Федеральное государственное бюджетное
образовательное учреждение высшего образования
«Кемеровский государственный университет»
Кузбасский гуманитарно-педагогический институт

Факультет информатики, математики и экономики
Кафедра математики, физики и математического моделирования

Ю.С. Гаврилова

ЯЗЫКИ И МЕТОДЫ ПРОГРАММИРОВАНИЯ

(часть 1)

Методические рекомендации по выполнению аудиторной и внеаудиторной самостоятельной работы
для обучающихся по направлениям подготовки
01.03.02 Прикладная математика и информатика, профиль «Математическое моделирование и информационные технологии»
02.03.03 Математическое обеспечение и администрирование информационных систем, профиль «Программное и математическое обеспечение информационных технологий»

Новокузнецк

2021

УДК [378.147.88:519.216](072)
ББК 74.484(2Рос-4Кем)я73+22.171 я73
Г 12

Гаврилова Ю.С.

Языки и методы программирования (часть 1): методические рекомендации по выполнению аудиторной и внеаудиторной самостоятельной работы для студентов факультета информатики, математики и экономики, обучающихся по направлениям подготовки 01.03.02 Прикладная математика и информатика, профиль «Математическое моделирование и информационные технологии»; 02.03.03 Математическое обеспечение и администрирование информационных систем, профиль «Программное и математическое обеспечение информационных технологий» / Ю.С. Гаврилова – Новокузнецк : КГПИ ФГБОУ ВО «КемГУ», 2021 – 57 с.

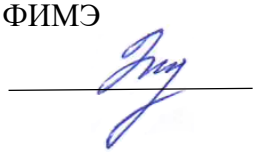
В работе представлены методические материалы по выполнению аудиторной и внеаудиторной самостоятельной работы по дисциплине «Языки и методы программирования»: основные теоретические сведения по синтаксису языков Go, Julia, Kotlin и Swift, примеры решения типовых задач; банк задач для лабораторных работ, методические рекомендации по решению и оформлению, оценивание работ в балльно-рейтинговой системе; список основной и дополнительной литературы.

Методические рекомендации предназначены для наиболее рациональной организации внеаудиторной и аудиторной самостоятельной работы студентов при подготовке к выполнению и выполнению лабораторных работ и теста.

Рекомендовано на заседании
кафедры математики, физики и
математического моделирования
Протокол № 3 от 16.10.2021
Заведующий каф. МФММ


/ Е.В. Решетникова

Утверждено методической комиссией
факультета информатики, математики и
экономики
Протокол № 3 от 11.11.2021
Председатель методической комиссии
ФИМЭ


/ И.А. Жибинова

УДК [378.147.88:519.216](072)
ББК 74.484(2Рос-4Кем)я73+22.171 я73
Г 12

текст представлен в авторской редакции

ОГЛАВЛЕНИЕ

ПРЕДИСЛОВИЕ

Настоящие методические рекомендации адресованы студентам, получающим квалификацию бакалавр по направлениям подготовки: 01.03.02 Прикладная математика и информатика, профиль «Математическое моделирование и информационные технологии» и 02.03.03 Математическое обеспечение и администрирование информационных систем, профиль «Программное и математическое обеспечение информационных технологий» и направлены на оказание помощи студентам в подготовке к выполнению четырех лабораторных работ по дисциплины «Языки и методы программирования».

При изучении различных языков в императивном стиле программирования необходимо учитывать структуру программы, типы данных; особенности объявления переменных и констант, организации ветвлений и циклов. Данные методические материалы содержат краткую теоретическую справку по четырем языкам программирования (Go, Julia, Kotlin, Swift), позволяют студенту подготовиться к лабораторным занятиям по соответствующим темам и выполнить их. Методические рекомендации могут оказаться полезными при написании курсовых и выпускных квалификационных работ.

1 КРАТКИЙ ТЕОРЕТИЧЕСКИЙ МАТЕРИАЛ

1.1 Лабораторная работа №1. Разработка простых алгоритмов на языке GO

Структура программы. Программа на языке Go определяется в виде пакетов (package), это подход языка Go к организации повторного использования кода. Программный код должен быть определен в каком-то определенном пакете, поэтому в самом начале файла с помощью оператора package указывается, к какому пакету будет принадлежать код.

При составлении программного кода может потребоваться функционал из других пакетов. В Go есть множество встроенных пакетов, которые содержат код, выполняющий определенные действия, например, выводить сообщение на консоль. Для подключения других пакетов необходимо использовать директиву import.

Блоками программы на языке Go являются функции. Они имеют входы, выходы и ряд действий, называемых операторами, расположенных в определенном порядке. Любая функция начинается с ключевого слова func, за которым следует имя функции, список из нуля и более параметров, возвращаемый тип (если есть) и само «тело», заключенное в фигурные скобки.

Рассмотрим пример использования встроенной функции, предназначенной для вывода сообщения на консоль. При выборе онлайн-компилятора для языка Go открывается файл «main.go», содержащий следующие команды:

```
package main
import "fmt"

func main() {
    fmt.Println("Hello World")
}
```

Команда import "fmt" подключает пакет fmt, который реализует форматирование для водных и выходных данных. В данном пакете содержится функция Println, которая выводит на экран сообщение, переданное ей в качестве аргумента.

Типы данных. Go – это язык со статической типизацией, т.е.

переменные всегда имеют определенный тип, который нельзя изменить.

В Go существуют следующие типы **целых** чисел: uint8, uint16, uint32, uint64, int8, int16, int32 и int64, числовые характеристики которых указывают, сколько бит использует каждый тип. Uint означает «unsigned integer» (беззнаковое целое), в то время как int означает «signed integer» (знаковое целое). Беззнаковое целое может принимать только положительные значения (или ноль). В дополнение к этому существуют два типа-псевдонима: byte (то же самое, что uint8) и rune (то же самое, что int32).

Также существует 3 машинно-зависимых целочисленных типа: uint, int и uintptr. Они машинно-зависимы, потому что их размер зависит от архитектуры используемого компьютера.

На начальном этапе изучения языка, для выполнения данной работы следует использовать тип int для целых чисел.

В Go есть два **вещественных** типа: float32 и float64 (соответственно, часто называемые вещественными числами с одинарной и двойной точностью). А также два дополнительных типа для представления комплексных чисел (чисел с мнимой частью): complex64 и complex128. Следует использовать тип float64, для работы с числами с плавающей точкой.

Основные математические операции над целыми и вещественными числами в языке Go представлены в таблице 1.1.

Таблица 1.1 – Основные математические операции в языке Go

Литерал	Пояснение
+	сложение
-	вычитание
*	умножение
/	деление
%	остаток от деления

Часто в программах на языке Go встречаются операторы «+=», который в контексте «x+=1» представляет собой «x=x+1», и «==», эквивалентный знаку

равенства в алгебре.

Строковые литералы (тип данных string) могут быть созданы с помощью двойных кавычек "Hello World" или с помощью апострофов `Hello World`. Различие между ними в том, что строки в двойных кавычках могут содержать новые строки (многострочный текст), а также, они позволяют использовать особые управляющие последовательности символов. Например, \n будет заменен символом новой строки и \t будет заменен символом табуляции.

Распространенные операции на строках включают в себя нахождение длины строки: len("Hello World"), доступ к отдельному символу в строке "Hello World"[1], и конкатенация двух строк вместе: "Hello " + "World".

Логические типы. Для взаимодействия с логическими типами используются следующие операторы: &&(и), || (или) и ! (не).

Массивы. Объявление одномерного массива целых чисел из пяти элементов выглядит следующим образом:

```
var x [5] int.
```

Заполнение данного одномерного массива числами может осуществляться путем непосредственного присваивания:

```
x[0] = 98
x[1] = 93
x[2] = 77
x[3] = 82
x[4] = 83
```

Объявление и заполнение двумерного массива:

```
var a=[3][3] int{
{0, 2, 5},
{1, 3, 1},
{2, 0, 8}}
```

Переменные. Go – регистрозависимый язык. Имена должны начинаться с буквы и могут содержать буквы, цифры и знак _ (знак подчеркивания). Переменные объявляются в разделе var, причем сделать это можно двумя способами:

1. Объявить переменную и сразу присвоить ей значение:

```
var x string = ”Hello World”
```

2. Сначала объявить переменную, а уже потом в тексте программы присвоить ей значение:

```
var x string
x = ”Hello World”
```

Знак «=» – это оператор присваивания.

Можно объявить переменную следующим образом: «x:=6» или «x:=Hello World», в данном случае типы данных будут определены автоматически.

Переменные в языке Go могут быть локальными и глобальными. Локальные переменные определяются внутри функции, а глобальные – внутри пакета.

Ветвление. Для того чтобы организовать выбор из двух альтернатив необходимо использовать конструкцию if:

```
if i % 2 == 0 {
     // even
} else {
     // odd
}
```

Если условие, записанное в ветвлении, выполняется, то выполнение программы переходит в раздел, отмеченный «even», если же нет – в раздел, отмеченный «odd».

Цикл for. Сначала создается переменная-счетчик, а затем записываются команды цикла. Заполнение массива элементами с помощью цикла выглядит следующим образом:

```
func main() {
    var x[5] int
    i:=0
    for i<=4 {
        x[i]=i
        fmt.Println(x[i])
        i+=1
    }
}
```

Данный программный код можно сократить, прописав в строке объявления цикла и первый индекс массива, и условие выхода из цикла, и

изменение счетчика:

```
func main() {
    var x[5] int
    for i:=0;i<=4; i++ {
        x[i]=i
        fmt.Println(x[i])
    }
}
```

Вывод двумерного массива на экран реализуется с помощью двух вложенных циклов:

```
for i:=0;i<=2; i++ {
     for j:=0;j<=2; j++{
        fmt.Print(a[i][j], " ")
     }
fmt.Println(" ")
}
```

Заполнение массива 3x3 случайными числами также реализуется с помощью двух вложенных циклов и встроенной функции rand.Intn():

```
package main
import ("fmt"
"math/rand")
func main() {
     var b[3][3] rune
          for i:=0;i<=2; i++ {
          for j:=0;j<=2; j++{
               b[i][j]=rune((rand.Intn(10)))-5
               fmt.Print(b[i][j], " ")
          }
          fmt.Println(" ")
}
```

Импорт "math/rand" необходим для того чтобы можно было использовать функцию rand из данного пакета.

1.2 Лабораторная работа №2. Разработка простых алгоритмов на языке Julia

Структура программы. Основной структурный блок программы – инструкция (statement). Каждая инструкция выполняет некоторое действие: вызов функции, объявление переменных, присвоение значений переменным и т.д. Каждая инструкция располагается на новой строке, без символа «;», однако если на одной строке все же необходимо разместить несколько

инструкций – они отделяются символом «;».

Переменные. Имя переменной представляет произвольный идентификатор, который может содержать алфавитно-цифровые символы или символ подчеркивания и должен начинаться либо с алфавитного символа, либо со знака подчеркивания.

Переменные автоматически создаются при присваивании, тип указывать не обязательно.

Julia – не является регистрозависимым языком программирования, поэтому запуск выполнения данного фрагмента кода не вызовет ошибку:

```
A="Tom"
print(a)
```

Типы данных. Иерархия типов данных представлена на рисунке 1. Самый верхний (общий) тип (супертип) – Any. Все остальные типы – подтипы (субтипы) этого типа. Тип Number является подтипом Any. В свою очередь, Number является супертипом для типов Complex и Real. Типы данных бывают абстрактные и конкретные. Переменная может иметь только конкретный тип.

В языке Julia есть несколько целочисленных типов: со знаком +/- Int8, Int16, Int32, Int64, Int128 и без знака UInt8, UInt16, UInt32, UInt64, UInt128 (8, 16, ...128 – число бит, которое отводится для хранения данных).

Тип Int соответствует Int64.

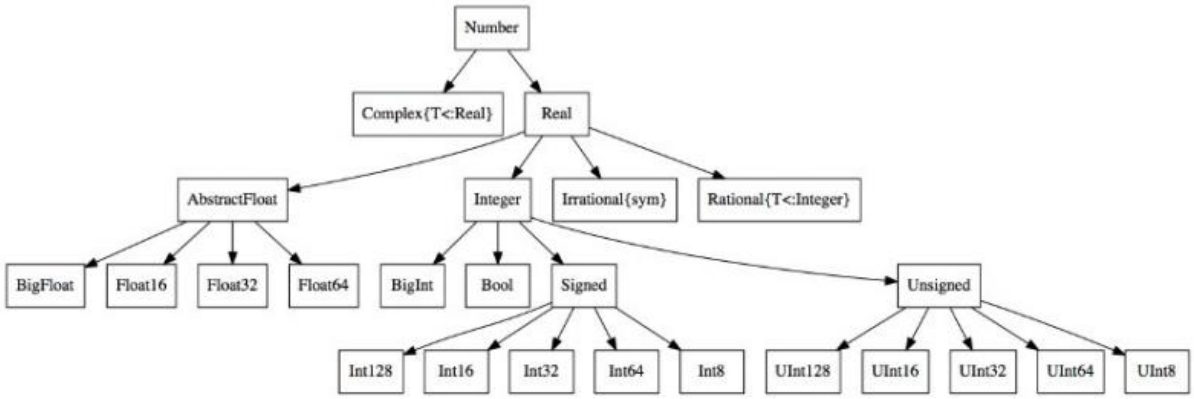


Рисунок 1 – Иерархия типов данных

Julia – это язык со статической типизацией, т.е. переменные всегда имеют определенный тип, который нельзя изменить.

В языке Julia есть четыре **вещественных** типа: Float16, Float32, Float64,

BigFloat.

Основные математические операции над целыми и вещественными числами в языке Julia представлены в таблице 1.2.

Таблица 1.2 – Основные математические операции в языке Julia

Литерал	Пояснение
+	сложение
-	вычитание
*	умножение
/	деление
//	деление для рациональных чисел, результатом является рациональное число
^	возведение в степень
+=	прибавляет к текущей переменной некоторое значение
-=	вычитает из текущей переменной некоторое значение
*=	умножает текущую переменную на некоторое значение, присваивая ей результат умножения
/=	делит значение текущей переменной на другое значение, присваивая результат деления

Символы. Символьные данные представлены типом Char (отдельный символ, который заключается в одинарные кавычки:

```
a = 'a'
```

Строки представлены типом данных string. Строка – это последовательность символов, заключенная в двойные кавычки, либо в тройные двойные кавычки (многострочный текст).

```
c = "Jo"
d = """
     SALT II was a series of talks between United States
     and Soviet negotiators from 1972 to 1979.
     It was a continuation of the SALT I talks.
"""
print(c)
print(d)
```

Для **конкатенации (объединения) строк** используется оператор * (звездочка):

```
c="Hello,"
d="world!"
e=c*d
print(e)
```

Интерполяция представляет собой объединение в строку различных литералов и значений, которые могут представлять другие типы данных. Для интерполяции значений переменных в строку используется символ $:

```
e="pi="
pi=3.14
print(e*"$pi")
println()
```

Логические типы. Для взаимодействия с логическими типами используются следующие операторы: &&(и), || (или) и ! (не).

Массивы. Для хранения набора значений в Julia, как и в других языках программирования, можно использовать массивы. При этом массив может хранить данные только одного и того же типа. В Julia одномерные массивы представлены типом Array или Vector:

a=Array{Float64}(undef,10) – одномерный массив с десятью элементами типа Float64, элементы массива не определены;

a=Vector{Float64}(undef,10) – одномерный массив с десятью элементами типа Float64, элементы массива не определены.

Заполнить одномерный массив случайными числами можно либо прямым перечислением элементов массива (x = [1, 2, 3]), либо с помощью функции rand:

a=rand(1:5,5) – одномерный массив из 5 элементов, заполненный случайными целыми числами от 1 до 5, распределение однородное.

Двумерные массивы. Одномерные массивы можно представить в виде ряда или строки значений. Однако кроме них можно использовать и многомерные массивы. К примеру, возьмем двумерный массив, т. е. такой массив, каждый элемент которого в свою очередь сам является массивом. Двумерный массив еще можно представить в виде таблицы, где каждая строка

– это отдельный массив, а ячейки строки – это элементы вложенного массива.

a=Matrix{Int64}(undef,2,5) – двумерный массив 2*5 – две строки, пять столбцов типа Int64, элементы массива не определены.

a=rand(3,5) – двумерный массив 3*5, заполненный случайными числами от 0 до 1.

Заполним двумерный массив случайными числами в промежутке от -5 до 5, затем в этом массиве первые 2 элемента главной диагонали заменим цифрой 9:

```
a=Matrix{Int64}(undef,5,5)
a=rand(-5:5,5,5)
print(a)
println()
a[2, 2]=9
a[1, 1]=9
print(a)
println()
```

Обратите внимание, что индексы массива начинаются с 1, а не с 0.

Ветвление. Для того чтобы организовать выбор из двух альтернатив необходимо использовать конструкцию if:

```
x=3
if x > 10
    println("X > 10")
else
    println("X < 10")
end
```

Если необходимо проверить несколько альтернативных вариантов, то можно добавить выражения else if:

```
x=3
y=30
if x > y
    println("X is more than Y")
elseif x == y
    println("X and Y are equal")
else
    println("X is less than Y")
end
```

Цикл for. Цикл for пробегается по всем элементам коллекции. В этом плане цикл for в Swift эквивалентен циклу for-each в ряде других языков программирования. Его формальная форма выглядит следующим образом:

for (переменная in последовательность){

выполняемые инструкции

}

Например, выведем все квадраты чисел от 1 до 9, используя цикл for:

```
for i in 1:10
     print("$(i*i) ")
     end
println()
```

Вывод элементов массива на экран с помощью двух вложенных циклов выглядит следующим образом:

```
a=Matrix{Int64}(undef,5,5)
a=rand(-5:5,5,5)
for i in 1:5
     for j in 1:5
          print("$(a[i,j]) ")
     end
println()
end
println()
```

1.3 Лабораторная работа №3. Разработка простых алгоритмов на языке Kotlin

Структура программы. Точкой входа в программу на языке Kotlin является функция main, которая отображается на экране при открытии онлайн-компилятора:

```
fun main(){
println("Hello Kotlin")
}
```

С данной функции начинается выполнение любой программы на языке Kotlin, поэтому она должна быть в любой программе.

Основной структурный блок программы – инструкция (statement). Каждая инструкция выполняет некоторое действие: вызов функции, объявление переменных, присвоение значений переменным и т.д. Каждая инструкция располагается на новой строке, без символа «;», однако если на одной строке все же необходимо разместить несколько инструкций – они отделяются символом «;».

Переменные. Имя переменной представляет произвольный

идентификатор, который может содержать алфавитно-цифровые символы или символ подчеркивания и должен начинаться либо с алфавитного символа, либо со знака подчеркивания. Для определения переменной можно использовать либо ключевое слово **val**, либо ключевое слово **var**.

С помощью ключевого слова val определяется неизменяемая переменная (immutable variable), т.е. можно присвоить значение такой переменной только один раз, а изменить его после первого присвоения уже нельзя.

Kotlin позволяет выводить тип переменной на основании данных, которыми переменная инициализируется. Поэтому при инициализации переменной тип можно опустить:

```
fun main(){
    val age = 5
}
```

Типы данных. Kotlin – это язык со статической типизацией, т.е. переменные всегда имеют определенный тип, который нельзя изменить.

В языке Kotlin существуют следующие типы **целых** чисел: byte, short, int, long, которые отличаются количеством байтов, используемых для хранения числа (1, 2, 4 и 8 соответственно). UInt означает «unsigned integer» (беззнаковое целое), в то время как int означает «signed integer»(знаковое целое). Беззнаковое целое может принимать только положительные значения (или ноль). Кроме UInt в языке есть следующие типы: UByte, UShort и ULong.

Объявление целых переменных и присваивание им значений осуществляется следующим образом:

```
fun main(){
val a: Byte = -10
val b: Short = 45
val c: UInt = 250U
val d: ULong = 30000U
println(a)
println(b)
println(c)
println(d)
}
```

Для передачи значений объектам, которые представляют беззнаковые целочисленные типы данных, после числа указывается суффикс U.

Любые целые числа в языке Kotlin воспринимаются как данные типа Int. Если нужно явно указать, что число представляет значение типа Long, то следует использовать суффикс L:

```
val sum = 45L
```

В Kotlin есть два **вещественных** типа: Float и Double. Объявление вещественных переменных и присваивание им значений осуществляется следующим образом:

```
fun main(){
val height: Double = 1.78
val pi: Float = 3.14F
println(height)
println(pi)
}
```

Чтобы присвоить число объекту типа Float после числа указывается суффикс f или F.

Основные математические операции над целыми и вещественными числами в языке Kotlin представлены в таблице 1.3.

Таблица 1.3 – Основные математические операции в языке Kotlin

Литерал	Пояснение
+	сложение
-	вычитание
*	умножение
/	деление
%	остаток от деления
++	++ увеличивает значение на 1 ++x возвращает увеличенное значение x++ возвращает значение до увеличения
--	-- уменьшает значение на 1 --x возвращает уменьшенное значение x-- возвращает значение до уменьшения
+=:	присваивание после сложения (присваивает левому операнду сумму левого и правого операндов)

-=:	присваивание после вычитания(присваивает левому операнду разность левого и правого операндов)
*=:	присваивание после умножения (присваивает левому операнду произведение левого и правого операндов)
/=:	присваивание после деления (присваивает левому операнду частное левого и правого операндов)
%=:	присваивание после деления по модулю (присваивает левому операнду остаток от целочисленного деления левого операнда на правый)

Если в операции деления оба операнда представляют целые числа, то результатом тоже будет целое число, а если в процессе деления образовалась дробная часть, то она отбрасывается:

```
fun main() {
val x = 11
val y = 5
val z = x / y
println(z)
}
```

Так в данном случае, хотя если согласно математике разделить 11 на 5, то получится 2.2, однако поскольку оба операнда представляют целочисленный тип, а именно тип Int, то дробная часть (0.2) отбрасывается, поэтому результатом будет число 2, а переменная z будет представлять тип Int.

Чтобы результатом было дробное число, один из операндов должен представлять число с плавающей точкой:

```
fun main() {
val x = 11
val y = 5.0
val z = x / y
println(z)
}
```

В данном случае переменная y представляет тип Double, поэтому результатом деления будет число 2.2, а переменная z также будет представлять тип Double.

Символы. Символьные данные представлены типом Char (отдельный

символ, который заключается в одинарные кавычки.

```
fun main(){
     val a: Char = 'A'
     val b: Char = 'B'
     val c: Char = 'T'
}
```

Строки представлены типом данных string. Строка – это последовательность символов, заключенная в двойные кавычки, либо в тройные двойные кавычки (многострочный текст).

```
fun main() {
    val name: String ="Jo"
    val text: String = """
     SALT II was a series of talks between United States
     and Soviet negotiators from 1972 to 1979.
     It was a continuation of the SALT I talks."""
println(text)
println(name)
}
```

Шаблоны строк (**stringtemplates**) представляют способ вставки в строку различных значений, в частности, значений переменных.

С помощью знака доллара $ можно вводить в строку значения различных переменных:

```
fun main() {
    val firstName = "Tom"
    val lastName = "Smith"
    val welcome = "Hello, $firstName $lastName"
    println(welcome)
}
```

Логические типы. Для взаимодействия с логическими типами используются следующие операторы: and(и), or (или) и ! (не).

Тип Any. Тип данных ограничивает набор значений, которые можно присвоить переменной. Например, нельзя присвоить переменной типа Double строку. После того, как тип переменной установлен, он не может быть изменен.

Однако в Kotlin также есть тип Any, который позволяет присвоить переменной данного типа любое значение:

```
fun main() {
var name: Any = "Tom"
```

```
println(name)
name = 6758
println(name)
}
```

Массивы. Для хранения набора значений в Kotlin, как и в других языках программирования, можно использовать массивы. При этом массив может хранить данные только одного того же типа. В Kotlin массивы представлены типом Array.

При определении массива после типа Array в угловых скобках необходимо указать, объекты какого типа могут храниться в массиве. Например, определим массив целых чисел. С помощью встроенной функции arrayOf() можно передать набор значений, которые будут составлять массив:

```
val numbers: Array<Int> = arrayOf(1, 2, 3, 4, 5)
```

С помощью индексов можно обратиться к определенному элементу в массиве. Индексация начинается с нуля, то есть первый элемент будет иметь индекс 0. Индекс указывается в квадратных скобках:

```
fun main() {
val numbers: Array<Int> = arrayOf(1, 2, 3, 4, 5)
val n = numbers[1]
numbers[2] = 7
println("numbers[2] = ${numbers[2]}")
}
```

Двумерные массивы. Одномерные массивы можно представить в виде ряда или строки значений. Однако кроме них можно использовать и многомерные массивы. К примеру, возьмем двумерный массив, т. е. такой массив, каждый элемент которого в свою очередь сам является массивом. Двумерный массив еще можно представить в виде таблицы, где каждая строка - это отдельный массив, а ячейки строки - это элементы вложенного массива.

Определение двумерных массивов менее понятно и может вызывать сложности. Например, двумерный массив чисел:

```
val table = Array(3, { Array(3, {0}) })
table[0] = arrayOf(1, 2, 3)
table[1] = arrayOf(4, 5, 6)
table[2] = arrayOf(7, 8, 9)
```

Для обращения к элементам подмассивов двумерного массива

необходимы два индекса. По первому индексу идет получение строки, а по второму индексу - столбца в рамках этой строки:

```
val table = Array(3, { Array(3, {0}) })
table[0][1] = 6
val n = table[0][1]
```

Ветвление. Для того чтобы организовать выбор из двух альтернатив необходимо использовать конструкцию if:

```
val a = 10
if (a == 10) {
     println("a равно 10")
}
else{
     println("a НЕ равно 10")
}
```

Если необходимо проверить несколько альтернативных вариантов, то можно добавить выражения else if:

```
val a = 10
if (a == 10) {
     println("a равно 10")
}
else if(a == 9){
     println("a равно 9")
}
else if(a == 8){
     println("a равно 8")
}
else{
     println("a <8 или a>10")
}
```

Конструкция if может возвращать значение. Например, найдем максимальное из двух чисел:

```
val a = 10
val b = 20
val c = if (a > b) a else b
println(c)
```

Цикл for. Цикл for пробегает по всем элементам коллекции. В этом плане цикл for в Kotlin эквивалентен циклу for-each в ряде других языков программирования. Его формальная форма выглядит следующим образом:

for (переменная in последовательность){

выполняемые инструкции

}

Например, выведем все квадраты чисел от 1 до 9, используя цикл for:

```
for(n in 1..9){
     print("${n * n} \t")
}
```

Вывод элементов массива на экран с помощью цикла выглядит следующим образом:

```
fun main() {
val numbers: Array<Int> = arrayOf(1, 2, 3, 4, 5)
for (i in 0..4){
     print("${numbers[i]}\t")}
}
```

Обработка двумерного массива реализуется с помощью двух вложенных циклов:

```
fun main() {
    val table: Array<Array<Int>> = Array(3,{Array(3, {0})})
    table[0] = arrayOf(1, 2, 3)
    table[1] = arrayOf(4, 5, 6)
    table[2] = arrayOf(7, 8, 9)
    for(row in table){
        for(cell in row){
            print("$cell \t")
        }
        println()
    }
}
```

Заполнение массива случайными числами реализуется с помощью следующего кода:

```
import kotlin.random.Random
fun main(args: Array<String>) {
     val table2: Array<Array<Int>> = Array(3,{Array(3,{0})})
     val n=table2.size;
     for (i in 0 until n){
     for(j in 0 until n){
        table2[i][j] = Random.nextInt(10)
        }
    }
     println()
     for(row2 in table2){
        for(cell2 in row2){
            print("$cell2 \t")
        }
```

```
            println()
        }
    }
```

Цикл while. Цикл while повторяет определенные действия пока истинно некоторое условие:

```
var i = 10
while(i> 0){
     println(i*i)
     i--;
}
```

Здесь пока переменная i больше 0, будет выполняться цикл, в котором на консоль будет выводиться квадрат значения i.

Можно применять данный тип циклов для перебора массива. Например:

```
fun main() {
val people = arrayOf("Tom", "Sam", "Bob")
var i = 0
while(i in people.indices){
     println(people[i])
     i++;
    }
}
```

Здесь определена дополнительная переменная i, которая представляет индекс элемента массива. У массива есть специальное свойство indices, которое содержит набор всех индексов. А выражение i inpeople.indices возвращает true, если значение переменной i входит в набор индексов массива.

В самом цикле по индексу организовано обращение к элементу массива: println(people[i]). Затем переход к следующему индексу с увеличением счетчика: i++.

1.4 Лабораторная работа №4. Разработка простых алгоритмов на языке Swift

Структура программы. Основной структурный блок программы – инструкция (statement). Каждая инструкция выполняет некоторое действие: вызов функции, объявление переменных, присвоение значений переменным и т.д. Каждая инструкция располагается на новой строке, без символа «;», однако если на одной строке все же необходимо разместить несколько

инструкций – они отделяются символом «;».

Переменные и константы. Имя переменной представляет произвольный идентификатор, который может содержать алфавитно-цифровые символы или символ подчеркивания и должен начинаться либо с алфавитного символа, либо со знака подчеркивания.

Swift – регистрозависимый язык программирования, поэтому запуск выполнения данного фрагмента кода вызовет ошибку:

```
var A="Tom"
print(a)
```

Для определения переменной нужно использовать ключевое слово **var**, для определения константы – ключевое слово **let**.

Константе **let** можно присвоить значение только один раз, а изменить его после первого присвоения уже нельзя.

Переменные и константы должны иметь уникальные имена. Нельзя использовать в программе несколько переменных и (или) констант с одними и теми же именами.

Причем хорошей практикой является использование названий в так называемом «верблюжьем регистре» или CamelCase. То есть названия начинаются с маленькой буквы.

Если название состоит из нескольких слов, то только первое из них начинается с маленькой буквы. Например: personName, personStreatAddress и т.д.

Swift позволяет выводить тип переменной на основании данных, которыми переменная инициализируется. Поэтому при инициализации переменной тип можно опустить.

Типы данных. Swift – это язык со статической типизацией, т.е. переменные всегда имеют определенный тип, который нельзя изменить.

В языке Swift существуют следующие типы **целых** чисел:

– Int8: представляет целые числа со знаком размером не более 8 бит (от -128 до 127);

– UInt8: представляет целые положительные числа размером не

более 8 бит (от 0 до 255);

– Int16: представляет целые числа со знаком размером не более 16 бит (от -32768 до 32767);

– UInt16: представляет целые положительные числа размером не более 16 бит (от 0 до 65535);

– Int32: представляет целые числа со знаком размером не более 32 бита (от -2147483648 до 2147483647);

– UInt32: представляет целые положительные числа размером не более 32 бита (от 0 до 4294967295);

– Int64: представляет целые числа со знаком размером не более 64 бита (от -9223372036854775808 до 9223372036854775807);

– UInt64: представляет целые положительные числа размером не более 64 бита (от 0 до 18446744073709551615);

– Int: представляет целые числа со знаком, например, 1, -30, 458. На 32-разрядных платформах эквивалентен Int32, а на 64-разрядных - Int64;

– UInt: представляет целые положительные числа, например, 1, 30, 458. На 32-разрядных платформах эквивалентен UInt32, а на 64-разрядных - UInt64.

Объявление целых переменных и присваивание им значений осуществляется следующим образом:

```
var age: Int = 32
```

или можно сначала определить переменную, а потом присвоить ей значение:

```
var name: String
name = "Tom".
```

Любые целые числа в языке Swift воспринимаются как данные типа Int.

В языке Swift есть три **вещественных** типа:

– Float: 32-битное число с плавающей точкой, содержит до 6 чисел в дробной части;

– Double: 64-битное число с плавающей точкой, содержит до 15

чисел в дробной части;

– Float80: 80-битное число с плавающей точкой.

Объявление вещественных переменных и присваивание им значений осуществляется следующим образом:

```
var d = 3.4            // тип Double
var f : Float = 1.2
```

Основные математические операции над целыми и вещественными числами в языке Swift представлены в таблице 1.4.

Таблица 1.4 – Основные математические операции в языке Swift

Литерал	Пояснение
+	сложение
-	вычитание
*	умножение
/	деление
%	остаток от деления
+=	прибавляет к текущей переменной некоторое значение
-=	вычитает из текущей переменной некоторое значение
*=	умножает текущую переменную на некоторое значение, присваивая ей результат умножения
/=	делит значение текущей переменной на другое значение, присваивая результат деления
%=	делит значение текущей переменной на другое значение, присваивая переменной остаток от деления

В арифметических операциях все операнды должны представлять один и тот же тип данных. Результатом операции является значение того же типа, что и тип операндов.

Если операнды представляют константы или переменные, или являются результатами каких-то других операций или выражений, то в этом случае

нужно явно выполнять преобразование типов. Для этого применяются специальные функции-инициализаторы типов данных. Все они совпадают с названиями типов данных: Int8(), Int(), Float(), Double() и т.д. Так, функция Double() преобразует значение к типу Double, а в скобки передается само значение. Например:

```
let d = 8         // представляет тип Int
let g = Double(d) // преобразуем в тип Double
print(g)  // 8.0
```

Символы. Символьные данные представлены типом Character (отдельный символ, который заключается в двойные кавычки:

```
var a: Character = "a"
```

Строки представлены типом данных string. Строка – это последовательность символов, заключенная в двойные кавычки, либо в тройные двойные кавычки (многострочный текст для версии языка после 4.0).

```
var name: String = "Jo"
var text: String = """
     SALT II was a series of talks between United States
     and Soviet negotiators from 1972 to 1979.
     It was a continuation of the SALT I talks.
"""
print(text)
print(name)
```

Для **преобразования значений** других типов данных к строке надо передать преобразуемое значение в инициализатор:

```
var number: Int = 22
var str: String = String(number)
```

Для **конкатенации (объединения) строк** используется оператор + (плюс):

```
var str: String = ""    // ""
str += "hello"          // "hello"
str = str + " world"    // "hello word"
```

Интерполяция представляет собой объединение в строку различных литералов и значений, которые могут представлять другие типы данных. Для интерполяции объекты других типов данных заключаются в скобки, перед которыми ставится слеш:

```
var age: Int = 22
var str: String = "Age: \(age)"
var name: String = "Ann"
str = "Age: \(age) and name: \(name)"
print(str)
```

Логические типы. Для взаимодействия с логическими типами используются следующие операторы: &&(и), || (или) и ! (не).

Массивы. Для хранения набора значений в Swift, как и в других языках программирования, можно использовать массивы. При этом массив может хранить данные только одного и того же типа. В Swift массивы представлены типом Array.

При определении массива после типа Array в угловых скобках необходимо указать, объекты какого типа могут храниться в массиве. Например, определим массив целых чисел:

```
var numbers: [Int] = [1, 2, 3, 4, 5]
```

С помощью индексов можно обратиться к определенному элементу в массиве. Индексация начинается с нуля, то есть первый элемент будет иметь индекс 0. Индекс указывается в квадратных скобках:

```
var numbers = [11, 12, 13, 14, 15]
print(numbers[0])
```

С помощью свойства count можно получить число элементов массива:

```
var numbers: [Int] = [1, 2, 3, 4, 5, 6, 7, 8]
print ("В массиве numbers \(numbers.count) элементов")
```

Двумерные массивы. Одномерные массивы можно представить в виде ряда или строки значений. Однако кроме них можно использовать и многомерные массивы. К примеру, возьмем двумерный массив, т. е. такой массив, каждый элемент которого в свою очередь сам является массивом. Двумерный массив еще можно представить в виде таблицы, где каждая строка - это отдельный массив, а ячейки строки – это элементы вложенного массива.

Определение двумерных массивов менее понятно и может вызывать сложности. Например, добавим 2 строки в двумерный массив чисел:

```
var arr = [[Int]](repeating: [Int](repeating: 0, count: 5), count: 5)
```

```
arr[0]=[1, 2, 3, 4, 5]
arr[1]=[1, 2, 3, 4, 5]
```

Для обращения к элементам подмассивов двумерного массива необходимы два индекса. По первому индексу идет получение строки, а по второму индексу – столбца в рамках этой строки:

```
var arr = [[Int]](repeating: [Int](repeating: 0, count: 5), count: 5)
arr[0]=[1, 2, 3, 4, 5]
arr[0][0]=9
```

Ветвление. Для того чтобы организовать выбор из двух альтернатив необходимо использовать конструкцию if:

```
var a = 10
if (a == 10) {
     print("a равно 10")
}
else{
     print("a НЕ равно 10")
}
```

Если необходимо проверить несколько альтернативных вариантов, то можно добавить выражения else if:

```
var a = 6
if (a == 10) {
     print("a равно 10")
}
else if (a == 9) {
     print("a равно 9")
}
else {
     print("a<9 или >10")
}
```

Конструкция if может возвращать значение (тернарный оператор). В зависимости от условия тернарный оператор возвращает второй или третий операнд: если условие равно true, то возвращается второй операнд; если условие равно false, то третий. Например: Например, найдем максимальное из двух чисел:

```
var a = 10
var b = 20
var c = a > b ? a : b
print(b)
```

Цикл for. Цикл for пробегается по всем элементам коллекции. В этом

плане цикл for в Swift эквивалентен циклу for-each в ряде других языков программирования. Его формальная форма выглядит следующим образом:

for (переменная in последовательность){

выполняемые инструкции

}

Например, выведем все квадраты чисел от 1 до 9, используя цикл for:

```
for i in 1...10
     print(i*i)
}
```

Вывод элементов массива на экран с помощью цикла выглядит следующим образом:

```
var numbers: [Int] = [1, 2, 3, 4, 5, 6, 7, 8]
for i in 0 ..< numbers.count {
    print(numbers[i])
}
```

Обработка двумерного массива реализуется с помощью двух вложенных циклов:

```
var arr = [[Int]](repeating: [Int](repeating: 0, count: 5), count: 5)
var str=""
for i in 0..<arr.count{
     for j in 0..<arr[0].count{
          arr[i][j]=i+j
          str=str+" "+String(arr[i][j])
     }
     print(str)
}
```

Заполнение массива случайными числами реализуется с помощью следующего кода:

```
var arr = [[Int]](repeating: [Int](repeating: 0, count: 5), count: 5)
var str=""
for i in 0..<arr.count{
     str=""
     for j in 0..<arr[0].count{
          arr[i][j]=Int.random(in: 1..<10)
          str=str+" "+String(arr[i][j])
     }
     print(str)
}
```

Цикл while. Цикл while повторяет определенные действия пока истинно некоторое условие:

```
var i = 10
repeat {
    print(i*i)
    i-=1
} while i > 0
```

Здесь пока переменная i больше 0, будет выполняться цикл, в котором на консоль будет выводиться квадрат значения i.

2 ИНДИВИДУАЛЬНЫЕ ЗАДАНИЯ ДЛЯ ВЫПОЛНЕНИЯ ЛАБОРАТОРНОЙ РАБОТЫ

№	Задание
Вариант 1	
1	Даны две строки s_1 и s_2. Посчитать, сколько раз в строке s_2 встречается первый символ строки s_1.
2	Дана строка s, состоящая из слов (последовательностей символов, не содержащих пробелов внутри себя), разделенных между собой одним или несколькими пробелами. Найти номер самого длинного слова. Если таких слов несколько, то указать все их номера.
3	Дан одномерный числовой массив. Если он упорядочен по убыванию, то вывести его на экран в обратном порядка; в противном случае вывести на экран номер первого элемента, нарушающего упорядоченность
4	Дан одномерный числовой массив. Все элементы массива, кратные 3, домножить на их индекс, а затем найти среднее арифметическое элементов преобразованного массива.
5	Дана целочисленная квадратная матрица четвертого порядка. Найти наименьшее из значений элементов столбца, который обладает наибольшей суммой модулей элементов. Если таких столбцов несколько, то взять первый из них.
6	Дана действительная квадратная матрица четвертого порядка, в которой не все элементы равны нулю. Получить новую матрицу путем деления всех элементов данной матрицы, лежащих ниже главной диагонали, на ее наибольший по модулю элемент.
Вариант 2	
1	Дана строка s. Известно, что среди символов данной строки имеется хотя бы один пробел и что первый символ строки s отличен от пробела. Преобразовать строку s, заменив все малые буквы латинского алфавита, предшествующие первому пробелу, одноименными большими.
2	Даны две строки s_1 и s_2, имеющие одинаковую длину. Образовать строку s, в которой должны чередоваться символы строк s_1 и s_2.
3	В одномерном числовом массиве поменять местами элементы с четными и нечетными индексами, если оба они кратны минимальному, и заменить эти элементы их индексами в обратном случае.
4	Дан одномерный числовой массив, все элементы которого различны. Если сумма всех элементов, расположенных между максимальным и минимальным элементами массива, меньше 10, то вычислить произведение ненулевых элементов одномерного числового массива; в противном случае найти количество элементов массива, кратных минимальному, и их произведение.

5	Дана действительная квадратная матрица четвертого порядка, в которой не все элементы равны нулю. Получить новую матрицу путем деления всех элементов данной матрицы, лежащих ниже главной диагонали, на ее наибольший по модулю элемент.
6	Дана целочисленная квадратная матрица четвертого порядка. Заменить нулями все неотрицательные элементы этой матрицы, находящиеся на ее побочной диагонали.
Вариант 3	
1	Дана строка *s*, состоящая из слов (последовательностей символов, не содержащих пробелов внутри себя), разделенных между собой одним или несколькими пробелами. Если количество слов в строке *s* четно, то поменять местами ее средние слова. В противном случае на место среднего слова строки *s* поместить пробелы.
2	Дана строка *s*. Преобразовать ее, заменив в ней каждую точку многоточием (то есть тремя точками), а все восклицательные знаки вопросительными.
3	Дан одномерный числовой массив, все элементы которого различны. Подсчитать в нем количество четных элементов, расположенных между наибольшим и наименьшим элементами массива.
4	Дан одномерный числовой массив, все элементы которого различны. Найти наименьшее положительное значение элемента массива и его номер. Если этот номер окажется больше 5, то заменить все элементы, расположенные между наибольшим и наименьшим элементами массива, их квадратами, иначе - кубами.
5	Дана действительная квадратная матрица четвертого порядка. Вычислить сумму тех из ее элементов, расположенных на главной диагонали и выше нее, которые превосходят по величине все элементы, расположенные ниже главной диагонали. Если на главной диагонали и выше нее нет элементов с указанным свойством, то вывести сообщение об этом.
6	Даны натуральные числа n, m, действительная матрица размера m×n. Найти среднее арифметическое каждого из столбцов, имеющих четные номера.
Вариант 4	
1	Дана строка *s*. Преобразовать эту строку так, чтобы ее символы шли в обратном порядке по отношению к их исходному расположению.
2	Дана строка *s*, состоящая из слов (последовательностей символов, не содержащих пробелов внутри себя), разделенных между собой одним или несколькими пробелами. Определить, сколько слов состоит из одинаковых символов, например: ааааа, еее.
3	Дан одномерный числовой массив, все элементы которого различны. Посчитать в нем количество соседств из двух положительных чисел.

	Если это количество кратно 3, то вывести на экран все элементы расположенные до максимального, иначе — после максимального.
4	Дан одномерный числовой массив, все элементы которого различны. Посчитать сумму всех неотрицательных элементов массива с четными индексами. Если эта сумма окажется больше 7, то заменить наибольший и наименьший элементы массива их произведением, иначе — поменять их местами.
5	Дана целочисленная квадратная матрица четвертого порядка. Заменить нулями все неотрицательные элементы этой матрицы, находящиеся на ее побочной диагонали.
6	В данной действительной матрице размера m×n поменять местами строку, содержащую элемент с наибольшим значением, со строкой, содержащий элемент с наименьшим значением. Предполагается, что эти элементы единственны.
Вариант 5	
1	Определить долю пробелов в данной строке *s*.
2	Даны две строки s_1 и s_2, имеющие различную длину. Посчитать в них количество попарно одинаковых символов.
3	Дан упорядоченный массив $a_1,\ a_2,\ \ldots a_n$. Известно, что число x принадлежит отрезку числовой оси, вмещающему заданный массив. Определить номер k, для которого $a_{k-1} < x < a_k$. Если k окажется кратным 3, то вывести на экран элементы массива в обратном порядке.
4	Дан одномерный числовой массив, все элементы которого различны. Посчитать сумму кубов элементов с нечетными индексами, расположенных до наибольшего элемента массива, и произведение индексов всех элементов, которые по числовому значению превосходят минимальный, но не превосходят максимальный элемент массива.
5	Дана действительная квадратная матрица порядка n. Переменной y присвоить значение, равное скалярному произведению строки и столбца (рассматриваемых как векторы), на пересечении которых находится наименьший элемент матрицы (в предположении, что такой элемент единственный).
6	Дана действительная квадратная матрица порядка n. Рассмотрим те ее элементы, которые расположены в строках, начинающихся с отрицательного элемента. Найти суммы тех из них, которые находятся соответственно ниже, выше и на побочной диагонали.
Вариант 6	
1	Дана строка *s*, состоящая из слов (последовательностей символов, не содержащих пробелов внутри себя), разделенных между собой одним или несколькими пробелами. Определить число слов, в которых последняя буква в слове не повторяется.

2	Дана строка *s*. Определить, является ли она ”палиндромом”. Если нет, то выяснить, станет ли строка *s* ”палиндромом” после удаления из нее всех пробелов.
3	Дан одномерный числовой массив. Определить, сколько в нем имеется пар совпадающих по величине соседних чисел; вывести на экран их индексы.
4	Дан одномерный числовой массив. Добавить к каждому элементу массива, кроме первого, значение предыдущего. После такого преобразования вычислить среднее арифметического наибольшего и наименьшего элементов массива.
5	Даны натуральные числа n, m, действительная матрица размера m×n. Найти среднее арифметическое каждого из столбцов, имеющих четные номера.
6	Дана действительная матрица размера m×n, все элементы которой различны. В каждой строке выбирается элемент с наименьшим значением, затем среди этих чисел выбирается наибольшее. Указать индексы этого элемента.
Вариант 7	
1	Дана строка s_1. Образовать строку s_2, включив в нее символы строки s_1, расположенные на нечетных позициях, кроме пробелов и знаков препинания.
2	Дана строка *s*, состоящая из слов (последовательностей символов, не содержащих пробелов внутри себя), разделенных между собой одним или несколькими пробелами. Определить, повторяется ли в тексте первое слово.
3	Найти наибольший и наименьший элементы одномерного числового массива, а также подсчитать количество элементов, кратных наименьшему элементу массива.
4	Дан одномерный числовой массив, все элементы которого различны. Заменить в нем четные элементы, расположенные после максимального, их квадратами, а нечетные элементы, расположенные после максимального — их кубами.
5	Даны натуральное число n, действительная квадратная матрица порядка n, действительные. Элементы последовательности домножить на 3, если наибольший элемент матрицы (в предположении, что такой элемент единственный) расположен на главной диагонали, и на 0.5 в противном случае.
6	В данной действительной квадратной матрице порядка n найти сумму элементов строки, в которой расположен элемент с наименьшим значением. Предполагается, что такой элемент единственный.
Вариант 8	
1	Дана строка *s*. Вывести ее на экран, подчеркивая (ставя минусы в соответствующих позициях следующей строки) все входящие в него цифры.

2	Дана строка s. Исключить из нее все лишние пробелы, то есть вместо каждой группы пробелов оставить только один. При исключении пробелов текст сдвигается влево.
3	Вычислить произведение всех ненулевых элементов одномерного числового массива с нечетными индексами. Заменить все элементы массива, равные нулю, отношением их индекса к значению максимального элемента.
4	Заменить в одномерном числовом массиве элементы, большие числа *M*, на число *a*, введенное с клавиатуры, а меньшие *M* — отношением $\frac{a}{i}$, где *i* — индекс элемента.
5	В данной действительной матрице размера m×n поменять местами строку, содержащую элемент с наибольшим значением, со строкой, содержащий элемент с наименьшим значением. Предполагается, что эти элементы единственны.
6	Дан двумерный целочисленный массив размером m×n. Некоторый элемент этого массива назовем «седловой точкой», если он является одновременно наименьшим в своей строке и наибольшим в своем столбце. Напечатать номера строки и столбца какой-нибудь «седловой точки» и напечатать число 0, если такой точки нет.
Вариант 9	
1	Дана строка *s*, состоящая из слов (последовательностей символов, не содержащих пробелов внутри себя), разделенных между собой одним или несколькими пробелами. Найти номер самого короткого слова. Если таких слов несколько, то указать все их номера.
2	Даны две строки s_1 и s_2, имеющие различную длину. Посчитать в них количество попарно различных символов.
3	В одномерном числовом массиве заменить отрицательные элементы их квадратами, а положительные разделить на числовое значение минимального элемента.
4	Дан одномерный числовой массив. Определить количество элементов этого массива, которые имеют четные индексы и являются кратными 3, а также вычислить сумму и произведение элементов, кратных либо первому, либо последнему элементам массива
5	Дана целочисленная квадратная матрица четвертого порядка. Переменной y присвоить значение наибольшего из элементов матрицы, расположенных на главной диагонали и ниже ее.
6	Даны натуральное число m и целочисленная квадратная матрица порядка m. Строку с номером i матрицы назовем отмеченной, если $a_i > 0$, и неотмеченной в противном случае. Посчитать число отрицательных элементов, расположенных в отмеченных строках.
Вариант 10	
1	Дана строка *s*, состоящая из слов (последовательностей символов, не

	содержащих пробелов внутри себя), разделенных между собой одним или несколькими пробелами. Определить, сколько раз стоящие рядом два слова начинаются на одну и ту же букву.
2	Дана строка *s*. Определить в ней долю символов, являющихся буквами латинского алфавита.
3	Определить в одномерном числовом массиве число соседств из двух чисел, сумма которых равна 5. Вычислить произведение всех элементов, удовлетворяющих этому условию.
4	Найти суммы и произведения отрицательных и положительных элементов одномерного числового массива. Подсчитать, сколько имеется в массиве элементов, равных нулю.
5	Дана действительная квадратная матрица порядка n. Рассмотрим те ее элементы, которые расположены в строках, начинающихся с отрицательного элемента. Найти суммы тех из них, которые находятся соответственно ниже, выше и на побочной диагонали.
6	Дана действительная квадратная матрица четвертого порядка. В строках с отрицательным элементом на побочной диагонали найти сумму всех элементов.
Вариант 11	
1	Дана строка *s*, состоящая из букв латинского алфавита. Верно ли, что в данной строке гласные буквы (*a, e, i, o, u*) чередуются с согласными?
2	Дана строка *s*. Известно, что среди символов данной строки есть по крайней мере один восклицательный знак и что первый символ этой строки отличен от восклицательного знака. Выяснить, имеется ли среди символов строки *s*, предшествующих первому восклицательному знаку, пара соседствующих одинаковых символов.
3	Дан одномерный числовой массив. Домножить на 3 все его положительные элементы с нечетными индексами; все отрицательные элементы, имеющие четные индексы, уменьшить в 5 раз.
4	Дан одномерный числовой массив. Вычислить сумму квадратов его элементов с четными индексами, а также произведение всех ненулевых элементов с нечетными индексами.
5	Дана действительная квадратная матрица порядка n. В строках с отрицательным элементом на главной диагонали найти наибольший из всех элементов.
6	Дана действительная квадратная матрица четвертого порядка, в которой не все элементы равны нулю. Получить новую матрицу путем деления всех элементов данной матрицы, лежащих ниже главной диагонали, на ее наибольший по модулю элемент.
Вариант 12	
1	Дана строка *s*, состоящая из слов (последовательностей символов, не содержащих пробелов внутри себя), разделенных между собой одним или несколькими пробелами. Если первое и последнее слова строки

	имеют одинаковую длину, то поменять их местами. В противном случае оставить строку *s* без изменения.
2	Дана строка *s*. Удалить из нее все группы букв вида *abcd*.
3	Определить в одномерном числовом массиве количество соседств из взаимно противоположных и не взаимно обратных чисел.
4	Добавить к каждому нечетному элементу одномерного целочисленного массива его номер, а затем все четные элементы заменить произведением номера элемента на отношение наибольшего элемента массива к наименьшему.
5	Дана действительная матрица размера m×n, все элементы которой различны. В каждой строке выбирается элемент с наименьшим значением, затем среди этих чисел выбирается наибольшее. Указать индексы этого элемента.
6	Дана целочисленная квадратная матрица четвертого порядка. Заменить нулями все неотрицательные элементы этой матрицы, находящиеся на ее побочной диагонали.
Вариант 13	
1	Дана строка *s*, состоящая из слов (последовательностей символов, не содержащих пробелов внутри себя), разделенных между собой одним или несколькими пробелами. Определить число слов в строке, состоящих из нечетного числа символов.
2	Дана строка *s*, состоящая из букв латинского алфавита. Преобразовать данную строку, упорядочив в ней буквы по алфавиту.
3	Найти наибольший элемент одномерного числового массива из стоящих на нечетных местах и наименьший из стоящих на четных местах. Если сумма этих элементов больше значения первого элемента массива, то уменьшить их в 2 раза, иначе — увеличить каждый из них на 3.
4	Дан одномерный числовой массив, все элементы которого различны. Посчитать, сколько в нем содержится элементов, совпадающих по абсолютной величине с номером. Если таких элементов нет, то вычислить сумму и произведение наибольшего и наименьшего элементов массива.
5	Дана действительная матрица размера m×n. Найти среднее арифметическое наибольшего и наименьшего элементов этой матрицы из лежащих на ее побочной диагонали
6	Даны натуральные числа n, m, действительная матрица размера m×n. Найти среднее арифметическое каждого из столбцов, имеющих четные номера.
Вариант 14	
1	Дана строка *s*, состоящая из слов (последовательностей символов, не содержащих пробелов внутри себя), разделенных между собой одним или несколькими пробелами. Определить, содержит ли данная строка

	слова, заканчивающиеся символом * .
2	Дана строка *s*. Известно, что среди символов данной строки имеется хотя бы один пробел и что первый символ строки *s* отличен от пробела. Преобразовать строку *s*, удалив из нее все символы, предшествующие первому пробелу и не являющиеся буквами латинского алфавита.
3	В одномерном числовом массиве найти количество и сумму элементов, кратных 3, а также среднее арифметического всех элементов, имеющих четные индексы и кратных 2.
4	Определить в одномерном числовом массиве количество соседств из чисел разного знака. Вывести на экран последовательность из чисел одного знака, имеющую наибольшую длину.
5	В данной действительной квадратной матрице порядка n найти сумму элементов строки, в которой расположен элемент с наименьшим значением. Предполагается, что такой элемент единственный.
6	В данной действительной матрице размера m×n поменять местами строку, содержащую элемент с наибольшим значением, со строкой, содержащий элемент с наименьшим значением. Предполагается, что эти элементы единственны.
Вариант 15	
1	Дана строка *s*. Выяснить, имеются ли такие натуральные *i* и *j*, что $i < j$ и *i*-тый символ строки *s* совпадает и (*i*+1)-ым символом, а *j*-тый — с (*j*+1)-ым символом строки *s*.
2	Дана строка *s*, состоящая из слов (последовательностей символов, не содержащих пробелов внутри себя), разделенных между собой одним или несколькими пробелами. Преобразовать строку s, удалив из нее все повторные вхождения слов.
3	Дан одномерный числовой массив. Найти сумму корней квадратных из абсолютных величин его элементов, а также произведение всех отрицательных элементов с четными номерами и произведение всех положительных элементов с нечетными номерами.
4	Посчитать в одномерном числовом массиве количество элементов, равных первому отрицательному. Вывести на экран самую длинную подпоследовательность из элементов, равных нулю.
5	Дана действительная квадратная матрица порядка n. Найти сумму элементов матрицы, расположенных в строках с отрицательным элементом на главной диагонали.
6	Дана действительная квадратная матрица порядка n. Рассмотрим те ее элементы, которые расположены в строках, начинающихся с отрицательного элемента. Найти суммы тех из них, которые находятся соответственно ниже, выше и на побочной диагонали.
Вариант 16	
1	Даны две строки s_1 и s_2. Посчитать, сколько раз в строке s_2 встречается первый символ строки s_1.

2	Дана строка *s*, состоящая из слов (последовательностей символов, не содержащих пробелов внутри себя), разделенных между собой одним или несколькими пробелами. Найти номер самого длинного слова. Если таких слов несколько, то указать все их номера.
3	Дан одномерный числовой массив. Если он упорядочен по убыванию, то вывести его на экран в обратном порядка; в противном случае вывести на экран номер первого элемента, нарушающего упорядоченность
4	Дан одномерный числовой массив. Все элементы массива, кратные 3, домножить на их индекс, а затем найти среднее арифметическое элементов преобразованного массива.
5	Дан двумерный целочисленный массив размером m×n. Некоторый элемент этого массива назовем «седловой точкой», если он является одновременно наименьшим в своей строке и наибольшим в своем столбце. Напечатать номера строки и столбца какой-нибудь «седловой точки» и напечатать число 0, если такой точки нет.
6	Дана действительная матрица размера m×n, все элементы которой различны. В каждой строке выбирается элемент с наименьшим значением, затем среди этих чисел выбирается наибольшее. Указать индексы этого элемента.
Вариант 17	
1	Дана строка *s*. Известно, что среди символов данной строки имеется хотя бы один пробел и что первый символ строки *s* отличен от пробела. Преобразовать строку *s*, заменив все малые буквы латинского алфавита, предшествующие первому пробелу, одноименными большими.
2	Даны две строки s_1 и s_2, имеющие одинаковую длину. Образовать строку *s*, в которой должны чередоваться символы строк s_1 и s_2.
3	В одномерном числовом массиве поменять местами элементы с четными и нечетными индексами, если оба они кратны минимальному, и заменить эти элементы их индексами в обратном случае.
4	Дан одномерный числовой массив, все элементы которого различны. Если сумма всех элементов, расположенных между максимальным и минимальным элементами массива, меньше 10, то вычислить произведение ненулевых элементов одномерного числового массива; в противном случае найти количество элементов массива, кратных минимальному, и их произведение.
5	Дана действительная квадратная матрица четвертого порядка. Получить целочисленную квадратную матрицу того же порядка, в которой элемент равен единице, если соответствующий ему элемент исходной матрицы больше элемента, расположенного в его строке на главной диагонали, и равен нулю в противном случае.
6	В данной действительной квадратной матрице порядка n найти сумму

	элементов строки, в которой расположен элемент с наименьшим значением. Предполагается, что такой элемент единственный.
Вариант 18	
1	Дана строка *s*, состоящая из слов (последовательностей символов, не содержащих пробелов внутри себя), разделенных между собой одним или несколькими пробелами. Если количество слов в строке *s* четно, то поменять местами ее средние слова. В противном случае на место среднего слова строки *s* поместить пробелы.
2	Дана строка *s*. Преобразовать ее, заменив в ней каждую точку многоточием (то есть тремя точками), а все восклицательные знаки вопросительными.
3	Дан одномерный числовой массив, все элементы которого различны. Подсчитать в нем количество четных элементов, расположенных между наибольшим и наименьшим элементами массива.
4	Дан одномерный числовой массив, все элементы которого различны. Найти наименьшее положительное значение элемента массива и его номер. Если этот номер окажется больше 5, то заменить все элементы, расположенные между наибольшим и наименьшим элементами массива, их квадратами, иначе - кубами.
5	Даны натуральное число m и целочисленная квадратная матрица порядка m. Строку с номером i матрицы назовем отмеченной, если $a_i > 0$, и неотмеченной в противном случае. Посчитать число отрицательных элементов, расположенных в отмеченных строках.
6	Дан двумерный целочисленный массив размером m×n. Некоторый элемент этого массива назовем «седловой точкой», если он является одновременно наименьшим в своей строке и наибольшим в своем столбце. Напечатать номера строки и столбца какой-нибудь «седловой точки» и напечатать число 0, если такой точки нет.
Вариант 19	
1	Дана строка *s*. Преобразовать эту строку так, чтобы ее символы шли в обратном порядке по отношению к их исходному расположению.
2	Дана строка *s*, состоящая из слов (последовательностей символов, не содержащих пробелов внутри себя), разделенных между собой одним или несколькими пробелами. Определить, сколько слов состоит из одинаковых символов, например: ааааа, еее.
3	Дан одномерный числовой массив, все элементы которого различны. Посчитать в нем количество соседств из двух положительных чисел. Если это количество кратно 3, то вывести на экран все элементы расположенные до максимального, иначе — после максимального.
4	Дан одномерный числовой массив, все элементы которого различны. Посчитать сумму всех неотрицательных элементов массива с четными индексами. Если эта сумма окажется больше 7, то заменить

	наибольший и наименьший элементы массива их произведением, иначе — поменять их местами.
5	Дана действительная квадратная матрица порядка n. Построить последовательность действительных чисел $a_1,\ a_2,\ \ldots a_n$ по правилу: если в i-той строке матрицы элемент, принадлежащий главной диагонали, отрицателен, то a_i равно сумме элементов i-той строки, предшествующих первому отрицательному элементу; в противном случае a_i равно сумме последних элементов i-той строки, начиная с первого по порядку неотрицательного элемента.
6	Даны натуральное число m и целочисленная квадратная матрица порядка m. Строку с номером i матрицы назовем отмеченной, если $a_i > 0$, и неотмеченной в противном случае. Посчитать число отрицательных элементов, расположенных в отмеченных строках.
Вариант 20	
1	Определить долю пробелов в данной строке *s*.
2	Даны две строки s_1 и s_2, имеющие различную длину. Посчитать в них количество попарно одинаковых символов.
3	Дан упорядоченный массив $a_1,\ a_2,\ \ldots a_n$. Известно, что число x принадлежит отрезку числовой оси, вмещающему заданный массив. Определить номер k, для которого $a_{k-1} < x < a_k$. Если k окажется кратным 3, то вывести на экран элементы массива в обратном порядке.
4	Дан одномерный числовой массив, все элементы которого различны. Посчитать сумму кубов элементов с нечетными индексами, расположенных до наибольшего элемента массива, и произведение индексов всех элементов, которые по числовому значению превосходят минимальный, но не превосходят максимальный элемент массива.
5	Дана действительная квадратная матрица четвертого порядка. В строках с отрицательным элементом на побочной диагонали найти сумму всех элементов.
6	Дана действительная квадратная матрица порядка n. Построить последовательность действительных чисел $a_1,\ a_2,\ \ldots a_n$ по правилу: если в i-той строке матрицы элемент, принадлежащий главной диагонали, отрицателен, то a_i равно сумме элементов i-той строки, предшествующих первому отрицательному элементу; в противном случае a_i равно сумме последних элементов i-той строки, начиная с первого по порядку неотрицательного элемента.
Вариант 21	
1	Дана строка *s*, состоящая из слов (последовательностей символов, не содержащих пробелов внутри себя), разделенных между собой одним

	или несколькими пробелами. Определить число слов, в которых последняя буква в слове не повторяется.
2	Дана строка *s*. Определить, является ли она «палиндромом». Если нет, то выяснить, станет ли строка *s* «палиндромом» после удаления из нее всех пробелов.
3	Дан одномерный числовой массив. Определить, сколько в нем имеется пар совпадающих по величине соседних чисел; вывести на экран их индексы.
4	Дан одномерный числовой массив. Добавить к каждому элементу массива, кроме первого, значение предыдущего. После такого преобразования вычислить среднее арифметического наибольшего и наименьшего элементов массива.
5	Даны натуральное число m и целочисленная квадратная матрица порядка m. Строку с номером i матрицы назовем отмеченной, если $a_i > 0$, и неотмеченной в противном случае. Посчитать число отрицательных элементов, расположенных в отмеченных строках.
6	Дана действительная квадратная матрица порядка n. Построить последовательность действительных чисел a_1, a_2, $\ldots a_n$ по правилу: если в i-той строке матрицы элемент, принадлежащий главной диагонали, отрицателен, то a_i равно сумме элементов i-той строки, предшествующих первому отрицательному элементу; в противном случае a_i равно сумме последних элементов i-той строки, начиная с первого по порядку неотрицательного элемента.
Вариант 22	
1	Дана строка s_1. Образовать строку s_2, включив в нее символы строки s_1, расположенные на нечетных позициях, кроме пробелов и знаков препинания.
2	Дана строка *s*, состоящая из слов (последовательностей символов, не содержащих пробелов внутри себя), разделенных между собой одним или несколькими пробелами. Определить, повторяется ли в тексте первое слово.
3	Найти наибольший и наименьший элементы одномерного числового массива, а также подсчитать количество элементов, кратных наименьшему элементу массива.
4	Дан одномерный числовой массив, все элементы которого различны. Заменить в нем четные элементы, расположенные после максимального, их квадратами, а нечетные элементы, расположенные после максимального — их кубами.
5	Дана действительная квадратная матрица четвертого порядка. Получить целочисленную квадратную матрицу того же порядка, в которой элемент равен единице, если соответствующий ему элемент исходной матрицы больше элемента, расположенного в его строке на

	главной диагонали, и равен нулю в противном случае.
6	Дан двумерный целочисленный массив размером m×n. Некоторый элемент этого массива назовем «седловой точкой», если он является одновременно наименьшим в своей строке и наибольшим в своем столбце. Напечатать номера строки и столбца какой-нибудь «седловой точки» и напечатать число 0, если такой точки нет.
Вариант 23	
1	Дана строка *s*. Вывести ее на экран, подчеркивая (ставя минусы в соответствующих позициях следующей строки) все входящие в него цифры.
2	Дана строка s. Исключить из нее все лишние пробелы, то есть вместо каждой группы пробелов оставить только один. При исключении пробелов текст сдвигается влево.
3	Вычислить произведение всех ненулевых элементов одномерного числового массива с нечетными индексами. Заменить все элементы массива, равные нулю, отношением их индекса к значению максимального элемента.
4	Заменить в одномерном числовом массиве элементы, большие числа *M*, на число *a*, введенное с клавиатуры, а меньшие *M* — отношением $\frac{a}{i}$, где *i* — индекс элемента.
5	Дана действительная матрица размера m×n. Найти среднее арифметическое наибольшего и наименьшего элементов этой матрицы из лежащих на ее побочной диагонали.
6	Даны натуральные числа n, m, действительная матрица размера m×n. Найти среднее арифметическое каждого из столбцов, имеющих четные номера.
Вариант 24	
1	Дана строка *s*, состоящая из слов (последовательностей символов, не содержащих пробелов внутри себя), разделенных между собой одним или несколькими пробелами. Найти номер самого короткого слова. Если таких слов несколько, то указать все их номера.
2	Даны две строки s_1 и s_2, имеющие различную длину. Посчитать в них количество попарно различных символов.
3	В одномерном числовом массиве заменить отрицательные элементы их квадратами, а положительные разделить на числовое значение минимального элемента.
4	Дан одномерный числовой массив. Определить количество элементов этого массива, которые имеют четные индексы и являются кратными 3, а также вычислить сумму и произведение элементов, кратных либо первому, либо последнему элементам массива
5	В данной действительной квадратной матрице порядка n найти сумму элементов строки, в которой расположен элемент с наименьшим

	значением. Предполагается, что такой элемент единственный.
6	Дана действительная квадратная матрица порядка n. Найти сумму элементов матрицы, расположенных в строках с отрицательным элементом на главной диагонали.
Вариант 25	
1	Дана строка *s*, состоящая из слов (последовательностей символов, не содержащих пробелов внутри себя), разделенных между собой одним или несколькими пробелами. Определить, сколько раз стоящие рядом два слова начинаются на одну и ту же букву.
2	Дана строка *s*. Определить в ней долю символов, являющихся буквами латинского алфавита.
3	Определить в одномерном числовом массиве число соседств из двух чисел, сумма которых равна 5. Вычислить произведение всех элементов, удовлетворяющих этому условию.
4	Найти суммы и произведения отрицательных и положительных элементов одномерного числового массива. Подсчитать, сколько имеется в массиве элементов, равных нулю.
5	Дана действительная квадратная матрица порядка n. Переменной y присвоить значение, равное скалярному произведению строки и столбца (рассматриваемых как векторы), на пересечении которых находится наименьший элемент матрицы (в предположении, что такой элемент единственный).
6	Дана действительная квадратная матрица порядка n. Рассмотрим те ее элементы, которые расположены в строках, начинающихся с отрицательного элемента. Найти суммы тех из них, которые находятся соответственно ниже, выше и на побочной диагонали.

3 МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ К ВЫПОЛНЕНИЮ ЛАБОРАТОРНЫХ РАБОТ

Онлайн-компилятор – это средство, преобразующее исходный текст программы, написанный на каком-либо языке программирования, в машинный код, называется компилятором. Онлайн-компиляторы применяются при необходимости запустить или быстро проверить какой-то код без запуска тяжелых десктопных IDE или прикладных компиляторов.

Лабораторные работы данного блока рекомендуется выполнять в онлайн-компиляторах, поскольку они не требуют установки или настройки, сохраняя при этом все необходимые для данных работ функциональные возможности.

Перед выполнением каждой лабораторной работы необходимо ознакомиться с кратким теоретическим материалом, представленным в п.1 настоящих методических указаний.

Отчет по лабораторной работе оформляется в соответствии с принятыми в КГПИ ФГБОУ ВО «КемГУ» Правилами оформления студенческих работ[1] виде документа формата *.docx, он должен содержать:

1. титульный лист с указанием ФИО студента и названием работы,
2. основную часть, включающую:
   - цель работы,
   - формулировку задания (из раздела 2),
   - краткое описание этапов выполнения работы по выполнению данного задания (рассуждения, схемы и т.д.),
   - программный код на языке Go, полученный в результате выполнения задания,

снимок окна экрана, полученный после запуска программы.

[1] Правила оформления учебных работ студентов : учебно-методическое пособие / И.А. Жибинова, А.Е. Аракелян, О.В. Соколова, Ю.Н. Соина-Кутищева. – Новокузнецк : НФИ КемГУ, 2018. – 124 с. – Текст : непосредственный.

3.1 Лабораторная работа №1. Разработка простых алгоритмов на языке GO

Цель работы: научиться применять язык программирования Go для решения задач профессиональной деятельности; научиться применять онлайн-компиляторы для разработки и отладки алгоритмов на языке Go.

Ход работы:

1. Открыть сайт онлайн-компилятора языка Go: https://www.onlinegdb.com/online_go_compiler.
2. Изучить краткий теоретический материал к лабораторной работе №1. Поочередно скопировать и запустить в онлайн-компиляторе каждый пример из данного материала, изучить особенности построения программ, объявления переменных, задания конструкций «ветвление» и «цикл». Исследовать корректность полученных в результате запуска программы данных.
3. Выполнить индивидуальные задания для лабораторной работы.
4. Составить отчет.

Пример выполнения индивидуального задания лабораторной работы №1:

Задание: *Дана целочисленная квадратная матрица четвертого порядка. Заменить нулями все неотрицательные элементы этой матрицы, находящиеся на ее главной диагонали.*

Решение*: Зададим матрицу размерности 4×4 и заполним ее случайными значениями таким образом, чтобы некоторые элементы матрицы были отрицательными. После этого выведем эту исходную матрицу на экран.*

Затем необходимо проверить каждый элемент матрицы на выполнение двух условий: элемент находится на главной диагонали и элемент является неотрицательным. Если данные условия для элемента выполняются, то элемент необходимо обнулить. После этого выведем новую матрицу на экран.

В результате выполнения задания был получен алгоритм:

```
package main
```

```
import ("fmt"
"math/rand")
func main() {
    var b[4][4] rune
    fmt.Println(" ")
    for i:=0;i<=len(b)-1; i++ {
        for j:=0;j<=len(b)-1; j++{
                b[i][j]=rune((rand.Intn(10)))-5
                fmt.Print(b[i][j], " ")
            }
            fmt.Println(" ")
        }
    fmt.Println(" ")
    for i:=0;i<=len(b)-1; i++ {
        for j:=0;j<=len(b)-1; j++{
            if i==j && b[i][j]>=0{
                b[i][j]=0
            }
            fmt.Print(b[i][j], " ")
        }
        fmt.Println(" ")
    }
}
```

На рисунке 1 представлен снимок окна экрана компилятора языка Go после запуска программы.

Рисунок 1. Окно компилятора Go после выполнения программы

3.2 Лабораторная работа №2. Разработка простых алгоритмов на языке Julia

Цель работы: научиться применять язык программирования Julia для решения задач профессиональной деятельности; научиться применять онлайн-компиляторы для разработки и отладки алгоритмов на языке Julia.

Ход работы:

1. Открыть сайт онлайн-компилятора языка Julia:

https://replit.com/languages/julia.

2. Изучить краткий теоретический материал к лабораторной работе №2. Поочередно скопировать и запустить в онлайн-компиляторе каждый пример из данного материала, изучить особенности построения программ, объявления переменных, задания конструкций «ветвление» и «цикл». Исследовать корректность полученных в результате запуска программы данных.

3. Выполнить индивидуальные задания для лабораторной работы.

4. Составить отчет.

Пример выполнения индивидуального задания лабораторной работы №2:

Задание: *Дана целочисленная квадратная матрица четвертого порядка. Заменить нулями все неотрицательные элементы этой матрицы, находящиеся на ее побочной диагонали.*

Решение*: Зададим матрицу размерности 4×4 и заполним ее случайными значениями таким образом, чтобы некоторые элементы матрицы были отрицательными. После этого выведем эту исходную матрицу на экран.*

Затем необходимо проверить каждый элемент матрицы на выполнение двух условий: элемент находится на побочной диагонали и элемент является неотрицательным. Если данные условия для элемента выполняются, то элемент необходимо обнулить. После этого выведем новую матрицу на экран.

Данный пример похож на тот, который был рассмотрен в примере к лабораторной работе №1, разница лишь в том, что обнуляются элементы не на главной диагонали, а на побочной. Для элементов, лежащих на побочно диагонали характерно следующее: индекс по строке i *(начинается с 1), а по столбцу –* n*-*i*+1 (где* n *– число столбцов в матрице).*

В результате выполнения задания был получен алгоритм:

```
a=Matrix{Int64}(undef,4,4)
a=rand(-5:5,4,4)
for i in 1:4
     for j in 1:4
          print("$(a[i,j]) ")
```

```
     end
println()
end
println()
for i in 1:4
     for j in 1:4
          if i==j && a[i,j]>=0
               a[i,j]=0
               end
     end
end
for i in 1:4
     for j in 1:4
          print("$(a[i,j]) ")
     end
println()
end
println()
```

На рисунке 2 представлен снимок окна экрана компилятора языка Julia после запуска программы.

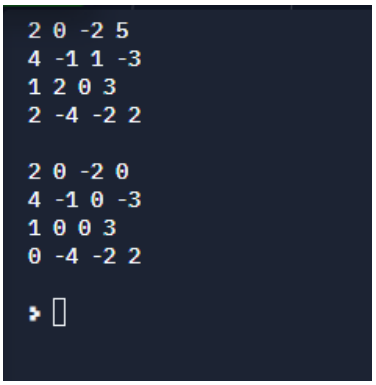


Рисунок 2. Окно компилятора Julia после выполнения программы

3.3 Лабораторная работа №3. Разработка простых алгоритмов на языке Kotlin

Цель работы: научиться применять язык программирования Kotlin для решения задач профессиональной деятельности; научиться применять онлайн-компиляторы для разработки и отладки алгоритмов на языке Kotlin.

Ход работы:

1. Открыть сайт онлайн-компилятора языка Kotlin: https://developer.android.com/training/kotlinplayground.

2. Изучить краткий теоретический материал к лабораторной работе №3. Поочередно скопировать и запустить в онлайн-компиляторе каждый

пример из данного материала, изучить особенности построения программ, объявления переменных, задания конструкций «ветвление» и «цикл». Исследовать корректность полученных в результате запуска программы данных.

3. Выполнить индивидуальные задания для лабораторной работы.
4. Составить отчет.

Пример выполнения индивидуального задания лабораторной работы №3:

Задание: *Дана целочисленная квадратная матрица четвертого порядка. Поменять местами столбец с максимальным элементом и столбец с минимальным элементом.*

Решение*: Зададим матрицу размерности 4×4 и заполним ее случайными значениями. После этого выведем эту исходную матрицу на экран.*

Затем необходимо взять за минимум и максимум самый первый элемент матрицы (первый элемент первой строки) для того чтобы потом с ним сравнивать остальные элементы. Логика сравнения заключается в следующем: как только будет найден элемент, который меньше минимального, его значение записывается в минимумом, а программа «запоминает» номер столбца, в котором содержится этот элемент. С максимальным элементом выполняются аналогичные действия.

После того, как мы «пройдем» по всем элементам матрицы, найдем минимальный и максимальный элементы, а также запомним номера их столбцов, можно будет поменять данные столбцы местами, для чего нужно будет ввести дополнительную переменную temp.

В результате выполнения задания был получен алгоритм:

```
import kotlin.random.Random
fun main() {
     val table: Array<Array<Int>> = Array(4,{Array(4,{0})})
     val n=table.size;
     val m=table[0].size;
     for (i in 0 until n){
```

```
            for(j in 0 until m){
                table[i][j] = Random.nextInt(100)-50
            }
        }
    for(row in table){
            for(cell in row){
                print("$cell \t")
            }
            println()
        }
    println()
    var max=table[0][0];
    var maxColumn=0;
    var min=table[0][0];
    var minColumn=0;
    for (i in 0 until n){
            for(j in 0 until m){
                if (table[i][j]<min){
                    min=table[i][j];
                    minColumn=j;
                }
                if (table[i][j]>max){
                    max=table[i][j];
                    maxColumn=j;
                }
            }
        }
    var temp=0;
    for (i in 0 until n){
        temp=table[i][minColumn];
        table[i][minColumn]=table[i][maxColumn];
        table[i][maxColumn]=temp;
    }
    for(row in table){
            for(cell in row){
                print("$cell \t")
            }
            println()
    }
}
```

На рисунке 3 представлен снимок окна экрана компилятора языка Kotlin после запуска программы.

```
14      0       21      -44
36      16      4       33
27      -37     -47     46
16      21      43      -45

14      0       -44     21
36      16      33      4
27      -37     46      -47
16      21      -45     43
```

Рисунок 3. Окно компилятора Kotlin после выполнения программы

3.4 Лабораторная работа №4. Разработка простых алгоритмов на языке Swift

Цель работы: научиться применять язык программирования Swift для решения задач профессиональной деятельности; научиться применять онлайн-компиляторы для разработки и отладки алгоритмов на языке Swift.

Ход работы:

1. Открыть сайт онлайн-компилятора языка Swift: https://replit.com/languages/swift.
2. Изучить краткий теоретический материал к лабораторной работе №4. Поочередно скопировать и запустить в онлайн-компиляторе каждый пример из данного материала, изучить особенности построения программ, объявления переменных, задания конструкций «ветвление» и «цикл». Исследовать корректность полученных в результате запуска программы данных.
3. Выполнить индивидуальные задания для лабораторной работы.
4. Составить отчет.

Пример выполнения индивидуального задания лабораторной работы №4:

Задание: *Дана целочисленная квадратная матрица четвертого порядка. Поменять местами ее главную и побочную диагонали.*

Решение*: Зададим матрицу размерности 4×4 и заполним ее случайными значениями. После этого выведем данную матрицу на экран.*

Пройдем по каждой строке матрицы, меняя элемент, принадлежащий главной диагонали с элементом, принадлежащим побочной диагонали. Элемент принадлежит главной диагонали в том случае, если его индексы по строке и по столбцу равны. Побочной же диагонали принадлежит элемент, у которого индекс строки – i, а столбца – n-i-1, где n- порядок матрицы. Как и в примере для предыдущей лабораторной работы, здесь для того чтобы поменять местами два элемента, нужно ввести промежуточную

переменную temp.

В результате выполнения задания был получен алгоритм:

```
var arr = [[Int]](repeating: [Int](repeating: 0, count: 5), count: 5)
var str=""
for i in 0..<arr.count{
     str=""
     for j in 0..<arr[0].count{
          arr[i][j]=Int.random(in: 1..<10)
          str=str+" "+String(arr[i][j])
     }
     print(str)
}
print()
var temp=0
for i in 0..<arr.count{
          temp=arr[i][i]
          arr[i][i]=arr[i][arr[0].count-i-1]
          arr[i][arr[0].count-i-1]=temp
}
for i in 0..<arr.count{
     str=""
     for j in 0..<arr[0].count{
          str=str+" "+String(arr[i][j])
     }
     print(str)
}
print()
```

На рисунке 4 представлен снимок окна экрана компилятора языка Swift после запуска программы.

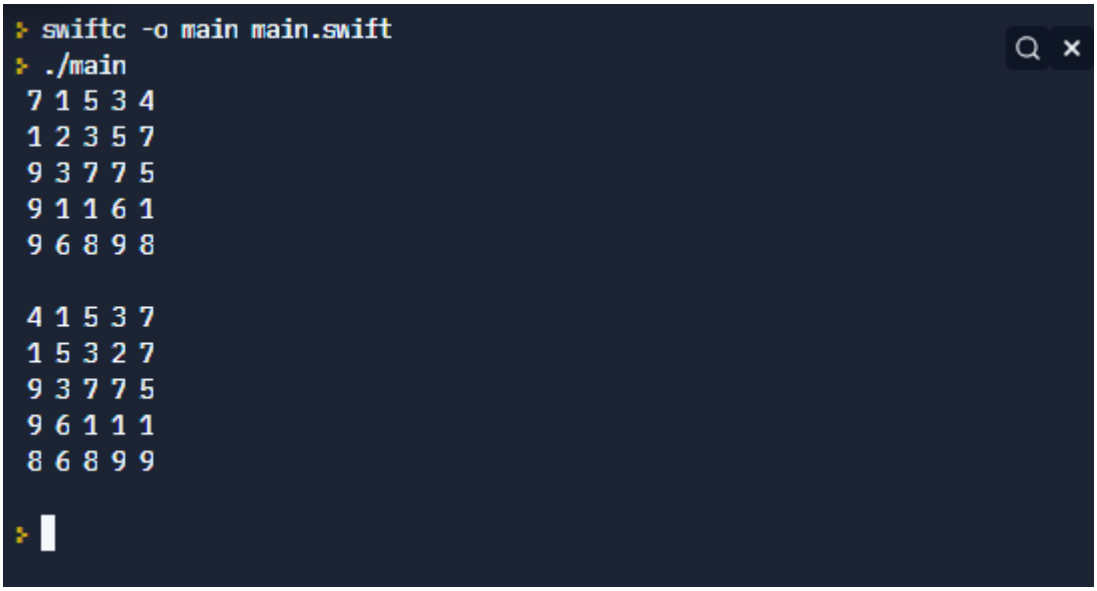


Рисунок 4. Окно компилятора Swift после выполнения программы

4 ОСОБЕННОСТИ ОЦЕНИВАНИЯ ЛАБОРАТОРНОЙ РАБОТЫ В БАЛЛЬНО-РЕЙТИНГОВОЙ СИСТЕМЕ

Лабораторные работы являются промежуточной формой контроля знаний студентов и представляют собой письменное выполнение определенных заданий. Они предназначены для проверки знаний студентов по учебной дисциплине «Языки и методы программирования», а также служат для закрепления полученных знаний, умений и навыков. В лабораторной работе студентам предлагаются задачи, сформулированные на основании материала, изложенного в лекциях или самостоятельно изученного студентами.

Перед тем как приступить к выполнению лабораторной работы, студентам следует ознакомиться с теоретическим материалом и разобраться с разобранными в нем типовыми задачами.

Система оценивания заданий лабораторных работ представлена в таблице 3.1.

Таблица 3.1. Оценивание лабораторных работ в БРС

Критерии оценивания задания	Количество баллов
Выполнение заданий лабораторной работы: логично и последовательно выполнены все шаги решения, рассуждения имеют четкое обоснование, получен верный ответ.	2
Ответы на вопросы преподавателя по лабораторной работе	1
Максимальное количество баллов за лабораторную работу	3

5 ОБРАЗЕЦ ТЕСТОВЫХ ЗАДАНИЙ ПО РАЗДЕЛУ «ПРОЦЕСС СОЗДАНИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ»

1. Выберите один верный ответ. Основной нормативный документ, регламентирующий состав процессов ЖЦ ПО – это ...

а) ISO/IEC 12207: 1995 «Information Technology -Software Life Cycle Processes»;

б) ISO/IEC 12210-1: 1998 «Cranes - Anchoring devices for in-service and out-of-service conditions»;

в) ISO/IEC 12205: 1996 «Petroleum products»;

г) ISO/IEC 12257: 1995 «Information Technology».

2. Выберите один верный ответ. К организационным процессам жизненного цикла ПО не относится ...

а) управление;

б) усовершенствование;

в) обучение;

г) аттестация.

3. Выберите один верный ответ. Приобретение ПО – это ... жизненного цикла ПО

а) основной процесс;

б) основа;

в) основа эффективности;

г) результат.

4. Выберите один верный ответ. Требованием не является ...

а) Архитектура программы должна основываться на паттерне проектирования «Фабрика»;

б) Программа должна обеспечивать три режима функционирования: штатный, режим регламентного обслуживания и режим восстановления системы после отказа или сбоя;

в) В качестве мер по повышению надежности программы должно быть предусмотрено резервирование наиболее важных информационных ресурсов;

г) Разработка программного приложения должна осуществляться на языке Python.

5. Студент получил задание: написать программу, которая заполняет квадратную матрицу случайными числами из множества Z. Данный программный код студент загрузил на образовательный портал как ответ на задание.

Проверьте данный код и укажите ошибки в выполнении задания (если они есть). Если задание выполнено правильно - напишите: "Ошибок нет".

```
package main
import "fmt"
func main() {
var b[4][4] rune
    for i:=0;i<=2; i++ {
    for j:=0;j<=2; j++{
    b[i][j]=rune((rand.Intn(10)))
    fmt.Print(b[i][j], " ")
    }
    fmt.Println(" ")
}
}
```

РЕКОМЕНДУЕМАЯ ЛИТЕРАТУРА

Основная учебная литература

1. Калентьев А.А. Новые технологии в программировании : учебное пособие / А.А. Калентьев, Д.В. Гарайс, А.Е. Горяинов – Томск : Эль Контент, 2014. – 176 с. – ISBN 978-5-4332-0185-9.- URL: http://biblioclub.ru/index.php?page=book_view_red&book_id=480503. - (дата обращения: 20.09.2021). – Текст : электронный.

Дополнительная литература

1. Мирошниченко, И.И. Языки и методы программирования: учебное пособие / И.И. Мирошниченко, Е.Г. Веретенникова, Н.Г. Савельева. – Ростов н/Д: Издательско-полиграфический комплекс Рост. гос. экон. ун-та (РИНХ), 2019. – 188 с. - ISBN 978-5-7972-2604-8. – URL: http://biblioclub.ru/index.php?page=book_view_red&book_id=567706. - (дата обращения: 20.09.2021). – Текст : электронный.

2. Android Studio : сайт. – 2020 - . - URL: https://developer.android.com/studio/intro (дата обращения: 20.09.2021). – Текст : электронный.

СОВРЕМЕННЫЕ ПРОФЕССИОНАЛЬНЫЕ БАЗЫ ДАННЫХ И СПРАВОЧНЫЕ СИСТЕМЫ

CITForum.ru : on-line библиотека свободно доступных материалов по информационным технологиям на русском языке : сайт. – 2001 – URL: http://citforum.ru (дата обращения: 20.09.2021). – Текст: электронный.

eLIBRARY.RU : научная электронная библиотека : сайт. – Москва, 2000 - . – URL: http://www.elibrary.ru (дата обращения: 20.09.2021). – Режим доступа: для зарегистрир. пользлвателей. – Текст: электронный.

Единое окно доступа к образовательным ресурсам : сайт. – Москва, 2005 - . – URL: http://window.edu.ru/ (дата обращения: 20.09.2021). –Текст: электронный.