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#** — **МАССИВЫ, БАЗА ДАННЫХ**

ЛАБОРАТОРНАЯ РАБОТА №6

(Вариант №5)

студента 4 курса 451 группы

направления 09.03.04 — Программная инженерия

факультета КНиИТ

Голикова Артема Олеговича

Проверил

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

Саратов 2020

**Часть 1-ая. Массивы.**

**Цель работы:** изучить основные принципы работы с массивами в F#, разработки приложений на F#.

**Теоретическая справка**

Массивы — это начинающиеся с нулевого индекса изменяемые коллекции фиксированного размера, содержащие последовательные элементы данных одного типа.

Массивы можно создавать несколькими способами. Небольшой массив можно создать, последовательно перечислив разделяемые точкой с запятой значения между символами [| и |], как показано в следующих примерах.

let array1 = [| 1; 2; 3 |]

val array1 : int [] = [|1; 2; 3|]

Также можно указать каждый элемент в отдельной строке (в этом случае точку с запятой можно опустить).

Тип элементов массива определяется по указанным литералам, и он должен быть одинаковым для всех элементов. Следующий код вызывает ошибку, поскольку 1.0 — это число с плавающей запятой, а 2 и 3 — целые числа.

// Causes an error.

// let array2 = [| 1.0; 2; 3 |]

Для создания массивов можно также использовать выражения последовательностей. Ниже приведен пример, в котором создается массив квадратов целых чисел от 1 до 10.

let array3 = [| for i in 1 .. 10 -> i \* i |]

Чтобы создать массив нулевых элементов, используйте функцию Array.zeroCreate.

let arrayOfTenZeroes : int array = Array.zeroCreate 10

Для доступа к элементам массива используется оператор точки (.) и квадратные скобки ([ и ]).

array1.[0]

Индексы массивов начинаются с 0.

Для доступа к элементам массива также можно использовать диапазонное представление, позволяющее указать поддиапазон массива. Ниже приведены примеры диапазонного представления.

// Accesses elements from 0 to 2.

array1.[0..2]

// Accesses elements from the beginning of the array to 2.

array1.[..2]

// Accesses elements from 2 to the end of the array.

array1.[2..]

// Accesses elements from 1 to the 3.

array1.[1..2]

При использовании диапазонного представления создается новая копия массива.

*Простые функции*

Функция Array.get получает элемент: Array.get array index

Функция Array.length возвращает длину массива: Array.length array

Также можно использовать свойство Length.

Array.length [| 1 .. 100 |] |> printfn "Length: %d"

Array.length [| |] |> printfn "Length: %d"

Array.length [| 1 .. 2 .. 100 |] |> printfn "Length: %d"

>

Length: 100

Length: 0

Length: 50

Функция Array.set задает для элемента указанное значение: Array.set array index value

Функция Array.create создает массив указанного размера и задает для всех элементов указанные значения.

let array1 = Array.create 10 "" ;;

val array1 : string [] = [|""; ""; ""; ""; ""; ""; ""; ""; ""; ""|]

Следующий пример кода иллюстрирует использование этих функций.

let array1 = Array.create 10 ""

for i in 0 .. array1.Length - 1 do

Array.set array1 i (i.ToString())

for i in 0 .. array1.Length - 1 do

printf "%s " (Array.get array1 i)

0 1 2 3 4 5 6 7 8 9

val array1 : string [] = [|"0"; "1"; "2"; "3"; "4"; "5"; "6"; "7"; "8"; "9"|]

val it : unit = ()

*Функции, создающие массивы*

Несколько функций создают массивы, не требуя наличия существующего массива.

Функция Array.empty создает новый массив, не содержащий никаких элементов.

Функция Array.create создает массив указанного размера и задает для всех элементов указанные значения.

Функция Array.init создает массив заданного размера, используя функцию для создания элементов: Array.init count initializer (>Array of squares: [|0; 1; 4; 9; 16; 25; 36; 49; 64; 81|])

Функция Array.zeroCreate создает массив, всем элементам которого присваивается нулевое значение, соответствующее типу массива. Следующий код демонстрирует эти функции.

let myEmptyArray = Array.empty

printfn "Length of empty array: %d" myEmptyArray.Length

printfn "Array of floats set to 5.0: %A" (Array.create 10 5.0)

printfn "Array of squares: %A" (Array.init 10 (fun index -> index \* index))

let (myZeroArray : float array) = Array.zeroCreate 10

>

Length of empty array: 0

Area of floats set to 5.0: [|5.0; 5.0; 5.0; 5.0; 5.0; 5.0; 5.0; 5.0; 5.0; 5.0|]

Array of squares: [|0; 1; 4; 9; 16; 25; 36; 49; 64; 81|]

Функция Array.copy создает новый массив, содержащий элементы, скопированные из существующего массива.

let array1 = [| 1 .. 10 |]

let array2 = Array.copy array1

printfn "%A\n%A" array1 array2

>

[|1; 2; 3; 4; 5; 6; 7; 8; 9; 10|] [|1; 2; 3; 4; 5; 6; 7; 8; 9; 10|]

Функция Array.sub создает новый массив из ограниченного диапазона элементов другого массива. Поддиапазон задается указанием начального индекса и длины. В следующем коде демонстрируется использование функции Array.sub.

let a1 = [| 0 .. 99 |]

let a2 = Array.sub a1 5 10

printfn "%A" a2

> val a2 : int [] = [|5; 6; 7; 8; 9; 10; 11; 12; 13; 14|]

Функция Array.append создает новый массив, объединяя два существующих массива.

Следующий код демонстрирует использование функции Array.append.

printfn "%A" (Array.append [| 1; 2; 3|] [| 4; 5; 6|])

>[|1; 2; 3; 4; 5; 6|]

Функция Array.choose выбирает элементы массива, включаемые в новый массив. Следующий код демонстрирует использование функции Array.choose. Обратите внимание, что тип элементов массива на обязательно должен соответствовать типу значения, возвращаемому в типе option. В данном примере элемент имеет тип int, а значение типа option является результатом выполнения полиномиальной функции elem\*elem - 1 и является числом с плавающей запятой.

printfn "%A" (Array.choose (fun elem -> if elem % 2 = 0 then

Some(float (elem\*elem - 1))

else

None) [| 1 .. 10 |])

[|3.0; 15.0; 35.0; 63.0; 99.0|]

Функция Array.collect применяет указанную функцию к каждому элементу существующего массива, а затем собирает элементы, созданные этой функцией, и объединяет их в новый массив. Следующий код демонстрирует использование функции Array.collect.

printfn "%A" (Array.collect (fun elem -> [| 0 .. elem |]) [| 1; 5; 10|])

>

[|0; 1; 0; 1; 2; 3; 4; 5; 0; 1; 2; 3; 4; 5; 6; 7; 8; 9; 10|]

Функция Array.concat принимает последовательность массивов и объединяет их в один массив. Следующий код демонстрирует использование функции Array.concat.

let multiplicationTable max = seq { for i in 1 .. max -> [| for j in 1 .. max -> (i, j, i\*j) |] }

printfn "%A" (Array.concat (multiplicationTable 3))

[|(1, 1, 1); (1, 2, 2); (1, 3, 3); (2, 1, 2); (2, 2, 4); (2, 3, 6); (3, 1, 3); (3, 2, 6); (3, 3, 9)|]

Array.sort - Сортирует элементы массива, возвращая новый массив. Для сравнения элементов используется оператор [Operators.compare](https://msdn.microsoft.com/ru-ru/library/ee353429(v=vs.110).aspx).

Array.sort array

let sortedArray1 = Array.sort [|1; 4; 8; -2; 5|]

printfn "%A" sortedArray1

[|-2; 1; 4; 5; 8|]

Array.sortBy - Сортирует элементы массива, используя заданную проекцию для ключей и возвращая новый массив.

Array.sortBy projection array

let sortedArray2 = Array.sortBy (fun elem -> abs elem) [|1; 4; 8; -2; 5|]

printfn "%A" sortedArray2

[|1; -2; 4; 5; 8|]

compare [|2;2;3|] [|1;4;3|]

>1 // разные

compare [|2;2;3|] [|2;2;3|]

>0 // одинаковые

Функция Array.filter принимает функцию логического условия и создает новый массив, содержащий только те элементы входного массива, для которых логическое условие истинно. Следующий код демонстрирует использование функции Array.filter.

printfn "%A" (Array.filter (fun elem -> elem % 2 = 0) [| 1 .. 10|])

"4234".ToCharArray() – разбитие строки на массив символов

val it : char [] = [|'4'; '2'; '3'; '4'|]

Функция Array.rev создает новый массив, изменяя порядок элементов существующего массива на обратный.

Array.rev ([|'4'; '2'; '3'; '4'|])

val it : char [] = [|'4'; '3'; '2'; '4'|]

let stringReverse (s: string) =

System.String(Array.rev (s.ToCharArray()))

printfn "%A" (stringReverse("!dlrow olleH"))

Функции модуля Array, преобразующие массивы, можно легко комбинировать с помощью оператора конвейера (|>), как показано в следующем примере.

[| 1 .. 10 |]

|> Array.filter (fun elem -> elem % 2 = 0)

|> Array.choose (fun elem -> if (elem <> 8) then Some(elem\*elem) else None)

|> Array.rev

|> printfn "%A"

Существует возможность создания многомерных массивов, но синтаксис литералов таких массивов не определен. Для создания массива из последовательности последовательностей элементов массива используйте оператор array2D.Последовательности могут быть литералами массивов или списков. Например, следующий код создает двумерный массив.

let my2DArray = array2D [ [ 1; 0]; [0; 1] ]

Функции Array.exists и Array.exists2 проверяют элементы в одном и в двух массивах соответственно. Эти функции принимают функцию проверки и возвращают значение true, если любой из элементов (или любая из пар элементов в случае функции Array.exists2) отвечает условию.

В следующем коде демонстрируется использование функций Array.exists и Array.exists2. В этих примерах новые функции создаются путем использования только одного из аргументов, в данных случаях — аргумента функции.

let allNegative = Array.exists (fun elem -> abs (elem) = elem)

printfn "%A" (allNegative [| -1; -2; -3 |])

printfn "%A" (allNegative [| -10; -1; 5 |])

printfn "%A" (allNegative [| 0 |])

let haveEqualElement = Array.exists2 (fun elem1 elem2 -> elem1 = elem2)

printfn "%A" (haveEqualElement [| 1; 2; 3 |] [| 3; 2; 1|])

>

false

true

true

true

Функция Array.forall проверяет, все ли элементы массива удовлетворяют заданному логическому условию. Вариант функции Array.forall2 выполняет такую же проверку с помощью логической функции, принимающей элементы двух массивов одинаковой длины. Следующий код иллюстрирует использование этих функций.

let allPositive = Array.forall (fun elem -> elem > 0)

printfn "%A" (allPositive [| 0; 1; 2; 3 |])

printfn "%A" (allPositive [| 1; 2; 3 |])

let allEqual = Array.forall2 (fun elem1 elem2 -> elem1 = elem2)

printfn "%A" (allEqual [| 1; 2 |] [| 1; 2 |])

printfn "%A" (allEqual [| 1; 2 |] [| 2; 1 |])

>false

true

true

false

Функция Array.find принимает логическую функцию и возвращает первый элемент, для которого функция возвращает значение true или вызывает исключение KeyNotFoundException, если элемент, удовлетворяющий условию, не найден. Функция Array.findIndex аналогична функции Array.find, но возвращает индекс элемента, а не сам элемент.

В следующем коде функции Array.find и Array.findIndex используются для поиска числа, которое одновременно является точным квадратом и точным кубом.

Функция Array.average возвращает среднее значение каждого элемента в массиве. Она ограничена типами элементов, поддерживающими точное деление на целое число, что включает типы с плавающей запятой, но не целочисленные типы. Функция Array.averageBy возвращает среднее значение результатов вызова функции для каждого элемента. Для массивов целого типа можно использовать функцию Array.averageBy и с помощью функции преобразовать каждый элемент массива в число с плавающей запятой для вычисления.

С помощью функции Array.max или Array.min можно получить максимальный и минимальный элемент, если тип элементов поддерживает такую возможность. Аналогичным образом функции Array.maxBy и Array.minBy позволяют сначала выполнить функцию, например для преобразования в тип, поддерживающий сравнение.

Функция Array.sum складывает элементы массива, а функция Array.sumBy вызывает вычислительную функцию для каждого элемента и складывает результаты.

Функция Array.set задает для элемента указанное значение. Функция Array.fill задает указанное значение диапазону элементов массива. В следующем примере кода приведен пример функции Array.fill.

let arrayFill1 = [| 1 .. 25 |]

Array.fill arrayFill1 2 20 0

printfn "%A" arrayFill1

[|1; 2; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 23; 24; 25|]

Функция Array.ofList создает новый массив из списка. Функция Array.ofSeq создает новый массив из последовательности. Функции Array.toList и Array.toSeq выполняют преобразование типа массива в типы этих других коллекций.

*Сортировка массивов*

Функция Array.sort позволяет упорядочить массив с помощью универсальной функции сравнения. Функция Array.sortBy позволяет указать функцию, создающую значение (называемое ключом), по которому выполняется упорядочение с помощью универсальной функции сравнения. Варианты функций Array.sortInPlace, Array.sortInPlaceBy и Array.sortInPlaceWith не возвращают новый массив, а изменяют существующий.

**Задание №1.**

На основе вещественного массива полученного от пользователя создать из него массивы целых положительных чисел и целых отрицательных чисел (отбросив дробную часть). Вывести результат и длины полученных массивов.

Код программы:

open System

Console.Title <- "Голиков А.О. - Массивы"

Console.ForegroundColor <- ConsoleColor.Cyan

Console.BackgroundColor <- ConsoleColor.Black

printfn "Введите массив вещественных чисел"

// Читаем массив с клавиатуры

let mutable inputmas = System.Console.ReadLine().Split[|' '|]

// Создаем массив вещественных чисел из массива типа string

let floatmas =

let mutable tmpmas = Array.create (Array.length inputmas) 0.0

for i = 0 to Array.length inputmas - 1 do

tmpmas.[i] <- float(inputmas.[i])

tmpmas

// Переводим массив типа float в массив типа int (отбрасываем дробную часть)

let intmas =

let mutable tmpmas = Array.create (Array.length inputmas) 0

for i = 0 to Array.length floatmas - 1 do

tmpmas.[i] <- int(floatmas.[i])

tmpmas

// Функции создания массивов положительных и отрицательных чисел

let plusmas =

Array.choose (fun x -> if x >= 0 then Some(x) else None) intmas

let minusmas =

Array.choose (fun x -> if x < 0 then Some(x) else None) intmas

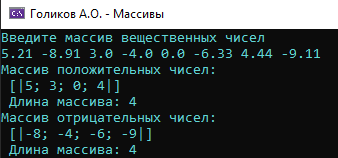
printfn "Массив положительных чисел: \n %A \n Длина массива: %A" plusmas (Array.length plusmas)

printfn "Массив отрицательных чисел: \n %A \n Длина массива: %A" minusmas (Array.length minusmas)

Console.ReadKey()

Console.Clear()

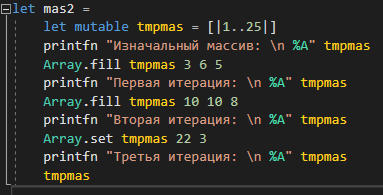
Результат выполнения:



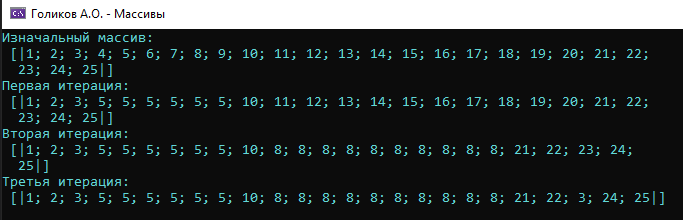
**Задание №2.**

В массиве чисел от 1 до 25 6 элементов, начиная с четвертого, обратить пятерками, 10 элементов, начиная с 11, обратить восьмерками, и двадцать третий элемент обратить в тройку.

Код программы:



Результат выполнения:



**Задание №3.**

В списке хранится информация о студентах (Фамилия, Имя, Отчество, Год Рождения, Рост).

1. Найдите всех студентов родившихся в 1998 году.

2. Найдите самых молодых студентов.

Код программы:

// Создадим тип Student с нужными параметрами

type Student = {lastname : string; firstname : string; middlename : string; birthyear : int; height : int}

// Создаем массив списков

let Students : Student [] = [| {lastname = "Иванов"; firstname = "Дмитрий"; middlename = "Сергеевич"; birthyear = 1994; height = 178};

{lastname = "Сидоров"; firstname = "Константин"; middlename = "Ильич"; birthyear = 1998; height = 180};

{lastname = "Одинцова"; firstname = "Юлия"; middlename = "Валерьевна"; birthyear = 1999; height = 165};

{lastname = "Воробьева"; firstname = "Анна"; middlename = "Александровна"; birthyear = 1998; height = 170};

{lastname = "Петров"; firstname = "Тарас"; middlename = "Борисович"; birthyear = 1996; height = 175}|]

// Функция находящих студентов 1998 года рождения

let Students1998 = Array.choose (fun s -> if s.birthyear = 1998 then Some(s) else None) Students

printfn "Студенты 1998 года рождения: \n %A \n" Students1998

// Функция находящего самого молодого студента

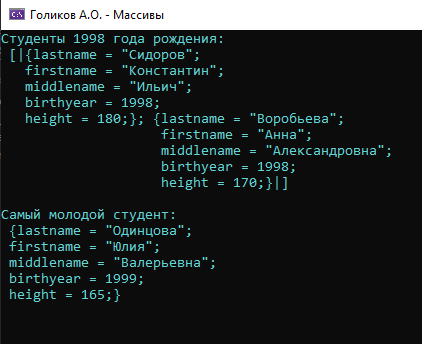
let youngest = Array.maxBy (fun s -> s.birthyear) Students

printfn "Самый молодой студент: \n %A" youngest

Console.ReadKey()

Console.Clear()

Результат работы:



**Контрольные вопросы:**

1. Перечислите способы создания массивов.

Массивы можно создавать несколькими способами. Небольшой массив можно создать, последовательно перечислив разделяемые точкой с запятой значения между символами [| и |]. Также можно указать каждый элемент в отдельной строке (в этом случае точку с запятой можно опустить). Несколько функций создают массивы, не требуя наличия существующего массива. Функция Array.empty создает новый массив, не содержащий никаких элементов. Функция Array.create создает массив указанного размера и задает для всех элементов указанные значения. Функция Array.init создает массив заданного размера, используя функцию для создания элементов. Функция Array.zeroCreate создает массив, всем элементам которого присваивается нулевое значение, соответствующее типу массива.

1. Данные каких типов могут располагаться в массивах?

Любых. Тип элементов массива определяется по указанным литералам, и он должен быть одинаковым для всех элементов.

1. Какие функции массивов называют простыми?

Функция Array.get получает элемент, функция Array.length возвращает длину массива, Функция Array.set задает для элемента указанное значение, функция Array.create создает массив указанного размера и задает для всех элементов указанные значения.

1. Перечислите способы создания новых массивов на основе старых.

Функция Array.copy создает новый массив, содержащий элементы, скопированные из существующего массива. Функция Array.sub создает новый массив из ограниченного диапазона элементов другого массива. Поддиапазон задается указанием начального индекса и длины. Функция Array.append создает новый массив, объединяя два существующих массива. Функция Array.choose выбирает элементы массива, включаемые в новый массив. Функция Array.collect применяет указанную функцию к каждому элементу существующего массива, а затем собирает элементы, созданные этой функцией, и объединяет их в новый массив. Функция Array.concat принимает последовательность массивов и объединяет их в один массив. А так же функции сортировок.

1. Каким образом можно отбирать значения в массивах?

Функция Array.choose выбирает элементы массива, включаемые в новый массив.

1. Каким образом можно поменять значения в массиве?

Функция Array.set задает для элемента указанное значение. Функция Array.fill задает указанное значение диапазону элементов массива.

1. Каким образом возможно все данные списка перекинуть в массив?

Функция Array.ofList создает новый массив из списка. Функция Array.ofSeq создает новый массив из последовательности. Функции Array.toList и Array.toSeq выполняют преобразование типа массива в типы этих других коллекций.

**Часть 2-ая. Модификация приложения F#: работа с базой данных Microsoft Access**

**Цель работы:** создать программу с использованием библиотеки WinForms, иллюстрирующую работу с базой данных.

**Задание:**

Отсортируйте все данные таблицы по фамилии работников. Добавьте поле возраст. Дополнительно отсортируйте работников по возрасту больше 30. Измените поиск – поиск по номеру, имени или фамилии (по выбору пользователя).

Код программы:

// Создание формы Windows Forms: Работа с базой данных Microsoft Access

open System

open System.Windows.Forms

open System.Drawing

open System.Data

open System.Data.OleDb

// Создаём шрифт

let ffont = new Font("Verdana", 9.75F,FontStyle.Regular, GraphicsUnit.Point)

// Создаём объект соединения с базой данных

let oleconn = new System.Data.OleDb.OleDbConnection("Provider = Microsoft.Jet.OLEDB.4.0;

Data Source = Работники.mdb")

// СоздаёмOleDbDataAdapter

let mutable dataAdpter4 = new System.Data.OleDb.OleDbDataAdapter("Select \* from [Главная таблица]", oleconn)

// Генерируем DataSet

let dataSet4 = new DataSet()

//fills the dataset with recod values

dataAdpter4.Fill(dataSet4,"[Главная таблица]") |> ignore

// Создаём форму и элементы управления

let dataForm = new Form(Text = "Различные примеры (F#) :: Работа с базой данных Microsoft Access", AutoScaleDimensions = new System.Drawing.SizeF(60.0F, 13.0F), ClientSize = new System.Drawing.Size(500, 550), StartPosition = FormStartPosition.CenterScreen)

let exitButton4 = new Button(Text = "Выход", Location = new System.Drawing.Point(300, 500))

let searchButton4 = new Button(Text = "Поиск", Location = new System.Drawing.Point(220, 500))

let searchButton5 = new Button(Text = "Sort", Location = new System.Drawing.Point(20, 500))

let searchButton6 = new Button(Text = "Sort 2", Location = new System.Drawing.Point(90, 500))

let label14 = new Label(Text = "Введите Номер или Фамилию:", Location = new System.Drawing.Point(0, 10), AutoSize = true)

let label24 = new Label(Text = "Номер:", Location = new System.Drawing.Point(0, 50), AutoSize = true)

let label34 = new Label(Text = "Имя:", Location = new System.Drawing.Point(0,100), AutoSize = true)

let label44 = new Label(Text = "Фамилия:", Location = new System.Drawing.Point(0,150), AutoSize = true)

let label54 = new Label(Text = "Возраст:", Location = new System.Drawing.Point(0,200), AutoSize = true)

let textBoxNumber4 = new TextBox(Location = new System.Drawing.Point(300,10))

let labelNumber4 = new Label(Location = new System.Drawing.Point(100,50), BorderStyle = BorderStyle.FixedSingle)

let labelFirstName4 = new Label(Location = new System.Drawing.Point(100,100), BorderStyle = BorderStyle.FixedSingle)

let labelLastName4 = new Label(Location = new System.Drawing.Point(100,150), BorderStyle = BorderStyle.FixedSingle)

let labelAge4 = new Label(Location = new System.Drawing.Point(100,200), BorderStyle = BorderStyle.FixedSingle)

// Создаём DataGridView

let dataGridView4 = new DataGridView(ColumnHeadersHeightSizeMode = DataGridViewColumnHeadersHeightSizeMode.AutoSize,Size = new System.Drawing.Size(480, 200), Location = new System.Drawing.Point(10, 250))

// Создаём столбцы DataGridView

let chrnumbercol = new DataGridViewTextBoxColumn()

let chrfnamecol = new DataGridViewTextBoxColumn()

let chrlnamecol = new DataGridViewTextBoxColumn()

// Добавляем столбцы в dataGridView4

dataGridView4.Columns.Add(chrnumbercol) |> ignore

dataGridView4.Columns.Add(chrfnamecol) |> ignore

dataGridView4.Columns.Add(chrlnamecol) |> ignore

// определяем связь

dataGridView4.DataSource <- dataSet4.Tables.["[Главная таблица]"]

// Применяем шрифт для формы

dataForm.Font <- ffont

// Применям связь данных базы с DataGridView (имя стоблца в базе / имя столбца в элементе)

chrnumbercol.DataPropertyName <- "Номер"

chrnumbercol.HeaderText <- "Номер"

chrfnamecol.DataPropertyName<-"Имя"

chrfnamecol.HeaderText<-"Имя работника"

chrlnamecol.DataPropertyName<-"Фамилия"

chrlnamecol.HeaderText<-"Фамилия работника"

// Добавляем элементы на форму

dataForm.Controls.Add(dataGridView4)

dataForm.Controls.Add(exitButton4)

dataForm.Controls.Add(searchButton4)

dataForm.Controls.Add(searchButton5)

dataForm.Controls.Add(searchButton6)

dataForm.Controls.Add(label14)

dataForm.Controls.Add(label24)

dataForm.Controls.Add(label34)

dataForm.Controls.Add(label44)

dataForm.Controls.Add(label54)

dataForm.Controls.Add(textBoxNumber4)

dataForm.Controls.Add(labelNumber4)

dataForm.Controls.Add(labelFirstName4)

dataForm.Controls.Add(labelLastName4)

dataForm.Controls.Add(labelAge4)

// Связываем Label'ы со столбцами базы данных

labelNumber4.Text <- Convert.ToString(dataSet4.Tables.["[Главная таблица]"].Rows.Item(0).Item(0))

labelFirstName4.Text <- Convert.ToString(dataSet4.Tables.["[Главная таблица]"].Rows.Item(0).Item(1))

labelLastName4.Text <- Convert.ToString(dataSet4.Tables.["[Главная таблица]"].Rows.Item(0).Item(2))

labelAge4.Text <- Convert.ToString(dataSet4.Tables.["[Главная таблица]"].Rows.Item(0).Item(3))

searchButton4.Click.Add(fun search->

// Обрабатываем номер строки индекса

let mutable introws = 0

// Определяем, была найдена запись или нет

let mutable blnfound = false

// Обрабатываем общее количество записей

let mutable inttotrec = Convert.ToInt32(dataSet4.Tables.["[Главная таблица]"].Rows.Count)

// Обрабатываем данные вводимые пользователем

let strtext = Convert.ToString(textBoxNumber4.Text)

// До тех пор, пока совпадений не найдено и конец записей не достигнут

while((blnfound = false) && (introws <= inttotrec-1)) do

let strempnum = Convert.ToString(dataSet4.Tables.["[Главная таблица]"].Rows.Item(introws).Item(0))

let strempnum1 = Convert.ToString(dataSet4.Tables.["[Главная таблица]"].Rows.Item(introws).Item(2))

// Сравниваем данные введённые в TextBox пользователем с нашей таблицей ("Номер")

// Если есть совпадений, отображаем результат запроса

if strtext.ToUpper() = strempnum.ToUpper() || strtext.ToUpper() = strempnum1.ToUpper() then

blnfound<-true

labelNumber4.Text <- Convert.ToString(dataSet4.Tables.["[Главная таблица]"].Rows.Item(introws).Item(0))

labelFirstName4.Text <- Convert.ToString(dataSet4.Tables.["[Главная таблица]"].Rows.Item(introws).Item(1))

labelLastName4.Text <- Convert.ToString(dataSet4.Tables.["[Главная таблица]"].Rows.Item(introws).Item(2))

labelAge4.Text <- Convert.ToString(dataSet4.Tables.["[Главная таблица]"].Rows.Item(introws).Item(3))

// Сравниваем со следующей запись до появления совпадений

introws<-introws + 1

// Если совпадения не найдены

if blnfound = false then

MessageBox.Show("Запись не найдена!", "Работа с базой данных Microsoft Access :: Сообщение об ошибке", MessageBoxButtons.OK, MessageBoxIcon.Information) |> ignore)

exitButton4.Click.Add(fun exit->

dataForm.Close()

oleconn.Close())

// Функции сортировки по фамилии и тех, кто старше 30

let clean1 (n : DataGridView) = while (n.Rows.Count > 1) do n.Rows.RemoveAt(0)

searchButton5.Click.Add (fun sort ->

clean1 dataGridView4

dataAdpter4 <- new System.Data.OleDb.OleDbDataAdapter("Select \* from [Главная таблица] order by [Фамилия],[Возраст]", oleconn)

dataAdpter4.Fill(dataSet4,"[Главная таблица]") |> ignore)

searchButton6.Click.Add (fun sort ->

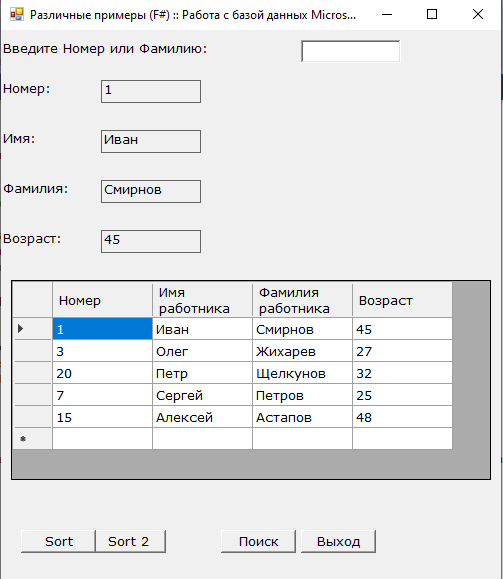
clean1 dataGridView4

dataAdpter4 <- new System.Data.OleDb.OleDbDataAdapter("Select \* from [Главная таблица] where [Возраст] > 30 order by [Фамилия]", oleconn)

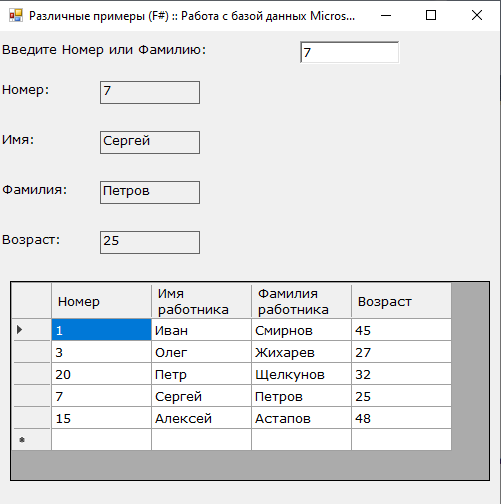
dataAdpter4.Fill(dataSet4,"[Главная таблица]") |> ignore)

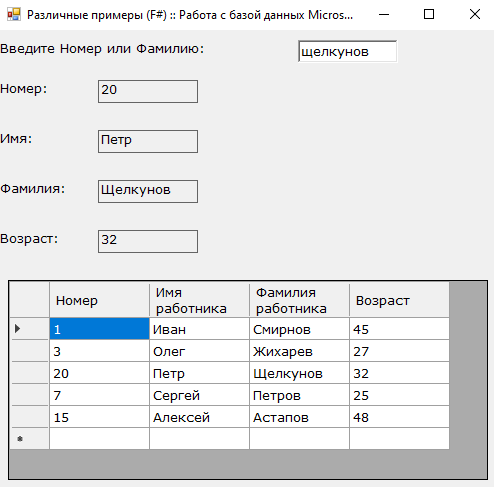
Application.Run(dataForm)

Результат выполнения:

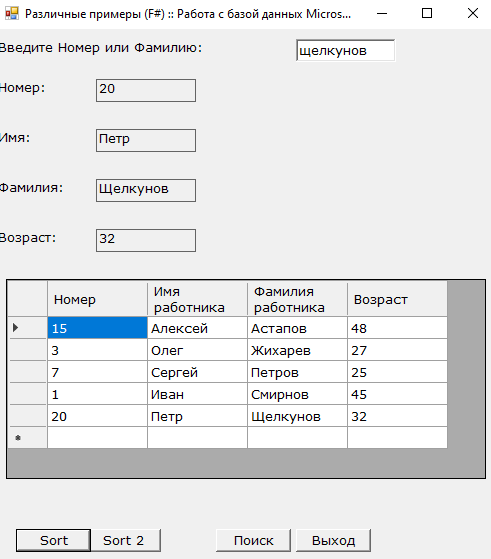


Поиск по номеру и фамилии:

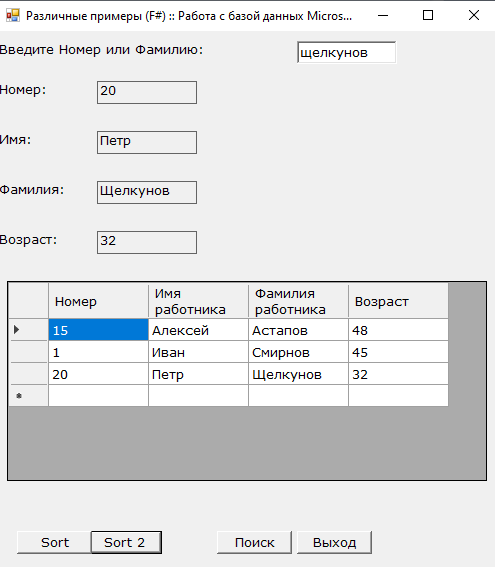




Сортировка по фамилии:



Сортировка по фамилии того, кто старше 30:



**Вывод:** изучил основные принципы работы с массивами в F#, разработки приложений на F#; создал программу с использованием библиотеки WinForms, иллюстрирующую работу с базой данных.