Рабочая программа дисциплины (модуля)*

Методы трансляции и формальные языки

Направление подготовки (специальность)

01.03.02 Прикладная математика и информатика

Направленность (профиль)

Прикладная математика и искусственный интеллект

Присваиваемая квалификация (степень)

бакалавр

Форма обучения

очная

Год набора 2024

*Рабочая программа дисциплины (модуля) адаптирована для инклюзивного обучения инвалидов и лиц с ограниченными возможностями здоровья

Челябинск 2023 г.

Содержание

1. ЦЕЛИ ОСВОЕНИЯ ДИСЦИПЛИНЫ

Преподаваемая дисциплина является средством решения некоторых прикладных и системных задач. Преподавание и изучение дисциплины следует рассматривать как важную составляющую профессиональной подготовки. Целью дисциплины является обучение студентов применению теории формальных языков и автоматов, методов и алгоритмов лексического и синтаксически управляемого разбора при создании системного и прикладного обеспечения.

Задачами дисциплины заключаются в том, чтобы ознакомить студентов с формальными методами описания структуры текстовой информации и синтаксиса языков программирования; дать представление о проблемах и направлениях исследований в области языков программирования; дать практические навыки по использованию современных библиотек и CASEсредств для обработки текстовой информации, для разработки компиляторов и интерпретаторов проблемно-ориентированных языков.

Результаты обучения по дисциплине направлены на достижение индикаторов, соответствующих компетенции ПК-1:

ПК-1.1. Обладает базовыми знаниями модели описания формальных языков, в том числе и языков программирования; задачи и этапы построения трансляторов;

ПК-1.2. Демонстрирует умения построения грамматики формального языка и преобразования её к требуемому виду для построения лексического и синтаксического анализаторов;

ПК-1.3. Имеет практический опыт проектирования компиляторов для архитектур семейства Intel;

2. МЕСТО ДИСЦИПЛИНЫ В СТРУКТУРЕ ОПОП

Цикл (раздел) ОПОП:	ФТД.02

2.1 Требования к предварительной подготовке обучающегося:

Знает: [УК-1.1. З-3.] современное состояние информационно-коммуникационных технологий в мире и перспективы их развития, принципы работы современных информационных технологий, [ОПК-1.1. З-1.] рынок информационных систем и информационно-коммуникационных технологий, автоматизирующих организационно-технические и экономические процессы, [ПК-2.1. З-1.] основные программные платформы и компоненты систем искусственного интеллекта: механизмы логического вывода (рассуждений), объяснений, приобретения знаний, интеллектуальных интерфейсов, принципы Data Ops и Dev Ops, современные компьютерные технологии разработки программных систем

Современные технологии разработки программных систем искусственного интеллекта

2.2 Дисциплины и практики, для которых освоение данной дисциплины (модуля) необходимо как предшествующее:

3. КОМПЕТЕНЦИИ ОБУЧАЮЩЕГОСЯ, ФОРМИРУЕМЫЕ В РЕЗУЛЬТАТЕ ОСВОЕНИЯ ДИСЦИПЛИНЫ (МОДУЛЯ)

ПК-1: Способен активно участвовать в разработке системного и прикладного программного обеспечения

Знать:

Для достижения ПК-1.1:
модели описания формальных языков, в том числе и языков программирования; задачи и этапы построения трансляторов

Уметь:

Для достижения ПК-1.2:
построить грамматику формального языка и преобразовать её к требуемому виду для построения лексического и синтаксического анализаторов

Владеть:

Для достижения ПК-1.3:
навыками проектирования компиляторов для архитектур семейства Intel

В результате освоения дисциплины обучающийся должен

3.1	**Знать:**
3.1.1	методы описания структуры текстовой информации и синтаксиса языков программирования;
3.1.2	этапы построения трансляторов и задачи их работы;
3.1.3	сведения о проблемах и направлениях исследований в области языков программирования.

3.2	**Уметь:**
3.2.1	пользоваться современными библиотеками и CASEсредств для обработки текстовой информации, для разработки компиляторов и интерпретаторов проблемно-ориентированных языков;
3.2.2	преобразовывать грамматику формальных языков к необходимому виду для построения анализаторов лексического и синтаксического вида.
3.3	**Владеть:**
3.3.1	применения теории формальных языков и автоматов, методов и алгоритмов лексического и синтаксически управляемого разбора при создании
3.3.2	системного и прикладного обеспечения.

4. ОБЪЕМ ДИСЦИПЛИНЫ (МОДУЛЯ)

Общая трудоемкость	**4 ЗЕТ**
Часов по учебному плану : 144 в том числе : аудиторные занятия : 64 самостоятельная работа : 71,5 : контактная работа: 72,5 ИКР: 8,5	Виды контроля в семестрах: зачеты с оценкой 7

5. СТРУКТУРА И СОДЕРЖАНИЕ ДИСЦИПЛИНЫ (МОДУЛЯ)

Код занятия	Наименование разделов и тем /вид занятия/	Семестр / Курс	Часов	Литература
	Раздел 1. Формальные языки и грамматики			
1.1	Определение транслятора, компилятора и интерпретатора. Универсальные и проблемно ориентированные языки программирования. Общая схема работы трансляторов. Способы определения языков. Язык как множество цепочек символов. Грамматики. Синтаксис и семантика языка программирования. Классификация грамматик и языков по Хомскому. /Лек/	7	2	Л1.1Л2.1 Л2.2 Э1
1.2	Задача распознавания принадлежности цепочки языку. Дерево вывода. Левосторонний и правосторонний выводы. Способы задания грамматик. Проблемы однозначности и эквивалентности грамматик. Классификация распознавателей. /Лек/	7	2	Л1.1Л2.1 Л2.2 Э1
1.3	Разработка грамматики для заданной конструкции языка. Отличия БНФ и РБНФ. /Пр/	7	2	Л1.1Л2.1 Л2.2 Э1
	Раздел 2. Регулярные языки и лексический анализ			
2.1	Регулярные языки и грамматики. Свойства регулярных языков. Лемма о разрастании для регулярного языка. Проблемы, разрешимые для регулярных языков. Регулярные и автоматные грамматики. Детерминированные и недетерминированные конечные автоматы. Преобразование конечного автомата к детерминированному виду. Минимизация конечных автоматов. /Лек/	7	2	Л1.1Л2.1 Л2.2 Э1
2.2	Регулярные множества и регулярные выражения. Основные свойства, алгебра регулярных выражений. Эквивалентность регулярных грамматик, конечных автоматов и регулярных выражений. /Лек/	7	2	Л1.1Л2.1 Л2.2 Э1
2.3	Применение регулярных выражений для обработки текста в Python и C++. Основные библиотечные функции. Особенности реализации регулярных выражений в языках программирования. /Лек/	7	2	Л1.1Л2.1 Л2.2 Э1
2.4	Средство для автоматизации построения лексических анализаторов (сканеров) FLEX (RE/flex), назначение и принципы. Автономное использование FLEX для обработки текста. /Лек/	7	2	Л1.1Л2.1 Э1

2.5	Построение конечного автомата (КА) для по заданной грамматике. /Пр/	7	2	Л1.1Л2.1 Э1
2.6	Тест-практика по регулярным выражениям. /Пр/	7	2	Л1.1Л2.1 Э1
2.7	Использование FLEX для обработки текста. Порядок правил, выделение подавтоматов. /Пр/	7	2	Л1.1Л2.1 Э1
2.8	Симуляция КА в Umlet. Генерация класса по диаграмме состояний. /Лаб/	7	2	Л1.1Л2.1 Э1
2.9	Преобразование недетерминированного КА к детерминированному. Минимизация конечных автоматов. /Лаб/	7	2	Л1.1Л2.1 Э1
2.10	Применение расширенных регулярных выражений в C++ и Python для обработки текста. /Лаб/	7	2	Л1.1Л2.1 Э1
2.11	Решение задач по обработке текста на FLEX. /Лаб/	7	2	Л1.1Л2.1 Э1
	Раздел 3. Контекстно-свободные языки и синтаксический анализ			
3.1	Контекстно-свободные грамматики. Свойства контекстно-свободных языков. Назначение и принципы работы синтаксических анализаторов. Автоматы с магазинной памятью. Преобразование КС грамматик. Классификация распознавателей для КС-языков. /Лек/	7	2	Л1.1Л2.1 Э1
3.2	Определение LL(k)-грамматики. Множества FIRST и и FOLLOW. Построение нисходящего распознавателя для LL(1)-грамматики методом рекурсивного спуска. /Лек/	7	2	Л1.1Л2.1 Э1
3.3	Определение LR(k)-грамматики. Построение восходящего распознавателя для LR(1)-грамматики. Алгоритм сдвиг-свертка. Средство для автоматизации построения синтаксических анализаторов (парсеров) BISON. /Лек/	7	2	Л1.1Л2.1 Э1
3.4	Разработка распознавателя для LL(1)-грамматик. Преобразование грамматики. Определение множеств FIRST и и FOLLOW. /Пр/	7	2	Л1.1Л2.1 Э1
3.5	Определение правил грамматики в BISON. /Пр/	7	2	Л1.1Л2.1 Э1
3.6	Определение правил грамматики в BISON. /Лаб/	7	2	Л1.1Л2.1 Э1
	Раздел 4. Структура компиляторов и интерпретаторов, этапы трансляции, CASE-стредства для разработки			
4.1	Основные принципы построения трансляторов. Этапы трансляции. Особенности построения интерпретаторов. Технология JIT. Трансляторы с языка ассемблера. Макроязыки и макрогенерация. Таблицы идентификаторов. /Лек/	7	2	Л1.1Л2.1 Э1
4.2	Проблемы интеграции FLEX и BISON. Атрибутные грамматики. Абстрактное синтаксическое дерево (AST), представление и обработка. /Лек/	7	2	Л1.1Л2.1 Э1
4.3	CASE-средства для автоматизации разработки трансляторов, построения и обработки AST. /Лек/	7	2	Л1.1Л2.1 Э1
4.4	Семантический анализ. Методы генерации и оптимизации кода. /Лек/	7	2	Л1.1Л2.1 Э1
4.5	Назначение, основные принципы организации LLVM. Интерфейс LLVM API, основные классы. /Лек/	7	2	Л1.1Л2.1 Э1
4.6	Назначение, основные принципы организации LLVM. Интерфейс LLVM API, основные классы. /Лек/	7	2	Л1.1Л2.1 Э1
4.7	JIT технология в JVM и LLVM. Инструменты LLVM JIT. Выполнение кода в LLVM. /Лек/	7	2	Л1.1Л2.1 Э1
4.8	Установка LLVM. Подключение LLVM к транслятору. Генерация кода для процессоров семейства Intel, выбор архитектуры. /Пр/	7	2	Л1.1Л2.1 Э1
4.9	Разработка интерпретатора в LLVM с использованием технологии JIT. Сравнение скорости работы с обычным интерпретатором. /Пр/	7	2	Л1.1Л2.1 Э1

4.10	Определение сканера для учебного языка. Проверка сканера. /Лаб/	7	2	Л1.1Л2.1 Э1
4.11	Определение парсера для учебного языка. Проверка парсера. /Лаб/	7	2	Л1.1Л2.1 Э1
4.12	Определение и обработка AST для учебного языка. Проверка интерпретатора. /Лаб/	7	2	Л1.1Л2.1 Э1
	Раздел 5. Иная контактная работа			
5.1	Консультации и промежуточная аттестация. /КурсР/	7	8,5	Л1.1Л2.1 Э1
	Раздел 6. Самостоятельная работа			
6.1	Установка LLVM, разработка компилятора с языка высокого уровня по вариантам с использованием LLVM. /Ср/	7	38,5	Л1.1Л2.1 Э1
6.2	Изучение материала лекций и литературы, подготовка к практическим занятиям. /Ср/	7	20	Л1.1Л2.1 Э1
6.3	Подготовка к диф.зачету. /Ср/	7	9	Л1.1Л2.1 Э1
6.4	Подготовка к тесту по регулярным выражениям. /Ср/	7	4	Л1.1Л2.1 Э1

6. ФОНД ОЦЕНОЧНЫХ СРЕДСТВ

6.1. Перечень видов оценочных средств

Лабораторные работы;
Решение тестов текущего контроля;
Вопросы для подготовки к зачету;

6.2. Типовые контрольные задания и иные материалы для текущей аттестации

Практическая работа 1. Разработка грамматики для заданной конструкции языка. Отличия БНФ и РБНФ.
Практическая работа 3. Тест-практика по регулярным выражениям.
Практическая работа 7. Установка LLVM. Подключение LLVM к транслятору. Генерация кода для процессоров семейства Intel, выбор архитектуры.
Практическая работа 8. Разработка интерпретатора в LLVM с использованием технологии JIT. Сравнение скорости работы с обычным интерпретатором.
Лабораторная работа 1. Симуляция КА в Umlet. Генерация класса по диаграмме состояний.
Лабораторная работа 2. Преобразование недетерминированного КА к детерминированному. Минимизация конечных автоматов.
Лабораторная работа 3. Применение расширенных регулярных выражений в C++ и Python для обработки текста.
Лабораторная работа 4. Решение задач по обработке текста на FLEX.
Лабораторная работа 5. Решение задач по синтаксическому анализу в BISON.
Лабораторная работа 6. Определение сканера для учебного языка. Проверка сканера.
Лабораторная работа 7. Определение парсера для учебного языка. Проверка парсера.
Лабораторная работа 8. Определение и обработка AST для учебного языка. Проверка интерпретатора.
Образец теста приведен в приложении.
Образец практической и лабораторной работы в приложении.

6.3. Типовые контрольные вопросы и задания для промежуточной аттестации

Вопросы к диф.зачету по дисциплине «Методы трансляции и формальные языки»
Вопрос 1 по теме «Теория формальных языков»
1. Алфавит. Цепочки символов в алфавите. Операции над цепочками символов. Формальное определение грамматики и языка.
2. Формальное определение языка. Операции над языками. Синтаксис и семантика языка.
3. Способы задания синтаксиса формальных языков.
4. Классификация языков и грамматик по Хомскому.
5. Разбор цепочек КС-грамматик. Вывод, цепочка вывода, сентенциальная форма. Дерево вывода.
6. Левосторонний и правосторонний выводы. Способы задания грамматик. Проблемы однозначности и эквивалентности грамматик.
7. Общая схема распознавателя. Классификация распознавателей.
8. Свойства регулярных языков. Лемма о разрастании для регулярного языка. Проблемы, разрешимые для

регулярных языков.
9. Регулярные и автоматные грамматики. Детерминированные и недетерминированные конечные автоматы.
10. Преобразование конечного автомата к детерминированному виду. Минимизация конечных автоматов.
11. Регулярные множества и регулярные выражения. Основные свойства, алгебра регулярных выражений. Эквивалентность регулярных грамматик, конечных автоматов и регулярных выражений.
12. Контекстно-свободные грамматики. Свойства контекстно-свободных языков. Назначение и принципы работы синтаксических анализаторов. Автоматы с магазинной памятью.
13. Преобразование КС-грамматик. Классификация распознавателей для КС-языков. Определение LL(k)-грамматики. Множества FIRST и и FOLLOW. Метод рекурсивного спуска.
14. Определение LR(k)-грамматики. Построение восходящего распознавателя для LR(1)-грамматики. Алгоритм сдвиг - свертка.

Вопрос 2 по теме “Теория перевода и построение трансляторов»
15. Лексика, синтаксис и семантика языка. Универсальные и проблемно-ориентированные языки программирования.
16. Определение транслятора, интерпретатора, компилятора и ассемблера.
17. Общая схема работы транслятора. Этапы трансляции
18. Особенности построения интерпретаторов. Технология JIT
19. Организация таблиц идентификаторов
20. Способы внутреннего представления программы.
21. Атрибутные грамматики. Абстрактное синтаксическое дерево (AST), представление и обработка.
22. Семантический анализ. Методы генерации и оптимизации кода.
23. Основные библиотечные функции для регулярных выражений. Особенности реализации регулярных выражений в языках программирования.
24. Генератор лексических анализаторов FLEX. Назначение и принципы
25. Генератор синтаксических анализаторов BISON.
26. Назначение, основные принципы организации LLVM.
27. Интерфейс LLVM API, основные классы
28. Промежуточное представление LLVM IR. Выбор целевой машины и генерация машинного кода.

6.4. Критерии оценивания

Учебным планом по данной дисциплине предусмотрен дифференцированный зачет.

В течение учебного семестра студенты за каждый вид работы получают баллы. Итоговая оценка складывается из суммы баллов, полученных за работу в семестре и за ответ на зачете.
Итоговая оценка выставляется, исходя из набранной суммы баллов:
От 0 до 49 баллов – «неудовлетворительно»
50-69 баллов – «удовлетворительно»
70-90 баллов – «хорошо»
91-100 баллов – «отлично».

Описание распределения баллов за выполнение работ контрольных мероприятий:

Текущий контроль:
Р1. Формальные языки и грамматики. (Максимально 10 баллов)
Задание выполнено вовремя - 2 балла, иначе 0 баллов;
Учтены все варианты синтаксиса в правилах - 3 балла, оценка снижается на 1 балл за каждый отсутствующий вариант;
Нет синтаксических ошибок в формате правил и логических ошибок - 5 баллов, оценка снижается на 1 балл за каждую ошибку.

Р2. Конечные автоматы для распознавания цепочек регулярного языка. (Максимально 10 баллов)
Выполнена проверка КА путем симуляции в UMLet- 3 балла, иначе 0 баллов;
Выполнена генерация класса для КА по диаграмме состояний - 2 балла, иначе 0 баллов;
Нет синтаксических и логических ошибок в схеме КА - 5 баллов, оценка снижается на 1 балл за каждую ошибку.

Р3 Регулярные выражения. (Максимально 10 баллов)
Решить 5 задач на C++ или Python с использованием функций библиотеки для регулярных выражений.
По 2 баллу за каждую решенную задачу, за частично правильное решение 1 балл, иначе 0 баллов.

Тест по регулярным выражениям. (Максимально 10 баллов)

Тест включает 15 вопросов разной сложности, для выполнения задания необходимо ответить правильно на не менее 10 вопросов.
По 1 баллу за правильный ответ на вопрос теста;
Баллы свыше 10 за правильные ответы переносятся в бонусы.

Р5. Синтаксический анализ. (Максимально 10 баллов)
Решить 5 задач с помощью инструмента для синтаксического анализа BISON.
По 2 баллу за каждую решенную задачу, за частично правильное решение 1 балл, иначе 0 баллов.

Р6. Разработка интерпретатора. (Максимально 20 баллов)
Решена подзадача 1 (лексический анализ) - 3 балла, оценка снижается на 1 балл за каждую ошибку;
Решена подзадача 2 (синтаксический анализ) - 3 балла, оценка снижается на 1 балл за каждую ошибку;
Решена подзадача 3 (построение AST, обход AST и интепретация) - 4 балла, оценка снижается на 1 балл за каждую ошибку.

Р7. Разработка компилятора на LLVM. (Максимально 10 баллов)
Выполнена установка библиотек LLVM и подключение к компилятору - 2 балла, иначе 0 баллов;
Выполнена генерация кода для процессора Intel в компиляторе для учебного языка - 4 балла, оценка снижается на 1 балл за каждую ошибку;
Выполнено тестирование компилятора - 2 балла, иначе 0 баллов;
Выполнено сравнение скорости программы на учебном языке и на языке С - 2 балла, иначе 0 баллов.

Р8. Использование JIT-технологии. (Максимально 10 баллов)
Выполнена генерация внутреннего представления кода для JIT интерпретатора учебного языка - 4 балла, оценка снижается на 1 балл за каждую ошибку;
Выполнено тестирование интерпретатора - 3 балла, иначе 0 баллов;
Выполнено сравнение скорости работы скомпилированного кода, простого и JIT интерпретатора - 3 балла, если выполнено сравнение только двух вариантов, то 2 балла, иначе 0 баллов.

Текущая аттестация:
Р4 Лексический анализ. (Максимально 10 баллов)
Решить 5 задач с помощью инструмента для лексического анализа FLEX.
По 2 баллу за каждую решенную задачу, за частично правильное решение 1 балл, иначе 0 баллов.

Дифференцированный зачет. (Максимально 10 баллов)
Это контрольное мероприятие проводится в форме собеседования. Задаются два вопроса по пройденным темам.
В первую очередь предлагаются вопросы по темам, которые были оценены на "неудовлетворительно" по текущему контролю.
Каждый ответ оценивается от 0 до 5 баллов в зависимости от полноты ответа, знания терминов.
Шкала оценивания:
Полный, правильный ответ - 5 баллов;
Одна неточность, неправильный термин - 4 балла;
Частичный ответ - 3 балла;
В ответе есть некоторые правильные определения - 2 балла;
Нет ответа - 0 баллов;
Оценка ставится как сумма баллов за оба ответа.

Бонус. Посещаемость и активность на занятиях. (Максимально 15 баллов)
Посещение занятий:
100% - 3 балла;
85% -2 балла;
менее 85% - 0 баллов;
Активность на занятиях:
Правильный ответ у доски - 1 балл;
Помощь с места - 0,2 балла;
Не более 3 баллов в сумме.
Выполнено расширенное задание Р6 (часть I) - 5 баллов.
Дополнительные баллы за тест (11+) - 1 - 5 баллов.

7. УЧЕБНО-МЕТОДИЧЕСКОЕ И ИНФОРМАЦИОННОЕ ОБЕСПЕЧЕНИЕ ДИСЦИПЛИНЫ (МОДУЛЯ)

7.1. Рекомендуемая литература

7.1.1. Основная литература

	Авторы, составители	Заглавие	Издательство, год	Ресурс
Л1.1	Бруно К. Л., Рафаэль А.	LLVM: инфраструктура для разработки компиляторов (https://e.lanbook.com/book/90119)	Москва : ДМК Пресс, 2015	ЭБС

7.1.2. Дополнительная литература

	Авторы, составители	Заглавие	Издательство, год	Ресурс
Л2.1	Вирт Н.	Построение компиляторов (http://e.lanbook.com/books/element.php? pl1_cid=25&pl1_id=1262)	Москва : ДМК Пресс, 2010	ЭБС
Л2.2	Джеффри К. Л.	Создаи? свои? собственныи? язык программирования. Руководство программиста по разработке компиляторов, интерпретаторов и доменно-ориентированных языков для решения современных вычислительных задач (https://e.lanbook.com/book/314933)	Москва : ДМК Пресс, 2023	ЭБС

7.2. Перечень ресурсов информационно-телекоммуникационной сети "Интернет"

Э1	Лань [Электронный ресурс] : электронно-библиотечная система (ЭБС) / издательство Лань. – URL: http://e.lanbook.com/.

7.3 Перечень информационных технологий

7.3.1 Программное обеспечение

LMS Moodle

7.3.2 Профессиональные базы данных и информационно-справочные системы

1. Научная электронная библиотека eLIBRARY.RU (https://elibrary.ru/defaultx.asp?) eLIBRARY.RU : научная электронная библиотека : сайт. – Москва, 2000 – . – URL: https://elibrary.ru. – Режим доступа: для зарегистрир. пользователей. – Текст : электронный.

2. Электронный каталог научной библиотеки ЧелГУ [Электронный ресурс] : база данных / Челяб. гос. ун-т. – Челябинск, 1992 .

8. МАТЕРИАЛЬНО-ТЕХНИЧЕСКОЕ ОБЕСПЕЧЕНИЕ ДИСЦИПЛИНЫ (МОДУЛЯ)

Для реализации дисциплины используются учебные аудитории для проведения занятий лекционного типа, групповых и индивидуальных консультаций, текущего контроля и промежуточной аттестации, компьютерные классы для проведения лабораторных работ, а также помещения для самостоятельной работы.

Учебные аудитории укомплектованы специализированной мебелью (подразумевается наличие стандартных рабочих (посадочных) мест) и техническими средствами обучения (переносное и / или стационарное мультимедийное оборудование: экран, ноутбук, проектор).

Для проведения занятий лекционного типа предлагаются наборы демонстрационного оборудования и учебно- наглядных пособий (мультимедийные презентации по отдельным темам, рисунки, таблицы, схемы и т.д).

Помещения для самостоятельной работы обучающихся оснащены компьютерной техникой с подключением к сети "Интернет" и обеспечением доступа в электронную информационно-образовательную среду университета

Для проведения лабораторных занятий в компьютерных классах должна быть установлена среда MinIDE (https://ipc.susu.ru/learn.html).

9. МЕТОДИЧЕСКИЕ УКАЗАНИЯ ДЛЯ ОБУЧАЮЩИХСЯ ПО ОСВОЕНИЮ ДИСЦИПЛИНЫ (МОДУЛЯ)

Учебным планом предусмотрена самостоятельная работа студентов. Самостоятельная работа проводится с целью углубления знаний по дисциплине и предусматривает:
• проработку теоретического материала по учебникам или конспекту лекций с обязательным разбором приведенных примеров;
• подготовку к лабораторным занятиям;
• подготовку к сдаче зачета.
При планировании времени на самостоятельную работу студентам необходимого предусмотреть регулярное повторение пройденного материала. Теоретический материал, законспектированный на лекциях, необходимо

дополнять сведениями из литературных источников, представленных в рабочей программе.

В случае применения при изучении дисциплины электронного обучения, дистанционных образовательных технологий общение обучающихся и преподавателя осуществляется в режиме реального или отложенного времени, при этом используются возможности системы дистанционного обучения Moodle и электронная почта.
Большую часть времени обучающиеся самостоятельно работают с учебно-методическими материалами. Студенты имеют возможность консультироваться с преподавателем по всем вопросам, возникающим в ходе самостоятельной работы, посредством электронной почты, сообщений системы дистанционного обучения Moodle.
Доступ обучающегося к учебным ресурсам в режиме отложенного времени, самостоятельной работы осуществляется через сеть Интернет в удобном для него месте, времени и темпе.

При обучении лиц с ограниченными возможностями здоровья электронное обучение, дистанционные образовательные технологии предусматривают возможность приема-передачи информации в доступных для них формах.
Реализация дисциплины с применением электронного обучения, дистанционных образовательных технологий (далее – ЭО, ДОТ) осуществляется на основании «Положения о реализации основных и дополнительных образовательных программ с применением электронного обучения и дистанционных образовательных технологий в федеральном государственном бюджетном образовательном учреждении высшего образования «Челябинский государственный университет», «Положения о порядке зачета обучающимся по основным профессиональным образовательным программам высшего образования в ФГБОУ ВО «ЧелГУ» результатов освоения в организациях, осуществляющих образовательную деятельность, учебных предметов, курсов, дисциплин (модулей), практик, дополнительных образовательных программ» посредством электронной информационно-образовательной среды ФГБОУ ВО «ЧелГУ». В исключительных случаях (форс-мажор и т.п.) при реализации образовательной деятельности с применением ЭО, ДОТ могут применять компоненты, не входящие в перечень электронной информационно- образовательной среды.

10. СПЕЦИАЛЬНЫЕ УСЛОВИЯ ОСВОЕНИЯ ДИСЦИПЛИНЫ ОБУЧАЮЩИМИСЯ С ИНВАЛИДНОСТЬЮ И ОГРАНИЧЕННЫМИ ВОЗМОЖНОСТЯМИ ЗДОРОВЬЯ

Освоение дисциплины инвалидами и лицами с ограниченными возможностями здоровья осуществляется с использованием специальных технических средств и голо информационных технологий, предоставляемых Ресурсным учебно-методическим центром по обучению инвалидов и лиц с ограниченными возможностями здоровья ЧелГУ по запросу обучающегося.
1. Мобильные специальные технические средства для лиц с нарушениями зрения: портативный компьютер с вводом/выводом шрифтом Брайля с синтезатором речи «ElBraile-W14J G2»; ноутбуки с программной экранного доступа NVDA; электронные увеличители для удаленного просмотра; видеоувеличители портативные; тифлоплеер; цифровые диктофоны.
2. Мобильные специальные технические средства для лиц с нарушениями слуха: система свободного звукового поля со встроенной совместимостью с FM-устройствами; радиоклассы «Сонет-РСМ» с передатчиком, заушным индуктором и индукционной петлей; система информационная для слабослышащих переносная «Исток» А2 со встроенным плеером – звуковым информатором; документ-камера; программируемые слуховые аппараты индивидуального пользования.
3. Ассистивные информационные технологии: программное обеспечение экранного доступа с синтезом речи NVDA; программы экранного увеличения; программы речевого синтеза для компьютеров и ноутбуков; программы речевого синтеза для мобильных устройств; экранная клавиатура; экранная лупа.
При необходимости для обучающихся с нарушениями зрения на рабочих местах для проведения практических или лабораторных занятий устанавливается специальное программное обеспечение (программа речевой навигации NVDA, речевые синтезаторы, экранные лупы).
В учебные аудитории обеспечивается беспрепятственный доступ для обучающихся инвалидов и обучающихся с ограниченными возможностями здоровья. В каждой аудитории, где обучаются инвалиды и лица с ограниченными возможностями здоровья, предусматривается соответствующее количество мест для обучающихся с учетом нарушений их здоровья.
Для освоения дисциплины инвалидам и лицам с ограниченными возможностями здоровья предоставляется доступ к печатным источникам, имеющимся в научной библиотеке ЧелГУ, с помощью специальных технических средств; доступ к электронным источникам, представленным в форме электронного документа в фонде научной библиотеки ЧелГУ или электронно-библиотечных системах, с помощью специальных технических и программных средств (рабочее место для незрячего пользователя с программным обеспечением экранного доступа с синтезом речи NVDA, рабочее место с компьютерным роллером и клавиатурой Clevy с большими кнопками и с разделяющей клавиши накладкой).
Учебно-методические материалы для обучающихся из числа инвалидов и лиц с ограниченными возможностями

здоровья предоставляются в формах, адаптированных к ограничениям их здоровья и восприятия информации:
Для лиц с нарушениями зрения:
- в печатной форме увеличенным шрифтом,
- в форме электронного документа,
- в форме аудиофайла,
- в печатной форме шрифтом Брайля.

Для лиц с нарушениями слуха:
- в печатной форме,
- в форме электронного документа.

Для лиц с нарушениями опорно-двигательного аппарата:
- в печатной форме,
- в форме электронного документа,
- в форме аудиофайла.

Данный перечень может быть конкретизирован в зависимости от контингента обучающихся.

Для инвалидов и лиц с ограниченными возможностями здоровья освоение дисциплины может быть частично или полностью осуществлено с использованием дистанционных образовательных технологий (Moodle, Adobe Connect Pro и пр.).

В освоении дисциплины инвалидами и лицами с ограниченными возможностями здоровья используется индивидуальная работа. Под индивидуальной работой подразумевается две формы взаимодействия с преподавателем: индивидуальная учебная работа (консультации), т.е. дополнительное разъяснение учебного материала и углубленное изучение материала с теми обучающимися, которые в этом заинтересованы, и индивидуальная воспитательная работа. Индивидуальные консультации направлены на индивидуализацию обучения и установлению воспитательного контакта между преподавателем и обучающимся инвалидом или обучающимся с ограниченными возможностями здоровья.

При проведении процедуры оценивания результатов обучения инвалидов и лиц с ограниченными возможностями здоровья по дисциплине обеспечивается выполнение следующих дополнительных требований в зависимости от индивидуальных особенностей, обучающихся:

а) инструкция по порядку проведения процедуры оценивания предоставляется в доступной форме (устно, в письменной форме, в письменной форме шрифтом Брайля, устно с использованием услуг сурдопереводчика);

б) доступная форма предоставления заданий оценочных средств (в печатной форме, в печатной форме увеличенным шрифтом, в печатной форме шрифтом Брайля, в форме электронного документа, задания зачитываются ассистентом, задания предоставляются с использованием сурдоперевода);

в) доступная форма предоставления ответов на задания (письменно на бумаге, набор ответов на компьютере, письменно шрифтом Брайля, с использованием услуг ассистента, устно).

При проведении процедуры оценивания результатов обучения инвалидов и лиц с ограниченными возможностями здоровья предусматривается использование технических средств, необходимых им в связи с их индивидуальными особенностями. Эти средства могут быть предоставлены ЧелГУ или могут использоваться собственные технические средства. При необходимости инвалидам и лицам с ограниченными возможностями здоровья предоставляется дополнительное время для подготовки ответа на задания, процедура оценивания результатов обучения по дисциплине может проводиться в несколько этапов.

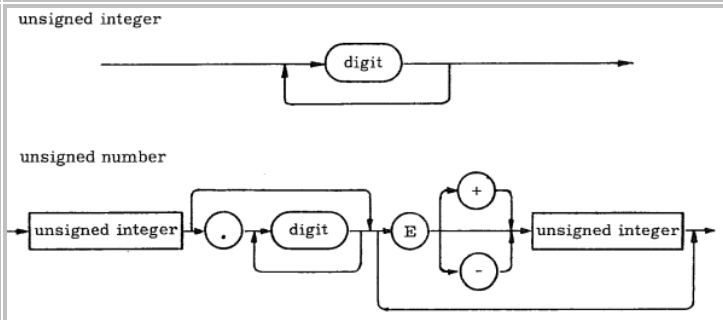
Проведение процедуры оценивания результатов обучения инвалидов и лиц с ограниченными возможностями здоровья допускается с использованием дистанционных образовательных технологий.

Приложения

Контрольное мероприятие №1

№ варианта	Что нужно сделать
1.	Вещественное число в Си (БНФ)
2.	Арифметическое выражение в Паскаль (без сравнений и логических операций) (БНФ)
3.	Логическое выражение в Паскаль (сравнения,and,or,not,без математических действий) (БНФ)
4.	Операторы ветвления в Си (БНФ)
5.	Операторы цикла в Си (БНФ)
6.	Операторы ветвления в Паскаль (БНФ)
7.	Операторы цикла в Паскаль (БНФ)
8.	Оператор объявления переменных в Си (БНФ)
9.	Вещественное число в Си (РБНФ)
10.	Арифметическое выражение в Паскаль (без сравнений и логических операций) (РБНФ)
11.	Логическое выражение в Паскаль (без математических действий) (РБНФ)
12.	Операторы ветвления в Си (РБНФ)
13.	Операторы цикла в Си (РБНФ)
14.	Операторы ветвления в Паскаль (РБНФ)
15.	Операторы цикла в Паскаль (РБНФ)
16.	Оператор объявления переменных в Си (РБНФ)
17.	Целое число (десятичное, шестнадцатеричное и восьмеричное) в Си (БНФ)
18.	Целое число (десятичное, шестнадцатеричное и восьмеричное) в Си (РБНФ)
19.	Операторы ветвления в Python (БНФ)
20.	Операторы цикла в Python (БНФ)
21.	Операторы ветвления в Python (РБНФ)
22.	Операторы цикла в Python (РБНФ)
23.	Односимвольная константа в Си (БНФ)
24.	Строковая константа в Си (БНФ)
25.	Односимвольная константа в Си (РБНФ)
26.	Строковая константа в Си (РБНФ)
27.	
28.	
29.	
30.	

№ варианта	Что нужно сделать
1.	Цепочка символов заканчивается на символ a и содержит четное количество символов a и четное количество символов b
2.	Цепочка символов содержит количество символов a кратное 3 и четное количество символов b
3.	Цепочка символов содержит хотя бы один символ c и содержит нечетное количество символов a и нечетное количество символов b
4.	Определить КА для грамматики $G=(\{a, b, c\}, \{S, A, B, C\}, P, S)$, где $P=\{ S\to aA \mid bB \mid aC; A\to bA \mid bB \mid c; B\to aA \mid cC \mid b; C\to bB \mid bC \mid a\}$
5.	Определить КА для грамматики $G=(\{a,b\}, \{S,A,B\}, P, S)$, где $P=\{S\to aA\|bA; A\to aB\|bB; B\to aB\|a\|bB\|b\}$
6.	Определить КА для грамматики $G=(\{a,b\}, \{S,A,B,C\}, P, S)$, где $P=\{S\to aB; A\to bB; B\to aB\|aC\|bB\|bC; C\to\varepsilon\}$
7.	Определить КА для грамматики $G=(\{a,b,c\}, \{S,A,B,C\}, P, S)$, где $P=\{S\to\varepsilon\|aA; A\to bB\|bS\|c; B\to bC; C\to bA\}$
8.	Определить КА для грамматики $G=(\{a, b, c, d, e\}, \{S, A, B, C, D\}, P, S)$, где $P=\{ S\to aA \mid bB; A\to cC \mid e; C\to cC \mid cA; B\to dD \mid e; D\to dD \mid dB\}$
9.	Определить КА для грамматики $G=(\{a,b\}, \{S,A,B,C\}, P, S)$, где $P=\{S\to aA\|bB\|bC; A\to aA\|bB\|\varepsilon; B\to aC; C\to aC\|\varepsilon\}$
10.	Определить КА для грамматики $G=(\{a,b,c\}, \{S,A\}, P, S)$, где $P=\{S\to abA\|\varepsilon; A\to bbA\|bS\|c\}$
11.	Определить КА для распознавания строки,

№ варианта	Что нужно сделать
	заданной RE для IP4 адреса: (0|[1-9][0-9]*)¥.(0|[1-9][0-9]*)¥.(0|[1-9][0-9]*)¥.(0|[1-9][0-9]*)
12.	Определить КА для распознавания хорошего пароля, который должен содержать хотя одну цифру, одну прописную и одну строчную латинскую букву.
13.	 Определить КА для распознавания числа
14.	Цепочка символов начинается с символа a и содержит нечетное количество символов a и нечетное количество символов b
15.	Цепочка символов содержит количество символов a кратное 3 и хотя бы один символ b
16.	Цепочка символов содержит хотя бы по одному символу b и c и содержит четное количество символов a.
17.	Цепочка из символов a и b не содержит 3 подряд одинаковых символов
18.	Определить КА для распознавания строки, заданной RE для даты: ([0-2][0-9]|3[0-1])¥.(0[0-9]|1[0-2])¥.([0-9]{4})
19.	Определить КА для грамматики G=({*0,1*}, {*S,A,B,C,D*}, *P*, *S*), где *P*={S→1A; A→0S|1B; B→0C; C→0C|1D|1|ε; D→0D}
20.	Определить КА для грамматики G=({*0,1*}, {*S,A,B*}, *P*, *S*),

№ варианта	Что нужно сделать
	где P={S→1S|0A|ε; A→1S|0B; B→0B|1B|ε}
21.	Определить КА для грамматики G=({*0,1*}, {*S,A,B*}, *P*, *S*), где P={S→0A|1B; A→0S|0; B→1S|1}
22.	Определить КА для грамматики G=({*0,1*}, {*S,A,B,C*}, *P*, *S*), где P={S→0A|0|1B; A→0C; B→1S|1; C→0A|0}
23.	Определить КА для грамматики G=({0, 1}, {*S*, *C*, *D*}, *P*, *S*), где P={ *S*→1*C* | 0*D*; *C*→0*D* | 0*S* | 1; *D*→1*C* | 1*S* | 0}
24.	Цепочка из символов a и b содержит 3 подряд одинаковых символов
25.	Определить КА для распознавания плохого пароля, в котором отсутствует цифра или прописная или строчная латинская буква.
26.	Определить КА для распознавания, что цепочка символов является правилом БНФ
27.	
28.	
29.	
30.	

1. Pig-Latin

You have decided that PGP encryption is not strong enough for your email. You have decided to use Pig Latin encryption.

Input and Output

You are to write a program that will take in an arbitrary number of lines of text and output it in Pig Latin. Each line of text will contain one or more words. A "word" is defined as a consecutive sequence of letters (upper and/or lower case). Words should be converted to Pig Latin according to the following rules (non-words should be output exactly as they appear in the input):

1. Words that begin with a vowel (a, e, i, o, or u, and the capital versions of these) should just have the string "`ay`" (not including the quotes) appended to it. For example, "`apple`" becomes "`appleay`".
2. Words that begin with a consonant (any letter than is not A, a, E, e, I, i, O, o, U or u) should have the first consonant removed and appended to the end of the word, and then appending "`ay`" as well. For example, "`hello`" becomes "`ellohay`".
3. Do not change the case of any letter.

Sample Input

```
This is the input.
```

Sample Output

```
hisTay isay hetay inputay.
```

2. Замена в тексте

Миша хочет устроиться на работу копирайтером. Суть этой работы заключается в том, чтобы брать готовые тексты и немного переделывать их.

В качестве отборочного задания Миша получил несколько текстов. Он придумал способ переделывать тексты – менять местами два слова. Т.е., например, из текста "Контора Рога и Копыта" при обмене слов "Рога" и "Копыта" должен получиться текст "Контора Копыта и Рога". При использовании этого метода слово, начинающееся с заглавной буквы, должно заменяться также словом начинающемся с заглавной буквой, а слово, начинающиеся со строчной буквы – также словом, начинающимся со строчной буквы. Слова должны заменяться только целиком. Например, при обмене слов "Рога" и "Копыта" текст "Рогатый" должен остаться без изменения.

Текст состоит только из заглавных и строчных латинских букв, пробелов и символов перевода строки.

Помогите Мише – сгенерируйте для него по исходным текстам результат, в котором будут произведены требуемые замены, а все остальные слова, пробелы и переводы строк останутся без изменения.

В первой строке ввода содержатся два слова, состоящие только из строчных латинских букв, которые нужно поменять. Длины слов не превышают 20 символов. Далее следует текст, не более 200 строк. Длина строк не превышает 2000 символов.

Пример ввода

```
you me
I sometimes wish you were a mermaid
I could raise you in the tub at home
We could take a swim together
On weekly day trips to the bay

Oh you and me
It would be only you and me
Oh you and me
It would be only
You and
Me
```

Пример вывода

```
I sometimes wish me were a mermaid
I could raise me in the tub at home
We could take a swim together
On weekly day trips to the bay

Oh me and you
It would be only me and you
Oh me and you
It would be only
Me and
You
```

3. Марсианская лингвистическая реформа

Алфавит марсианского языка состоит из строчных латинских букв. Буквам a, e, i, o, u, y соответствуют гласные звуки, остальным – согласные.

В результате опроса марсиан выяснилось, что им трудно произносить слова, в которых присутствуют два или более гласных звука подряд. Марсианскими лингвистами было принято решение: во всех словах, где есть два или более гласных звука подряд, оставить только последний из них.

Марсиане просят написать программу, переводящую слова из старого в новый форматы.

Например, если старое слово имело вид `eeaaeeuinfaormaaiatoyuaoics`, то новое слово будет таким: `informatics`.

Формат входного файла

Во входном файле содержится марсианское слово старого формата.

Формат выходного файла

Выходной файл должен содержать марсианское слово нового формата.

Ограничения

Длина слова находится в диапазоне от 1 до 100 символов.

Пример ввода 1

```
eeaaeeuinfaormaaiatoyuaoics
```

Пример вывода 1

```
informatics
```

Пример ввода 2

```
aeeaaayyaaauaaiaaiaacm
```

Пример вывода 2

```
acm
```

Пример ввода 3

```
bzzt
```

Пример вывода 3

```
bzzt
```

4. Слова и числа

Главный бюрократ компании "Планетарная экспресс-доставка" Гермес Конрад после написания очередного отчета или распоряжения обязательно подсчитывает количество слов и чисел в документе, так как владелец компании, профессор Х.Дж.Фарнсворт, не любит лишних слов, а любит только числа. Напишите программу, которая поможет Гермесу подсчитать количество слов и чисел в документе.
Во входном файле содержится от 1 до 100 строк длиной не более 80 символов. Словом или числом будем называть непрерывную последовательность печатных символов. При этом число в отличие от слова содержит одну или более цифр от 0 до 9. Слова и числа в документе разделены последовательностями пробелов, табуляций и переходов на новую строку.
В выходной файл вывести количество слов и чисел в документе в виде "`2 word(s), 1 number(s)`".

Пример ввода

```
             REPORT
In 1st quarter the margin was $20.02.
```

Вывод для примера

```
6 word(s), 2 number(s)
```

5. **Форматирование программы**

Регистр букв в программе на языке Паскаль не имеет значения, но для лучшего выделения структуры программы рекомендуется писать все резервированные слова (AND, ARRAY, BEGIN, CASE, CONST, DIV, DO, DOWNTO, ELSE, END, FILE, FOR, FUNCTION, GOTO, IF, IN, LABEL, MOD, NIL, NOT, OF, OR, PACKED, PROCEDURE, PROGRAM, RECORD, REPEAT, SET, THEN, TO, TYPE, UNTIL, VAR, WHILE, WITH) прописными буквами, а остальные идентификаторы, включая имена встроенных типов и функций, – строчными буквами.
Напишите программу, которая изменяет регистр букв в программе на языке Паскаль в соответствии с этой рекомендацией, оставляя без изменений регистр в комментариях, которые записываются в `{}`, и в строковых константах, которые записываются в `''`.
Ввод содержит синтаксически правильную программу на языке Паскаль. Количество строк не превышает 100. Длина строк не превышает 80 символов. Длина идентификаторов не превышает 20 символов.
Вывод содержит программу с изменением регистра букв резервированных слов и идентификаторов, остальные символы копируются из ввода без изменений.

Пример ввода

```
{ My first program }
Program Hello;
BeGiN
   WRITELN('Hello, world!');
end.
```

Пример вывода

```
{ My first program }
PROGRAM hello;
BEGIN
   writeln('Hello, world!');
END.
```

6. **Комментарии**

Джон, хотя и пишет на языке C, дает файлам расширение CPP, чтобы использовать в своих программах комментарии в C++-стиле (от // до конца строки). Обычный C-комментарий, который начинается с символов "/*" и заканчивается символами "*/" , Джон также иногда использует, обычно для многострочных комментариев. Для участия в конкурсе программ необходимо, чтобы программа соответствовала стандартам языка ANSI C, и Джону нужно заменить все C++-комментарии на стандартные. Для этого в C++-комментарии можно заменить "//" на "/*" и добавить "*/" в конце строки. Иногда в C++-комментарии может встретиться последовательность символов "*/", в этом случае нужно вставить пробел между двумя этими символами: "* /". К счастью внутри строковых констант в программе Джона не встречаются последовательностей "//", "/*" и "*/".

Напишите программу, которая преобразует в программе Джона C++-комментарии в C-комментарии.

Во входном файле содержится программа Джона, не более 100 строк длиной не более 100 символов.

В выходной файл вывести программу из входного файла, изменив стиль комментариев.

Пример ввода

```
#include <stdio.h>
/* Пример программы
 */
int main()
{ printf( //Печать
     "Hello, world");
  return 0; //*/*
}
```

Вывод для примера

```
#include <stdio.h>
/* Пример программы
 */
int main()
{ printf( /*Печать*/
     "Hello, world");
  return 0; /** /**/
}
```

Контрольное мероприятие №4

Темы вопросов теста
1. Поиск текста
2. Предопределенные классы символов
3. Метасимвол .
4. Определяемые классы символов []
5. Метасимволы для количества + * ? {}
6. Варианты
7. Метасимволы для позиции текста
8. Подвыражения
9. Ленивые повторения
10. Игнорируемые подвыражения
11. Специальные подвыражения
12. Обратные ссылки
13. Замена всего текста
14. Замена с использованием подвыражений
15. Замена с трюками

Примеры вопросов
1

Напишите регулярное выражение, которое определяет наличие символа (в строке. Длина выражения не должна превышать 2 символов.

Есть текст	Нет текста
:-(	:-)
text(?	a+b
	a space

2

Напишите регулярное выражение, которое проверяет, что в строке есть две или более цифры подряд. Длина выражения не должна превышать 5 символов.

Есть текст	Нет текста
x*87	a_5
test:1324	E2-E4
900	9th
	2.7.1
	word
	<--

3

Напишите регулярное выражение, которое проверяет, что строка содержит 3 или более символов. Длина выражения не должна превышать 4 символов.

Есть текст	Нет текста
abc	#1
abcdef	#2
1+42	a
#12	c

4

Напишите регулярное выражение, которое проверяет, что строка содержит один из 4 знаков арифметических операций. Длина выражения не должна превышать 6 символов.

Есть текст	Нет текста
1+2	1.20
30/a	1, 2, 3, 4
width*height	(no operators)
-789	

5

Напишите регулярное выражение, которое проверяет, что строка содержит две буквы `b`, между которыми одна или более букв `a`. Длина выражения не должна превышать 4 символов.

Есть текст	Нет текста
baaaaaaab	bb
baobab	bob
vbaaabh	baaa
	aaab

6

Напишите регулярное выражение, которое проверяет, что строка содержит слово `dog` или `cat` (не как часть другого слова). Длина выражения не должна превышать 15 символов.

Есть текст	Нет текста
The cat run	The fox catch mice
A dog barks	A bulldog
dog	Many dogs
cat	warcat run

7

Напишите регулярное выражение, которое проверяет, что строка состоит только из прописных латинских букв. Длина выражения не должна превышать 8 символов.

Есть текст	Нет текста
TEST	_TEST_
Z	A Z
QWERTYUIOPASDFGHJKLZXCVBNM	7
	Test
	QWERTYU!OPA$DFGHJKLZXCVBNM

8

Напишите регулярное выражение, с помощью которого можно из имени файла извлечь расширение файла (группа символов после последней точки, включая точку). Оставшаяся часть имени файла не может быть пустой. Длина выражения не должна превышать 12 символов.

Есть текст	Выделенные части	Нет текста
image1.png	.png	sample
module@DB_model(v1).c++	.c++	.noname
noextenstion.	.	.
very.long.name.test	.test	

9

Напишите регулярное выражение, с помощью которого можно из строки извлечь текст внутри первого тега p. Длина выражения не должна превышать 15 символов.

Есть текст	Выделенные части	Нет текста
a text b	text	no close
1 2	1	no open
ab	ab	sample</p
		p>sample

10

Напишите регулярное выражение, с помощью которого можно из URL получить протокол, имя сервера и номер порта. Номер порта является опциональным. Длина выражения не должна превышать 40 символов.

Есть текст	Выделенные части	Нет текста
http://example.com:8080/	http, example.com, 8080	http:/example.com:8080/
ftp://localhost	ftp, localhost,	http://example.com::8080
https://ipc.susu.ru/learn.html	https, ipc.susu.ru,	http//example.com:8080/index.html
wais://db.com:1234/person?UID=1	wais, db.com, 1234	http://example.com:/index.html
		file:///C:/

11

Напишите регулярное выражение, с помощью которого можно найти в строке первое слово, содержащее текст abc, но не более чем один раз. Словом называется последовательность символов из букв, цифр и символов подчеркивания. Длина выражения не должна превышать 35 символов.

Есть текст	Выделенные части	Нет текста
abc	abc	abcdabc
abcabc rabcabck uabcabd	uabcabd	abbat
abctabc uabwabcdnm a	uabwabcdnm	dbcabd
fabcsabcr ubcaabcdn abc	ubcaabcdn	axc
u iabcopabck labwnmabcd	labwnmabcd	abd

12

Напишите регулярное выражение, с помощью которого можно проверить, что строка состоит из четного количества букв a. Длина выражения не должна превышать 8 символов

Есть текст	Нет текста
aa	a
aaaa	bb
aaaaaa	aaa
aaaaaaaaaa	abab
	baab
	aaaaa

13

Напишите регулярное выражение и шаблон для замены, с помощью которых все числа в строке будут выделены квадратными скобками. Длина выражения не должна превышать 4 символов.

Есть текст	После замены	Нет текста
2nd	[2]nd	no numbers
100+25	[100]+[25]	
1,12,789	[1],[12],[789]	

14

Напишите регулярное выражение и шаблон для замены, с помощью которых можно найти и заменить в строке последовательности из двух или более гласных букв на последнюю букву из этой последовательности. Длина выражения не должна превышать 20 символов.

Есть текст	После замены	Нет текста
caeyuiat	cat	zzzz
oyiinteeaernouuuet	internet	one
colour	colur	

Регулярное выражение:

Шаблон для замены:

15

Напишите регулярное выражение и шаблон для замены, с помощью которых можно разбить слово на слога. Символ минус ставится после последовательности гласных букв, только если после следующей за ней последовательности согласных есть еще гласные буквы. Длина выражения не должна превышать 32 символов.

Есть текст	После замены	Нет текста
colour	co-lour	cat
acceptable	a-cce-pta-ble	kayak
aeronautical	ae-ro-nau-ti-cal	cwm
keynote	key-no-te	nth
kangaroo	ka-nga-roo	

Регулярное выражение:

Шаблон для замены:

1. Pig-Latin

You have decided that PGP encryption is not strong enough for your email. You have decided to use Pig Latin encryption.

Input and Output

You are to write a program that will take in an arbitrary number of lines of text and output it in Pig Latin. Each line of text will contain one or more words. A "word" is defined as a consecutive sequence of letters (upper and/or lower case). Words should be converted to Pig Latin according to the following rules (non-words should be output exactly as they appear in the input):

1. Words that begin with a vowel (a, e, i, o, or u, and the capital versions of these) should just have the string "`ay`" (not including the quotes) appended to it. For example, "`apple`" becomes "`appleay`".
2. Words that begin with a consonant (any letter than is not A, a, E, e, I, i, O, o, U or u) should have the first consonant removed and appended to the end of the word, and then appending "`ay`" as well. For example, "`hello`" becomes "`ellohay`".
3. Do not change the case of any letter.

Sample Input

```
This is the input.
```

Sample Output

```
hisTay isay hetay inputay.
```

2. XML-преобразователь

XML-формат используется для хранения различной структурной информации и обмена информацией между программами. В некоторой программе этот формат использовался для хранения конфигурационных файлов. В новой версии программы для увеличения ее быстродействия было решено использовать упрощенный XML-формат.

Обычный XML-формат	Упрощенный XML-формат
<ЭЛЕМЕНТ />	<ЭЛЕМЕНТ></ЭЛЕМЕНТ>
<ЭЛЕМЕНТ АТРИБУТ1 АТРИБУТ2='ЗНАЧЕНИЕ2' АТРИБУТ3="ЗНАЧЕНИЕ3" />	<ЭЛЕМЕНТ><АТРИБУТ1></АТРИБУТ1> <АТРИБУТ2>ЗНАЧЕНИЕ2</АТРИБУТ2> <АТРИБУТ3>ЗНАЧЕНИЕ3</АТРИБУТ3></ЭЛЕМЕНТ>
<ЭЛЕМЕНТ АТРИБУТ1 АТРИБУТ2='ЗНАЧЕНИЕ2' АТРИБУТ3="ЗНАЧЕНИЕ3">ТЕКСТ</ЭЛЕМЕНТ>	<ЭЛЕМЕНТ><АТРИБУТ1></АТРИБУТ1> <АТРИБУТ2>ЗНАЧЕНИЕ2</АТРИБУТ2> <АТРИБУТ3>ЗНАЧЕНИЕ3</АТРИБУТ3> ПРЕОБРАЗОВАННЫЙ ТЕКСТ</ЭЛЕМЕНТ>

Имена элементов и атрибутов состоят только из латинских букв, цифр и символов подчеркивание, точка и минус. Длина имен не более 100 символов. ЗНАЧЕНИЕ2 в ' не содержит символа '. ЗНАЧЕНИЕ3 в " не содержит символа ". За этим исключением ЗНАЧЕНИЕ2 и ЗНАЧЕНИЕ3 могут содержать любые символы, включая переход на новую строку. ТЕКСТ может содержать вложенные элементы, которые подвергаются аналогичному преобразованию. После имени элемента и после атрибутов и может быть несколько пробелов, символов табуляции и перехода на новую строку. Эти символы в выходной файл не выводятся.

При установке новой версии программы необходимо выполнить преобразование конфигурационных файлов в новый упрощенный формат. Напишите программу, выполняющую такое преобразование.
Во входном файле содержится одна или несколько строк текста, содержащего XML-элементы.
В выходной файл вывести преобразованный текст, в котором XML-элементы записаны в новом упрощенном XML-формате.

Пример ввода

```
<book genre="Science Fiction">
<autor>Asimov</autor><title>Foundation</title>
<info
  publisher="Booble's doc"
  year='1975' />
</book>
```

Пример вывода

```
<book><genre>Science Fiction</genre>
<autor>Asimov</autor><title>Foundation</title>
<info><publisher>Booble's doc</publisher><year>1975</year></info>
</book>
```

3. Марсианская лингвистическая реформа

Алфавит марсианского языка состоит из строчных латинских букв. Буквам a, e, i, o, u, y соответствуют гласные звуки, остальным – согласные.
В результате опроса марсиан выяснилось, что им трудно произносить слова, в которых присутствуют два или более гласных звука подряд. Марсианскими лингвистами было принято решение: во всех словах, где есть два или более гласных звука подряд, оставить только последний из них.
Марсиане просят написать программу, переводящую слова из старого в новый форматы.
Например, если старое слово имело вид `eeaaeeuinfaormaaiatoyuaoics`, то новое слово будет таким: `informatics`.
Формат входного файла
Во входном файле содержится марсианское слово старого формата.
Формат выходного файла
Выходной файл должен содержать марсианское слово нового формата.
Ограничения
Длина слова находится в диапазоне от 1 до 100 символов.

Пример ввода 1

```
eeaaeeuinfaormaaiatoyuaoics
```

Пример вывода 1

```
informatics
```

Пример ввода 2

```
aeeaaayyaaauaaiaaiaacm
```

Пример вывода 2

```
acm
```

Пример ввода 3

```
bzzt
```

Пример вывода 3

```
bzzt
```

4. Слова и числа

Главный бюрократ компании "Планетарная экспресс-доставка" Гермес Конрад после написания очередного отчета или распоряжения обязательно подсчитывает количество слов и чисел в документе, так как владелец компании, профессор Х.Дж.Фарнсворт, не любит лишних слов, а любит только числа. Напишите программу, которая поможет Гермесу подсчитать количество слов и чисел в документе.

Во входном файле содержится от 1 до 100 строк длиной не более 80 символов. Словом или числом будем называть непрерывную последовательность печатных символов. При этом число в отличие от слова содержит одну или более цифр от 0 до 9. Слова и числа в документе разделены последовательностями пробелов, табуляций и переходов на новую строку.

В выходной файл вывести количество слов и чисел в документе в виде "`2 word(s), 1 number(s)`".

Пример ввода

```
              REPORT
In 1st quarter the margin was $20.02.
```

Вывод для примера

```
6 word(s), 2 number(s)
```

5. **Форматирование программы**

Регистр букв в программе на языке Паскаль не имеет значения, но для лучшего выделения структуры программы рекомендуется писать все резервированные слова (AND, ARRAY, BEGIN, CASE, CONST, DIV, DO, DOWNTO, ELSE, END, FILE, FOR, FUNCTION, GOTO, IF, IN, LABEL, MOD, NIL, NOT, OF, OR, PACKED, PROCEDURE, PROGRAM, RECORD, REPEAT, SET, THEN, TO, TYPE, UNTIL, VAR, WHILE, WITH) прописными буквами, а остальные идентификаторы, включая имена встроенных типов и функций, – строчными буквами.

Напишите программу, которая изменяет регистр букв в программе на языке Паскаль в соответствии с этой рекомендацией, оставляя без изменений регистр в комментариях, которые записываются в `{ }`, и в строковых константах, которые записываются в `' '`.

Ввод содержит синтаксически правильную программу на языке Паскаль. Количество строк не превышает 100. Длина строк не превышает 80 символов. Длина идентификаторов не превышает 20 символов.

Вывод содержит программу с изменением регистра букв резервированных слов и идентификаторов, остальные символы копируются из ввода без изменений.

Пример ввода

```
{ My first program }
Program Hello;
BeGiN
   WRITELN('Hello, world!');
end.
```

Пример вывода

```
{ My first program }
PROGRAM hello;
BEGIN
   writeln('Hello, world!');
END.
```

6. **Комментарии**

Джон, хотя и пишет на языке C, дает файлам расширение CPP, чтобы использовать в своих программах комментарии в C++-стиле (от // до конца строки). Обычный C-комментарий, который начинается с символов "/*" и заканчивается символами "*/" , Джон также иногда использует, обычно для многострочных комментариев. Для участия в конкурсе программ необходимо, чтобы программа соответствовала стандартам языка ANSI C, и Джону нужно заменить все C++-комментарии на стандартные. Для этого в C++-комментарии можно заменить "//" на "/*" и добавить "*/" в конце строки. Иногда в C++-комментарии может встретиться последовательность символов "*/", в этом случае нужно вставить пробел между двумя этими символами: "* /". К счастью внутри строковых констант в программе Джона не встречаются последовательностей "//", "/*" и "*/".

Напишите программу, которая преобразует в программе Джона C++-комментарии в C-комментарии.

Во входном файле содержится программа Джона, не более 100 строк длиной не более 100 символов.

В выходной файл вывести программу из входного файла, изменив стиль комментариев.

Пример ввода

```
#include <stdio.h>
/* Пример программы
 */
int main()
{ printf( //Печать
     "Hello, world");
  return 0; //*/*
}
```

Вывод для примера

```
#include <stdio.h>
/* Пример программы
 */
int main()
{ printf( /*Печать*/
     "Hello, world");
  return 0; /** /**/
}
```

1. Распаковка

Строка длиной не более 2· 109 символов, состоящая из прописных латинских букв A..Z, была упакована по следующим правилам:

- букве в исходной строке соответствует та же буква в упакованной строке
- для последовательности из N > 1 одинаковых букв в упакованной строке записывается число N , затем буква
- для последовательности из N > 1 одинаковых подстрок в упакованной строке записывается число N , затем в круглых скобках упакованная подстрока.

Напишите программу, определяющую по упакованной форме количество вхождений каждой из букв A..Z в исходную строку.

Ввод

В первой строке входного файле содержится упакованная строка длиной не более 256 символов.

Вывод

В выходной файл для каждой буквы A..Z, встречающейся в тексте, вывести строку, содержащую букву, затем пробел и количество вхождений этой буквы в исходную строку. Информация выводится в алфавитном порядке.

Пример ввода

```
10(5XH)X
```

Пример вывода

```
H 10
X 51
```

2. Simply syntax

In the land of Hedonia the official language is Hedonian. A Hedonian professor had noticed that many of her students still did not master the syntax of Hedonian well. Tired of correcting the many syntactical mistakes, she decided to challenge the students and asked them to write a program that could check the syntactical correctness of any sentence they wrote. Similar to the nature of Hedonians, the syntax of Hedonian is also pleasantly simple. Here are the rules:

0. The only characters in the language are the characters p through z and N, C, D, E, and I.
1. Every character from p through z is a correct sentence.
2. If s is a correct sentence, then so is Ns.
3. If s and t are correct sentences, then so are Cst, Dst, Est and Ist.
4. Rules 0. to 3. are the only rules to determine the syntactical correctness of a sentence.

You are asked to write a program that checks if sentences satisfy the syntax rules given in Rule 0. – Rule 4.

Input: The input consists of a number of sentences consisting only of characters p through z and N, C, D, E, and I. Each sentence is ended by a new-line character. The collection of sentences is terminated by the end-of-file character. If necessary, you may assume that each sentence has at most 256 characters and at least 1 character.

Output: The output consists of the answers `YES` for each well-formed sentence and `NO` for each not-well-formed sentence. The answers are given in the same order as the sentences. Each answer is followed by a new-line character, and the list of answers is followed by an end-of-file character.

Sample Input

```
Cp
Isz
NIsz
Cqpq
```

Sample Output

```
NO
YES
YES
NO
```

3. Slurpys

Dr. Zoidberg is a ET.
A Slurpy is a word in his language that has certain properties.
Your program will read in strings of characters and output whether or not they are Slurpys.

A *Slump* is a character string that has the following properties:

- Its first character is either a '`D`' or an '`E`'.
- The first character is followed by a string of one or more '`F`'s.
- The string of one or more '`F`'s is followed by either a Slump or a '`G`'. The Slump or 'G' that follows the F's ends the Slump. For example '`DFFEFFFG`' is a Slump since it has a '`D`' for its first character, followed by a string of two '`F`'s, and ended by the Slump '`EFFFG`'.
- Nothing else is a Slump.

A *Slimp* is a character string that has the following properties:

- Its first character is an '`A`'.
- If it is a two character Slimp then its second and last character is an '`H`'.
- If it is not a two character Slimp then it is in one of these two forms:
  a) '`A`' followed by '`B`' followed by a Slimp followed by a '`C`'.
  b) '`A`' followed by a Slump (see above) followed by a '`C`'.
- Nothing else is a Slimp.

A *Slurpy* is a word that consists of a Slimp followed by a Slump.
Examples
Slumps: DFG, EFG, DFFFFFG, DFDFDFDFG, DFEFFFFFG
Not Slumps: DFEFF, EFAHG, DEFG, DG, EFFFFDG
Slimps: AH, ABAHC, ABABAHCC, ADFGC, ADFFFFGC, ABAEFGCC, ADFDFGC
Not Slimps: ABC, ABAH, DFGC, ABABAHC, SLIMP, ADGC
Slurpys: AHDFG, ADFGCDFFFFFG, ABAEFGCCDFEFFFFFG
Not Slurpys: AHDFGA, DFGAH, ABABCC
The first line of input contains an integer N
between 1 and 10 describing how many strings of characters are represented. The next N

lines each contain a string of 1 to 60 alpha characters.
Each of N
lines of output should consist of either `YES` or `NO` depending on whether or not the corresponding input line is a Slurpy.

Input Sample

```
2
AHDFG
DFGAH
```

Output Sample

```
YES
NO
```

4. NOT

Логические выражения, содержащие отрицания, трудны для понимания, поэтому программисты, особенно начинающие, должны их избегать. Напишите программу, которая выполняет рефакторинг логического выражения на языке Паскаль, преобразуя его к виду, не содержащему операции NOT, используя правила де Моргана, удаление двойного отрицания и замену операций отношения на инверсные.

Логическое выражение в этой задаче имеет следующий синтаксис:

выр ::= выр1 | выр4

выр1 ::= выр2 | выр1 «OR» выр2

выр2 ::= выр3 | выр2 «AND» выр3

выр3 ::= « (» выр «) » | «NOT» выр3

выр4 ::= идент «=» идент | идент «<>» идент | идент «>» идент | идент «<» идент | идент «>=» идент | идент «<=» идент

идент ::= «A» | «B» | … | «Z»

Для удаления отрицания используются следующие правила:

NOT(A OR B) →NOT(A) AND NOT(B)

NOT(A AND B) →NOT(A) OR NOT(B)

NOT(NOT A) →A

NOT(A<B) →A>=B

NOT(A>B) →A<=B

NOT(A<=B) →A>B

NOT(A>=B) →A<B

NOT(A=B) →A<>B

NOT(A<>B) →A=B

Формат ввода

В первой строке ввода логическое выражение, соответствующее указанному синтаксису. Между лексемами могут быть пробелы. Длина строки не превышает 200 символов. Строчные буквы в записи выражений не используются.

Формат вывода

Вывести логическое выражение после удаления отрицаний. Других преобразований в логическом выражении кроме вышеуказанных делать нельзя. Скобки должны использоваться только там, где они необходимы для указания порядка выполнения операций. Самый низкий

приоритет в языке Паскаль у операций сравнения, выше у операции OR, еще выше у операции AND. Пробелы в выходном логическом выражении должны отсутствовать.

Пример ввода

```
NOT((A>=B) OR (C<X) AND (D<>X))
```

Пример вывода

```
(A<B)AND((C>=X)OR(D=X))
```

5. Статическая сложность

Анализ временной сложности алгоритмов – важный инструмент создания эффективных программ. Алгоритмы, выполняемые за линейное время, как правило, значительно быстрее алгоритмов, требующих для выполнения той же задачи квадратичного времени, так что предпочтение должно быть отдано первым. Обычно определяют время выполнения алгоритма по отношению к n – "размеру" входных данных. Это может быть число объектов, которые нужно отсортировать, число точек многоугольника и т.п. Поскольку определение формулы зависимости временной сложности алгоритма от n – непростая задача, было бы замечательно, если бы её можно было автоматизировать. К сожалению, в общем случае это невозможно. Но в этой задаче мы будем рассматривать программы очень простой природы, над которыми это можно проделать. Рассматриваемые программы записаны согласно следующим правилам БНФ, где *<число>* может быть любым неотрицательным целым числом:

- <Программа> ::= "`BEGIN`" <Список операторов> "`END`"
- <Список операторов> ::= <Оператор> | <Оператор> <Список операторов>
- <Оператор> ::= <Оператор LOOP> | <Оператор OP>
- <Оператор LOOP> ::= <Заголовок LOOP> <Список операторов> "`END`"
- <Заголовок LOOP> ::= "`LOOP`" <число> | "`LOOP n`"
- <Оператор OP> ::= "`OP`" <число>

Время выполнения такой программы может быть вычислено следующим образом: выполнение оператора `OP` требует столько единиц времени, сколько указано в его параметре. Список операторов, заключённый в оператор `LOOP`, выполняется столько раз, сколько указано в параметре оператора, то есть или заданное константное число раз, если задано число, или n раз, если параметром является n. Время выполнения списка операторов равно сумме времени выполнения его частей. Таким образом, время выполнения программы в целом зависит от n.

Ввод

В первой строке находится целое число k – число программ во входном файле. Затем идут k программ, удовлетворяющих приведённой грамматике. Пробелы и переводы строк могут встречаться везде в программе, но не в ключевых словах `BEGIN`, `END`, `LOOP` и `OP`, нет их и в целых числах.

Вывод

Для каждой программы сначала идёт строка с номером программы. В следующей строке записывается время работы программы в терминах n – многочлен степени не более 10. Многочлен должен быть записан обычным способом, то есть подобные слагаемые должны быть приведены, слагаемое с большей степенью должно предшествовать слагаемому с

меньшей степенью, слагаемые с коэффициентом 0 не записываются, множители 1 не записываются. Общий вид второй строки "Runtime = a*n^10+b*n^9+...+i*n^2+j*n+k". Если время выполнения нулевое, нужно вывести "Runtime = 0". Между результатами должна быть пустая строка.
Ограничения: вложенность операторов LOOP не превышает 10, размер входного файла не более 2 Кбайт, коэффициенты многочлена в ответе не превышают 50 000
.

Пример ввода

```
2
BEGIN
  LOOP n
    OP 4
    LOOP 3 LOOP n OP 1 END OP 2 END
    OP 1
  END
  OP 17
END

BEGIN
 OP 1997 LOOP n LOOP n OP 1 END END
END
```

Пример вывода

```
Program #1
Runtime = 3*n^2+11*n+17

Program #2
Runtime = n^2+1997
```

6. Транслятор для языка Basic-

Транслятор переводит программу на языке высокого уровня в программу на машинном языке. Напишите транслятор для упрощенного языка Basic-. В языке Basic- всего 26 переменных, обозначаемых прописными латинскими буквами от A до Z. Так как все переменные целые, то операторы объявления переменных отсутствуют. Для упрощения задачи в языке также отсутствуют операторы ввода-вывода, ветвления и цикла, комментарии, а также все операции и функции, кроме бинарных операций + и –. В результате программа на языке Basic- состоит из последовательности операторов присваивания вида:

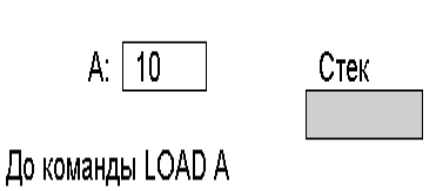
<*переменная*> **=** <*выражение*>
где <*выражение*> это <*операнд*> или <*выражение*> **+** <*операнд*> или <*выражение*> **–** <*операнд*>
а <*операнд*> это <*переменная*> или **(** <*выражение*> **)**

Каждый оператор присваивания записывается на отдельной строке, не содержит пробелов, длина оператора не превышает 80 символов. Пример программы:

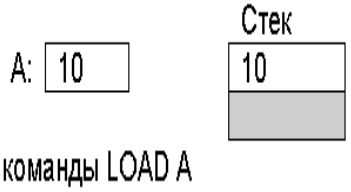
```
A=B
D=A+(B-C)
```

Машинный язык, на который транслируется программа, состоит всего из 4 команд:

LOAD <*переменная*>
Загружает значение переменной в стек.

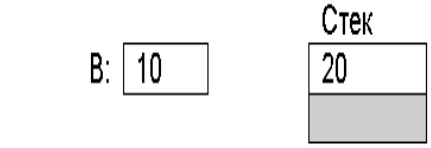


До команды LOAD A

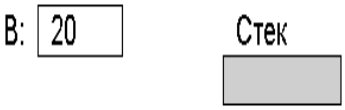


После команды LOAD A

SAVE <*переменная*>
Сохраняет вершину стека в переменной.

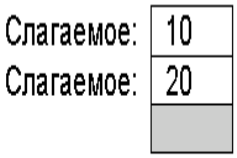


До команды SAVE B

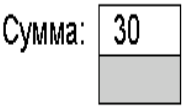


После команды SAVE B

ADD
Складывает два числа на вершине стеке, сумма помещается в стек.

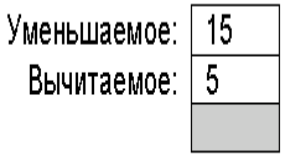


Состояние стека до команды ADD

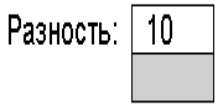


После команды ADD

SUB
Вычисляет разность на двух чисел на вершине стека, разность помещается в стек.



Состояние стека до команды SUB



После команды SUB

Результат трансляции вышеуказанной программы из двух операторов может выглядеть следующим образом:

```
LOAD B
SAVE A
LOAD C
LOAD A
LOAD B
ADD
SUB
SAVE D
```

Это не единственный вариант машинного кода. Результат вычислений не изменится, если поменять местами, например, команды `LOAD A` и `LOAD B`.

Во входном файле программа на языке Basic- (один или более операторов присваивания).
В выходной файл вывести результат трансляции программы на машинный язык. Не делайте оптимизирующий транслятор! Последовательность блоков кода для операторов присваивания должна совпадать с последовательностью этих операторов в программе на языке Basic-. Для каждого оператора присваивания должна быть команда `SAVE`, которая должна быть последней в соответствующем блоке кода. Каждому появлению переменной в правой части оператора присваивания должна соответствовать команда `LOAD` в машинном коде. В пределах блока кода порядок машинных команд может быть достаточно произвольным, требуется лишь соответствие результата вычислений выражению в правой части оператора присваивания.

Пример ввода

```
A=B
D=A+(B-C)
```

Пример вывода

```
LOAD B
SAVE A
LOAD A
LOAD C
LOAD B
SUB
ADD
SAVE D
```

Контрольное мероприятие №7

Написать интерпретатор для учебного языка.

Задача 1. Используя CAIO, выделить лексемы учебного языка.

Для каждой лексемы, кроме пробельных (игнорируемых) символов и комментариев, нужно напечатать на отдельной строке тип лексемы и саму лексему. Тип лексемы выбирается из "резервированное слово", "идентификатор", "операция", "строка", "число", "символ" (скобки, разделитель операторов, запятая). Например, для программы

```
input b
a=b
```

должно быть выведено

```
?резервированное-слово? : "input"
?идентификатор? : "b"
?символ? : "\n"
?идентификатор? : "a"
?операция? : "="
?идентификатор? : "b"
?символ? : "\n"
```

Пример решения:

```
%option lexprint
%type <string> ?идентификатор? ?символ? ?резервированное-слово? ?операция?
%%
input ?резервированное-слово?
[a-zA-Z]\w* ?идентификатор?
[ \t] ;
= ?операция?
.|\n ?символ?
%%
```

При запуске программы используйте команду `zadacha1.exe -d1 test.txt`

Задача 2. Используя CAIO, написать программу для проверки корректности синтаксиса программы на учебном языке.

Задача 3. Добавьте в правила грамматики операторы для построения AST и напишите интерпретатор для учебного языка.

Описание учебного языка (пример варианта)

A. Комментарии:

/* текст до ***/**

B. Числа:
целые десятичные и

шестнадцатиричные, начинающиеся с цифры и заканчивающиеся буквой **H**

C. Строки (для оператора вывода):

'*текст*'
если в строке необходим символ ', он удваивается
Пример: `'It''s sample'`

Символ перехода на новую строку в строке недопустим, и специального обозначения для него как в C нет.

D. Переменные и регистр резервированных слов:

Идентификатор переменной начинается с буквы или символа _, затем идут буквы, цифры или символ _, длиной до 10 символов
Регистр букв в резервированных словах и идентификаторах игнорируется.

E. Выражения:
Операции **+**, **-**, *****, **/** и сравнения **>**, **<**, **=**, **!=**, **<=**, **>=**, круглые скобки, операция смена знака **-**, обычный приоритет

F. Оператор присваивания:

*переменная***=***выражение*

G. Операторы ввода-вывода:

READ(*переменная*$_1$**,** *переменная*$_2$**...)**
WRITE(*строка*$_1$|*переменная*$_1$**,** *строка*$_2$|*переменная*$_2$**...)**
WRITELN(*строка*$_1$|*переменная*$_1$**,** *строка*$_2$|*переменная*$_2$**...)**
Оператор WRITELN завершает вывод символом перехода на новую строку, а WRITE - нет.

H. Сложные операторы:

Операторы разделяются символом **;**
IF *выражение* **THEN**
  операторы
END

IF *выражение* **THEN**
  операторы
ELSE
  операторы
END

WHILE *выражение* **DO**

операторы
END

I. Расширения:

Логические операции **&& || !** и условная операция *выр*1**?***выр*2**:***выр*3 как в языке C.
Для тех, кто забыл C: условная операция выглядит так: max := a>b?a:b, а выражения с && и || могут вычисляться частично.

Контрольное мероприятие №8

Написать компилятор для учебного языка с помощью LLVM

Задача 1. Установить LLVM, подключить к компилятору

Задача 2. Выполнить обход AST и генерацию кода

Задача 3. Выполнить проверку генерированной программы

Задача 4. Сравнить скорость программы на учебном языке и на языке C

Описание учебного языка (пример варианта)

A. Комментарии:

/* текст до ***/**

B. Числа:
целые десятичные и

шестнадцатиричные, начинающиеся с цифры и заканчивающиеся буквой **H**

C. Строки (для оператора вывода):

'*текст*'
если в строке необходим символ ', он удваивается
Пример: `'It''s sample'`

Символ перехода на новую строку в строке недопустим, и специального обозначения для него как в C нет.

D. Переменные и регистр резервированных слов:

Идентификатор переменной начинается с буквы или символа _, затем идут буквы, цифры или символ _, длиной до 10 символов
Регистр букв в резервированных словах и идентификаторах игнорируется.

E. Выражения:
Операции **+**, **-**, ***, /** и сравнения **>**, **<**, **=**, **!=**, **<=**, **>=**, круглые скобки, операция смена знака **-**, обычный приоритет

F. Оператор присваивания:

переменная=выражение

G. Операторы ввода-вывода:

READ(*переменная*$_1$**,** *переменная*$_2$**...)**
WRITE(*строка*$_1$|*переменная*$_1$**,** *строка*$_2$|*переменная*$_2$**...)**
WRITELN(*строка*$_1$|*переменная*$_1$**,** *строка*$_2$|*переменная*$_2$**...)**
Оператор WRITELN завершает вывод символом перехода на новую строку, а WRITE - нет.

H. Сложные операторы:

Операторы разделяются символом **;**
IF *выражение* **THEN**
  операторы
END

IF *выражение* **THEN**
  операторы
ELSE
  операторы
END

WHILE *выражение* **DO**
  операторы
END

I. Расширения:

Логические операции **&& || !** и условная операция *выр*$_1$**?***выр*$_2$**:***выр*$_3$ как в языке C.
Для тех, кто забыл C: условная операция выглядит так: max := a>b?a:b, а выражения с && и || могут вычисляться частично.

Контрольное мероприятие №9

Написать интерпретатор для учебного языка с помощью JIT-технологии LLVM

Задача 1. Выполнить обход AST и генерацию внутреннего представления для JIT

Задача 2. Выполнить проверку интерпретатора

Задача 3. Сравнить скорость компилятора и интерпретатора учебного языка

Описание учебного языка (пример варианта)

A. Комментарии:

/* текст до ***/**

B. Числа:
целые десятичные и

шестнадцатиричные, начинающиеся с цифры и заканчивающиеся буквой **H**

C. Строки (для оператора вывода):

'*текст*'
если в строке необходим символ ', он удваивается
Пример: `'It''s sample'`

Символ перехода на новую строку в строке недопустим, и специального обозначения для него как в C нет.

D. Переменные и регистр резервированных слов:

Идентификатор переменной начинается с буквы или символа _, затем идут буквы, цифры или символ _, длиной до 10 символов
Регистр букв в резервированных словах и идентификаторах игнорируется.

E. Выражения:
Операции **+**, **-**, ***, /** и сравнения >, <, =, **!**=, <=, >=, круглые скобки, операция смена знака **-**, обычный приоритет

F. Оператор присваивания:

переменная=выражение

G. Операторы ввода-вывода:

READ(*переменная*$_1$**,** *переменная*$_2$**...)**
WRITE(*строка*$_1$|*переменная*$_1$**,** *строка*$_2$|*переменная*$_2$**...)**
WRITELN(*строка*$_1$|*переменная*$_1$**,** *строка*$_2$|*переменная*$_2$**...)**
Оператор WRITELN завершает вывод символом перехода на новую строку, а WRITE - нет.

H. Сложные операторы:

Операторы разделяются символом **;**
IF *выражение* **THEN**
  операторы
END

IF *выражение* **THEN**
  операторы
ELSE
  операторы
END

WHILE *выражение* **DO**
  операторы
END

I. Расширения:

Логические операции **&& || !** и условная операция *выр*$_1$**?***выр*$_2$**:***выр*$_3$ как в языке C.
Для тех, кто забыл C: условная операция выглядит так: max := a>b?a:b, а выражения с && и || могут вычисляться частично.

Контрольное мероприятие №10

Вопросы к диф.зачету по дисциплине «Методы трансляции и формальные языки»

Вопрос 1 по теме «Теория формальных языков»

1. Алфавит. Цепочки символов в алфавите. Операции над цепочками символов. Формальное определение грамматики и языка.
2. Формальное определение языка. Операции над языками. Синтаксис и семантика языка.
3. Способы задания синтаксиса формальных языков.
4. Классификация языков и грамматик по Хомскому.
5. Разбор цепочек КС-грамматик. Вывод, цепочка вывода, сентенциальная форма. Дерево вывода.
6. Левосторонний и правосторонний выводы. Способы задания грамматик. Проблемы однозначности и эквивалентности грамматик.
7. Общая схема распознавателя. Классификация распознавателей.
8. Свойства регулярных языков. Лемма о разрастании для регулярного языка. Проблемы, разрешимые для регулярных языков.
9. Регулярные и автоматные грамматики. Детерминированные и недетерминированные конечные автоматы.
10. Преобразование конечного автомата к детерминированному виду. Минимизация конечных автоматов.
11. Регулярные множества и регулярные выражения. Основные свойства, алгебра регулярных выражений. Эквивалентность регулярных грамматик, конечных автоматов и регулярных выражений.
12. Контекстно-свободные грамматики. Свойства контекстно-свободных языков. Назначение и принципы работы синтаксических анализаторов. Автоматы с магазинной памятью.
13. Преобразование КС-грамматик. Классификация распознавателей для КС-языков. Определение LL(k)-грамматики. Множества FIRST и и FOLLOW. Метод рекурсивного спуска.
14. Определение LR(k)-грамматики. Построение восходящего распознавателя для LR(1)-грамматики. Алгоритм сдвиг-свертка.

Вопрос 2 по теме “Теория перевода и построение трансляторов»

15. Лексика, синтаксис и семантика языка. Универсальные и проблемно-ориентированные языки программирования.
16. Определение транслятора, интерпретатора, компилятора и ассемблера.
17. Общая схема работы транслятора. Этапы трансляции
18. Особенности построения интерпретаторов. Технология JIT
19. Организация таблиц идентификаторов
20. Способы внутреннего представления программы.
21. Атрибутные грамматики. Абстрактное синтаксическое дерево (AST), представление и обработка.
22. Семантический анализ. Методы генерации и оптимизации кода.
23. Основные библиотечные функции для регулярных выражений. Особенности реализации регулярных выражений в языках программирования.
24. Генератор лексических анализаторов FLEX. Назначение и принципы
25. Генератор синтаксических анализаторов BISON.
26. Назначение, основные принципы организации LLVM.
27. Интерфейс LLVM API, основные классы
28. Промежуточное представление LLVM IR. Выбор целевой машины и генерация машинного кода.

01.03.02 Прикладная математика и информатика, направленность (профиль) Прикладная математика и искусственный интеллект
Рабочая программа дисциплины "Методы трансляции и формальные языки", 2024 год набора, очная форма обучения

Проректор по учебной работе	утверждено	В.Е. Федоров

Ученым советом математического факультета

Протокол заседания № 3 от 19.10.2023

Председатель Ученого совета математического факультета	согласовано	Е.А. Сбродова

Заседанием кафедры теории управления и оптимизации

Протокол заседания № 4 от 12.10.2023

Заведующий кафедрой	согласовано	И. В. Изместьев
Автор (составитель)		Е.А. Скрипов

Структура рабочей программы соответствует приказу ректора ФГБОУ ВО «ЧелГУ» от «13» апреля 2021 г. № 247-1