

Moduły służą do grupowania funkcji.
Deklaracja modułu: ***module*** nazwa_modułu = *deklaracje funkcji*
Wywołanie: **nazwa_modułu.nazwa_funkcji** argumenty
Pamiętajmy o odpowiednim formatowaniu (wcięcia)!

Wypisywanie na konsolę

Do wypisywania na konsolę służą funkcje `printf` i `printfn`. Przyjmują one co najmniej jeden argument typu string. Jeśli pierwszy argument zawiera specjalne znaczniki formatujące, to należy podać dodatkowe argumenty, np.:

```
printfn "x=%f, s=%s" 2.0 "Ala ma kota"
```

Wybrane znaczniki formatujące:

- %d lub %i liczba całkowita
- %f liczba zmiennoprzecinkowa
- %s napis
- %b wartość logiczna
- %O obiekt
- %A dowolny typ

Podobnie działa funkcja `sprintf`, ale napis nie jest wypisywany na konsolę, tylko zwracany jako wynik funkcji.

Zadanie 1. **[3p]**

Napisz funkcje obliczające pola i obwody figur:

a) Kwadratu
b) Prostokąta
c) Koła

Funkcje mają operować na liczbach typu float.

Zadanie 2. **[1p]**

Napisz funkcję obliczającą wartość wielomianu $(-3y^4 + ½ x^2 + 2y - 7)$, dla $x,y \in R$.

Zadanie 3. **[3p]**

Napisz funkcje **rekurencyjne** w języku F# obliczające:

a) n-ty wyraz ciągu $\mathbf{a_n = a_{n-1} * q}$, dla $\mathbf{a_1}$ i **q** będących argumentami funkcji
b) największy wspólny dzielnik dwóch liczb naturalnych
c) sumę cyfr podanej liczby

Zadanie 4.

Napisz funkcję rekurencyjną, która sprawdza, czy podana liczba jest liczbą pierwszą. **[1p]**

Zadanie 5. **[2p]**

Korzystając z rekurencji napisz funkcje w języku F# **wypisujące na ekran**:

a) Kolejne liczby naturalne z przedziału **<a, b>** , gdzie $a,b \in \mathbf{N}$ i $a > b$
b) N-krotną konkatenację napisu podanego jako argument funkcji