МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

**«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»**

Кафедра математической кибернетики и компьютерных наук

**ОСНОВЫ РАБОТЫ С F# - ФУНКЦИИ (ФУНКЦИОНАЛЬНЫЙ СТИЛЬ) - КЛАССЫ (ОБЪЕКТНО - ОРИЕНТИРОВАННЫЙ СТИЛЬ)**

ЛАБОРАТОРНАЯ РАБОТА №3

(Вариант №5)

студента 3 курса 351 группы

направления 09.03.04 — Программная инженерия

факультета КНиИТ

Голикова Артема Олеговича

Проверил

к. пед. н., доцент \_\_\_\_\_\_\_\_\_\_\_\_ В. А. Векслер

Саратов 2020

**Часть 1-я**

**«Перегрузка операций, циклы и математические функции»**

**Цель работы:** ознакомиться с понятием перегрузки операций, циклов и математических функций в F#, на примере нескольких тренировочных задач закрепить знания.

**Краткие теоритические сведения:**

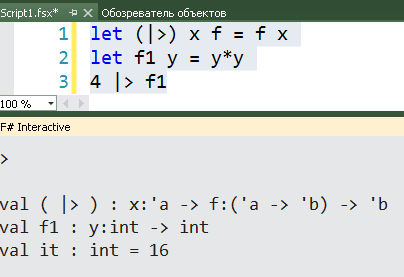
Перегрузка операторов — это функция, доступная во многих языках программирования. В F # вы можете не только перегрузить существующие операторы, но и определить новые. Выполнить перегрузку можно в унарном и бинарном вариантах. Возможность определять операторы обеспечивает более высокий уровень контроля над синтаксисом ваших программ, позволяя создавать выражения, которые лучше подходят для проблемы.

Ярким примером этого является вездесущий оператор |>.

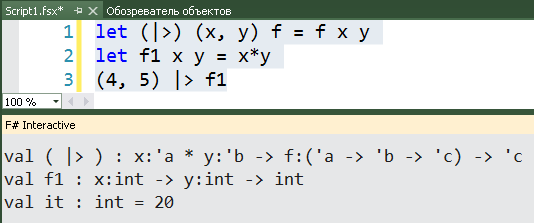
let (|>) x f = f x

Оператор указывает, что аргумент предшествует функции, которая должна его получить (это и определено по умолчанию).

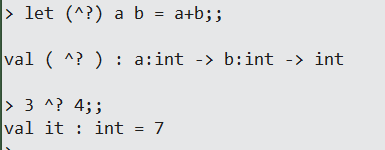
Пример:



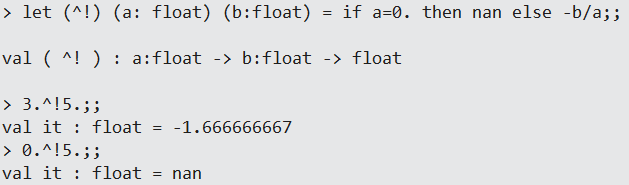
Переопределим операцию, таким образом, чтобы она работала только с парой чисел (картежом) и функцией с двумя аргументами. Как видим, операция определилась для любых типов.



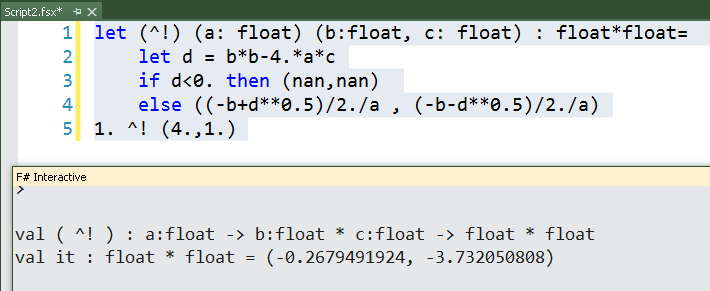
*Создадим новую операцию:*

**

*Создадим оператор для решения линейного уравнения:*

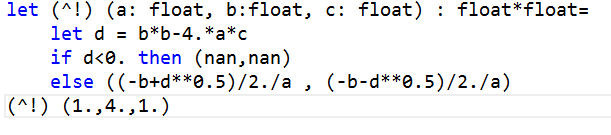
**

*Создадим бинарную операцию для решения квадратного уравнения*

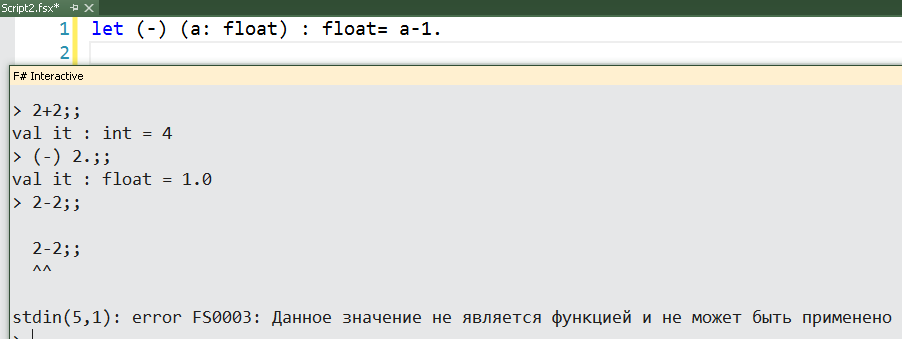
**

*float\*float – тип картеж двух вещественных чисел*

*Откажемся от бинарности. «Унарный» вариант:*

**

*Пример. Перепрядённую операцию можно использовать только в новом контексте:*

**

Общие соглашения об именах заключаются в следующем:

* “a”, “b”, “c” т.д., это типы
* “f”, “g”, “h” т.д., это функции
* “x”, “y”, “z” т.д., это аргументы функции
* Списки обозначаются добавлением суффикса «s», так что «xs» - это список x, «fs» - это список функций и т. Д. Чрезвычайно распространено, что «x :: xs» означает голову (первый элемент) и хвост (остальные элементы) списка.
* «\_» используется всякий раз, когда вы не заботитесь о значении. Итак, «x :: \_» означает, что вы не заботитесь о остальной части списка, и «let f\_ = something» означает, что вам не нужен аргумент f.

***Пример:***

***let*** *(|>) x f = f x*

*Мы не знаем, что такое f и x, f может быть любой функцией, а x может быть любой величиной. Явное выражение этого кода не нужно,. все понимание основывается на общих соглашениях*

***Циклы***

*Выражение for...to*

Выражение for...to используется для прохождения в цикле диапазона значений переменной цикла

for identifier = start [ to | downto ] finish do

body-expression

Несмотря на то, что технически for...to является выражением, оно больше похоже на традиционный оператор в императивном языке программирования. В следующих примерах показаны различные варианты использования выражения for...to.

let function1() =

for i = 1 to 10 do

printf "%d " i

printfn ""

let function2() =

for i = 10 downto 1 do

printf "%d " i

printfn ""

function1()

function2()

let beginning x y = x - 2\*y

let ending x y = x + 2\*y

let function3 x y =

for i = (beginning x y) to (ending x y) do

printf "%d " i

printfn ""

function3 10 4

Результат выполнения приведенного кода будет следующим.

1 2 3 4 5 6 7 8 9 10

10 9 8 7 6 5 4 3 2 1

2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18

*Выражение for...in*

Эта конструкция для организации циклов используется для перебора в перечислимой коллекции при соответствии шаблона, например в выражении диапазона, в последовательности, в списке, в массиве или в другой конструкции, поддерживающей перечисление.

for pattern in enumerable-expression do

body-expression

Перечислимое выражение можно задать в виде перечислимой коллекции или с помощью оператора .. в виде диапазона целочисленных значений. К перечислимым коллекциям относятся списки, последовательности, массивы, наборы, сопоставления и т. д.

При определении диапазона с помощью оператора .. можно использовать следующий синтаксис.

start .. finish

Можно также использовать версию, в которой инкремент имеет значение пропуска, как показано в следующем примере.

start .. skip .. finish

Если в качестве шаблона в целочисленном диапазоне используется простая переменная-счетчик, то обычно счетчик при каждой итерации увеличивается на 1, но если диапазон содержит значение пропуска, то счетчик будет увеличиваться на это значение.

Значения, соответствующие шаблону, также можно использовать в теле цикла.

В следующих примерах кода показано использование выражения for...in.

let list1 = [ 1; 5; 100; 450; 788 ]

for i in list1 do

printfn "%d" i

Выходные данные выглядят следующим образом.

1

5

100

450

788

В следующем примере показано применение цикла к последовательности и использование шаблона в виде кортежа вместо обычной переменной.

*Тип seq, представляет упорядочен­ную последовательность элементов.*

*> let seqOfNumbers = seq { 1 .. 5 };;*

*// Sequence that has an increment.*

*seq { 0 .. 10 .. 100 }*

*let sequence2 = seq { 10 .. -1 .. 1 }*

let seq1 = seq { for i in 1 .. 10 -> (i, i\*i) }

for (a, asqr) in seq1 do

printfn "%d squared is %d" a asqr

Выходные данные выглядят следующим образом.

1 squared is 1

2 squared is 4

3 squared is 9

4 squared is 16

5 squared is 25

6 squared is 36

7 squared is 49

8 squared is 64

9 squared is 81

10 squared is 100

В следующем примере показано применение цикла к простому целочисленному диапазону.

let function1() =

for i in 1 .. 10 do

printf "%d " i

printfn ""

function1()

Выходные данные функции function1 выглядят следующим образом.

1 2 3 4 5 6 7 8 9 10

В следующем примере показано применение цикла со значением пропуска 2, в результате чего выбирается каждый второй элемент диапазона.

let function2() =

for i in 1 .. 2 .. 10 do

printf "%d " i

printfn ""

function2()

Выходные данные функции function2 выглядят следующим образом.

1 3 5 7 9

В следующем примере показано использование диапазона знаков.

let function3() =

for c in 'a' .. 'z' do

printf "%c " c

printfn ""

function3()

Выходные данные функции function3 выглядят следующим образом.

a b c d e f g h i j k l m n o p q r s t u v w x y z

В следующем примере показано использование отрицательного значения пропуска для перебора в обратном порядке.

let function4() =

for i in 10 .. -1 .. 1 do

printf "%d " i

printfn " ... Lift off!"

function4()

Выходные данные функции function4 выглядят следующим образом.

10 9 8 7 6 5 4 3 2 1 ...Lift off!

Начало и конец диапазона также могут представлять собой выражения, например функции, как показано в следующем примере кода.

let beginning x y = x - 2\*y

let ending x y = x + 2\*y

let function5 x y =

for i in (beginning x y) .. (ending x y) do

printf "%d " i

printfn ""

function5 10 4

Выходные данные функции function5 со входными значениями выглядят следующим образом.

2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18

В следующем примере показано использование подстановочного знака (\_), если элемент не требуется использовать в цикле.

let mutable count = 0

for \_ in list1 do

count <- count + 1

printfn "Number of elements in list1: %d" count

Выходные данные выглядят следующим образом.

Number of elements in list1: 5

*Циклы: выражение while...do (F#)*

Выражение while...do используется для выполнения итерации (в цикле), пока заданное проверяемое условие истинно.

while test-expression do

body-expression

Проверяется истинность условия test-expression. Если результат проверки — значение true, выполняется выражение body-expression и снова проверяется истинность условия. Если результат проверки истинности условия — значение false, итерация заканчивается.

В следующем примере демонстрируется использование выражения while...do.

*Дополнительно:*

*let rng = new Random();; - создание генератора случайных чисел*

*> rng.Next();;*

*val it : int = 776780816*

*> rng.Next(10);;*

*val it : int = 8*

*rng.Next(дипазон);; - нахождение случайного числа в указанном диапазоне*

open System //подключение библиотеки

let lookForValue value maxValue =

let mutable continueLooping = true

let randomNumberGenerator = new Random()

while continueLooping do

// Generate a random number between 1 and maxValue.

let rand = randomNumberGenerator.Next(maxValue)

printf "%d " rand

if rand = value then

printfn "\nFound a %d!" value

continueLooping <- false

lookForValue 10 20

Выходные данные приведенного выше кода — серия случайных чисел от 1 до 20, последнее из которых — 10.

13 19 8 18 16 2 10

Found a 10!

> // Цикл while

let mutable i = 0

while i < 5 do

i <- i + 1

printfn “i = %d” i;;

i = 1

i = 2

i = 3

i = 4

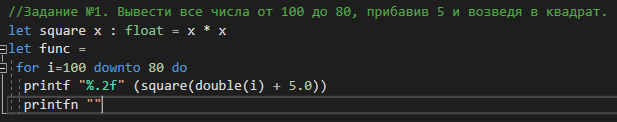
i = 5

val mutable i : int = 5

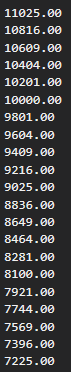
**Ход работы:**

**Задание №1.**

Числа от 100 до 80 увеличить на 5 и найти квадрат.



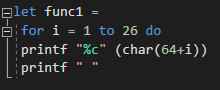
Результат выполнения:



**Задание №2.**

Выведите все заглавные буквы английского алфавита (2-мя способами).

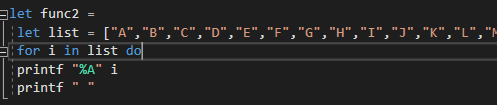
1 способ:



Результат выполнения:



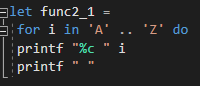
2 способ:



Результат выполнения:



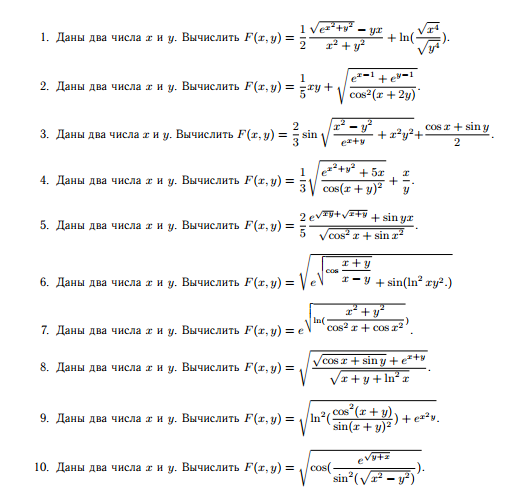
3 способ:



Результат выполнения:



**Задание №3.**



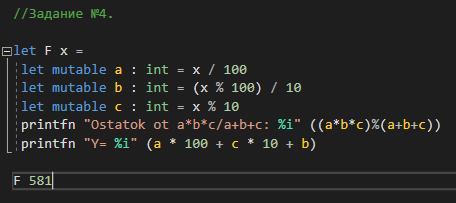


Результат выполнения:

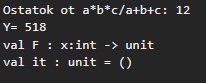


**Задание №4.**

Дано трехзначное целое число 𝑋. Найти остаток от деления произведения всех его цифр на сумму всех его цифр. Вывести новое число 𝑌, поменяв местами вторую и третью цифру числа 𝑋.



Результат выполнения:



**Контрольные вопросы:**

1. Для чего используется выражение for...to?

Выражение for...to используется для прохождения в цикле диапазона значений переменной цикла.

1. Для чего используется выражение for...in?

Эта конструкция для организации циклов используется для перебора в перечислимой коллекции при соответствии шаблона, например в выражении диапазона, в последовательности, в списке, в массиве или в другой конструкции, поддерживающей перечисление.

1. Для чего используется выражение while...do?

Выражение while...do используется для выполнения итерации (в цикле), пока заданное проверяемое условие истинно.

1. Опишите принцип работы выражения while...do.

while test-expression do

body-expression

Проверяется истинность условия test-expression. Если результат проверки — значение true, выполняется выражение body-expression и снова проверяется истинность условия. Если результат проверки истинности условия — значение false, итерация заканчивается.

1. Каким образом обозначается шаг в выражении for...in?

Если в качестве шаблона в целочисленном диапазоне используется простая переменная-счетчик, то обычно счетчик при каждой итерации увеличивается на 1, но если диапазон содержит значение пропуска, то счетчик будет увеличиваться на это значение.

1. Как можно выразить начало и конец диапазона в выражении for...in?

При определении диапазона с помощью оператора .. можно использовать следующий синтаксис.

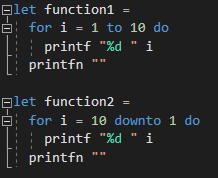
start .. finish

Можно также использовать версию, в которой инкремент имеет значение пропуска.

start .. skip .. finish

1. Перечислите варианты использования выражения for... to.

Примеры использование прямого цикла и обратного. Функции выводят числа от 1 до 10 и от 10 до 1.



Результат выполнения:



**Вывод:** ознакомился с понятием перегрузки операций, циклов и математических функций в F#, закрепил знания на примере нескольких тренировочных задач.

**Часть 2-я**

**Классы: Свойства и методы. Наследование. Обработка исключений.**

**Краткие теоритические сведения:**

Объявление метода или свойства начинается с ключевого слова member, за которым следует имя self-идентификатора. Это имя используется для обращения к членам класса внутри метода или свойства. В случае, если метод или свойство статические, self-идентификатор указывать не нужно. В отличие от C#, здесь нет неявной переменной this, которая передаётся в каждый нестатический метод класса и предоставляет доступ к объекту, для которого вызван этот метод. Вместо этого, разработчик волен сам выбирать удобное имя, под которым будет известен текущий объект.

Как и в C #, классы могут иметь статические члены, и это обозначается ключевым словом static. Статический модификатор предшествует ключевому слову участника. Статические элементы, не могут иметь самоидентификатор, такой как «это», потому что для них нет экземпляра.

В F# полностью поддерживается перегрузка методов. Можно определить несколько методов, имеющих одинаковое имя, но отличающихся типом и количеством параметров.

Как и любой объектно-ориентированный язык, F# позволяет создать новый класс, наследующий свойства уже существующего.

При явном описании класса-наследника необходимо после ключевого слова inherit указать имя базового класса, и вызвать его конструктор в constructor expression с помощью того же ключевого слова.

Преобразование:

:> оператор выполняет статическое приведение, что означает, что успешность приведения определяется во время компиляции.

Можно также использовать upcast оператор для выполнения такого преобразования. Следующее выражение задает преобразование вверх по иерархии:

upcast expression

:?> Оператор выполняет динамическое приведение, что означает, что успешность приведения определяется во время выполнения.

Можно также использовать downcast оператор для выполнения преобразования динамического типа. Следующее выражение задает преобразование к типу, который выводится из контекста программы вниз по иерархии:

downcast expression

match - выражение позволяет управлять ветвлением основывается на сравнении выражения с набором шаблонов.

:? возвращает true, если значение соответствует указанному типу; в противном случае возвращает false (оператор проверки типа).

as используется для присвоения текущему объекту объекта имя объекта. Также используется, чтобы дать имя целому шаблону в соответствии с шаблоном.

**Ход работы:**

Построить иерархию классов, начиная с абстрактного класса. Предусмотреть абстрактные методы в проектируемых классах, а также переопределение этих методов в классах-потомках. Приведите пример использование индексатора и свойств. Продемонстрируйте использование конструкции для обработки ошибок.

Класс 1. Объёмная фигура (Определить объём, Определить площадь поверхности).

Класс 2. Прямой круговой цилиндр (радиус основания, высота).

Класс 3. Полый цилиндр (внутренний радиус).

//Класс фигуры

type Figure =

class

val a : float

val b : float

val c : float

new (A: float, B: float, C: float) = {

a = A

b = B

c = C

}

member this.Volume = this.a \* this.b \* this.c

member this.Square = 2.0 \* ((this.a \* this.b) + (this.a \* this.c) + (this.c \* this.b))

member this.Item (ix:int) =

match ix with

| 0 -> this.a

| 1 -> this.b

| 2 -> this.c

| \_ -> failwith "Index out of range"

end

let a = new Figure(2.0, 3.0, 4.0)

try

printfn "Parameters of Figure: (%.2f, %.2f, %.2f)" a.[0] a.[1] a.[2]

printfn "Volume of Figure: (%.2f)" a.Volume

printfn "Square of Figure: (%.2f)" a.Square

with

| \_ as e -> printfn "Error: %s" e.Message

//Класс цилиндра

type Cylinder =

inherit Figure

val r : float

val h : float

new (A: float, B: float, C: float, R: float, H: float) = {

inherit Figure(A,B,C)

r = R

h = H

}

member this.Volume = Math.PI \* (this.r \* this.r) \* this.h

member this.Square = 2.0 \* Math.PI \* this.r \* (this.r + this.h)

member this.Item (ix:int) =

match ix with

| 0 -> this.r

| \_ -> failwith "Index out of range"

let b = new Cylinder(2.0, 3.0, 4.0, 5.0, 8.0)

try

printfn "Volume of Cylinder: (%.2f)" b.Volume

printfn "Square of Cylinder: (%.2f)" b.Square

with

| \_ as e -> printfn "Error: %s" e.Message

//Класс полого цилиндра

type HollowCylinder =

inherit Cylinder

val ir : float

new (A: float, B: float, C:float, R: float, H: float, iR: float) = {

inherit Cylinder(A, B, C, R, H)

ir = iR

}

member this.Volume = Math.PI \* this.h \* (this.r \* this.r - this.ir \* this.ir)

member this.Square = 2.0 \* Math.PI \* (this.r \* this.r - this.ir \* this.ir) + 2.0 \* Math.PI \* (this.r + this.ir) \* this.h

member this.Item (ix:int) =

match ix with

| 0 -> this.ir

| \_ -> failwith "Index out of range"

let c = new HollowCylinder(2.0, 3.0, 4.0, 5.0, 8.0, 3.0)

try

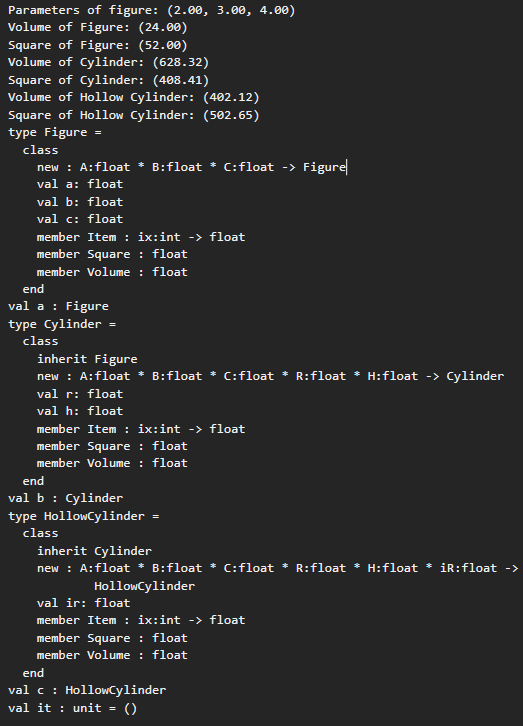
printfn "Volume of Hollow Cylinder: (%.2f)" c.Volume

printfn "Square of Hollow Cylinder: (%.2f)" c.Square

with

| \_ as e -> printfn "Error: %s" e.Message

Результат выполнения:



Значения верны. Программа работает корректно.

**Вывод:** ознакомился с понятием наследования и исключениями, применил их в задании; закрепил навыки работы с ООП, классами в языке F#.