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. ВЫБОР ИНСТРУМЕНТАЛЬНОЙ ЯЗЫКОВОЙ СРЕДЫ

Студент Н. С. Сенько

Проверяющий Н. Ю. Гриценко

Минск 2025

СОДЕРЖАНИЕ

1 ЦЕЛЬ РАБОТЫ

Задача данной лабораторной работы заключается в выявлении ключевых компонентов языка программирования, необходимых для создания простых программ. В этот набор должны входить числовые и строковые константы, все типы переменных, циклы, условные операторы, функции и структуры данных. Такой набор должен быть достаточным для разработки программ, которые иллюстрируют основные конструкции языка и позволяют изучить их взаимодействие в контексте конкретных задач.

Кроме того, для успешного выполнения работы необходимо выбрать подходящую инструментальную среду, которая включает язык программирования, операционную систему и тип компьютера. Это обеспечит правильную работу созданных программ и продемонстрирует их функциональные возможности. При выборе среды разработки важно учитывать поддержку всех необходимых элементов языка, таких как типы данных и операторы, а также удобство для выполнения поставленных задач.

В рамках работы планируется разработать три программы, каждая из которых будет включать все элементы выбранного набора языка. Эти программы должны продемонстрировать использование числовых и строковых констант, работу с переменными различных типов, применение циклов и условных операторов, а также использование структур данных и функций. Все программы должны быть реализованы в выбранной инструментальной среде и корректно функционировать на указанной операционной системе.

2 ПОДМНОЖЕСТВО ЯЗЫКА ПРОГРАММИРОВАНИЯ

Kotlin – это современный статически типизированный язык программирования, который работает на платформе *Java Virtual Machine (JVM)*, а также может быть скомпилирован в *JavaScript* и используется в разработке нативных приложений для различных платформ. Он разработан с целью объединить простоту и выразительность синтаксиса с мощной поддержкой объектно-ориентированного и функционального программирования. *Kotlin* предлагает ряд современных возможностей, таких как *null*-безопасность, *lambdas* (функции высших порядков), расширения функций и свойства, что позволяет писать более безопасный и лаконичный код. Эти особенности делают *Kotlin* идеальным выбором для разработки сложных приложений, где важна чёткость и понятность кода, а также возможность избежать распространённых ошибок, связанных с нулевыми ссылками.

В таблице 2.1 представлено подмножество языка программирования *Kotlin*, включающее основные операторы, функции и структуры данных, которые являются фундаментальными для понимания и использования *Kotlin*.

Таблица 2.1 – Подмножество языка *Kotlin*

№	Категория	Пример
1	Типы переменных	Boolean (true, false) Byte Short Int Long Float Double Char

2	Операторы циклов	for (i in (0..n) {}
		for (j in (0 until m) {}
		for (k in n downTo 0) {}
		while(i < n) { i++; }

3	Структуры данных	// Массив Array<T>
4	Функции	// Функция с типизацией fun fin1(a: Int, b: Float) : Float {}

5	Условные операторы	if (a < b) {
		  return 1;
		} else {
		  return 2;

Продолжение таблицы 2.1

		}
		
		when(i) {
		1 -> {
		println(“one”)
		}
		2 -> {
		println(“two”)
		}
		else -> {
		println(“A lot of”)
		}
		}

В заключение, *Kotlin* представляет собой мощный и современный язык программирования, который сочетает в себе простоту и выразительность, а также предлагает множество возможностей для повышения безопасности и эффективности кода. Его поддержка различных платформ и интеграция с *Java* делают его универсальным инструментом для разработчиков. Благодаря таким особенностям, как *null*-безопасность и функциональные возможности, *Kotlin* становится предпочтительным выбором для создания сложных приложений, где важны как качество, так и читаемость кода.

3 ИНСТРУМЕНТАЛЬНАЯ ЯЗЫКОВАЯ СРЕДА

Для успешного выполнения транслирования программ, написанных на *Fortran*, важен выбор инструментальной языковой среды, которая включает в себя язык программирования, операционную систему и используемые инструменты [1].

3.1 Исполняющее устройство

В качестве базовой платформы используется компьютер на архитектуре *x86* с процессором *Intel Core i5*. Эти процессоры обеспечивают высокую производительность в вычислительных задачах, что критически важно для работы с языком Fortran, ориентированным на научные расчёты и численное моделирование. *Архитектура x86* предоставляет широкую совместимость с компиляторами и библиотеками *Fortran*, а также поддерживает оптимизацию для параллельных вычислений. Надёжность и масштабируемость *x86*-систем делают их идеальным выбором для ресурсоёмких задач, таких как трансляция и исполнение программ на *Fortran*.

3.2 Операционная система

В качестве операционной системы выбрана *Arch Linux* — легковесный и гибкий дистрибутив *Linux* с *rolling-release* моделью обновлений. *Arch Linux* предоставляет полный контроль над системой, доступ к актуальным пакетам через менеджер pacman, а также возможность тонкой настройки окружения под конкретные задачи. Для разработки используется редактор *Visual Studio Code* с расширением *Modern Fortran*, которое обеспечивает подсветку синтаксиса, автодополнение, интеграцию с отладчиком (например, *gdb*) и компиляторами. Дополнительно устанавливаются инструменты для работы с системами сборки (*CMake, Make*) и профилирования (*gprof*), что упрощает разработку и оптимизацию кода на *Fortran*.

3.3 Язык программирования

Основным языком программирования выбран *Fortran*. Этот язык исторически доминирует в высокопроизводительных вычислениях (*HPC*), физическом моделировании и инженерных расчётах благодаря своей эффективности при работе с массивами данных и поддержке параллельных вычислений (через *OpenMP*, *MPI*). Для компиляции используется *gfortran* — свободный компилятор из набора *GCC*, совместимый с современными стандартами *Fortran*.

Выбор связки *x86, Arch Linux* и *Fortran* обеспечивает баланс между производительностью, гибкостью настройки и доступностью инструментов для разработки сложных вычислительных систем.

ЗАКЛЮЧЕНИЕ

Данная лабораторная работа позволила определить ключевые элементы языка *Fortran*, необходимые для разработки базовых программ, ориентированных на вычислительные задачи. Использование архитектуры *x86* в сочетании с операционной системой *Arch Linux* создало оптимальную среду для реализации проекта, обеспечив доступ к современным компиляторам (таким как *gfortran*), инструментам анализа кода и профилирования.

Выполнение работы подтвердило теоретические аспекты языка, включая строгую типизацию, управление памятью и оптимизацию для высокопроизводительных расчётов. Практическая реализация задач на *Arch Linux* позволила освоить инструменты разработки (*Visual Studio Code* с поддержкой *Modern Fortran*, системы сборки *CMake/Make*), а также оценить преимущества связки *x86* и *Fortran* для ресурсоёмких вычислений.

Таким образом, работа стала важным этапом в освоении *Fortran*, продемонстрировав его потенциал в решении инженерных и научных задач, а также гибкость экосистемы *Arch Linux* для настройки специализированных рабочих окружений.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] Fortran Высокопроизводительный язык [Электронный ресурс]. – Режим доступа: https://fortran-lang.org/ru – Дата доступа: 27.01.2025.

ПРИЛОЖЕНИЕ А
(ОБЯЗАТЕЛЬНОЕ)
ИСХОДНЫЙ КОД ПРОГРАММНОГО ПРОДУКТА

```
// Программа 1

fun checkEven(n : Int) {
    if(n % 2 == 0) {
        print(n);
        println(" - even");
    } else {
        print(n);
        println(" – odd");

    }
}


fun main() {
    val numbers = Array<Int>(5) { 0 };

    numbers[0] = 4;
    numbers[1] = 5;
    numbers[2] = 6;
    numbers[3] = 7;
    numbers[4] = 8;

    for(i in 0 until 5) {
        checkEven(numbers[i]);
    }

    val n = 3;
    var ans = 1;

    for(i in n downTo 1) {
        ans *= i;
    }

    println(ans)
}

// Программа 2

fun main() {
    var n = 6;

    var someBool : Boolean = n % 3 == 0;

    if(someBool == true) {
        println("n divide by 3");
    } else {
        println("n divide by 3");
    }

}

// Программа 3

fun main() {
    var i : Int = 10;

    while (i > 3) {
```

```
        i -= 3;
    }

    when (i) {
        1 -> {
            println("one");
        }

        2 -> {
            println("two");
        }

        else -> {
            println("A lot of");
        }
    }
}
```