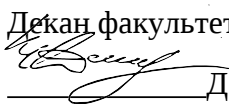
МИНОБРНАУКИ РОССИИ
Ярославский государственный университет им. П.Г. Демидова

Кафедра дискретного анализа

УТВЕРЖДАЮ

Декан факультета ИВТ


_____Д.Ю. Чалый

«_23_»____мая_____2023 г.

Рабочая программа дисциплины

«Алгоритмы и алгоритмические языки»

Направление подготовки

01.03.02 Прикладная математика и информатика

Профиль

«Искусственный интеллект»

Квалификация выпускника

Бакалавр

Форма обучения

очная

Программа рассмотрена на заседании кафедры	Программа одобрена НМК факультета ИВТ
от 11 апреля 2023 г.,	протокол № 6 от 28 апреля 2023 г.
протокол № 4	

Ярославль

1. Цели освоения дисциплины

2. Место дисциплины в структуре образовательной программы бакалавриата *(магистратуры, специалитета)*

Дисциплина «Алгоритмы и алгоритмические языки» согласно учебному плану входит в модуль «Алгоритмизация и программирование» и реализуется в 1-2 семестрах. Изучается на основе знаний, полученных при изучении дисциплин модулей «Аппаратное и программное обеспечение компьютера», «Современные цифровые технологии».

Результаты изучения дисциплины «» востребованы при освоении дисциплин модулей: «Машинное обучение и анализ данных» и «Искусственный интеллект», преддипломной практике и выпускной квалификационной работе.

3. Планируемые результаты обучения по дисциплине, соотнесенные с планируемыми результатами освоения образовательной программы бакалавриата *(магистратуры, специалитета)*

Процесс изучения дисциплины направлен на формирование следующих компетенций в соответствии с ФГОС ВО и приобретения следующих знаний, умений, навыков и (или) опыта деятельности:

Формируемая компетенция (код и формулировка)	**Индикатор достижения компетенции[1] (код и формулировка)**	**Перечень планируемых результатов обучения**
Общепрофессиональные компетенции		
ОПК-2. Способен использовать и адаптировать существующие математические методы и системы программирования для разработки и реализации алгоритмов решения прикладных задач.	ИОПК2.1 Осуществляет выбор и адаптацию математических методов и систем программирования для разработки и реализации алгоритмов решения прикладных задач.	Знать: Уметь: Владеть навыками:

[1] Для образовательных программ, реализуемым в соответствии с ФГОС ВО, актуализированными с учетом профессиональных стандартов

ОПК-5. Способен разрабатывать алгоритмы и компьютерные программы, пригодные для практического применения	ИОПК5.1 Обладает знаниями в области алгоритмизации и программирования. ИОПК5.2 Демонстрирует умение выбрать и обосновать выбор языка и среды программирования для разработки компьютерных программ. ИОПК5.3 Владеет навыками разработки алгоритмов и компьютерных программ, пригодных для практического применения.	Знать: Уметь: Владеть навыками:
ПК-1. Способен проектировать компьютерное программное обеспечение.	ИПК1.1 Разрабатывает и изменяет архитектуру компьютерного программного обеспечения. ИПК1.2 Проектирует структуру данных, баз данных и программных интерфейсов. ИПК1.3 Разрабатывает техническую документацию на компьютерное программное обеспечение с использованием существующих стандартов, оценивает и согласовывает сроки выполнения поставленных задач.	

ПК-9. Способен оценить качество разрабатываемого программного обеспечения путем проверки соответствия продукта заявленным требованиям, сбора и передачи информации о несоответствиях.	ИПК9.1 Демонстрирует умение определять и описывать тестовые случаи на основе требований, заявленных к программному обеспечению. ИПК9.2 Проводит тестирование по разработанным тестовым случаям, осуществляет сбор информации о несоответствиях заявленным требованиям. ИПК9.3 Анализирует результаты тестирования и дает оценку качеству разрабатываемого программного обеспечения.	

4. Объем, структура и содержание дисциплины

Общая трудоемкость дисциплины составляет _8__ зачетных единиц, __288__ акад. часов.

№ п/п	Темы (разделы) дисциплины, их содержание	Семестр	Виды учебных занятий, включая самостоятельную работу студентов, и их трудоемкость (в академических часах)						Формы текущего контроля успеваемости Форма промежуточной аттестации *(по семестрам)*
			Контактная работа					самостоятельная работа	
			лекции	практические	лабораторные	консультации	аттестационные испытания		
1.	Основы алгоритмизации и программирования		4					40	
	Понятие алгоритма, свойства и способы описания алгоритмов, базовые алгоритмические конструкции. Понятие программирования. Общая характеристика языков программирования и их классификация. Понятие синтаксиса и семантики языка программирования. Этапы разработки программного обеспечения.								
	в том числе с ЭО и ДОТ							*44*	
2.	Введение в программирование на языке Python		4	8	22			52	
	Особенности Python, версии языка, установка Python на компьютер, интерактивный режим Python, IPython; выбор редактора. Типы данных. Операции с целыми числами. Операции с вещественными								

	числами. Переменные. Стандартный ввод/вывод. Логические операции, операции сравнения. Условия:if; else; elif. Блоки, отступы. Циклwhile. Операторыbreak, continue. Циклfor. Строки и символы. Списки.								
	в том числе с ЭО и ДОТ							86	
3.	Программирование на языке Python		24	8	22			104	
	Функции. Словари. Интерпретатор: установка; запуск скрипта. Файловый ввод/вывод. Модули, подключение модулей. Установка дополнительных модулей. Библиотеки для анализа данных: NumPy; Matplotlib.								
	в том числе с ЭО и ДОТ							158	
	ИТОГО		32	16	44			196	Экзамен
	в том числе с ЭО и ДОТ							288	

4.1 Информация о реализации дисциплины в форме практической подготовки

Информация о разделах дисциплины и видах учебных занятий, реализуемых в форме практической подготовки

5. Образовательные технологии, в том числе технологии электронного обучения и дистанционные образовательные технологии, используемые при осуществлении образовательного процесса по дисциплине

6. Перечень лицензионного и (или) свободно распространяемого программного обеспечения, используемого при осуществлении образовательного процесса по дисциплине

7. Перечень современных профессиональных баз данных и информационных справочных систем, используемых при осуществлении образовательного процесса по дисциплине (при необходимости)

1. ОС семейства MicrosoftWindows
2. LibreOffice
3. MozillaFirefox
4. Microsoft Office (ауд.616)
5. Microsoft Office 365(онлайн)

8. Перечень основной и дополнительной учебной литературы, ресурсов информационно-телекоммуникационной сети «Интернет», рекомендуемых для освоения дисциплины

а) основная литература

1. Косицин, Д. Ю. Язык программирования Python : учебно-методическое пособие / Д. Ю. Косицин. — Минск : БГУ, 2019. — 136 с. — ISBN 978-985-566-746-0. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/180546
1. Шкаберина, Г. Ш. Программирование. Основы языка Python : учебное пособие / Г. Ш. Шкаберина, Н. Л. Резова. — Красноярск : СибГУ им. академика М. Ф. Решетнёва, 2018. — 92 с. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/147450

б) дополнительная литература

1. Борзунов, С. В. Алгебра и геометрия с примерами на Python / С. В. Борзунов, С. Д. Кургалин. — 3-е изд., стер. — Санкт-Петербург : Лань, 2022. — 444 с. — ISBN 978-5-8114-9980-9. — Текст : электронный // Лань :электронно-библиотечная система. — URL: https://e.lanbook.com/book/202154
2. Бузина, Т. С. Информатика : учебное пособие / Т. С. Бузина. — Иркутск : Иркутский ГАУ, 2020. — 161 с. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/183501
3. Забелин, А. А. Реализация алгоритмов вычислительной математики на языке Python : учебное пособие / А. А. Забелин. — Чита : ЗабГУ, 2020. — 130 с. — ISBN 978-5-9293-2575- — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/173632

в) ресурсы сети «Интернет»

1. Интерактивная доска.
2. http://www.ois.org.ua/spravka/mat/index.htm - электронная библиотека по математике.
3. http://eqworld.ipmnet.ru/ru/library.htm- учебно-образовательная физико-математическая библиотека
4. http://www.exponenta.ru/- образовательный математический сайт.

9. Материально-техническая база, необходимая для осуществления образовательного процесса по дисциплине

Приложение № 1 к рабочей программе дисциплины
« Алгоритмы и алгоритмические языки **»**
наименование дисциплины

Фонд оценочных средств
для проведения текущего контроля успеваемости
и промежуточной аттестации студентов
по дисциплине

1. Типовые контрольные задания и иные материалы,
используемые в процессе текущего контроля успеваемости

Образцы заданий для практического занятия по теме «Введение в программирование на языке Python»

1. Познакомьтесь с документацией разработчика, используемой Вами среды программирования. Выясните, каким образом осуществляется создание новой программы в данной среде программирования, как описываются переменные величины и константы, организуется ввод/вывод информации, строятся выражения с использованием математических функций.
2. Напишите программу, которая считывает три числа и выводит их сумму. Каждое число записано в отдельной строке.
3. N человек делят K яблок. Неделящийся остаток остается в корзине. Сколько яблок достанется каждому человеку? Сколько яблок останется в корзине? Напишите программу, которая получает на вход N и K, затем выводит искомое число яблок (два числа).
4. В то далёкое время, когда Паша ходил в школу, ему очень не нравилась формула Герона для вычисления площади треугольника, так как казалась слишком сложной. В один прекрасный момент Павел решил избавить всех школьников от страданий и написать и распространить по школам программу, вычисляющую площадь треугольника по трём сторонам. Одна проблема: так как эта формула не нравилась Павлу, он её не запомнил. Помогите ему завершить доброе дело и напишите программу, вычисляющую площадь треугольника по переданным длинам трёх его сторон по формуле Герона: $S = \sqrt{p(p-a)(p-b)(p-c)}$, где $p = \frac{a+b+c}{2}$ – полупериметр треугольника. На вход программе подаются целые числа, выводом программы должно являться вещественное число, соответствующее площади треугольника.
5. Напишите программу, принимающую на вход целое число, которая выводит True, если переданное значение попадает в интервал $(-15,12] \cup (14,17) \cup [19,+\infty)$ и False в противном случае (регистр символов имеет значение). Обратите внимание на разные скобки, используемые для обозначения интервалов.
6. Напишите простой калькулятор, который считывает с пользовательского ввода три строки: первое число, второе число и операцию, после чего применяет операцию к введённым числам («первое число» «операция» «второе число») и выводит результат на экран. Поддерживаемые операции: +, -, /, *, mod, pow, div, где mod — это взятие остатка от деления, pow — возведение в степень, div — целочисленное деление. Если выполняется деление и второе число равно 0, необходимо выводить строку "Деление

на 0!". Обратите внимание, что на вход программе приходят вещественные числа. **Образцы заданий для практического занятия по теме «Программирование на языке Python»**

1. Имеется файл с данными по успеваемости абитуриентов. Он представляет из себя набор строк, где в каждой строке записана следующая информация:

Фамилия;Оценка_по_математике;Оценка_по_информатике;Оценка_по_русскому_языку. Поля внутри строки разделены точкой с запятой, оценки — целые числа. Напишите программу, которая считывает файл с подобной структурой и для каждого абитуриента выводит его среднюю оценку по этим трём предметам на отдельной строке,

```
print('First;Second-1 Second-2;Third'.split(';'))
# ['First', 'Second-1 Second-2', 'Third']
```

Sample Input:

```
Петров;85;92;78
Сидоров;100;88;94
Иванов;58;72;85
```

Sample Output:

```
85.0
94.0
71.666666667
81.0 84.0 85.666666667
```

1. Напишите программу, которая подключает модуль math и, используя значение числа π из этого модуля, находит для переданного ей на стандартный ввод радиуса круга периметр этого круга и выводит его на стандартный вывод.

Sample Input:

```
10.0
```

Sample Output:

```
62.83185307179586
```

2. Напишите программу, которая запускается из консоли и печатает значения всех переданных аргументов на экран (имя скрипта выводить не нужно). Не изменяйте порядок аргументов при выводе. Для доступа к аргументам командной строки программы подключите модуль sys и используйте переменную argv из этого модуля. Пример работы программы:

```
> python3 my_solution.py arg1 arg2
arg1 arg2
```

3. Напишите программу, которая принимает на стандартный вход список игр футбольных команд с результатом матча и выводит на стандартный вывод сводную таблицу результатов всех матчей. За победу команде начисляется 3 очка, за поражение — 0, за ничью — 1. Формат ввода следующий: В первой строке указано целое число n — количество завершенных игр. После этого идет nn строк, в которых записаны результаты игры в следующем формате:

Первая_команда;Забито_первой_командой;Вторая_команда;Забито_второй_командой

Вывод программы необходимо оформить следующим образом: Команда: Всего_игр Побед Ничьих Поражений Всего_очков. Конкретный пример ввода-вывода приведён ниже. Порядок вывода команд произвольный.

Sample Input:

```
3
Зенит;3;Спартак;1
Спартак;1;ЦСКА;1
ЦСКА;0;Зенит;2
```

Sample Output:

```
Зенит:2 2 0 0 6
ЦСКА:2 0 1 1 1
Спартак:2 0 1 1 1
```

Образцы заданий для лабораторных работ по теме «Введение в программирование на языке Python»

1. Напишите программу, которая приветствует пользователя, выводя слово Hello, введенное имя и необходимые знаки препинания.
2. Жители страны Малевии часто экспериментируют с планировкой комнат. Комнаты бывают треугольные, прямоугольные и круглые. Чтобы быстро вычислять жилплощадь, требуется написать программу, на вход которой подаётся тип фигуры комнаты и соответствующие параметры, которая бы выводила площадь получившейся комнаты. Для числа π в стране Малевии используют значение 3.14. Формат ввода, который используют Малевийцы: треугольник (a, b, c), где a, b и c — длины сторон треугольника; прямоугольник (a, b), где a и b — длины сторон прямоугольника; круг (r), где r — радиус окружности.
3. Напишите программу, которая получает на вход три целых числа, по одному числу в строке, и выводит на консоль в три строки сначала максимальное, потом минимальное, после чего оставшееся число. На ввод могут подаваться и повторяющиеся числа.
4. В университете по офису передвигается робот. Недавно студенты из группы программистов написали для него программу, по которой робот, когда заходит в комнату, считает количество программистов в ней и произносит его вслух: «n программистов». Для того, чтобы это звучало правильно, для каждого n нужно использовать верное окончание слова. Напишите программу, считывающую с пользовательского ввода целое число n (неотрицательное), выводящее это число в консоль вместе с правильно изменённым словом «программист», чтобы робот мог нормально общаться с людьми, например: 1 программист, 2 программиста, 5 программистов. В комнате может быть очень много программистов. Проверьте, что ваша программа правильно обработает все случаи, как минимум до 1000 человек.
5. Паша очень любит кататься на общественном транспорте, а получая билет, сразу проверяет, счастливый ли ему попался. Билет считается счастливым, если сумма первых трех цифр совпадает с суммой последних трех цифр номера билета. Однако Паша очень плохо считает в уме, поэтому попросил вас написать программу, которая проверит равенство сумм и выведет «Счастливый», если суммы совпадают, и «Обычный», если суммы различны. На вход программе подаётся строка из шести цифр. Выводить нужно только слово «Счастливый» или «Обычный», с большой буквы.
6. Какое значение будет у переменной i после выполнения фрагмента программы?

```
i = 0
while i <= 10:
    i = i + 1
    if i > 7:
        i = i + 2
```

7. Сколько итераций цикла будет выполнено в этом фрагменте программы?

```
i = 0
while i <= 10:
    i = i + 1
    if i > 7:
        i = i + 2
```

8. Сколько всего знаков * будет выведено после исполнения фрагмента программы:

```
i = 0
while i < 5:
    print('*')
    if i % 2 == 0:
        print('**')
    if i > 2:
        print('***')
    i = i + 1
```

9. Напишите программу, которая считывает со стандартного ввода целые числа, по одному числу в строке, и после первого введенного нуля выводит сумму полученных на вход чисел.
10. В университете между информатиками и биологами устраивается соревнование. Победителям соревнования достанется большой и вкусный пирог. В команде биологов a человек, а в команде информатиков — b человек. Нужно заранее разрезать пирог таким образом, чтобы можно было раздать кусочки пирога любой команде, выигравшей соревнование, при этом каждому участнику этой команды должно достаться одинаковое число кусочков пирога. И так как не хочется резать пирог на слишком мелкие кусочки, нужно найти минимальное подходящее число. Напишите программу, которая помогает найти это число. Программа должна считывать размеры команд (два положительных целых числа a и b, каждое число вводится на отдельной строке) и выводить наименьшее число d, которое делится на оба этих числа без остатка.
11. Определите, какое значение будет иметь переменная i после выполнения следующего фрагмента программы:

```
i = 0
s = 0
while i < 10:
    i = i + 1
    s = s + i
    if s > 15:
        break
    i = i + 1
```

12. Определите, какое значение будет иметь переменная i после выполнения следующего фрагмента программы:

```
i = 0
s = 0
while i < 10:
    i = i + 1
    s = s + i
    if s > 15:
        continue
    i = i + 1
```

13. Напишите программу, которая считывает целые числа с консоли по одному числу в строке. Для каждого введённого числа нужно проверить: если число меньше 10, то пропускаем это число; если число больше 100, то прекращаем считывать числа; в остальных случаях следует вывести это число обратно на консоль в отдельной строке.
14. Когда Павел учился в школе, он запоминал таблицу умножения прямоугольными блоками. Для тренировок ему бы очень пригодилась программа, которая показывала бы блок таблицы умножения. Напишите программу, на вход которой даются четыре числа a, b, c и d, каждое в своей строке. Программа должна вывести фрагмент таблицы умножения для всех чисел отрезка [a;b] на все числа отрезка [c;d]. Числа a, b, c и d являются натуральными и не превосходят 10, a≤b, c≤d. Следуйте формату вывода из примера, для разделения элементов внутри строки используйте '\t' — символ табуляции. Заметьте, что левым столбцом и верхней строкой выводятся сами числа из заданных отрезков — заголовочные столбец и строка таблицы.

Sample Input 1:

```
7
10
5
6
```

Sample Output 1:

```
        5       6
7       35      42
8       40      48
9       45      54
10      50      60
```

15. Напишите программу, которая считывает с клавиатуры два числа a и b, считает и выводит на консоль среднее арифметическое всех чисел из отрезка [a;b], которые делятся на 33. В приведенном ниже примере среднее арифметическое считается для чисел на отрезке [−5;12]. Всего чисел, делящихся на 33, на этом отрезке 6: −3,0,3,6,9,12. Их среднее арифметическое равно 4.5. На вход программе подаются интервалы, внутри которых всегда есть хотя бы одно число, которое делится на 3.

Sample Input:

```
-5
12
```

Sample Output:

```
4.5
```

16. GC-состав является важной характеристикой геномных последовательностей и определяется как процентное соотношение суммы всех гуанинов и цитозинов к общему числу нуклеиновых оснований в геномной последовательности. Напишите программу, которая вычисляет процентное содержание символов G (гуанин) и C (цитозин) в введенной строке (программа не должна зависеть от регистра вводимых символов).

Например, в строке "acggtgttat" процентное содержание символов G и C равно $\frac{4}{10} \cdot 100 = 40.0$, где 4 - это количество символов G и C, а 10 - это длина строки.

Sample Input:

```
acggtgttat
```

Sample Output:

```
40.0
```

17. Есть строка s = "abcdefghijk".В поле ответа через пробел запишите строки (без кавычек), полученные в результате следующих операций:

```
# s = 'abcdefghijk'
s[3:6]
s[:6]
s[3:]
s[::-1]
s[-3:]
s[:-6]
s[-1:-10:-2]
```

Итоговый формат ответа должен выглядеть следующим образом:

```
abcd efg hijklmnop qrst uvw xy z
```

18. Узнав, что ДНК не является случайной строкой, студенты группы информатиков предложили использовать алгоритм сжатия, который сжимает повторяющиеся символы в строке. Кодирование осуществляется следующим образом: s = 'aaaabbcaa' преобразуется в 'a4b2c1a2', то есть группы одинаковых символов исходной строки заменяются на этот символ и количество его повторений в этой позиции строки. Напишите программу, которая считывает строку, кодирует её предложенным алгоритмом и выводит закодированную последовательность на стандартный вывод. Кодирование должно учитывать регистр символов.

Sample Input 1:

```
aaaabbcaa
```

Sample Output 1:

```
a4b2c1a2
```

Sample Input 2:

```
abc
```

Sample Output 2:

```
a1b1c1
```

19. Сколько элементов будет содержать список students после следующих операций?

```
students = ['Ivan', 'Masha', 'Sasha']
students += ['Olga']
students += 'Olga'
```

Введите в поле ответа одно число. Подумайте, почему так происходит.

20. Имеется программа, код которой указан ниже. Укажите, какие значения будут содержать списки в помеченных участках:

```
a = [1, 2, 3]
b = a
# значения списка b?

a[1] = 10
# значения списка b?

b[0] = 20
# значения списка a?

a = [5, 6]
# значения списка b?
```

Запишите значения списков в одну строку, разделяя списки точкой с запятой, а элементы внутри списка — пробелом, например:1 1 1; 2 2 2; 3 3 3; 4 4 4

21. Напишите программу, на вход которой подается одна строка с целыми числами. Программа должна вывести сумму этих чисел. Используйте метод split строки.

Sample Input:

```
4 -1 9 3
```

Sample Output:

```
15
```

22. Напишите программу, на вход которой подаётся список чисел одной строкой. Программа должна для каждого элемента этого списка вывести сумму двух его соседей. Для элементов списка, являющихся крайними, одним из соседей считается элемент, находящий на противоположном конце этого списка. Например, если на вход подаётся список "1 3 5 6 10", то на выход ожидается список "13 6 9 15 7" (без кавычек). Если на вход пришло только одно число, надо вывести его же. Вывод должен содержать одну строку с числами нового списка, разделёнными пробелом.

Sample Input 1:

```
1 3 5 6 10
```

Sample Output 1:

```
13 6 9 15 7
```

Sample Input 2:

```
10
```

Sample Output 2:

```
10
```

23. Напишите программу, которая принимает на вход список чисел в одной строке и выводит на экран в одну строку значения, которые повторяются в нём более одного раза. Для решения задачи может пригодиться метод sort списка. Выводимые числа не должны повторяться, порядок их вывода может быть произвольным.

Sample Input 1:

```
4 8 0 3 4 2 0 3
```

Sample Output 1:

```
0 3 4
```

Sample Input 2:

```
10
```

Sample Output 2:

Sample Input 3:

```
1 1 2 2 3 3
```

Sample Output 3:

```
1 2 3
```

24. Напишите программу, которая считывает с консоли числа (по одному в строке) до тех пор, пока сумма введённых чисел не будет равна 0 и сразу после этого выводит сумму квадратов всех считанных чисел. Гарантируется, что в какой-то момент сумма введённых чисел окажется равной 0, после этого считывание продолжать не нужно. В примере мы считываем числа 1, -3, 5, -6, -10, 13; в этот момент замечаем, что сумма этих чисел равна нулю и выводим сумму их квадратов, не обращая внимания на то, что остались ещё не прочитанные значения.

Sample Input:

```
1
-3
5
-6
-10
13
4
-8
```

Sample Output:

```
340
```

25. Напишите программу, которая выводит часть последовательности 1 2 2 3 3 3 4 4 4 4 5 5 5 5 5 ... (число повторяется столько раз, чему равно). На вход программе передаётся неотрицательное целое число n — столько элементов последовательности должна отобразить программа. На выходе ожидается последовательность чисел, записанных через пробел в одну строку. Например, если n = 7, то программа должна вывести 1 2 2 3

3 3 4. 26. Напишите программу, которая считывает список чисел lst из первой строки и число x из второй строки, которая выводит все позиции, на которых встречается число x в переданном списке lst. Позиции нумеруются с нуля, если число x не встречается в списке, вывести строку «Отсутствует» (без кавычек, с большой буквы). Позиции должны быть выведены в одну строку, по возрастанию абсолютного значения.

Sample Input 1:

```
5 8 2 7 8 8 2 4
8
```

Sample Output 1:

```
1 4 5
```

Sample Input 2:

```
5 8 2 7 8 8 2 4
10
```

Sample Output 2:

```
Отсутствует
```

27. Напишите программу, на вход которой подаётся прямоугольная матрица в виде последовательности строк, заканчивающихся строкой, содержащей только строку «end» (без кавычек). Программа должна вывести матрицу того же размера, у которой каждый элемент в позиции i, j равен сумме элементов первой матрицы на позициях (i-1, j), (i+1, j), (i, j-1), (i, j+1). У крайних символов соседний элемент находится с противоположной стороны матрицы. В случае одной строки/столбца элемент сам себе является соседом по соответствующему направлению.

Sample Input 1:

```
9 5 3
0 7 -1
-5 2 9
end
```

Sample Output 1:

```
3 21 22
10 6 19
20 16 -1
```

Sample Input 2:

```
1
end
```

Sample Output 2:

```
4
```

Образцы заданий для лабораторных работ по теме «Программирование на языке Python»

1. Есть функция f, которая определена следующим образом:

```
def f(n):
    return n * 10 + 5
```

Введите её в интерпретаторе и посчитайте, чему равно значение следующего выражения:

```
f(f(f(10)))
```

Разберитесь, почему получается именно такое значение.

2. Напишите функцию f(x), которая возвращает значение следующей функции, определённой на всей числовой прямой:

$$f(x) = \begin{cases} 1 - (x+2)^2, & \text{при } x \le -2 \\ -\frac{x}{2}, & \text{при } -2 < x \le 2 \\ (x-2)^2 + 1, & \text{при } 2 < x \end{cases}$$

3. Напишите функцию modify_list(l), которая принимает на вход список целых чисел, удаляет из него все нечётные значения, а чётные нацело делит на два. Функция не должна ничего возвращать, требуется только изменение переданного списка, например:

```
lst = [1, 2, 3, 4, 5, 6]
print(modify_list(lst))  # None
print(lst)               # [1, 2, 3]
modify_list(lst)
print(lst)               # [1]

lst = [10, 5, 8, 3]
modify_list(lst)
print(lst)               # [5, 4]
```

4. Напишите функцию update_dictionary(d, key, value), которая принимает на вход словарь d и два числа: key и value. Если ключ key есть в словаре d, то добавьте значение value в список, который хранится по этому ключу. Если ключа key нет в словаре, то нужно добавить значение в список по ключу 2·key. Если и ключа 2·key нет, то нужно добавить ключ 2·key в словарь и сопоставить ему список из переданного элемента

   [value].Требуется реализовать только эту функцию, кода вне неё не должно быть. Функция не должна вызывать внутри себя функции input и print. Пример работы функции:

```
d = {}
print(update_dictionary(d, 1, -1))  # None
print(d)                            # {2: [-1]}
update_dictionary(d, 2, -2)
print(d)                            # {2: [-1, -2]}
update_dictionary(d, 1, -3)
print(d)                            # {2: [-1, -2, -3]}
```

5. Когда Антон прочитал «Войну и мир», ему стало интересно, сколько слов и в каком количестве используется в этой книге. Помогите Антону написать упрощённую версию такой программы, которая сможет подсчитать слова, разделённые пробелом и вывести получившуюся статистику. Программа должна считывать одну строку со стандартного ввода и выводить для каждого уникального слова в этой строке число его повторений (без учёта регистра) в формате «слово количество» (см. пример вывода). Порядок вывода слов может быть произвольным, каждое уникальное слово должно выводиться только один раз.

Sample Input 1:

```
a aa abC aa ac abc bcd a
```

Sample Output 1:

```
ac 1
a 2
abc 2
bcd 1
aa 2
```

Sample Input 2:

```
a A a
```

Sample Output 2:

```
a 3
```

6. Имеется реализованная функция f(x), принимающая на вход целое число x, которая вычисляет некоторое целочисленное значение и возвращает его в качестве результата

работы. Функция вычисляется достаточно долго, ничего не выводит на экран, не пишет в файлы и зависит только от переданного аргумента x. Напишите программу, которой на вход в первой строке подаётся число n — количество значений x, для которых требуется узнать значение функции f(x), после чего сами эти n значений, каждое на отдельной строке. Программа должна после каждого введённого значения аргумента вывести соответствующие значения функции f на отдельной строке. Для ускорения вычисления необходимо сохранять уже вычисленные значения функции при известных аргументах.

Sample Input:

```
5
5
12
9
20
12
```

Sample Output:

```
11
41
47
61
41
```

7. Напишите программу, которая считывает из файла строку, соответствующую тексту, сжатому с помощью кодирования повторов, и производит обратную операцию, получая исходный текст. Запишите полученный текст в файл.

Sample Input:

```
a3b4c2e10b1
```

Sample Output:

```
aaabbbbcceeeeeeeeeeb
```

8. Напишите программу, которая считывает текст из файла (в файле может быть больше одной строки) и выводит самое частое слово в этом тексте и через пробел то, сколько раз оно встретилось. Если таких слов несколько, вывести лексикографически первое (можно использовать оператор < для строк). В качестве ответа укажите вывод программы. Слова, написанные в разных регистрах, считаются одинаковыми.

Sample Input:

```
abc a bCd bC AbC BC BCD bcd ABC
```

Sample Output:

```
abc 3
```

9. Простейшая система проверки орфографии основана на использовании списка известных слов. Каждое слово в проверяемом тексте ищется в этом списке и, если такое слово не найдено, оно помечается, как ошибочное. Напишите подобную систему. Через стандартный ввод подаётся следующая структура: первой строкой — количество d записей в списке известных слов, после передаётся d строк с одним словарным словом на строку, затем — количество l строк текста, после чего — l строк текста. Напишите программу, которая выводит слова из текста, которые не встречаются в словаре. Регистр слов не учитывается. Порядок вывода слов произвольный. Слова, не встречающиеся в словаре, не должны повторяться в выводе программы.

Sample Input:

```
3
a
bb
cCc
2
a bb aab aba ccc
c bb aaa
```

Sample Output:

```
aab
aba
c
aaa
```

10. Группа биологов в институте биоинформатики завела себе черепашку. После дрессировки черепашка научилась понимать и запоминать указания биологов следующего вида:

```
север 10
запад 20
юг 30
восток 40
```

где первое слово — это направление, в котором должна двигаться черепашка, а число после слова — это положительное расстояние в сантиметрах, которое должна пройти черепашка. Но команды даются быстро, а черепашка ползёт медленно, и программисты догадались, что можно написать программу, которая определит, куда в итоге биологи приведут черепашку. Для этого программисты просят вас написать программу, которая выведет точку, в которой окажется черепашка после всех команд. Для простоты они решили считать, что движение начинается в точке (0, 0), и движение на восток увеличивает первую координату, а на север — вторую. Программе подаётся на вход число команд *n*, которые нужно выполнить черепашке, после чего *n* строк с самими командами. Вывести нужно два числа в одну строку: первую и вторую координату конечной точки черепашки. Все координаты целочисленные.

Sample Input:

```
4
север 10
запад 20
юг 30
восток 40
```

Sample Output:

```
20 -20
```

1. Список вопросов и (или) заданий для проведения промежуточной аттестации

Образцы вопросов для защиты лабораторных работ по теме «Введение в программирование на языке Python»

1. Какие задачи решаются с помощью языка программирования Python?
2. Как установить Python на компьютер? Какие редакторы используются для программирования на языке Python?
3. Перечислите встроенные типы данных языка программирования Python.
4. Перечислите основные операции над целыми числами, определенными в Python.

5. Перечислите основные операции над вещественными числами, определенными в Python.
6. Как в Python определяются переменные?
7. Как осуществить стандартный ввод/вывод в Python?
8. Перечислите логические операции и операции сравнения, определенные в Python.
9. Как в Python записывается оператор if? Какие формы записи данного оператора существуют?
10. Для каких целей в Python используются блоки и отступы?
11. Приведите синтаксис оператора while.
12. С какой целью в циклах используются операторы break и continue?
13. Приведите синтаксис оператора for.
14. Как в Python задаются строки? Как задаются символы?
15. Перечислите основные операции над строками.
16. Как в Python задать список?
17. Какие операции выполняются над списками?

Образцы вопросов для защиты лабораторных работ по теме «Программирование на языке Python»

1. Как в Python задаются функции?
2. Как описать рекурсивную функцию?
3. Как вызвать функцию?
4. Для решения каких задач используются словари? Чем словари отличаются от списков? Как определить словарь?
5. Как установить интерпретатор Python на компьютер? Как запустить скрипт, написанный на языке Python?
6. Как в Python организуется файловый ввод/вывод.
7. Что такое модуль? Как подключить модуль в программу, написанную на языке Python?
8. Как установить дополнительные модули?
9. Какой функционал предоставляет программисту библиотека NumPy? Каким образом она подключается к программе?
10. Какой функционал предоставляет программисту библиотека Matplotlib? Каким образом она подключается к программе?

Самостоятельна работа по теме «Основы алгоритмизации и программирования»

Изучите теорию по теме, представьте краткий конспект:

1. Понятие алгоритма, свойства и способы описания алгоритмов.
2. Базовые алгоритмические конструкции.
3. Понятие программирования.
4. Общая характеристика языков программирования и их классификация.
5. Понятие синтаксиса и семантики языка программирования.

6. Этапы разработки программного обеспечения.

Самостоятельна работа по теме «Введение в программирование на языке Python»
Изучите теорию по теме, представьте краткий конспект:

1. Задачи, решаемые с помощью языка Python.
2. Особенности языка программирования Python. Версии Python.
3. Установка Python на компьютер. Интерактивный режим Python, IPython. Выбор редактора.
4. Типы данных Python.
5. Операции с целыми числами.
6. Операции с вещественными числами.
7. Переменные в Python.
8. Стандартный ввод/вывод в Python.
9. Логические операции, операции сравнения.
10. Условия: if; else; elif.
11. Блоки, отступы.
12. Цикл while.
13. Операторы break, continue.
14. Цикл for.
15. Строки и символы.
16. Списки.

Самостоятельна работа по теме «Программирование на языке Python»
Изучите теорию по теме, подготовьте краткий конспект:

1. Функции.
2. Словари.
3. Интерпретатор: установка; запуск скрипта.
4. Файловый ввод/вывод.
5. Модули, подключение модулей.
6. Установка дополнительных модулей.
7. Библиотеки для анализа данных: NumPy; Matplotlib.

Вопросы к экзамену:

1. Особенности языка программирования Python. Версии Python.
2. Установка Python на компьютер. Интерактивный режим Python, IPython. Выбор редактора.
3. Типы данных Python.
4. Операции с целыми числами в Python.
5. Операции с вещественными числами в Python.
6. Переменные в Python.
7. Стандартный ввод/вывод в Python.
8. Логические операции, операции сравнения в Python.

9. Условия: if; else; elif.
10. Блоки, отступы.
11. Цикл while.
12. Операторы break, continue.
13. Цикл for.
14. Строки и символы.
15. Списки.
16. Функции.
17. Словари.
18. Интерпретатор: установка; запуск скрипта.
19. Файловый ввод/вывод.
20. Модули, подключение модулей.
21. Установка дополнительных модулей.
22. Библиотека NumPy.
23. Библиотека Matplotlib.

Уровни оценки компетенций следующие: базовый – 55-69 баллов, повышенный – 70-100 баллов.

Преподаватель проводит систематический контроль знаний студентов, ориентируясь на перечень вопросов для проведения зачета/экзамена.

Критерии оценки лабораторных работ / практических занятий / самостоятельной работы студента (от 0 до 10 баллов):

- ***9-10 баллов*** выставляется студенту, если работа выполнена самостоятельно и полностью верно; представлен отчет, содержащий результаты выполнения заданий работы и ответы на контрольные вопросы; студент анализирует результаты, полученные в ходе выполнения работы, делает выводы.
- ***7-8 баллов*** выставляется студенту, если работа выполнена самостоятельно, в целом правильно, но имеются некоторые неточности в выполнении заданий или ответах на контрольные вопросы; представлен отчет, содержащий результаты выполнения заданий и ответы на контрольные вопросы; студент анализирует результаты, полученные в ходе выполнения работы, делает выводы.
- ***5-6 баллов*** выставляется студенту, если работа выполнена самостоятельно, в целом правильно, но имеются некоторые неточности в выполнении заданий или ответах на контрольные вопросы; представлен отчет, содержащий результаты выполнения заданий лабораторной работы и ответы на контрольные вопросы; студент испытывает затруднения при проведении анализа результатов, полученных в ходе выполнения лабораторной работы, и формулировке выводов.
- ***3-4 балла*** выставляется студенту, если студент не до конца справился с заданием, не совсем верно ответил на контрольные вопросы, однако оформил отчет по результатам работы.

- ***1-2 балла*** выставляется студенту, если студент не до конца справился с заданием, не совсем верно ответил на контрольные вопросы, не оформил отчет по результатам работы.
- ***0 баллов*** выставляется студенту, если студент не справился с заданием, неверно ответил на представленные вопросы.

Ответ на зачете/экзамене оценивается исходя из 40 баллов (максимум). Билет содержит теоретический вопрос и практическое задание, преподаватель может задавать дополнительные вопросы. Полный ответ на основной вопрос оценивается максимум в 20 баллов, предполагает свободное изложение (не чтение) всего необходимого материала, ответы студента на уточняющие вопросы, если они есть. Правильный ответ на дополнительный вопрос оценивается максимум в 5 баллов. Правильное выполнение практического задания оценивается в 20 баллов.

5.3 Шкала и критерии оценивания компетенций на различных этапах их формирования

Шкала оценивания компетенций:

Оценка в 100-балльной шкале	Оценка в 5-ти балльной шкале	Уровень сформированности компетенций
0-54 баллов	неудовлетворительно (не зачтено)	недостаточный
55-69 баллов	удовлетворительно (зачтено)	базовый
70-85 баллов	хорошо (зачтено)	повышенный
86-100 баллов	отлично (зачтено)	

Критерии оценивания компетенций:

Индикаторы достижения компетенций	**Критерии оценивания компетенций**		
	Недостаточный уровень	**Базовый уровень**	**Повышенный уровень**

ИОПК2.1Осуществляет выбор и адаптацию математических методов и систем программирования для разработки и реализации алгоритмов решения прикладных задач.	Испытывает серьезные затруднения при осуществлении выбора и адаптации математических методов и систем программирования для разработки и реализации алгоритмов решения прикладных задач.	Демонстрирует умение осуществлять выбор и адаптацию математических методов и систем программирования для разработки и реализации алгоритмов решения прикладных задач.	Самостоятельно и грамотно осуществляет выбор и адаптацию математических методов и систем программирования для разработки и реализации алгоритмов решения прикладных задач, в том числе в новой или нестандартной ситуации.
ИОПК5.1 Обладает знаниями в области алгоритмизации и программирования.	Не обладает знаниями в области алгоритмизации и программирования.	Демонстрирует знания в области алгоритмизации и программирования.	Демонстрирует глубокие знания в области алгоритмизации и программирования.
ИОПК5.2 Демонстрирует умение выбрать и обосновать выбор языка и среды программирования для разработки компьютерных программ.	Не умеет выбрать и обосновать выбор языка и среды программирования для разработки компьютерных программ.	Умеет выбирать и обосновать выбор языка и среды программирования для разработки компьютерных программ в стандартной	Умеет выбирать и обосновать выбор языка и среды программирования для разработки компьютерных программ, в том числе в новой или нестандартной

Индикаторы достижения компетенций	**Критерии оценивания компетенций**		
	Недостаточный уровень	**Базовый уровень**	**Повышенный уровень**
		ситуации.	ситуации.
ИОПК5.3 Владеет навыками разработки алгоритмов и компьютерных программ, пригодных для практического применения.	Не владеет навыками разработки алгоритмов и компьютерных программ, пригодных для практического применения.	Демонстрирует владение навыками разработки алгоритмов и компьютерных программ, пригодных для практического применения.	Полностью самостоятельно и верно разрабатывает алгоритмы и компьютерные программы, пригодные для практического применения.

ИПК1.1 Разрабатывает и изменяет архитектуру компьютерного программного обеспечения.	Не знает принципы построения и виды архитектуры компьютерного программного обеспечения, типовые решения, библиотеки программных модулей, шаблоны, классы объектов, используемые при разработке компьютерного программного обеспечения, методы и средства проектирования компьютерного программного обеспечения. Не умеет использовать существующие типовые решения и шаблоны проектирования компьютерного программного обеспечения, применять методы и средства проектирования компьютерного программного обеспечения.	Знает принципы построения и виды архитектуры компьютерного программного обеспечения, типовые решения, библиотеки программных модулей, шаблоны, классы объектов, используемые при разработке компьютерного программного обеспечения, методы и средства проектирования компьютерного программного обеспечения. Умеет использовать существующие типовые решения и шаблоны проектирования компьютерного программного обеспечения, применять методы и средства проектирования компьютерного программного обеспечения.	Демонстрирует глубокое знание и понимание принципов построения и видов архитектуры компьютерного программного обеспечения, типовых решений, библиотек программных модулей, шаблонов, классов объектов, используемых при разработке компьютерного программного обеспечения, методов и средств проектирования компьютерного программного обеспечения. Полностью верно и самостоятельно использует существующие типовые решения и шаблоны проектирования компьютерного программного обеспечения, применяет методы и средства

Индикаторы достижения компетенций	**Критерии оценивания компетенций**		
	Недостаточный уровень	**Базовый уровень**	**Повышенный уровень**
			проектирования компьютерного программного обеспечения.

ИПК1.2 Проектирует структуру данных, баз данных и программных интерфейсов.	Не знает и умеет применять методы и средства проектирования баз данных и программных интерфейсов.	Знает и умеет применять методы и средства проектирования баз данных и программных интерфейсов.	Глубоко знает, полностью верно и самостоятельно умеет применять методы и средства проектирования баз данных и программных интерфейсов.
ИПК1.3 Разрабатывает техническую документацию на компьютерное программное обеспечение с использованием существующих стандартов, оценивает и согласовывает сроки выполнения поставленных задач.	Не знает стандарты в области разработки компьютерного программного обеспечения. Не умеет разрабатывать техническую документацию, оценивать и согласовывать сроки выполнения поставленных задач.	Знает стандарты в области разработки компьютерного программного обеспечения. Умеет разрабатывать техническую документацию, оценивать и согласовывать сроки выполнения поставленных задач.	Демонстрирует свободное владение стандартами в области разработки компьютерного программного обеспечения. Полностью верно и самостоятельно разрабатывает техническую документацию, оценивает и согласовывает сроки выполнения поставленных задач.
ИПК9.1 Демонстрирует умение определять и описывать тестовые случаи на основе требований, заявленных к программному обеспечению.	Не знает классификацию видов и типов тестирования программного обеспечения, техники проектирования и комбинаторики тестов, тестовые данные, обеспечивающие проверку безопасности программного обеспечения. Не	Знает классификацию видов и типов тестирования программного обеспечения, техники проектирования и комбинаторики тестов, тестовые данные, обеспечивающие проверку безопасности программного обеспечения. Умеет	Глубоко знает и понимает классификацию видов и типов тестирования программного обеспечения, техники проектирования и комбинаторики тестов, тестовые данные, обеспечивающие проверку безопасности программного

Индикаторы достижения компетенций	Критерии оценивания компетенций		
	Недостаточный уровень	**Базовый уровень**	**Повышенный уровень**
	умеет применять техники проектирования тестов, анализировать тестовые случаи на предмет полноты учета покрытия, документировать тесты, разрабатывать скрипты и/или программные модули для автоматизации тестирования программного обеспечения, в том числе для проверки информационной безопасности разрабатываемого программного обеспечения.	применять техники проектирования тестов, анализировать тестовые случаи на предмет полноты учета покрытия, документировать тесты, разрабатывать скрипты и/или программные модули для автоматизации тестирования программного обеспечения, в том числе для проверки информационной безопасности разрабатываемого программного обеспечения.	обеспечения. Полностью верно и самостоятельно применяет техники проектирования тестов, анализирует тестовые случаи на предмет полноты учета покрытия, документирует тесты, разрабатывает скрипты и/или программные модули для автоматизации тестирования программного обеспечения, в том числе для проверки информационной безопасности разрабатываемого программного обеспечения.
ИПК9.2 Проводит тестирование по разработанным тестовым случаям, осуществляет сбор информации о несоответствиях заявленным требованиям.	Не умеет выполнять начальные настройки для проведения тестирования, выполнять необходимые виды тестирования.	Умеет выполнять начальные настройки для проведения тестирования, выполнять необходимые виды тестирования.	Самостоятельно и полностью верно выполняет начальные настройки для проведения тестирования, необходимые виды тестирования.

ИПК9.3 Анализирует результаты тестирования и дает оценку качеству разрабатываемого программного обеспечения.	Не знает типы дефектов программного обеспечения, их классификацию и статистику возникновения. Не умеет определять уровень критичности дефектов программного	Знает типы дефектов программного обеспечения, их классификацию и статистику возникновения. Умеет определять уровень критичности дефектов	Демонстрирует глубокое знание и понимание типов дефектов программного обеспечения, их классификации и статистики возникновения. Полностью верно и самостоятельно
Индикаторы достижения компетенций	**Критерии оценивания компетенций**		
	Недостаточный уровень	**Базовый уровень**	**Повышенный уровень**
	обеспечения, составлять отчеты об анализе результатов тестирования программного обеспечения.	программного обеспечения, составлять отчеты об анализе результатов тестирования программного обеспечения.	определяет уровень критичности дефектов программного обеспечения, составляет отчеты об анализе результатов тестирования программного обеспечения.

Приложение № 2 к рабочей программе дисциплины
« Алгоритмы и алгоритмические языки **»**
наименование дисциплины

Методические указания для студентов по освоению дисциплины

Текст