acm.mipt.ru

олимпиады по программированию на Физтехе

Программа и задания по курсу "Сравнительный анализ языков программирования"

- по курсу: Сравнительный анализ языков программирования
- по направлению: 552800
- факультет: ФРТК
- кафедра: Прикладной информатики
- курс: 2
- семестр: 3, 4
- лекции: 66 часов
- ВСЕГО ЧАСОВ: 66
- Программу составил: Ворожцов А.В.

ПРОГРАММА

1. Общие вопросы.
   1. Сферы применения языков программирования (научные исследования, коммерческие применения, искусственный интеллект, системное программирование и др.).
   2. Характеристики языков программирования (базовая парадигма языка, способ организации программы, типы решаемых задач)
   3. Критерии оценки языков программирования (удобность, читаемость, скорость разработки приложений, поддержка поиска ошибок на этапе компиляции и работы программы, модульность и интегрируемость, эффективность по времени и памяти).
   4. Эволюция языков программирования. Краткий исторический очерк.
2. Синтаксис и семантика языков программирования.
   1. Понятия синтаксиса и семантики. Задача описания синтаксиса языка программирования. Методы формального описания синтаксиса. Форма Бэкуса-Наура. Контекстно-свободные грамматики. Деревья синтаксического анализа. Расширенные формы БНФ.
   2. Введение в контекстно-свободные грамматики. LR(k) грамматики. Задача определения принадлежности к грамматике, её вычислительная сложность.
   3. Лексемы. Лексический анализ. Продукционный подход к созданию трансляторов (Lex &amp Yacc). Мета-языки.
3. Основные понятия языков программирования.
   1. Имена. Основные вопросы их реализации. Переменные и их атрибуты: имя, адрес, значение, тип. Время жизни переменной. Область видимости. Концепция связывания (способы связывания, статические, динамические переменные). Понятие глобальных и локальных данных.
   2. Типы данных. Простые типы данных. Проверка типов, строгая типизация. Динамические типы данных. Полиморфизм объектов.
   3. Процедуры и функции. Параметры. Методы передачи параметров. Полиморфизм функций. Побочные эффекты функций.
   4. Работа с памятью. Статическое и динамическое выделение памяти. Сборка мусора.
4. Парадигмы программирования.
   1. Абстрактные вычислители и парадигмы программирования. Конечные автоматы, алгоритмы Маркова и машины Тьюринга и соответствующие им парадигмы программирования.
   2. Классификация и эволюция языков программирования. Данные, функции, объекты, состояния и условия. Структурное и объектно-ориентированное

Поиск
Поиск
Раздел «Парадигмы»
Инструменты:
Поиск
Изменения
Index
Статистика
Разделы
Информация
Алгоритмы
Язык Си
Язык Ruby
Язык Ассемблера
El Judge
Парадигмы
Образование
Сети
Objective C
Logon>>

программирование. Функциональное программирование. Ленивые вычисления. Программирование от состояний – автоматное программирование. Событийное программирование. Продукционное программирование. Сентенциальное программирование. Гибридные и чистые модели. Недостатки и преимущества различных гибридных и чистых моделей. Выбор лучшей парадигмы программирования при решении практических задач. Рассмотрение языков C, C++, Java, Python, Ruby, Lisp, Haskell, Prolog, Рефал и их отличительных характеристик. Языки для проектирования электронных устройств и логических схем.
   3. Языки и виртуальные вычислительные машины.
5. Языки разметки и языки запросов.
   1. Проблемы соединения данных с логикой работы программы (правилами интерпретации данных).
   2. Иерархическая модель хранения данных. XML-языки разметки, и язык XSLT трансляции XML-данных.
   3. Другие модели хранения данных и соответствующие им языки. Триплетная модель данных RDF. Языки обработки данных, записанных в модели RDF.
   4. Базы данных и языки запросов. Реляционная, иерархическая и сетевая модели данных. Модель данных в языке Prolog.
6. Инструменты разработки языков программирования. Выделение лексем с помощью Lex. Продукционный подход к созданию трансляторов (Yacc). Язык как расширение существующего языка. Разработка макро-языков на основе языков Tcl и Python и Lisp. Реализация калькуляторов, автоматов и конвертаторов с помощью Lex и Yacc. Методы и инструменты создания функциональных языков программирования. Основные вопросы проектирования языков программирования: основные принципы разработки языков программирования; цели разработки; модели данных; модели конструкций управления; механизмы абстракции Виртуальные машины и промежуточные языки.
   Введение в теорию трансляции: сравнение интерпретаторов и компиляторов; стадии трансляции; машинно-зависимая и машинно-независимая части транслятора.

Список литературы:

1. М. Бен-Ари. Языки программирования. Практический сравнительный анализ – М.: Мир, 2000.
2. Н.Н. Непейвода, Стили и методы программирования, – М.:Интернет-Ун-т Информ. Технологий,- 320 с., 2005
3. Р. Себеста. Основные концепции языков программирования – М., Вильямс, 2001.
4. А. Ахо, Дж. Ульман "Теория синтаксического анализа, перевода и компиляции", Т.1 "Синтаксический анализ", М.: Мир, 1978

Задание 1

1.1. Напишите три программы, связанные с правильными скобочными структурами: а) проверка введённой скобочной структуры на правильность б) вывод всех правильных скобочных структур длины 2*n*; в) вычисление числа правильных скобочных структур длины 2*n*. Правильные скобочные структуры задаются правилами:

```
S       a
S       a ( S ) S
```

1.2. Напишите программу, которая распознает язык, задаваемый правилами в форме EBNF:

```
S =    0 |
   1 S |
   2 S S |
   3 S S S.
```

1.3. Напишите программу-калькулятор выражений в польской обратной нотации. Например, для выхода “2 3 * 5 6 * +” программа должна вывести число 36.

1.4. Напишите программу на языке C, которая определяет, подходит ли слово W к регулярному выражению R. Под регулярным выражением понимается регулярное выражение Perl, в котором можно использовать только круглые и квадратные скобки, а также мета-символы “+” “*” и “?”.

1.5. Напишите программу-калькулятор на основе сентенциальной парадигмы программирования используя операцию замены по регулярному выражению. А именно, запишите в цикле замены регулярных выражений, соответствующих арифметическим действиям над элементарными объектами (числами), на их значение. Оцените время работы алгоритма.

1.6. Создайте простейший язык описания конечных автоматов и напишите для него транслятор. Напишите на нём конечный автомат, который принимает на вход двоичную запись скольк угодно большого натурального числа и выводит двоичную запись этого числа, умноженного на 5.

1.7. Напишите калькулятор на Lex & Yacc (или Bison) основанный на следующей грамматике

```
пусто   =
выр     = слаг оствыр
оствыр  = '+' выр | пусто
слаг    = множ остслаг
остслаг = '*' слаг | пусто
множ    = NUMBER | '(' выр ')'
```

1.8. Напишите программу, которая по данному списку слов возвращает описание конечного автомата. Этот конечный автомат должен работать следующим образом: получал на вход текст из пробелов и английских букв в нижнем регистре и все слова, из данного списка, превращал в аналогичные слова в верхнем регистре.

Задание 2

Второе задание подразумевает работу в проектах. Каждому студенту предлагается поучаствовать в одном - двух проектах. Предлагаемые ниже задачи следует рассматривать как проекты.

Результатом каждого проекта должно стать

- рабочие программы
- документация
- наглядная яркая презентация результатов

Основные этапы работы в проекте

- определение функциональности системы
- проектирование системы
- выделение ключевых подзадач и распределение между участниками проекта
- контроль руководителя проекта за ходом выполнения подзадач
- подготовка тестов и тестирование системы
- создание документации и презентации системы

Во многих проектах ярко выражены следующие компоненты: графический интерфейс, логический блок, тесты, дополнительные модули. Такое деление на компоненты можно положить в основу распределения задач между участниками проекта.

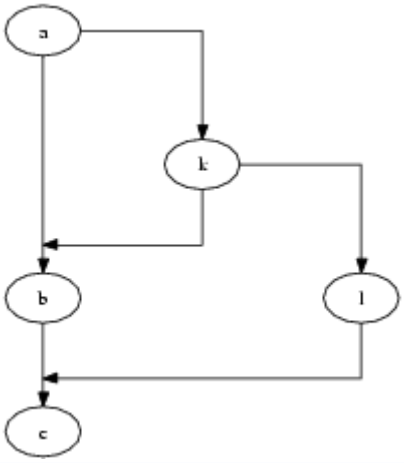
2.1. (Python + Tk турнирная система) Используя Python и графическую библиотеку Tk создайте среду для проведения соревнований между программами по игре в крестики-нолики "5 в ряд" на поле 15x15. Вместо Python+Tk можно использовать Tcl+Tk, Python+WX, ... Вместо игры крестики нолики можно взять любую другую игру, где нет простой выйгрышной стратегии. Напишите несколько простых программ-игроков, играющую в эту игру. Программы-игроки должны взаимодействовать с соперником через стандартные потоки ввода/вывода.

2.1'. (Python + Tk террариум) На базе Python + Tk разработайте среду, в которой могут жить искусственные организмы. Реализуйте возможность эволюции, то есть сделайте часть параметров алгоритма жизни особей вынесите в изменяемый "генотип".

2.2. (dot -> dragon) Напишите программу, которая позволяет по описанию dragon-схемы на языке dot получать её визуальное представление. Например:

Описание на языке dot:	Визуальное представление:

```
digraph {
    graph [root="a"];
    edge [dtype="next"];
    a->b->c;
    k->b;
    l->c;
    edge [dtype="nextright"];
    a->k;
    k->l;
}
```



2.3. (FA <--> visual representation, FA translator) Создайте язык описания конечных автоматов. На основе библиотеки GraphViz (алгоритм neato) и поставляемого с ним визуального инструмента tcldot разработайте визуальный инструмент редактирования конечных автоматов, который позволяет

- редактировать логику конечного автомата через редактирование его визуального представления
- из текстового описания конечного автомата создавать визуальное представление конечного автомата и наоборот
- экспортировать описание конечного автомата в программу на C, его реализующего.
- визуализировать работу конечного автомата

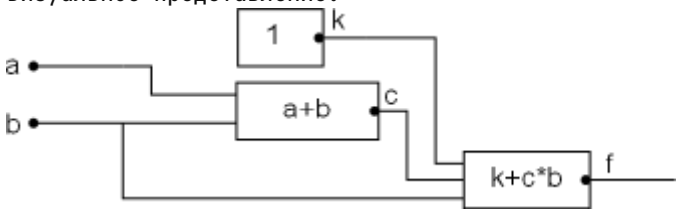
Напишите также транслятор описаний конечных автоматов (на языке C/C++ или любом другом языке).

2.4. (Arithmetic schemes <--> visual representation) Создайте язык для описания логических схем, в которых сигнал, передаваемый по проводам, представляет собой целое число, а роль элементарные логических элементов играют произвольные арифметические выражения от входных сигналов. В качестве языка описания схемы можно выбрать один из существующих языков: Haskell, Schema, VeryLog. Разработайте визуальный редактор схем, который позволял бы сохранять результат в виде тектового описания на выбранном языке. Добавьте также возможность получать более-менее наглядное визуальное представление по данному текстовому описанию. Для создания визуального инструмента можно использовать GraphViz и поставляемый с ним визуальный инструмент tcldot.

Текстовое описание:

```
input: a b
output: f
k = 1;
c = a + b;
f = k + c*b;
```

Визуальное представление:



2.5. (Statement-based database) Придумайте язык запросов к базе данных, в которой хранится неупорядоченное множество кортежей длины 4. При этом четыре элемента кортежа играют следующие роли:

контекст субъект предикат объект

Например, хранилище может содержать такие кортежи (квадруплеты):

контекст	**субъект**	**предикат**	**объект**
Дарвин	человек	произошел	обезьяна
Религия	бог	создал	человек
Магазин1	РучкаN123	стоит	10 руб
Магазин1	РучкаN123	имеет_класс	канц.товар
Магазин1	Вася	купил	РучкаN123

Язык запросов должен, в частности, позволять писать такие запросы:

1) Найти все объекты класса “канцелярский товар”, которые купил Вася и которые стоят больше 10 рублей.

2) Найти все объекты, которые используются в роли объекта в обоих контекстах “религия” и “Дарвин”.

3) Найти все объекты класса "Продукт", для которых более чем один раз указан размер скидки, то есть в базе есть более одного предложения с предикатом "имеет скидку".

Напишите простейшую реализацию квадруплетного хранилища и языка запросов к нему.

2.6. (переводчик на Рефал) Напишите на языке Рефал программу, которая осуществляет перевод простейших предложений с английского на русский (10 глаголов, 10 прилагательных, 10 наречий, предлоги и местоимения). Попробуйте заложить в неё как можно больше возможностей. Сделайте так, чтобы присутствовало слово, которое переводится по-разному в зависимости от контекста, в котором оно встретилось.

2.7. (Универсальный язык разметки) Создайте расширение языка Tcl, которое применялась бы для создания структурированных текстов и позволяло бы легко конвертировать записанные тексты в различные форматы (XML, HTML, LaTeX, ...). Важной особенность этого языка должно стать интеграция императивного и декларативного подходов. В частности данный язык должен позволить писать алгоритмы генерации серии задач и получать различные условные компиляции учебников, и др.