Министерство образования Республики Беларусь

Учреждение образования
“Белорусский государственный университет информатики и радиоэлектроники”

Кафедра интеллектуальных информационных технологий
Факультет информационных технологий и управления

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

к лабораторному практикуму

ПО КУРСУ:

« Языковые процессоры интеллектуальных систем и интеллектуализация CASE-технологий »

МИНСК

Целью изучения дисциплины «Языковые процессоры интеллектуальных систем» является рассмотрение теории построения языковых процессоров, методов анализа и проектирования языков программирования различного назначения, в том числе используемых в интеллектуальных системах.

В результате изучения дисциплины студенты должны

знать:

- основные концепции языков программирования;
- теорию построения компиляторов;
- методы и способы формального определения синтаксиса и семантики языков различного назначения;

уметь:

- разрабатывать основные блоки языковых процессоров;
- разрабатывать формальные спецификации языков различного назначения.

1. СТРУКТУРА ЛАБОРАТОРНОГО ПРАКТИКУМА

Лабораторный практикум состоит из двух частей:

1. Реализовать компилятор для языка программирования.
2. Разработать формальную спецификацию языка программирования с заданными характеристиками.

Тематический план лабораторного практикума следующий:

№	Тема	Состав
1.	Уточнение задания. Разработка языка	подгруппа
2.	Уточнение задания. Разработка языка	группа
3.	Разработка лексического анализатора	подгруппа
4.	Решение задач. Конечные автоматы. Регулярные выражения.	группа
5.	Разработка синтаксического анализатора. Формирование грамматики языка	подгруппа
6.	Решение задач. Построение таблицы предиктивного анализатора	группа
7.	Разработка синтаксического анализатора. Синтаксически управляемая трансляция	подгруппа
8.	Решение задач. Построение таблицы SLR-анализатора	группа
9.	Генерация промежуточного кода.	подгруппа
10.	Решение задач. Синтаксически управляемая трансляция	группа
11.	Выполнение промежуточного кода	подгруппа
12.	Защита первой части практикума (Разработка компилятора)	группа
13.	Разработка языка программирования.	подгруппа
14.	Решение задач.	группа

15.	Разработка языка программирования.	подгруппа
16.	Защита лабораторного практикума.	группа

2. ОПИСАНИЕ ПЕРВОЙ ЧАСТИ ЛАБОРАТОРНОГО ПРАКТИКУМА

2.1. Требования к разрабатываемому языку

1. Встроенные типы
2. Возможность инициализация переменных всех типов при объявлении: <тип> <имя_переменной> = <выражение>
   2.1. Инициализирующее выражение может быть константным
3. Встроенные операции
4. Встроенные функции
   4.1. Встроенные функции ввода\вывода для работы со встроенными типами
5. Использование сложных выражений (составных и со скобками)
6. Блочный оператор
7. Управляющие структуры
   7.1. Условный оператор (if-then-else)
   7.2. Операторы цикла (while и until)
   7.3. Оператор цикла с итерациями (for)
8. Пользовательские подпрограммы
   8.1. Передача и возврат параметров
   8.2. Задание локальной и глобальной области видимости для имен переменных

2.2. Варианты свойств языка

1. Объявление переменных
   1.1. Явное
   1.2. Не явное
2. Преобразование типов
   2.1. Явное, например, a = (int) b
   2.2. Не явное
3. Оператор присваивания
   3.1. Одноцелевой, например, a = b
   3.2. Многоцелевой, например, a, b = c, d
4. Структуры, ограничивающие область видимости
   4.1. Подпрограммы
   4.2. Подпрограммы и блочные операторы

5. Маркер блочного оператора
   5.1. Явные, например, { } или begin end
   5.2. Не явный, например как в python
6. Условные операторы
   6.1. Двух вариантный оператор if-then-else
   6.2. Двух вариантный оператор и многовариантный switch-case
7. Перегрузка подпрограмм
   7.1. Отсутствует
   7.2. Присутствует
8. Передача параметров в подпрограмму
   8.1. Только по значению и возвращаемому значению
   8.2. По значению и результату
   8.3. По ссылке
9. Допустимое место объявления подпрограмм:
   9.1. В начале программы
   9.2. В любом месте программы, также и внутри другой подпрограммы.

2.3. Варианты языков

1. Язык, описывающий математические вычисления
1.1. Встроенные типы: int, float
1.2. Операции: +, -, *, \, %, ^, ==, !=, <, >, <=, >=
2. Язык для работы с векторами и матрицами
2.1. Встроенные типы: vector, matrix
2.2. Операции: v + v, v - v, n * v, v * v, |v|, m + m, m - m, m * n, m * m, m[n], m[n] * n, |m|, v * m
3. Язык для работы с графовыми структурами
3.1. Встроенные типы: node, arc, graph
3.2. Операции: переопределить +, -, *, \ и т.д. для встроенных типов
4. Язык для работы с множествами
4.1. Встроенные типы: element, set
4.2. Операции: переопределить +, -, *, \ и т.д. для встроенных типов
5. Язык для работы со строками
5.1. Встроенные типы: char, string, массив string
5.2. Операции: переопределить +, -, *, \ и т.д. для встроенных типов
6. Язык для работы со списковыми структурами
6.1. Встроенные типы: element, list
6.2. Операции: переопределить +, -, *, \ и т.д. для встроенных типов
7. Язык для работы с xml данными
7.1. Встроенные типы: document, node, attribute
8. Язык для работы с реляционными данными
8.1. Встроенные типы: table, row, column

2.4. Варианты целевого кода

1. Исполняемый файл (.exe), формат промежуточного кода – язык C++, генерация целевого кода стандартным компилятором.
2. Байт-код JVM, формат промежуточного кода – язык Java, генерация целевого кода стандартным компилятором (javac).
3. Байт-код JVM, формат промежуточного кода ассемблер для JVM
4. Байт-код .NET, формат промежуточного кода ассемблер для .NET (CIL)
5. Исполняемый файл (.exe), для генерации целевого кода использовать LLVM (http://llvm.org)
6. Исполняемый файл (.exe), для генерации целевого код использовать GCC Front end (входит в состав GNU GCC)

2.5. Варианты заданий

Вариант	Язык	Свойства									Целевой код
		1	2	3	4	5	6	7	8	9	
1	1	1	1	1	2	1	1	1	1	1	1
2	1	1	1	1	1	2	1	1	1	1	2
3	1	1	1	1	1	1	1	1	1	2	3
4	2	1	1	1	1	1	1	1	1	1	1
5	3	1	1	1	1	1	1	1	1	1	2
6	4	1	1	1	2	1	1	1	1	1	3
7	4	1	1	1	1	2	1	1	1	1	1
8	4	1	1	1	1	1	1	1	1	2	2
9	5	1	1	1	1	1	1	1	1	1	3
10	6	1	1	1	2	1	1	1	1	1	1
11	6	1	1	1	1	2	1	1	1	1	2
12	6	1	1	1	1	1	1	1	1	2	3
13	1	2	1	1	2	1	1	1	1	1	1
14	1	2	1	1	1	2	1	1	1	1	2
15	1	2	1	1	1	1	1	1	1	2	3
16	2	1	1	1	1	1	1	1	1	1	1
17	3	1	1	1	1	1	1	1	1	1	2
18	4	2	1	1	2	1	1	1	1	1	3
19	4	2	1	1	1	2	1	1	1	1	1
20	4	2	1	1	1	1	1	1	1	2	2
21	5	2	1	1	1	1	2	1	1	1	3
22	6	2	1	1	2	1	1	1	1	1	1
23	6	2	1	1	1	2	1	1	1	1	2
24	6	2	1	1	1	1	1	1	1	2	3
25	2	1	1	1	1	1	1	1	1	1	1

26	3	1	1	1	1	1	1	1	1	1	2
27	4	2	2	1	1	1	2	1	1	1	3
28	7	1	1	1	1	1	1	1	1	2	1
29	8	1	1	1	1	1	1	1	1	1	2
30	8	2	1	1	1	1	2	1	1	1	3

2.6. Порядок выполнения и промежуточная отчетность

№ занятия	Тема
1.	**Что должно быть**: **Что делаем на занятии**: Выбираем вариант разрабатываемого языка Пишем пример какого-нибудь алгоритма на придуманном языке. **Что делаем дома**: Разрабатываем минимум 3 (три) примера логически цельных алгоритма на придуманном языке Начинаем описывать синтаксис языка, используя формат ANTLR
2.	**Что должно быть**: Три (3) текстовых файла с примерами на придуманном языке Первая версия синтаксиса языка, описанная в формате ANTLR **Что делаем на занятии**: Дополняем синтаксис языка. Генерируем код синтаксического анализатора и добавляем “main” **Что делаем дома**: Дописываем синтаксический анализатор и тестируем его на имеющихся примерах.
3.	**Что должно быть**: Примеры, файл грамматики, программный код синтаксического анализатора. **Что делаем на занятии**: Определяем набор семантических правил для языка. Добавляем в грамматику действия по выполнению семантических правил. Выбираем вариант целевого кода **Что делаем дома**: Дописываем семантический анализатор и проверяем на примерах выдачу синтаксических и семантических ошибок. Изучаем формат выбранного целевого кода.
4.	**Что должно быть**: Примеры, файл грамматики, программный код синтаксического и

	семантического анализатора **Что делаем на занятии**: Добавляем в грамматику действия по формированию целевого кода. **Что делаем дома**: Доделываем программный код компилятора и проверяем его на примерах.
5.	**Что должно быть**: Версия компилятора с некоторыми небольшими недоделками **Что делаем на занятии**: Устраняем недоделки в компиляторе. Тестируем компилятор. Формируем отчет по лабораторной работе. **Что делаем дома**: Формируем отчет по лабораторной работе. Подготавливаемся к защите лабораторной работы.
6.	**Что должно быть**: Программный код компилятора. Собранный компилятор. Примеры. Отчет. **Что делаем на занятии**: Защищаем лабораторную работу. Выбираем вариант задания по второй лабораторной работе **Что делаем дома**: Готовимся ко второй лабораторной работе.

2.7. Форма итоговой отчётности

Защита лабораторного практикума состоит из следующих частей:

1. Демонстрация практических результатов и проверка корректность работы и соответствие документации.
2. Проверка документации и результатов аналитического задания.
3. Ответ на контрольные вопросы.

Содержание отчета по первой части лабораторного практикума следующее:

1. Спецификация разработанного языка программирования.
   1.1. Синтаксис объявления переменных и подпрограмм
   1.2. Синтаксис операций над данными (их должно быть не менее 10 штук)
   1.3. Синтаксис всех управляющих конструкций
2. Оформленный (отформатированный и прокомментированный) файл грамматики (.g).
3. Описание дополнительно разработанных классов.
4. Перечень генерируемых ошибок.
5. Примеры работы компилятора.

Требования к оформлению отчета аналогичны требованиям оформления научно-технической документации. Допускается оформления фрагментов исходных текстов и примеров шрифтом меньшего размера.