Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет   
информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра информатики

Дисциплина: Методы трансляции

ОТЧЁТ

к лабораторной работе №1

на тему

**ОПРЕДЕЛЕНИЕ МОДЕЛИ ЯЗЫКА**

Выполнил: студент гр.253505 Снежко М.А.

Проверил: ассистент кафедры информатики Гриценко Н.Ю.

Минск 2025

**СОДЕРЖАНИЕ**

[1 Цель работы 3](#_30j0zll)

[2 Теоретические сведения 4](#_1fob9te)

3 Инструментальная языковая среда [6](#_2et92p0)

Заключение 7

[Приложение А](#_1t3h5sf) (обязательное) Листинг программного кода  [9](#_1t3h5sf)

# 1 ЦЕЛЬ РАБОТЫ

Целью данной лабораторной работы является определение подмножества языка программирования, которое включает ключевые элементы, необходимые для разработки базовых программ. В подмножество должны быть включены числовые и текстовые константы, все типы переменных, операторы цикла, структуры данных, функции и условные операторы. Это подмножество должно быть достаточным для создания программ, демонстрирующих работу с основными конструкциями языка, и позволит изучить их взаимодействие в рамках конкретных задач.

Для выполнения работы необходимо также определить инструментальную языковую среду, включающую язык программирования, операционную систему и тип компьютера. Это позволит обеспечить корректную работу разработанных программ и продемонстрировать их функциональность. Выбор среды разработки должен учитывать поддержку всех необходимых элементов языка, таких как типы данных, операторы и структуры, а также удобство использования для реализации поставленных задач. Удобство разработки может включать наличие отладчиков, встроенных средств анализа кода и документации.

Таким образом, в рамках данной работы требуется разработать три программы, каждая из которых будет включать все элементы выбранного подмножества языка. Эти программы должны демонстрировать использование числовых и текстовых констант, работу с переменными различных типов, применение операторов цикла и условных операторов, а также использование структур данных и функций. Программы должны быть реализованы в выбранной инструментальной среде и корректно работать на указанной операционной системе. Дополнительно, необходимо провести вручную тестирование каждой программы, чтобы убедиться в их корректности и эффективности. Тестирование должно включать проверку всех возможных сценариев использования, а также оценку устойчивости программ к ошибкам. Важно также задокументировать результаты тестирования, чтобы иметь возможность в будущем анализировать найденные ошибки и улучшать код. Наконец, для успешного завершения лабораторной работы необходимо подготовить отчет, который будет содержать описание процесса разработки, выбор инструментальной среды, описание каждой программы и результаты тестирования.

# 2 ПОДМНОЖЕСТВО ЯЗЫКА ПРОГРАММИРОВАНИЯ

Язык программирования *F#* является мощным инструментом для разработки, который сочетает функциональный, императивный и объектно-ориентированный подходы. *F#* отличается лаконичностью синтаксиса, поддержкой сопоставления с образцом, неизменяемыми данными и функциями высшего порядка. Эти особенности делают его идеальным выбором для создания высоконадежного и производительного программного обеспечения. *F#* позволяет писать неуклюжий самодокументируемый код, где фокус остается на вашем домене проблемы, а не сведения о программировании. Делается без ущерба для скорости и совместимости [1].

В таблице 2.1 представлено подмножество языка программирования *F#*, включающее ключевые типы данных, операторы циклов, структуры данных, функции и условные операторы. Эти элементы являются основой для понимания и использования *F#* в самых различных областях.

Таблица 2.1 – Подмножество языка *F#*

|  |  |  |
| --- | --- | --- |
| № | Категория | Пример |
| 1 | Типы переменных | *let x* = 10 (числовой)  *let y* = *3L* (целое)  *let z* = 1.0 + 2.0 + 1j (комплексное)  *let name* = "*Alice*"(строковый)  *let flag* = *true* (логический)  *let date* = *System.DateTime.Now* (дата) |
| 2 | Операторы циклов | *for i in* 1 .. 10 *do { }*  *while condition do { }*  *for .i* = 1 *to* 5 *{ }* |
| 3 | Структуры данных | [1; 2; ...] (список)  *Array.init* 10 (*fun i* -> *i*) (массив)  *Set.ofList* [1; 2; 3] (множество)  *tuple* = (1, "*text*") (кортеж) |
| 4 | Классы и структуры | *type Person*(*name*: *string, age*: *int*) = *member this.Name* = *name* (класс)  *type Point* = { *X*: *int*; *Y*: *int* } (структура) |
| 5 | Функции | *let* *myFunction* *x* = *x* + 1 (обычная функция)  *let rec factorial n* = *if n* <= 1 *then* 1 *else n* \* *factorial* (*n* - 1) (рекурсивная функция) |
| 6 | Условные операторы | *if condition then* { } *else* { } |

Как видно из таблицы 2.1 данное подмножество охватывает основные элементы языка *F#*, необходимые для выполнения широкого спектра задач.

# 3 ИНСТРУМЕНТАЛЬНАЯ ЯЗЫКОВАЯ СРЕДА

Инструментальная языковая среда, используемая для разработки, основана на языке программирования *Swift*. *Swift* представляет собой современный язык, который сочетает в себе высокую производительность и простоту написания кода.

*Swift* – это результат самых новейших исследований в области языков программирования в сочетании с десятилетиями опыта создания программного обеспечения, работающего на миллиардах устройств. Именованные параметры выражены в чистом синтаксисе, что делает *API* в *Swift* простыми для чтения и поддержки. Более того, вам даже не нужно вводить точки с запятой. Выведенные типы делают код чище и менее подверженным ошибкам, в то время как модули устраняют заголовки и предоставляют пространства имен [2].

Разработка ведётся в операционной системе *Windows* 10, которая предоставляет стабильную и надёжную платформу для программирования. Несмотря на то, что *Swift* изначально разрабатывался для экосистемы Apple, его использование на *Windows* стало возможным благодаря портированию компилятора *Swift* и других инструментов. *Windows* 10 обеспечивает поддержку популярных редакторов кода, таких как *Visual Studio Code*, и систем управления версиями, что делает процесс разработки комфортным и продуктивным. Кроме того, система предоставляет доступ к богатому набору инструментов для тестирования и отладки, что особенно важно при создании качественного программного обеспечения.

Основным рабочим устройством является ноутбук *MSI* *Pulse*. Этот ноутбук оснащён современным процессором и мощной графической картой, что обеспечивает высокую производительность и позволяет работать с ресурсоёмкими задачами, такими как компиляция кода, тестирование приложений и работа с большими проектами. Большой объём оперативной памяти и быстрый накопитель обеспечивают комфортную работу даже при многозадачности. Надёжность и мобильность ноутбука делают его отличным выбором для выполнения сложных задач и продуктивной работы над проектами. Надежная сборка и продуманная система охлаждения обеспечивают стабильную работу даже при выполнении сложных задач.

Таким образом, выбор инструментальной языковой среды для разработки на языке *Swift*, в сочетании с использованием операционной системы *Windows* 10 и ноутбука *MSI* *Pulse*, создает оптимальные условия для эффективного программирования. *Swift*, как современный язык с высокой производительностью и удобством в использовании, предоставляет разработчикам все необходимые инструменты для реализации разнообразных проектов. *Windows* 10, благодаря своей стабильности и поддержке популярных инструментов, дополняет этот процесс, обеспечивая комфортную и продуктивную среду для работы.

# ЗАКЛЮЧЕНИЕ

В ходе выполнения лабораторной работы было проведено определение подмножества языка программирования F#, включая все основные элементы, необходимые для разработки программ. F# – это функциональный язык, который предлагает разработчикам мощные инструменты для построения приложений. В рамках работы были изучены ключевые аспекты языка, такие как типы переменных, операторы циклов, структуры данных, функции и условные операторы. Это подмножество охватывает все стандартные возможности F#, что позволяет использовать его для решения широкого круга задач, начиная от простых арифметических вычислений и заканчивая сложной обработкой данных и созданием масштабируемых приложений.

Особое внимание было уделено выбору инструментальной среды. Выбранная среда разработки *Visual Studio* обеспечивает стабильную работу всех компонентов языка F# и предоставляет удобные инструменты для написания, тестирования и отладки программ. Данное *IDE* поддерживают функциональные возможности языка, такие как автозаполнение кода, встроенные средства отладки и управление версиями, что значительно упрощает процесс разработки и улучшает производительность.

Разработанные в рамках работы программы демонстрируют применение различных элементов подмножества языка *F#*. Они включают работу с числовыми и текстовыми константами, использование различных типов переменных, применение операторов циклов и условных операторов, а также манипуляции со структурами данных и функциями. Примеры программ иллюстрируют, как данные конструкции могут быть использованы для выполнения задач, таких как обработка списков, работа с коллекциями и реализация алгоритмов.

В заключение, выполнение данной лабораторной работы позволило не только изучить основные элементы языка *F#*, но и получить практический опыт их применения. Это подмножество языка, а также выбранная инструментальная среда, могут быть успешно использованы для решения задач в области анализа данных, разработки программного обеспечения, автоматизации бизнес-процессов и других направлений. Полученные знания и навыки станут прочной основой для дальнейшего изучения языка *F#* и его применения в профессиональной деятельности, открывая новые горизонты в мире программирования и разработки высококачественных приложений.

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

[1] Microsoft "Что такое F#" – Электронный ресурс. Режим доступа: https://learn.microsoft.com/ru-ru/dotnet/fsharp/what-is-fsharp.  
 [2] Developer Apple "Swift" – Электронный ресурс. Режим доступа: https://developer.apple.com/swift.

# ПРИЛОЖЕНИЕ А

# (обязательное)

# Листинг программного кода

Листинг А.1 – Первая программа

open System

// Пример работы с различными типами переменных

let x = 10

let y = 3L

let z = 1.0 + 2.0 \* 1j

let name = "Alice"

let flag = true

let date = DateTime.Now

// Обычная функция

let square x = x \* x

// Рекурсивная функция для вычисления факториала

let rec factorial n =

if n <= 1 then 1

else n \* factorial (n - 1)

// Использование цикла for

let printSquares n =

for i in 1 .. n do

printfn "Square of %d: %d" i (square i)

// Основная функция

[<EntryPoint>]

let main argv =

printfn "Hello, %s! Today's date is %A." name date

printfn "Factorial of 5: %d" (factorial 5)

printfn "Squares from 1 to 10:"

printSquares 10

0

Листинг А.2 – Вторая программа

open System

// Определение структуры

type Point = { X: int; Y: int }

// Определение класса

type Person(name: string, age: int) =

member this.Name = name

member this.Age = age

member this.Greet() = printfn "Hello, my name is %s, and I am %d years old." this.Name this.Age

// Работа с массивами и кортежами

let points = [ { X = 1; Y = 2 }; { X = 3; Y = 4 }; { X = 5; Y = 6 } ]

let sumCoordinates (x, y) = x + y

[<EntryPoint>]

let main argv =

let person = Person("Bob", 25)

person.Greet()

printfn "Coordinates of points:"

points |> List.iter (fun p -> printfn "Point: (%d, %d)" p.X p.Y)

let tupleExample = (10, 20)

printfn "Sum of tuple elements: %d" (sumCoordinates tupleExample)

let arrayExample = Array.init 5 (fun i -> i \* i)

printfn "Squares in array: %A" arrayExample

0

Листинг А.3 – Третья программа

open System

// Определение функции для фильтрации множества

let isEven x = x % 2 = 0

// Работа с множествами

let numbers = Set.ofList [1 .. 20]

let evenNumbers = Set.filter isEven numbers

// Использование условных операторов

let classifyNumber x =

if x % 2 = 0 then "Even"

else "Odd"

// Функция высшего порядка

let applyToAll f set =

set |> Set.map f

[<EntryPoint>]

let main argv =

printfn "Original set: %A" numbers

printfn "Filtered even numbers: %A" evenNumbers

printfn "Classifying numbers from 1 to 10:"

[1 .. 10] |> List.iter (fun n -> printfn "%d is %s" n (classifyNumber n))

let doubledNumbers = applyToAll (fun x -> x \* 2) numbers

printfn "Doubled set: %A" doubledNumbers

0