МИНОБРНАУКИ РОССИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
(ФГБОУ ВО «ВГУ»)

УТВЕРЖДАЮ
Заведующий кафедрой
теоретической и прикладной лингвистики

Шилихина К.М.
03.06.2024 г.

РАБОЧАЯ ПРОГРАММА УЧЕБНОЙ ДИСЦИПЛИНЫ

Б1.О.25 Информатика и основы программирования

1. Код и наименование направления подготовки/специальности:

45.03.03 Фундаментальная и прикладная лингвистика

2. Профиль подготовки/специализация:

Экспертно-аналитическая деятельность

3. Квалификация выпускника: бакалавр

4. Форма обучения: очная

5. Кафедра, отвечающая за реализацию дисциплины: кафедра теоретической и прикладной лингвистики

6. Составители программы: Донина Ольга Валерьевна, кандидат филол. наук, доцент кафедры теоретической и прикладной лингвистики

7. Рекомендована: Научно-методическим советом факультета РГФ, протокол № 8 от 01.04.2024 г.

8. Учебный год: 2024/2025 **Семестр:** 1, 2

9. Цели и задачи учебной дисциплины

научить слушателей применять компьютерные технологии (в первую очередь, язык программирования Python) для решения возникающих на практике лингвистических задач (автоматическая обработка и анализ текстовых данных, поиск информации и др.).

10. Место учебной дисциплины в структуре ООП:

Настоящая дисциплина является частью математического и естественно-научного цикла. Основные положения дисциплины должны быть использованы в дальнейшем при изучении следующих дисциплин: Проектирование баз данных, Автоматическая обработка естественного языка, Компьютерная лингвистика, Анализ данных для лингвиста, Искусственные интеллектуальные системы, Семантический WEB.

11. Планируемые результаты обучения по дисциплине/модулю (знания, умения, навыки), соотнесенные с планируемыми результатами освоения образовательной программы (компетенциями) и индикаторами их достижения:

Код	Название компетенции	Коды	Индикаторы	Планируемые результаты обучения
ОПК–7	Способен понимать принципы работы современных информационных технологий и использовать их для решения задач профессиональной деятельности	ОПК-7.1	Осуществляет поиск, сбор, хранение, обработку, представление информации при решении задач профессиональной деятельности	Знать: принципы работы современных информационных технологий. Уметь: использовать современные информационные технологии для решения задач профессиональной деятельности. Владеть: принципами создания программных продуктов, ориентированных на автоматическую обработку естественного языка.
		ОПК-7.2	Подбирает и использует информационные технологии при решении задач профессиональной деятельности	
		ОПК-7.3	Создает программные продукты, ориентированные на автоматическую обработку естественного языка	
ПК-7	Владеет принципами создания электронных языковых ресурсов (текстовых, речевых и мультимодальных корпусов; словарей, тезаурусов, онтологий; фонетических, лексических, грамматических и иных баз данных и баз знаний) и умеет	ПК-7.1	Разрабатывает и документирует программные интерфейсы	Знать: основные принципы создания электронных языковых ресурсов. Уметь: пользоваться основными методами, способами и средствами получения, хранения, переработки информации; пользоваться лингвистически ориентированными программными продуктами Владеть: навыком разработки электронных языковых ресурсов; опытом применения основных методов, способов и средств получения,
		ПК-7.3	Анализирует требования к программному обеспечению	

	пользоваться такими ресурсами.			хранения, переработки информации

12. Объем дисциплины в зачетных единицах/час. — 7 з.е. / 252 ч.

Форма промежуточной аттестации: зачет с оценкой; экзамен

13. Трудоемкость по видам учебной работы

Вид учебной работы		Трудоемкость		
		Всего	По семестрам	
			1 семестр	2 семестр
Аудиторные занятия		**94**	**50**	**44**
в том числе:	лекции	30	16	14
	практические	-	-	-
	лабораторные	64	34	30
Самостоятельная работа		**122**	**76**	**46**
в том числе: курсовая работа (проект)		-	-	-
Форма промежуточной аттестации *(зачет)*		**-**	Зачет с оценкой	Экзамен - 36
Итого:		**252**	**126**	**126**

13.1. Содержание дисциплины

п/п	Наименование раздела дисциплины	Содержание раздела дисциплины	Реализация раздела дисциплины с помощью онлайн-курса, ЭУМК
		1. Лекции	
1.1	Системы контроля версий	Gut. GitHub.	https://edu.vsu.ru/course/view.php?id=25878
1.2	Основы языка программирования Python	Условные инструкции в Python. Циклы. Iterables. Списки. Итерация по списку и его индексам. Кортежи. Множества. Словари. Работа с файлами. Кодировки, формат CSV/TSV. Строковые методы. Функции. Регулярные выражения.	https://edu.vsu.ru/course/view.php?id=25878
1.3	Основы обработки естественного языка в Python	Морфология, лемматизация, морфоанализ. Pandas. Matplotlib. Seaborn. Веб-скрейпинг. HTML, парсинг HTML. XML, парсинг XML. JSON, многострочный JSON. Дистрибутивная семантика. Word2vec, fasttext. Эмбеддинги. Тематическое моделирование. Работа с файловой системой. Сети в Python, network.	https://edu.vsu.ru/course/view.php?id=25878

		IDE VSCode; Jupyter.	
2. Практические занятия			
2.1			
2.2			
3. Лабораторные занятия			
3.1	Системы контроля версий	Gut. GitHub.	https://edu.vsu.ru/course/view.php?id=25878
3.2	Основы языка программирования Python	Условные инструкции в Python. Циклы. Iterables. Списки. Итерация по списку и его индексам. Кортежи. Множества. Словари. Работа с файлами. Кодировки, формат CSV/TSV. Строковые методы. Функции. Регулярные выражения.	https://edu.vsu.ru/course/view.php?id=25878
3.3	Основы обработки естественного языка в Python	Морфология, лемматизация, морфоанализ. Pandas. Matplotlib. Seaborn. Веб-скрейпинг. HTML, парсинг HTML. XML, парсинг XML. JSON, многострочный JSON. Дистрибутивная семантика. Word2vec, fasttext. Эмбеддинги. Тематическое моделирование. Работа с файловой системой. Сети в Python, network. IDE VSCode; Jupyter.	https://edu.vsu.ru/course/view.php?id=25878

13.2. Темы (разделы) дисциплины и виды занятий

№ п/п	Наименование темы (раздела) дисциплины	Виды занятий (количество часов)				
		Лекции	Практические	Лабораторные	Самостоятельная работа	Всего
1	Системы контроля версий	2	-	4	8	14
2	Основы языка программирования Python	14	-	30	57	101
3	Основы обработки естественного языка в Python	14	-	30	57	101
	Итого:	30	-	64	122	216

14. Методические указания для обучающихся по освоению дисциплины

Необходимо регулярное посещение лекционных и лабораторных занятий, работа с литературой по дисциплине, выполнение индивидуальных лабораторных работ. Самостоятельная работа обучающихся предусматривает подготовку к аудиторным и лабораторным занятиям; выполнение домашних заданий.

При изучении материала учебной дисциплины по учебнику нужно, прежде всего, уяснить существо каждого излагаемого там вопроса. Главное – это понять изложенное в учебнике, а не «заучить».

Изучать материал рекомендуется по темам конспекта лекций и по главам (параграфам) учебника (учебного пособия). Сначала следует прочитать весь материал темы (параграфа), особенно не задерживаясь на том, что показалось не совсем понятным: часто это становится понятным из

последующего. Затем надо вернуться к местам, вызвавшим затруднения и внимательно разобраться в том, что было неясно.

Особое внимание при повторном чтении необходимо обратить на формулировки соответствующих определений, формулы и т.п. (они обычно бывают набраны в учебнике курсивом); в точных формулировках, как правило, существенно каждое слово и полезно понять, почему данное положение сформулировано именно так. Однако не следует стараться заучивать формулировки; важно понять их смысл и уметь изложить результат своими словами.

Закончив изучение раздела, полезно составить краткий конспект, по возможности, не заглядывая в учебник (учебное пособие).

При изучении учебной дисциплины особое внимание следует уделить приобретению навыков решения профессионально-ориентированных задач. Для этого, изучив материал данной темы, надо сначала обязательно разобраться в решениях соответствующих задач, которые рассматривались на практических занятиях, приведены в учебно-методических материалах, пособиях, учебниках, ресурсах Интернета, обратив особое внимание на методические указания по их решению.

Закончив изучение раздела, нужно проверить умение ответить на все вопросы программы курса по этой теме (осуществить самопроверку).

Все вопросы, которые должны быть изучены и усвоены, в программе перечислены достаточно подробно. Однако очень полезно составить перечень таких вопросов самостоятельно (в отдельной тетради) следующим образом:

– начав изучение очередной темы программы, выписать сначала в тетради последовательно все перечисленные в программе вопросы этой темы, оставив справа широкую колонку;

– по мере изучения материала раздела (чтения учебника, учебно-методических пособий, конспекта лекций) следует в правой колонке указать страницу учебного издания (конспекта лекции), на которой излагается соответствующий вопрос.

В результате в этой тетради будет полный перечень вопросов для самопроверки, который можно использовать и при подготовке к экзамену. Кроме того, ответив на вопрос или написав соответствующую формулу, можете по учебнику (конспекту лекций) быстро проверить, правильно ли это сделано, если в правильности своего ответа Вы сомневаетесь. Наконец, по тетради с такими вопросами Вы можете установить, весь ли материал, предусмотренный программой, Вами изучен.

Следует иметь в виду, что в различных учебных изданиях материал может излагаться в разной последовательности. Поэтому ответ на какой-нибудь вопрос программы может оказаться в другой главе, но на изучении курса в целом это, конечно, никак не скажется.

15. Перечень основной и дополнительной литературы, ресурсов интернет, необходимых для освоения дисциплины

а) основная литература:

№ п/п	Источник
1	Златопольский Д. М. Основы программирования на языке Python / Д. М. Златопольский. — 2-ое изд., испр. и доп. — Москва : ДМК Пресс, 2018. — 396 с. — ISBN 978-5-97060-641-4. — Текст : электронный // Лань : электронно- библиотечная система. — URL: https://e.lanbook.com/book/131683 (дата обращения: 06.01.2021). — Режим доступа: для авториз. пользователей.
2	Интеллектуальные информационные системы и технологии : учебное пособие / Ю. Ю. Громов, О. Г. Иванова, В. В. Алексеев и др. – Тамбов : Тамбовский государственный технический университет (ТГТУ), 2013. – 244 с. – URL: https://biblioclub.ru/index.php?page=book&id=277713

б) дополнительная литература:

№ п/п	Источник
3	Лутц М. Изучаем Python / М. Лутц. – Символ-Плюс. – 2020. – 1280 с.
4	Саммерфилд М. Программирование на Python 3 / М. Саммерфилд. – Символ-Плюс. – 2020. – 608 с.
5	Бизли Д. Python. Подробный справочник / Д. Бизли. – Символ-Плюс. – 2020. – 864 с.
6	Сузи, Р. А. Язык программирования Python : учебное пособие / Р. А. Сузи. — 2-е изд. — Москва : ИНТУИТ, 2016. — 350 с. — ISBN 5-9556-0058-2. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/100546 (дата обращения: 06.01.2021). — Режим доступа: для авториз. пользователей.
7	Северенс, Ч. Введение в программирование на Python : учебное пособие / Ч. Северенс. — 2-е изд. — Москва : ИНТУИТ, 2016. — 231 с. — Текст : электронный // Лань : электроннобиблиотечная система. — URL: https://e.lanbook.com/book/100703 (дата обращения: 06.01.2021). — Режим доступа: для авториз. пользователей.

8	Практикум по программированию на языке Python. Учебное пособие. — М.: Финансовый университет, департамент анализа данных и машинного обучения, 2023. — 320 с.
9	Марченко А. Л. Python: большая книга примеров / А. Л. Марченко. — Москва : Издательство Московского университета, 2023. — 361 с.
10	Хахаев И. А. Практикум по алгоритмизации и программированию на Python: / И. А. Хахаев - М. : Альт Линукс, 2010. - 126 с.
11	Сысоева М. В., Сысоев И. В. Программирование для «нормальных» с нуля на языке Python : Учебник. / М.В. Сысоева, И.В. Сысоев ; отв. ред. В. Л. Черный. – 2-е изд., испр. и доп. – Москва : Базальт СПО; МАКС Пресс, 2023. – 184 с.
12	Каипова А.Д., Казагачев В.Н. Программирование на Python: учебное пособие / А.Д. Каипова, В.Н. Казагачев - Алматы: Эверо, 2022 - 288 с.
13	Амоа К. А., Рындин Н. А., Скворцов Ю. С. Разработка программных пакетов на языке Python: учеб. пособие / К. А. Амоа, Н. А. Рындин, Ю. С. Скворцов; ФГБОУ ВО «Воронежский государственный технический университет». – Воронеж: Изд-во ВГТУ, 2020. – 61 с.
14	Лутц М. Карманный справочник / М. Лутц. – М.: ООО «И.Д. Вильямс», 2015. – 320 с.
15	Бейдер Д. Чистый Python. Тонкости программирования для профи. / Д. Бейдер. — СПб.: Питер, 2018. — 288 с.
16	Лутц М. Программирование на Python. / М. Лутц – СПб.: Символ-Плюс, 2011. – 992 с.
17	Mueller J.P. Beginning Programming with Python / J.P. Mueller. – John Wiley & Sons, Inc. – 2018. – 411 p.
18	Мэтиз Э. Изучаем Python. Программирование игр, визуализация данных, веб-приложения. / Э. Мэтиз — СПб.: Питер, 2017. — 496 с.
19	Стивенсон Б. Python. Сборник упражнений / Б. Стивенсон – М.: ДМК Пресс, 2021. – 238 с.
20	Жуковский, О.И. Информационные технологии и анализ данных : учебное пособие / О.И. Жуковский ; Томский Государственный университет систем управления и радиоэлектроники (ТУСУР). – Томск : Эль Контент, 2014. – 130 с. : схем., ил. – Режим доступа: по подписке. – URL: https://biblioclub.ru/index.php?page=book&id=480500
21	Серегин, М.Ю. Интеллектуальные информационные системы : учебное пособие / М.Ю. Серегин, М.А. Ивановский, А.В. Яковлев ; Тамбовский государственный технический университет. – Тамбов : Тамбовский государственный технический университет (ТГТУ), 2012. – 205 с. : ил. – Режим доступа: по подписке. – URL: https://biblioclub.ru/index.php?page=book&id=277790
22	Хныкина, А.Г. Информационные технологии : учебное пособие / А.Г. Хныкина, Т.В. Минкина ; Северо-Кавказский федеральный университет. – Ставрополь : Северо-Кавказский Федеральный университет (СКФУ), 2017. – 126 с. : схем., ил. – Режим доступа: по подписке. – URL: https://biblioclub.ru/index.php?page=book&id=494703
23	Силен Д. Основы Data Science и Big Data. Python и наука о данных / Д. Силен, А. Мейсман, М. Али. – Спб.: Питер, 2017. – 336 с.
24	Бенгфорт Б., Билбро Р., Охеда Т. Прикладной анализ текстовых данных на Python. Машинное обучение и создание приложений обработки естественного языка. — СПб.: Питер, 2019. — 368 с.

в) информационные электронно-образовательные ресурсы (официальные ресурсы интернет)*:

№ п/п	Ресурс
25	ЭБС Лань. – Режим доступа: по подписке. – URL: ЭБС Лань (lanbook.com)
26	ЭБС «Университетская библиотека онлайн». – Режим доступа: по подписке. – URL: ЭБС "Университетская библиотека онлайн" читать электронные книги (biblioclub.ru)
27	ЭБС ЮРАЙТ.– Режим доступа: по подписке. – URL: Образовательная платформа Юрайт. Для вузов и ссузов. (urait.ru)

16. Перечень учебно-методического обеспечения для самостоятельной работы

№ п/п	Источник
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 (дата обращения: 06.01.2021). — Режим доступа: для авториз. пользователей.
2	Донина О.В. Автоматизация лингвистических исследований: учебное пособие / О.В.

	Донина. – Воронеж : Издательский дом ВГУ, 2022. – 125 с.
3	Пушкарёва, Т.П. Основы компьютерной обработки информации : учебное пособие / Т.П. Пушкарёва ; Сибирский федеральный университет. – Красноярск : Сибирский федеральный университет (СФУ), 2016. – 180 с. : ил. – Режим доступа: по подписке. – URL: https://biblioclub.ru/index.php?page=book&id=497475
4	Исакова, А.И. Информационные технологии : учебное пособие / А.И. Исакова ; Томский Государственный университет систем управления и радиоэлектроники (ТУСУР), Кафедра автоматизированных систем управления (АСУ). – Томск : ТУСУР, 2013. – 207 с. : ил. – Режим доступа: по подписке. – URL: https://biblioclub.ru/index.php?page=book&id=480610
5	Гасанов, Э. Э. Интеллектуальные системы. Теория хранения и поиска информации : учебник для вузов / Э. Э. Гасанов, В. Б. Кудрявцев. — 2-е изд., испр. и доп. — Москва : Издательство Юрайт, 2020. — 271 с. — (Высшее образование). — ISBN 978-5-534-08684-3. — Текст : электронный // ЭБС Юрайт [сайт]. — URL: https://urait.ru/bcode/452220

17. Образовательные технологии, используемые при реализации учебной дисциплины, включая дистанционные образовательные технологии (ДОТ), электронное обучение (ЭО), смешанное обучение):

При реализации дисциплины могут проводиться различные типы лекций (вводная, обзорная и т.д.). При проведении лабораторных работ предпочтение отдается применению классических технологий: обсуждение со студентами заранее подготовленных ими тем и разбор практических задач и лабораторных работ.

18. Материально-техническое обеспечение дисциплины:

/ауд. 12/ - компьютерный класс: Компьютер Arbyte Tempo/AOC (12 шт.), Проектор Benq MW523 (1 шт.), Сканер Canon Canoscan LiDE 120 (5 шт.) Экран проекционный (1 шт.)

---

19. Оценочные средства для проведения текущего контроля успеваемости и промежуточной аттестации

Порядок оценки освоения обучающимися учебного материала определяется содержанием следующих разделов дисциплины:

№ п/п	Наименование раздела дисциплины (модуля)	Компетенция	Индикаторы достижения компетенции	Оценочные средства
1	1. Системы контроля версий 2. Основы языка программирования Python 3. Основы обработки естественного языка в Python	ОПК-7	Осуществляет поиск, сбор, хранение, обработку, представление информации при решении задач профессиональной деятельности (ОПК-7.1)	Выполнение индивидуальных лабораторных работ, тестовые задания
			Подбирает и использует информационные технологии при решении задач профессиональной деятельности (ОПК-7.2)	
			Создает программные продукты, ориентированные на автоматическую обработку естественного языка (ОПК-7.3)	
		ПК-7	Разрабатывает и документирует программные интерфейсы (ПК-7.1)	
			Пользуется электронными языковыми ресурсами для	

№ п/п	Наименование раздела дисциплины (модуля)	Компетенция	Индикаторы достижения компетенции	Оценочные средства
			решения прикладных задач (ПК-7.2)	
			Анализирует требования к программному обеспечению (ПК-7.3)	
Промежуточная аттестация форма контроля – зачет с оценкой, экзамен				Практические задания Лабораторные работы КИМ

20 Типовые оценочные средства и методические материалы, определяющие процедуры оценивания

20.1 Текущий контроль успеваемости

Контроль успеваемости по дисциплине осуществляется с помощью следующих оценочных средств:

Лабораторные работы (в том числе домашние задания), состоящие из упражнений; тестовые задания.

Описание технологии проведения: Обучающиеся выполняют и сдают лабораторные работы.

Требования к выполнению заданий (или шкалы и критерии оценивания): обучающийся в полной мере должен выполнить предлагаемые ему упражнения лабораторных работ и ответить на теоретические вопросы по сдаваемому материалу

Упражнение 1

Напишите программу, запрашивающую у пользователя его имя. В ответ на ввод на экране должно появиться приветствие с обращением по имени, введенному с клавиатуры ранее.

Упражнение 2

Напишите программу, запрашивающую у пользователя целое число и выводящую на экран информацию о том, является введенное число четным или нечетным.

Упражнение 3

Разработайте программу, запрашивающую у пользователя букву латинского алфавита. Если введенная буква входит в следующий список (a, e, i, o или u), необходимо вывести сообщение о том, что эта буква гласная. Если была введена буква y, программа должна написать, что эта буква может быть как гласной, так и согласной. Во всех других случаях должно выводиться сообщение о том, что введена согласная буква.

Упражнение 4

Строка называется палиндромом, если она пишется одинаково в обоих направлениях. Например, палиндромами в английском языке являются слова «anna», «civic», «level», «hannah». Напишите программу, запрашивающую у пользователя строку и при помощи цикла определяющую, является ли она палиндромом.

Упражнение 5

Помимо слов, существуют целые фразы, являющиеся палиндромами, если не обращать внимания на пробелы. Вот лишь несколько примеров на английском: «go dog», «flee to me remote elf» and «some men interpret nine memos». Русские варианты есть следующие: «А кобыле цена дана, да не целы бока», «А Луна канула» и другие. При вынесении решения о том, является ли строка палиндромом, должны игнорироваться пробелы и знаки препинания, а заглавные и прописные буквы должны считаться эквивалентными.

Упражнение 6

«Двенадцать дней Рождества» (The Twelve Days of Christmas) – старая английская песня, построение которой базируется на постоянно увеличивающемся списке подарков в каждый из 12 дней Рождества. В первый день был послан один подарок, в следующий – второй и т. д. Первые три куплета песни приведены ниже. Полностью текст песни можно без труда найти в интернете.

On the first day of Christmas
my true love sent to me:
A partridge in a pear tree.

On the second day of Christmas
my true love sent to me:
Two turtle doves,
And a partridge in a pear tree.

On the third day of Christmas
my true love sent to me:
Three French hens,
Two turtle doves,
And a partridge in a pear tree

Напишите программу, которая будет сама строить куплеты этой песенки. В программе должна присутствовать функция для отображения одного куплета. В качестве входного параметра она должна принимать порядковый номер дня, а в качестве результата возвращать готовый куплет. Далее в основной программе эта функция должна быть вызвана 12 раз подряд. Каждая строка с очередным подарком должна присутствовать в вашей программе лишь раз, за исключением строки «A partridge in a pear tree». В этом случае вы можете отдельно хранить такой вид строки для первого куплета и слегка измененный («And a partridge in a pear tree») – для всех последующих.

Упражнение 7

Многие в своих сообщениях не ставят заглавные буквы, особенно если используют для набора мобильные устройства. Создайте функцию, которая будет принимать на вход исходную строку и возвращать строку с восстановленными заглавными буквами. По существу, ваша функция должна:

- сделать заглавной первую букву в строке, не считая пробелы;
- сделать заглавной первую букву после точки, восклицательного или вопросительного знака, не считая пробелы;
- если текст на английском языке, сделать заглавными буквы «i», которым предшествует пробел или за которыми следует пробел, точка, восклицательный или вопросительный знак.

Реализация такого рода автоматической корректуры исключит большую часть ошибок с регистром букв. Например, строку «what time do i have to be there? what’s the address? this time i’ll try to be on time!» ваша функция должна преобразовать в более приемлемый вариант «What time do I have to be there? What’s the address? This time I’ll try to be on time!». В основной программе запросите у пользователя исходную строку, обработайте ее при помощи своей функции и выведите на экран итоговый результат.

Упражнение 8

В данном упражнении вам предстоит разработать программу, в которой у пользователя будет запрошен список слов, пока он не оставит строку ввода пустой. После этого на экране должны быть показаны слова, введенные пользователем, но без повторов, – каждое по одному разу. При этом слова должны быть отображены в том же порядке, в каком их вводили с клавиатуры. Например, если пользователь на запрос программы введет следующий список слов:

first
second

first
third
second

программа должна вывести:

first
second
third

Упражнение 9

В данном упражнении вы напишете программу, которая будет выделять слова из строки, введенной пользователем. Начните с создания функции, принимающей на вход единственный строковый параметр. В качестве результата она должна возвращать список слов из строки с удаленными знаками препинания, в число которых должны входить точки, запятые, восклицательный и вопросительный знаки, дефисы, апострофы, двоеточия и точки с запятыми. При этом не нужно избавляться от знаков препинания, стоящих внутри слова, таких как апостроф, служащий в английском языке для обозначения сокращений. Например, если на вход функции дать строку "Contractions include: don't, isn't, and wouldn't.", функция должна вернуть следующий список: ["Contractions", "include", "don't", "isn't", "and", "wouldn't"]. В основной программе должна происходить демонстрация вашей функции. Запросите у пользователя строку и выведите на экран все составляющие ее слова с удаленными знаками препинания.

Упражнение 10

Обычно при написании перечислений и списков мы разделяем их элементы запятыми, а перед последним ставим союз «и», как показано ниже:

яблоки
яблоки и апельсины
яблоки, апельсины и бананы
яблоки, апельсины, бананы и лимоны

Напишите функцию, которая будет принимать на вход список из строк и возвращать собранную строку из его элементов в описанной выше манере. Хотя в представленном примере количество элементов списка ограничивается четырьмя, ваша функция должна уметь обрабатывать списки любой длины. В основной программе запросите у пользователя несколько элементов списка, отформатируйте их должным образом при помощи функции и выведите на экран.

Упражнение 11

«Поросячьей латынью» называют молодежный жаргонный язык, производный от английского. И хотя корни этого новообразованного языка неизвестны, упоминание о нем есть как минимум в двух документах, датированных XIX веком, а это значит, что ему уже больше сотни лет.

Для перевода слова с английского на «поросячью латынь» нужно сделать следующее:

- если слово начинается с согласной буквы (включая y), то все буквы с начала слова и до первой гласной (за исключением y) переносятся в конец слова и дополняются сочетанием букв ay. Например, слово computer будет преобразовано в omputercay, а слово think – в inkthay;
- если слово начинается с гласной буквы (не включая y), к концу слова просто добавляется way. К примеру, слово algorithm превратится в algorithmway, а office – в officeway.

Напишите программу, которая будет запрашивать у пользователя строку. После этого она должна переводить введенный текст на «поросячью латынь» и выводить его на экран.

Расширьте свое решение, чтобы ваш анализатор корректно обрабатывал символы в верхнем регистре и знаки препинания, такие как запятая, точка, а также восклицательный и вопросительный знаки.

Если в оригинале слово начинается с заглавной буквы, то в переводе на «поросячью латынь» оно также должно начинаться с заглавной буквы, тогда как буквы, перенесенные в конец слов, должны стать строчными.

Например, слово Computer должно быть преобразовано в Omputercay. Если в конце слова стоит знак препинания, он там же и должен остаться после выполнения перевода. То есть слово в конце предложения Science! необходимо трансформировать в Iencescay!.

Упражнение 12

Токенизация (Tokenizing) представляет собой процесс преобразования исходной строки в список из подстрок, называемых токенами (token). Зачастую со списком токенов работать бывает проще, чем со всей исходной строкой, поскольку в ней могут присутствовать неравномерные разрывы. Кроме того, иногда бывает непросто на лету определить, где заканчивается один токен и начинается другой.

В математических выражениях токенов являются, например, операторы, числа и скобки. Здесь и далее мы будем причислять к списку операторов следующие: *, /, ^, - и +. Операторы и скобки легко идентифицировать, поскольку эти токены всегда состоят ровно из одного символа и никогда не являются составной частью других токенов. Числа выделить бывает сложнее, поскольку эти токены могут состоять из нескольких символов. Любая непрерывная последовательность цифр должна восприниматься как один числовой токен.

Напишите функцию, принимающую в качестве единственного входного параметра строку, содержащую математическое выражение, и преобразующую ее в список токенов. Каждый токен должен быть либо оператором, либо числом, либо скобкой. Для простоты реализации в данном упражнении мы будем оперировать только целочисленными значениями. Функция должна возвращать созданный список токенов.

При решении поставленной задачи вы можете принять допущение о том, что входная строка всегда будет содержать математическое выражение, состоящее из скобок, чисел и операторов. При этом в вашей функции должно быть предусмотрено, что токены могут отделяться друг от друга разным количеством пробелов, а могут и не отделяться вовсе. В основной программе продемонстрируйте работу функции, запросив у пользователя исходную строку и выведя на экран список составляющих ее токенов.

Упражнение 13

Если помните, на старых мобильных телефонах текстовые сообщения набирались при помощи цифровых кнопок. При этом одна кнопка была ассоциирована сразу с несколькими буквами, а выбор зависел от количества нажатий на кнопку. Однократное нажатие приводило к появлению первой буквы в соответствующем этой кнопке списке, последующие нажатия меняли ее на следующую. Список символов, ассоциированных с цифровой панелью, приведен в таблице ниже.

Символы, соответствующие кнопкам на старых телефонах:

Кнопка	Символы
1	. , ? ! :
2	A B C
3	D E F
4	G H I
5	J K L
6	M N O
7	P Q R S
8	T U V
9	W X Y Z
0	Пробел

Напишите программу, отображающую последовательность кнопок, которую необходимо нажать, чтобы на экране телефона появился текст, введенный пользователем. Создайте словарь, сопоставляющий символы с кнопками, которые необходимо нажать, а затем воспользуйтесь им для вывода на экран последовательности кнопок в соответствии с введенным пользователем сообщением по запросу. Например, на ввод строки Hello, World! ваша программа должна откликнуться следующим выводом: 4433555555666110966677755531111. Удостоверьтесь, что ваша программа корректно обрабатывает строчные и прописные буквы. При преобразовании букв в цифры игнорируйте символы, не входящие в указанный перечень, такие как точка с запятой или скобки.

Упражнение 14

Азбука Морзе зашифровывает буквы и цифры при помощи точек и тире. Она была изобретена в XIX веке для передачи информации посредством телеграфа, но широко используется и сегодня.

В данном упражнении вам необходимо написать программу, в которой соответствие символов из азбуки Морзе будет храниться в виде словаря. В таблице ниже приведена та часть азбуки, которая вам понадобится при решении этого задания.

В основной программе вам необходимо запросить у пользователя строку. После этого программа должна преобразовать его в соответствующую последовательность точек и тире, вставляя пробелы между отдельными символами. Символы, не представленные в таблице, можно игнорировать. Например, сообщение Hello, World! может быть представлено следующей последовательностью: .... . .–.. .–.. – – – .–– – – – .–. .–.. –..

Азбука Морзе

Символ	Код	Символ	Код	Символ	Код	Символ	Код
A	.–	J	.– – –	S	...	1	.– – – –
B	–...	K	–.–	T	–	2	..– – –
C	–.–.	L	.–..	U	..–	3	...– –
D	–..	M	– –	V	...–	4	….–
E	.	N	–.	W	.– –	5	…..
F	..–.	O	– – –	X	–..–	6	–….
G	– –.	P	.– –.	Y	–.– –	7	– –...
H	….	Q	– –.–	Z	– –..	8	– – –..
I	..	R	.–.	0	– – – – –	9	– – – –.

Упражнение 15

Первый, третий и пятый символы в канадском почтовом индексе представляют собой буквы, а второй, четвертый и шестой – цифры. Провинцию или территорию, которой принадлежит индекс, можно определить по первому символу индекса, как показано в таблице. Символы D, F, I, O, Q, U, W и Z в настоящее время не используются в почтовых индексах Канады.

Второй символ в почтовом индексе определяет, расположен ли интересующий нас адрес в городе или в сельской местности. Если на этом месте стоит ноль, значит, это сельская местность, иначе город.

Напишите программу, которая будет запрашивать почтовый индекс у пользователя и отображать провинцию или территорию, которой он принадлежит, с указанием того, городская это территория или сельская. Например, если пользователь введет индекс T2N1N4, программа должна определить, что речь идет о городе на территории провинции Альберта. А индекс X0A1B2 соответствует сельской местности в провинции Нунавут или в Северо-Западных территориях. Используйте словарь для хранения информации о соответствии первого символа индекса конкретной провинции или территории. Выведите на экран соответствующее сообщение об ошибке, если индекс начинается с символа, который не используется для этих целей, или второй символ не является цифрой.

Почтовые индексы Канады:

Провинция/территория	**Первая буква (буквы) индекса**
Ньюфаундленд	A
Остров Принца Эдуарда	C
Новая Шотландия	B
Нью-Брансуик	E
Квебек	G, H и J
Онтарио	K, L, M, N и P
Манитоба	R
Саскачеван	S
Альберта	T
Британская Колумбия	V
Нунавут	X
Северо-Западные территории	X

Юкон	Y

Упражнение 16

Несмотря на то что популярность оплаты по чекам за последние годы серьезно снизилась, некоторые компании до сих пор используют этот способ для ведения взаиморасчетов с сотрудниками и поставщиками. Сумма на чеках обычно указывается дважды: один раз цифрами, второй – прописью на английском языке. Повторение суммы двумя разными формами записи призвано не позволить недобросовестным сотрудникам или поставщикам изменять сумму на чеках перед их обналичиванием.

В данном упражнении вам необходимо написать функцию, принимающую в качестве входного параметра число от 0 до 999 и возвращающую строку прописью. Например, если значение параметра будет равно 142, функция должна вернуть следующую строку: «one hundred forty two». Используйте один или несколько словарей вместо условных конструкций if/elif/else для выработки решения этой задачи. Напишите основную программу, в которой пользователь будет вводить числовое значение, а на экран будет выводиться соответствующая сумма прописью.

Упражнение 17

Напишите программу, определяющую и выводящую на экран количество уникальных символов во введенной пользователем строке. Например, в строке Hello, World! содержится десять уникальных символов, а в строке zzz – один. Используйте словарь или набор для решения этой задачи.

Упражнение 18

Анаграммами называются слова, образованные путем взаимной перестановки букв. В английском языке, например, анаграммами являются слова «live» и «evil», а в русском – «выбор» и «обрыв». Напишите программу, которая будет запрашивать у пользователя два слова, определять, являются ли они анаграммами, и выводить на экран ответ.

Понятие анаграмм не ограничивается словами, а может быть расширено до целых предложений. Например, строки «William Shakespeare» и «I am a weakish speller» являются полными анаграммами, если игнорировать пробелы и заглавные буквы.

Расширьте свою программу, добавив возможность проверки на анаграммы целых фраз. При анализе не обращайте внимания на знаки препинания, заглавные буквы и пробелы.

Упражнение 19

В известной игре Эрудит (Scrabble) каждой букве соответствует определенное количество очков. Общая сумма очков, которую получает игрок, составивший это слово, складывается из очков за каждую букву, входящую в его состав. Чем более употребимой является буква в языке, тем меньше очков начисляется за ее использование. В таблице ниже приведены все соответствия букв и очков из английской версии игры.

Стоимость букв в английской версии игры Эрудит:

Очки	Буквы
1	A, E, I, L, N, O, R, S, T и U
2	D и G
3	B, C, M и P
4	F, H, V, W и Y
5	K
8	J и X
10	Q и Z

Напишите программу, рассчитывающую и отображающую количество очков за собранное слово. Создайте словарь для хранения соответствий между буквами и очками и используйте его в своем решении.

Упражнение 20

Карточка для игры в лото состоит из пяти колонок, в каждой из которых – пять номеров. Колонки помечены буквами B, I, N, G и O. Под каждой буквой могут быть номера в своем диапазоне из 15 чисел. А именно под буквой B могут присутствовать числа от 1 до 15, под I – от 16 до 30, под N – от 31 до 45 и т. д.

Напишите функцию, которая будет создавать случайную карточку лото и сохранять ее в словаре. Ключами словаря будут буквы B, I, N, G и O, а значениями – списки из пяти чисел, располагающихся в колонке под каждой буквой. Создайте еще одну функцию для отображения созданной карточки лото на экране со столбцами с заголовками. В основной программе создайте карту лото случайным образом и выведите ее на экран.

Упражнение 21

Карточка для игры в лото считается выигравшей, если в ней на одной линии расположились пять выпавших номеров. Обычно игроки зачеркивают номера на своих карточках. В данном упражнении мы будем обнулять в словаре выпавшие номера.

Напишите функцию, принимающую на вход карточку в качестве параметра. Если карточка содержит последовательность из пяти нулей (по вертикали, горизонтали или диагонали), функция должна возвращать True, в противном случае – False.

В основной программе вы должны продемонстрировать на примере работу функции, создав и отобразив несколько карточек с указанием того, какие из них выиграли. В вашем примере должно быть как минимум по одной карточке с выигрышем по вертикали, горизонтали и диагонали, а также карточки, на которые выигрыш не выпал.

Упражнение 22

В данном упражнении мы напишем программу, выполняющую симуляцию игры в лото с одной картой. Начните с генерирования списка из всех возможных номеров для выпадения (от B1 до O75). После этого перемешайте номера в хаотичном порядке, воспользовавшись функцией shuffle из модуля random. Вытаскивайте по одному номеру из списка и зачеркивайте номера, пока карточка не окажется выигравшей. Проведите 1000 симуляций и выведите на экран минимальное, максимальное и среднее количество извлечений номеров, требующееся для выигрыша.

Упражнение 23

В операционных системах на базе Unix обычно присутствует утилита с названием head. Она выводит первые десять строк содержимого файла, имя которого передается в качестве аргумента командной строки. Напишите программу на Python, имитирующую поведение этой утилиты. Если файла, указанного пользователем, не существует, или не задан аргумент командной строки, необходимо вывести соответствующее сообщение об ошибке.

Упражнение 24

В тех же операционных системах на базе Unix обычно есть и утилита с названием tail, которая отображает последние десять строк содержимого файла, имя которого передается в качестве аргумента командной строки. Реализуйте программу, которая будет делать то же самое. В случае отсутствия файла, указанного пользователем, или аргумента командной строки вам нужно вывести соответствующее сообщение.

Данную задачу можно решить сразу несколькими способами. Например, можно все содержимое файла целиком загрузить в список и затем выбрать из него последние десять элементов. А можно дважды прочитать содержимое файла: первый раз, чтобы посчитать количество строк, а второй – чтобы отобразить последние десять из них. При этом оба перечисленных подхода нежелательны, если речь идет о файлах достаточного объема. Существует решение, требующее единственного чтения файла и сохранения всех десяти строк за раз. В качестве дополнительного задания разработайте такой алгоритм.

Упражнение 25

Продолжаем тему операционных систем на базе Unix, в которых обычно также есть утилита с названием cat, что является сокращением от concatenate (сцепить). Эта утилита выводит на экран объединенное содержимое нескольких файлов, имена которых передаются ей в качестве аргументов командной строки. При этом файлы сцепляются в том порядке, в котором указаны в аргументах.

Напишите программу на Python, имитирующую работу этой утилиты. В процессе работы программа должна выдавать сообщения о том, какие файлы открыть не удается, и переходить к следующим файлам. Если программа была запущена без аргументов командной строки, на экран должно быть выведено соответствующее сообщение об ошибке.

Упражнение 26

Напишите программу, которая будет считывать содержимое файла, добавлять к считанным строкам порядковый номер и сохранять их в таком виде в новом файле. Имя исходного файла необходимо запросить у пользователя, так же, как и имя целевого файла. Каждая строка в созданном файле должна начинаться с ее номера, двоеточия и пробела, после чего должен идти текст строки из исходного файла.

Упражнение 27

В данном упражнении вы должны написать программу, которая будет находить самое длинное слово в файле. В качестве результата программа должна выводить на экран длину самого длинного слова и все слова такой длины. Для простоты принимайте за значимые буквы любые непробельные символы, включая цифры и знаки препинания.

Упражнение 28

Одна из техник декодирования простейших алгоритмов шифрования заключается в применении частотного анализа. Иными словами, вы просто анализируете зашифрованный текст, подсчитывая частоту употребления всех букв. Затем можно использовать операции подстановки для замены наиболее популярных символов на часто используемые в языке буквы (в английском это, например, буквы E и T).

Напишите программу, которая будет способствовать дешифрации текста путем вывода на экран частоты появления разных букв. При этом пробелы, знаки препинания и цифры должны быть проигнорированы. Также не должен учитываться регистр, то есть символы a и A должны восприниматься как одна буква. Имя файла для анализа пользователь должен передавать программе посредством аргумента командной строки. Если программе не удастся открыть файл для анализа или аргументов командной строки будет слишком много, на экране должно быть отображено соответствующее сообщение об ошибке.

Упражнение 29

Разработайте программу, которая будет показывать слово (или слова), чаще остальных встречающееся в текстовом файле. Сначала пользователь должен ввести имя файла для обработки. После этого вы должны открыть файл и проанализировать его построчно, разделив при этом строки по словам и исключив из них пробелы и знаки препинания. Также при подсчете частоты появления слов в файле вам стоит игнорировать регистры.

Упражнение 30

Напишите программу, выполняющую перевод из буквенных оценок в числовые и обратно. Программа должна позволять пользователю вводить несколько значений для перевода – по одному в каждой строке. Для начала предпримите попытку сконвертировать введенное пользователем значение из числового в буквенное. Если возникнет исключение, попробуйте выполнить обратное преобразование – из буквенного в числовое. Если и эта попытка окончится неудачей, выведите предупреждение о том, что введенное значение не является допустимым. Пусть ваша программа конвертирует оценки до тех пор, пока пользователь не оставит ввод пустым.

Упражнение 31

Как вы знаете, в языке Python для создания комментариев в коде используется символ #. Комментарий начинается с этого символа и продолжается до конца строки – без возможности остановить его раньше.

В данном упражнении вам предстоит написать программу, которая будет удалять все комментарии из исходного файла с кодом на языке Python. Пройдите по всем строкам в файле на предмет поиска символа #. Обнаружив его, программа должна удалить все содержимое, начиная с этого символа и до конца строки. Для простоты не будем рассматривать ситуации, когда знак решетки встречается в середине строки. Сохраните новое содержимое в созданном файле. Имена файла источника и файла назначения должны быть запрошены у пользователя. Удостоверьтесь в том, что программа корректно обрабатывает возможные ошибки при работе с обоими файлами.

Упражнение 32

Создание пароля посредством генерирования случайных символов может обернуться сложностью в запоминании полученной относительно надежной последовательности. Некоторые системы создания паролей рекомендуют сцеплять вместе два слова на английском языке, тем самым упрощая запоминание заветного ряда символов – правда, в ущерб его надежности.

Напишите программу, которая будет открывать файл со списком слов, случайным образом выбирать два из них и сцеплять вместе для получения итогового пароля. При создании пароля исходите из следующего требования: он должен состоять минимум из восьми символов и максимум из десяти, а каждое из используемых слов должно быть длиной хотя бы в три буквы. Кроме того, сделайте заглавными первые буквы обоих слов, чтобы легко можно было понять, где заканчивается одно и начинается другое. По завершении процесса полученный пароль должен быть отображен на экране.

Упражнение 33

Ученикам, желающим запомнить правила написания слов в английском языке, часто напоминают следующее рифмованное одностишие: «I before E except after C» (I перед E, если не после C). Это правило позволяет запомнить, в какой последовательности писать буквы I и E, идущие в слове одна за другой, а именно: буква I должна предшествовать букве E, если непосредственно перед ними не стоит буква C. Если стоит – порядок гласных будет обратным. Примеры слов, на которые действует это правило: believe, chief, fierce, friend, ceiling и receipt. Но есть и исключения из этого правила, и одним из них является слово weird (странный).

Напишите программу, которая будет построчно обрабатывать текстовый файл. В каждой строке может присутствовать много слов, а может и не быть ни одного. Слова, в которых буквы E и I не соседствуют друг с другом, обработке подвергать не следует. Если же такое соседство присутствует, необходимо проверить, соответствует ли написание анализируемого слова указанному выше правилу. Создайте и выведите на экран два списка. В первом должны располагаться слова, следующие правилу, а во втором – нарушающие его. При этом списки не должны содержать повторяющиеся слова. Также отобразите на экране длину каждого списка, чтобы пользователю было понятно, сколько слов в файле не отвечает правилу.

Упражнение 34

Липограмма представляет собой литературный прием, состоящий в написании текста без использования одной из букв (или группы букв). Часто отвергнутой буквой является одна из распространенных гласных, хотя это условие и не обязательно. Например, в стихотворении Эдгара Аллана По «Ворон» («The Raven»), состоящем из более тысячи слов, ни разу не встречается буква Z, что делает его полноценной липограммой. Еще одним примером липограммы является роман «Исчезание» («La Disparition»). Во французской и английской версиях этого произведения общим объемом примерно в 300 страниц не употребляется буква E, за исключением фамилии автора.

Истории литературы известен случай написания романа объемом около 50 тыс. слов, в котором ни разу не была употреблена самая популярная в английском алфавите буква E. Название его – «Gadsby».

Напишите программу, которая будет считывать список слов из файла и собирать статистику о том, в каком проценте слов используется каждая буква алфавита. Выведите результат для всех 26 букв английского алфавита и отдельно отметьте букву, которая встречалась в словах наиболее редко. В вашей программе должны игнорироваться знаки препинания и регистр символов.

Упражнение 35

Автоматическая проверка орфографии не помешала бы многим из нас. В данном упражнении мы напишем простую программу, сверяющую слова из текстового файла со словарем. Неправильно написанными будем считать все слова, которых не нашлось в словаре.

Имя файла, в котором требуется выполнить орфографическую проверку, пользователь должен передать при помощи аргумента командной строки. В случае отсутствия аргумента должна выдаваться соответствующая ошибка. Сообщение об ошибке также должно появляться,

если не удается открыть указанный пользователем файл. Вам следует игнорировать регистр символов и знаки препинания при выполнении проверки.

Упражнение 36

Проверка орфографии – лишь составная часть расширенного текстового анализа на предмет наличия ошибок. Одной из самых распространенных ошибок в текстах является повторение слов. Например, автор может по ошибке дважды подряд написать одно слово, как в следующем примере:

At least one value must be entered
entered in order to compute the average.

Некоторые текстовые процессоры умеют распознавать такой вид ошибок при выполнении текстового анализа. В данном упражнении вам предстоит написать программу для определения наличия дублей слов в тексте. При нахождении повтора на экран должен выводиться номер строки и дублирующееся слово. Удостоверьтесь, что программа корректно обрабатывает случаи, когда повторяющиеся слова находятся на разных строках, как в предыдущем примере. Имя файла для анализа должно быть передано программе в качестве единственного аргумента командной строки. При отсутствии аргумента или невозможности открыть указанный файл на экране должно появляться соответствующее сообщение об ошибке.

Упражнение 37

Перед публикацией текста или документа обычно принято удалять или изменять в них служебную информацию.

В данном упражнении вам необходимо написать программу, которая будет заменять все служебные слова в тексте на символы звездочек (по количеству символов в словах). Вы должны осуществлять регистрозависимый поиск служебных слов в тексте, даже если эти слова входят в состав других слов. Список служебных слов должен храниться в отдельном файле. Сохраните отредактированную версию исходного файла в новом файле. Имена исходного файла, файла со служебными словами и нового файла должны быть введены пользователем.

В качестве дополнительного задания расширьте свою программу таким образом, чтобы она выполняла замену служебных слов вне зависимости от того, какой регистр символов используется в тексте. Например, если в списке служебных слов будет присутствовать слово exam, то все следующие варианты слов должны быть заменены звездочками: exam, Exam, ExaM и EXAM.

Упражнение 38

При написании функций хорошей практикой считается предварение ее блоком комментариев с описанием назначения функции, ее входных параметров и возвращаемого значения. Но некоторые разработчики просто не пишут комментарии к своим функциям. Другие честно собираются написать их когда-нибудь в будущем, но руки так и не доходят.

Напишите программу, которая будет проходить по файлу с исходным кодом на Python и искать функции, не снабженные блоком комментариев.кодом Можно принять за аксиому, что строка, начинающаяся со слова def, следом за которым идет пробел, будет считаться началом функции. И если функция документирована, предшествующая строчка должна начинаться со знака #. Перечислите названия всех функций, не снабженных комментариями, вместе с именем файла и номером строки, с которой начинается объявление функции.

Одно или несколько имен файлов с кодом на языке Python пользователь должен передать в функцию в качестве аргументов командной строки. Для файлов, которые не существуют или не могут быть открыты, должны выдаваться соответствующие предупреждения, после чего должна быть продолжена обработка остальных файлов.

Упражнение 39

Существует как минимум одно слово в английском языке, содержащее все гласные буквы в том порядке, в котором они расположены в алфавите, а именно A, E, I, O, U и Y. Напишите программу, которая будет просматривать текстовый файл на предмет поиска и отображения таких слов. Имя исходного файла должно быть запрошено у пользователя. Если имя файла окажется неверным или возникнут иные ошибки при чтении файла, выведите соответствующее сообщение об ошибке.

Упражнение 40

Ширина окна терминала обычно составляет 80 символов, хотя есть более широкие и узкие терминалы. Это затрудняет отображение текстов, разбитых на абзацы. Бывает, что строки в исходном файле оказываются слишком длинными и автоматически переносятся, тем самым затрудняя чтение, или слишком короткими, что приводит к недостаточному заполнению свободного места на экране.

Напишите программу, которая будет открывать файл и выводить его на экран с постоянной длиной строк. Если в исходном файле строка оказывается слишком длинной, «лишние» слова должны быть перенесены на следующую строку, а если слишком короткой, слова со следующей строки должны перекочевать в конец текущей до полного ее заполнения.

Убедитесь, что ваша программа корректно обрабатывает тексты, в которых присутствуют абзацы. Идентифицировать конец одного абзаца и начало другого можно по наличию пустой строки в тексте после удаления символа конца строки.

Упражнение 41

Фонетический алфавит представляет собой таблицу обозначений букв, каждой из которых соответствует то или иное слово. Широкое распространение такие алфавиты приобретают в условиях повышенной зашумленности каналов передачи информации, когда собеседник может просто не расслышать конкретную букву. В таких случаях вместо букв используются целые слова. Один из наиболее распространенных фонетических алфавитов был разработан в военном блоке НАТО. Соответствие букв и слов в нем приведено в таблице ниже.

Напишите программу, которая будет запрашивать слово у пользователя и отображать его на экране в виде шифра из соответствующих слов, обозначающих буквы исходного текста. Например, если пользователь введет слово Hello, на экране должна быть отображена следующая последовательность слов: Hotel Echo Lima Lima Oscar. Для решения этой задачи вам предстоит использовать рекурсивную функцию, а не циклы. При этом все небуквенные символы, введенные пользователем, можно игнорировать.

Фонетический алфавит НАТО:

Буква	Слово	Буква	Слово	Буква	Слово
A	Alpha	J	Juliet	S	Sierra
B	Bravo	K	Kilo	T	Tango
C	Charlie	L	Lima	U	Uniform
D	Delta	M	Mike	V	Victor
E	Echo	N	November	W	Whiskey
F	Foxtrot	O	Oscar	X	Xray
G	Golf	P	Papa	Y	Yankee
H	Hotel	Q	Quebec	Z	Zulu
I	India	R	Romeo		

Упражнение 42

Как ясно из названия, римские цифры появились еще в Древнем Риме. Но даже после падения Римской империи они продолжали использоваться на территории Европы вплоть до позднего Средневековья, а в определенных случаях применяются и сегодня.

Римские цифры базируются на обозначениях M, D, C, L, X, V и I, соответствующих числам 1000, 500, 100, 50, 10, 5 и 1. В основном римские цифры в составляющих их числах располагаются в порядке убывания – от больших к меньшим. В этом случае итоговое число равно сумме всех составляющих его римских цифр. Если цифры меньшего номинала предшествуют цифрам большего номинала, то первые вычитаются из вторых и итог прибавляется к общей сумме. В такой манере могут использоваться римские цифры C, X и I. При этом вычитание производится только из чисел, максимум в десять раз превосходящих вычитаемое. Таким образом, цифра I может предшествовать V или X, но не может вычитаться из L, C, D или M. А значит, число 99 должно быть написано как XCIX, а не IC. Создайте рекурсивную функцию, которая будет переводить римскую запись чисел в десятичную. Функция должна обрабатывать один или два символа в начале строки, после чего будет производиться ее рекурсивный вызов для оставшихся символов. Используйте пустую строку с возвращаемым значением 0 в качестве базового случая. Также напишите основную программу, которая будет запрашивать у

пользователя число, введенное римскими цифрами, и отображать его десятичный эквивалент. При этом можно сделать допуск о том, что пользователь всегда вводит корректное число, так что обработку ошибок вам реализовывать не нужно.

Упражнение 43

В данном упражнении вам предстоит написать рекурсивную функцию, определяющую, является ли переданная ей в виде аргумента строка палиндромом. Стоит отметить, что пустая строка является палиндромом по определению, как и строка, состоящая из одного символа. Более длинные строки можно считать палиндромами, если их первый и последний символы одинаковые, а подстрока, исключающая эти символы, также является палиндромом.

Напишите основную программу, которая будет запрашивать у пользователя слово и при помощи рекурсивной функции определять, является ли оно палиндромом. В результате на экране должно появиться соответствующее сообщение.

Упражнение 44

Редакционное расстояние между двумя строками представляет собой меру их схожести. Чем меньше между строками редакционное расстояние, тем более похожими они могут считаться. Это означает, что одна строка может быть преобразована в другую с минимальным количеством операций вставки, удаления и замены символов.

Рассмотрим слова *kitten* и *sitting*. Первое слово может быть трансформировано во второе путем выполнения следующих операций: заменить k на s, заменить e на i и добавить g в конец строки. Это минимально возможное количество операций, необходимое для преобразования слова *kitten* в *sitting*. Таким образом, редакционное расстояние между этими строками составляет 3.

Напишите рекурсивную функцию, вычисляющую редакционное расстояние между двумя строками по следующему алгоритму.

Имеем исходные строки s и t
Если длина строки s равна нулю, тогда
    Возвращаем длину строки t
Если длина строки t равна нулю, тогда
    Возвращаем длину строки s
Иначе
    Устанавливаем переменную cost равной 0
    Если последний символ в s не совпадает с последним символом в t, тогда
        Устанавливаем cost равной 1
    Устанавливаем d1 равной редакционному расстоянию между строкой s без последнего
символа и строкой t с прибавлением единицы
    Устанавливаем d2 равной редакционному расстоянию между строкой t без последнего
символа и строкой s с прибавлением единицы
    Устанавливаем d3 равной редакционному расстоянию между строкой s без последнего
символа и строкой t без последнего символа с прибавлением cost
    Возвращаем минимальное значение среди d1, d2 и d3

Используйте написанную вами рекурсивную функцию в основной программе, запрашивающей у пользователя две строки и выводящей на экран редакционное расстояние между ними.

Упражнение 45

Каждый химический элемент из таблицы Менделеева характеризуется своим обозначением, состоящим из одного, двух или трех символов. Есть такая игра – пытаться выразить то или иное слово через химические эле- менты. Например, слово silicon может быть записано при помощи следующих химических элементов: Si, Li, C, O и N. В то же время слово hydrogen не может быть составлено из названий элементов.

Напишите рекурсивную функцию, способную определять, можно ли выразить переданное ей слово исключительно через обозначения химических элементов. Ваша функция должна

принимать два параметра: слово, которое нужно проверить, и список символов, которые можно при этом использовать. Возвращать функция должна строку, состоящую из использованных символов, если собрать искомое слово можно, и пустую строку в противном случае. При этом регистры символов учитываться не должны.

В основной программе должна быть использована ваша функция для проверки всех элементов таблицы Менделеева на возможность составить их названия из обозначений химических элементов. Отобразите на экране названия элементов вместе с обозначениями, которые были использованы для их написания. Например, одна из строчек может выглядеть так:

Silver может быть представлен как SiLvEr

Упражнение 46

Существует такая игра, в которой химические элементы выстраиваются в длинную цепочку таким образом, чтобы каждое очередное слово начиналось на букву, на которую заканчивается предыдущее. Например, если последовательность начинается с элемента Hydrogen, то следующий элемент должен начинаться на N, и это может быть Nickel. Следом может идти Lithium. При этом элементы в ряду не должны повторяться. При игре в одиночку целью является составление списка элементов максимально возможной длины. Если в игре принимают участие двое, то целью становится назвать такой химический элемент, чтобы оппонент не смог продолжить последовательность.

Напишите программу, которая будет запрашивать у пользователя химический элемент и при помощи рекурсивной функции определять максимально возможную последовательность слов. Выведите на экран полученный ряд. Удостоверьтесь, что программа выводит соответствующее сообщение об ошибке, если пользователь укажет несуществующий химический элемент.

Упражнение 47

Кодирование на основе длин серий представляет собой простую технику сжатия информации, которая демонстрирует свою эффективность при наличии множества соседствующих друг с другом повторяющихся значений. Сжатие достигается за счет замены целой группы повторяющихся значений на однократное его упоминание с отдельно хранящимся счетчиком повторов. Например, список ["A", "A", "A", "A", "A", "A", "A", "A", "A", "A", "A", "A", "A", "B", "B", "B", "B", "A", "A", "A", "A", "A", "A", "B"] может быть закодирован в следующем виде: ["A", 12, "B", 4, "A", 6, "B", 1]. Процесс декодирования заключается в размножении каждого элемента в соответствии со счетчиком.

Напишите рекурсивную функцию для декодирования списка, кодированного на основе длин серий. В качестве единственного аргумента функция должна принимать закодированный соответствующим образом список. На выходе должен получиться расшифрованный список элементов. В основной программе продемонстрируйте работу алгоритма декодирования на примере представленного здесь списка.

Напишите рекурсивную функцию, реализующую алгоритм кодирования на основе длин серий. На вход функции должен поступать список или строка, а на выходе будет закодированный список. В основной программе запросите у пользователя строку, сожмите ее при помощи своей функции и отобразите на экране кодированный список.

20.2 Промежуточная аттестация

Промежуточная аттестация по дисциплине осуществляется с помощью следующих оценочных средств: Практико-ориентированные задания

Пример контрольно-измерительного материала 1

Напишите программу, которая находит рекордное количество вхождений (не обязательно подряд) символа в строку.

Формат ввода Вводится одна строка.

Формат вывода Выводится одно целое число — максимальное количество раз, которое встречается какая-либо буква (без учёта регистра) или иной символ во введённой строке.

Пример

Ввод	Вывод
Длинношееееед	4

Описание технологии проведения

Обучающемуся выдаётся КИМ, содержащий практическое задание и блок теоретических вопросов

Требования к выполнению заданий (или шкалы и критерии оценивания)

Для оценивания результатов обучения на экзамене используются следующие содержательные показатели:

1. знание теоретических основ учебного материала, основных определений, понятий и используемой терминологии;
2. умение проводить обоснование и представление основных теоретических и практических результатов (теорем, алгоритмов, методик) с использованием математических выкладок, блок-схем, структурных схем и стандартных описаний к ним;
3. умение связывать теорию с практикой, иллюстрировать ответ примерами, в том числе, собственными, умение выявлять и анализировать основные закономерности, полученные, в том числе, в ходе выполнения лабораторно-практических заданий;
4. умение обосновывать свои суждения и профессиональную позицию по излагаемому вопросу;
5. владение навыками проведения компьютерного эксперимента, тестирования алгоритмов.

Различные комбинации перечисленных показателей определяют критерии оценивания результатов обучения (сформированности компетенций) на экзамене: высокий (углубленный) уровень сформированности компетенций; повышенный (продвинутый) уровень сформированности компетенций; пороговый (базовый) уровень сформированности компетенций.

Для оценивания результатов обучения на дифференцированном зачете используется – зачтено с оценкой / не зачтено по результатам сдачи лабораторных работ и ответов на тестовые задания.

Для оценивания результатов обучения на экзамене используется 4-балльная шкала: «отлично», «хорошо», «удовлетворительно», «неудовлетворительно».

Соотношение показателей, критериев и шкалы оценивания результатов обучения на экзамене представлено в следующей таблице.

Критерии оценивания компетенций	Уровень сформированности компетенций	Шкала оценок
Полное соответствие ответа обучающегося всем перечисленным критериям. Продемонстрировано знание принципов создания электронных языковых ресурсов, умение пользоваться такими ресурсами, владение принципами создания электронных языковых ресурсов.	Повышенный уровень	Отлично
Ответ на контрольно-измерительный материал не соответствует одному (двум) из перечисленных показателей, но обучающийся дает правильные ответы на дополнительные вопросы. Недостаточно продемонстрировано знание принципов создания электронных языковых ресурсов, умение пользоваться такими ресурсами, владение принципами создания электронных языковых ресурсов.	Базовый уровень	Хорошо
Ответ на контрольно-измерительный материал не соответствует любым двум (трем) из перечисленных показателей, обучающийся дает неполные ответы на дополнительные вопросы. Демонстрирует частичные знания принципов создания электронных языковых ресурсов, умение пользоваться такими ресурсами, владение принципами создания электронных языковых ресурсов.	Пороговый уровень	Удовлетворительно
Ответ на контрольно-измерительный материал не	–	Неудовлетворительно

соответствует любым трем (четырем) из перечисленных показателей. Обучающийся демонстрирует отрывочные, фрагментарные знания, допускает грубые ошибки.		

Задания разделов рекомендуются к использованию при проведении диагностических работ с целью оценки остаточных знаний по результатам освоения данной дисциплин.