Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра информатики

Дисциплина: Метод­­­­ы трансляции

ОТЧЕТ

к лабораторной работе №1

на тему

**ОПРЕДЕЛЕНИЕ МОДЕЛИ ЯЗЫКА. ВЫБОР ИНСТРУМЕНТАЛЬНОЙ ЯЗЫКОВОЙ СРЕДЫ.**

Студент П.С. Глытов

Преподаватель Н. Ю. Гриценко

Минск 2024

**СОДЕРЖАНИЕ**

[1 Постановка задачи 3](#_Toc158056567)

[2 Определение подмножества языка программирования Kotlin 4](#_Toc158056568)

[3 Определение инструментальной языковой среды 10](#_Toc158056569)

[Выводы 11](#_Toc158056570)

[Список использованных источников 12](#_Toc158056571)

[Приложение А (обязательное) Пример реализации программ на языке   
                           программирования Kotlin 13](#_Toc158056572)

# 1 ПОСТАНОВКА ЗАДАЧИ

Целью выполнения данной лабораторной работы является определить подмножество языка программирования (типы констант, переменных, операторов и функций), определение языка программирования и операционной системы для разработки, а также тексты нескольких программ.

# 2 ОПРЕДЕЛЕНИЕ ПОДМНОЖЕСТВА ЯЗЫКА ПРОГРАММИРОВАНИЯ KOTLIN

Kotlin – статически типизированный язык программирования, созданный для JVM. Он призван увеличить производительность разработчика и улучшить читаемость кода. Синтаксис минималистичен, но функционален. Стандартная библиотека обладает обширным набором функций, упрощающих разработку приложений. Kotlin подходит для создания приложений разного типа – от мобильных до веб-серверов и настольных приложений.

Целочисленные константы в Kotlin могут быть представлены значениями, такими как -3, 0 и 5. Они используются для хранения целых чисел без десятичной части и могут быть как положительными, так и отрицательными [1].

Числа с плавающей точкой, такие как 3.14 и -2.5, являются числовыми константами в формате с плавающей точкой. В Kotlin они могут быть типа float или double и представляют значения с десятичной частью.

Строковые константы, такие как "Привет, мир!" или "", представляют последовательности символов, заключенные в кавычки. Они могут содержать любые символы, включая буквы, цифры, знаки препинания и пробелы.

Unicode-литералы в Kotlin позволяют представлять символы Unicode в строках с помощью специального синтаксиса `\uXXXX`, где `XXXX` - это код Unicode символа в шестнадцатеричном формате. Например, "\u03B1" представляет символ α.

В Kotlin существует разнообразие типов переменных для различных нужд. Синтаксис инициализации переменной показан на рисунке 2.1.



Рисунок 2.1­­ – Инициализация переменной.

Тип Int предназначен для хранения целочисленных значений без дробной части, таких как 1, -5, 100.

Double используется для хранения чисел с плавающей точкой двойной точности, что позволяет представлять как целые, так и дробные числа с высокой точностью, например, 3.14, -0.5, 10.0.

String представляет строки символов и используется для хранения текстовой информации, такой как "Привет, мир!", "abc123", "Это строка".

Boolean – это логический тип данных, который может принимать только два значения: true или false. Он используется для хранения булевых (логических) значений.

Кроме этих основных типов, в Kotlin также имеются типы переменных, такие как Float (для чисел с плавающей точкой одинарной точности), Long (для больших целых чисел), Byte (для очень маленьких целых чисел) и Short (для небольших целых чисел). Эти типы предоставляют дополнительные возможности для работы с данными в зависимости от требований конкретной задачи. Пример с данными типами изображен на рисунке 2.2.

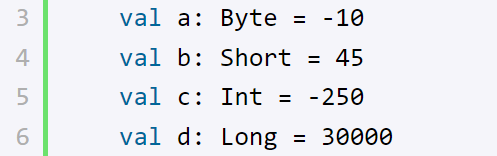


Рисунок 2.2 – Использование различных типов.

Кotlin предоставляет разнообразные структуры данных, обеспечивающие эффективное управление и организацию информации в приложениях.

Массив – структура данных, представляющая собой упорядоченную коллекцию элементов фиксированного размера. Он обеспечивает быстрый доступ к элементам по индексу и удобен для хранения последовательных данных одного типа.

Список – структура данных, представляющая собой упорядоченную коллекцию элементов, которые могут повторяться. Элементы списка доступны по индексу, что обеспечивает быстрый доступ и возможность изменения элементов.

Множество – структура данных, которая содержит уникальные элементы без определенного порядка. В Kotlin множества могут использоваться для удаления дубликатов из коллекции или проверки наличия элементов.

Отображение – структура данных, представляющая собой коллекцию пар ключ-значение, где каждый ключ уникален. Это обеспечивает эффективный доступ к значениям по ключу и позволяет реализовывать словари и таблицы соответствия.

Стек – структура данных, работающая по принципу "последний вошел, первый вышел" (LIFO – Last In, First Out). Это означает, что последний добавленный элемент будет первым, который извлечен из стека.

Очередь – структура данных, которая работает по принципу "первый вошел, первый вышел" (FIFO – First In, First Out). Элементы добавляются в конец очереди, а извлекаются из начала.

Каждая из этих структур данных предоставляет определенный способ организации и доступа к данным, что делает их эффективными инструментами для работы с информацией в Kotlin.

В Kotlin доступны различные операторы цикла для повторения выполнения определенных блоков кода в зависимости от условий.

Оператор for используется для итерации по коллекциям, диапазонам или любым объектам, поддерживающим итерацию. Он обычно используется, когда заранее известно количество итераций. Пример использования данного цикла изображен на рисунке 2.3.

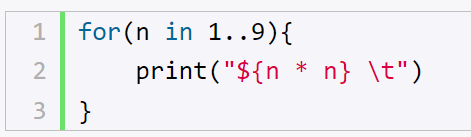


Рисунок 2.3 – Использование цикла for.

Оператор while выполняет блок кода, пока указанное условие истинно. Он применяется в ситуациях, когда количество итераций неизвестно заранее.

Пример использования представлен на рисунке 2.4.

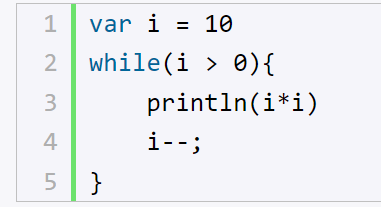


Рисунок 2.4 – Использование цикла while.

Оператор do..while также выполняет блок кода, но он проверяет условие после каждой итерации. Это гарантирует, что блок кода выполнится хотя бы один раз, даже если условие изначально ложно. Пример использования изображен на рисунке 2.5.

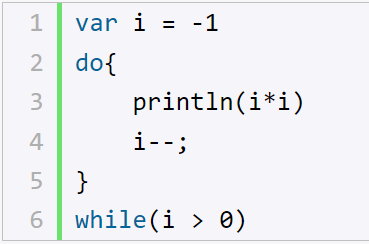


Рисунок 2.5 – Использование цикла do while.

Оператор repeat выполняет блок кода определенное количество раз. В отличие от оператора for, который итерирует по коллекциям или диапазонам, repeat просто повторяет выполнение блока заданное количество раз.

Эти операторы цикла позволяют эффективно управлять повторяющимися задачами в Kotlin, предоставляя различные методы для обхода итераций в зависимости от конкретных потребностей программы.

Функции в Kotlin - это блок кода, который выполняет определенную задачу. Они объявляются с помощью ключевого слова "fun" и могут принимать аргументы и возвращать значения.

Функции в Kotlin могут быть простыми, принимая аргументы и возвращая значения, или они могут быть более сложными, включая управление потоком выполнения с помощью условий и циклов.

Параметры функций определяются в скобках после имени функции и могут иметь значения по умолчанию, что делает функции более гибкими в использовании. Пример функции приведен на рисунке 2.6.

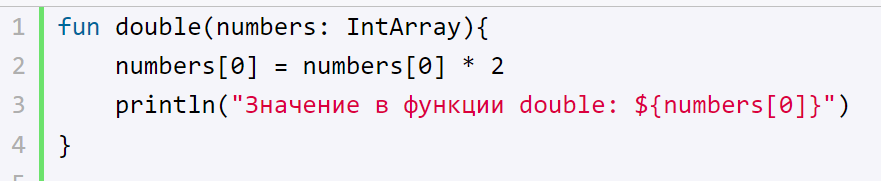


Рисунок 2.6 – Пример функции.

В Kotlin также поддерживаются анонимные функции, которые могут быть переданы в качестве аргументов или сохранены в переменных.

Функции могут быть объявлены как часть класса или объекта, что позволяет создавать методы, специфичные для этого класса или объекта.

Расширения функций позволяют добавлять новое поведение к существующим классам без изменения их исходного кода, что делает код более модульным и легко читаемым.

Условные операторы в Kotlin позволяют выполнять различные действия в зависимости от выполнения определенных условий.

Оператор if..else используется для выполнения блока кода, если условие истинно, и другого блока кода, если условие ложно. Он может быть использован как с одним условием, так и с несколькими условиями при помощи операторов "или" и "и". Пример использования конструкции if..else показан на рисунке 2.7.

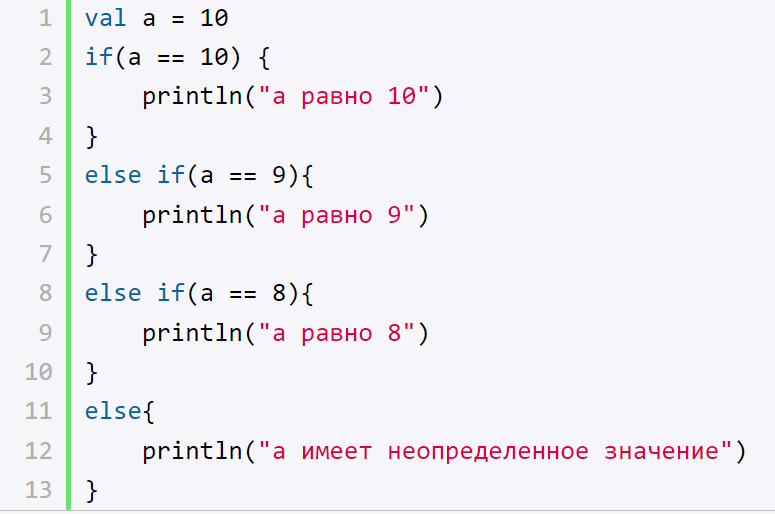


Рисунок 2.7 – Пример if..else.

Оператор when (аналог switch/case в других языках программирования) предоставляет более удобный способ выполнения блоков кода в зависимости от значений переменной. Он позволяет сопоставлять переменную с несколькими значениями и выполнять соответствующий блок кода для каждого случая. Пример использования when изображен на рисунке 2.8.

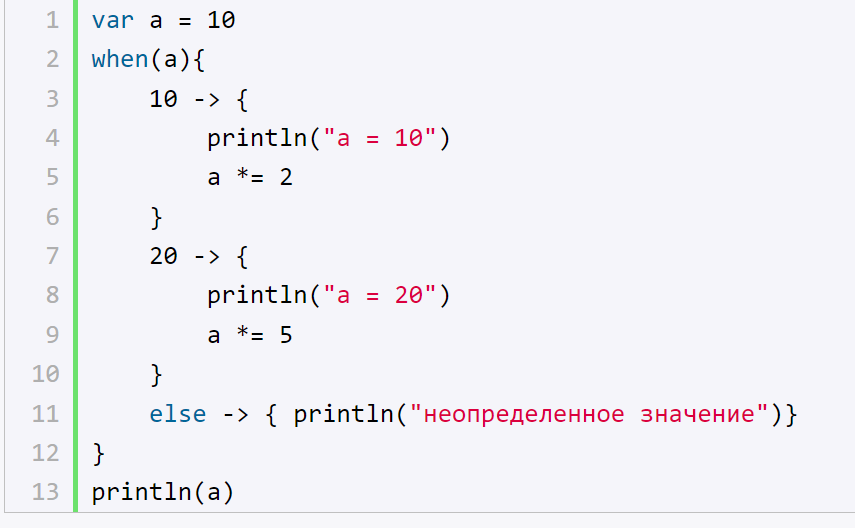


Рисунок 2.8 – Пример when.

Эти условные операторы обеспечивают гибкость в управлении потоком выполнения программы в зависимости от различных условий, что позволяет разработчикам создавать более сложные и мощные приложения.

# 3 ОПРЕДЕЛЕНИЕ ИНСТРУМЕНТАЛЬНОЙ ЯЗЫКОВОЙ СРЕДЫ

Python 3.10 – это интерпретируемый, высокоуровневый язык программирования, который предназначен для разработки разнообразных приложений в различных сферах. Он обладает простым и понятным синтаксисом, что делает его идеальным выбором как для начинающих, так и для опытных разработчиков. Python предоставляет широкий набор инструментов и библиотек, позволяющих решать широкий спектр задач, включая веб-разработку, научные вычисления, обработку данных, искусственный интеллект, анализ данных и многое другое.

Одной из основных особенностей Python является его гибкость и расширяемость. В Python легко интегрировать библиотеки, написанные на других языках программирования, что позволяет разработчикам использовать уже существующие решения для своих проектов. Кроме того, Python имеет активное сообщество разработчиков, которые создают и поддерживают множество библиотек и фреймворков, расширяя его функциональные возможности.

Python также известен своей мультипарадигмальной природой, поддерживая объектно-ориентированное, процедурное, функциональное и аспектно-ориентированное программирование. Это делает Python гибким и универсальным инструментом для решения различных задач.

Благодаря своей простоте, мощности и гибкости Python остается одним из самых популярных языков программирования в мире. Он используется во многих областях, включая веб-разработку, научные исследования, анализ данных, разработку игр, автоматизацию задач и многое другое.

В качестве интегрированной среды разработки был выбран PyCharm Professional Edition 2023 от JetBrains.

Операционная система инструментальной языковой среды является Windows.

# ВЫВОДЫ

В ходе выполнения данной лабораторной было определено подмножество языка программирования (типы констант, переменных, операторов и функций), определены язык программирования и операционная система для разработки, а также реализованы тексты нескольких программ.

## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Переменные [Электронный ресурс]. – Режим доступа: https://metanit.com/kotlin/tutorial/2.1.php. – Дата доступа: 07.02.24
2. Циклы [Электронный ресурс]. – Режим доступа: https://metanit.com/kotlin/tutorial/2.7.php. – Дата доступа: 07.02.24
3. Условные конструкции [Электронный ресурс]. – Режим доступа: https://metanit.com/kotlin/tutorial/2.6.php. – Дата доступа: 07.02.24

# ПРИЛОЖЕНИЕ А

# (обязательное)

# Листинг кода

Листинг 1 – Файл ex1.kt

fun main() {

var continueCalculation = true

while (continueCalculation) {

println("Введите первое число:")

val number1 = readLine()?.toDoubleOrNull()

println("Введите второе число:")

val number2 = readLine()?.toDoubleOrNull()

println("Выберите операцию (+, -, \*, /):")

val operator = readLine()

if (number1 == null || number2 == null || operator == null) {

println("Ошибка: Неверный формат ввода чисел или операции")

continue

}

val result = when (operator) {

"+" -> number1 + number2

"-" -> number1 - number2

"\*" -> number1 \* number2

"/" -> if (number2 != 0.0) number1 / number2 else "Ошибка: деление на ноль"

else -> "Ошибка: неверная операция"

}

println("Результат: $result")

println("Хотите продолжить? (да/нет)")

val choice = readLine()

continueCalculation = choice.equals("да", ignoreCase = true)

}

println("Спасибо за использование калькулятора!")

}

Листинг 2 – Файл ex2.kt

fun main() {

val array = intArrayOf(1, 2, 3, 4, 5)

val shiftAmount = 2

println("Исходный массив:")

printArray(array)

cyclicShiftRight(array, shiftAmount)

println("\nМассив после циклического сдвига вправо на $shiftAmount элемента(ов):")

printArray(array)

}

fun cyclicShiftRight(arr: IntArray, k: Int) {

val n = arr.size

val temp = IntArray(n)

for (i in 0 until n) {

temp[(i + k) % n] = arr[i]

}

for (i in 0 until n) {

arr[i] = temp[i]

}

}

fun printArray(arr: IntArray) {

for (element in arr) {

print("$element ")

}

}