МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное учреждение
высшего профессионального образования
"Казанский (Приволжский) федеральный университет"
Институт вычислительной математики и информационных технологий



*подписано электронно-цифровой подписью*

# Программа дисциплины

Языки программирования и методы трансляции Б3.В.8

Направление подготовки: 010400.62 - Прикладная математика и информатика
Профиль подготовки: Математическое и программное обеспечение вычислительных машин и сетей
Квалификация выпускника: бакалавр
Форма обучения: очное
Язык обучения: русский

**Автор(ы):**

Еникеев А.И.

**Рецензент(ы):**

Бухараев Н.Р.

**СОГЛАСОВАНО:**

Заведующий(ая) кафедрой: Еникеев А. И.
Протокол заседания кафедры No ___ от "____" _____ 201__г
Учебно-методическая комиссия Института вычислительной математики и информационных технологий:
Протокол заседания УМК No ____ от "____" _____ 201__г

Регистрационный No 9170714

Казань
2014

## Содержание

Программу дисциплины разработал(а)(и) доцент, к.н. (доцент) Еникеев А.И. кафедра технологий программирования отделение фундаментальной информатики и информационных технологий , a_eniki@inbox.ru

## 1. Цели освоения дисциплины

Цель курса - дать студентам концептуальное понимание систем программирования ,принципов их разработки и реализации, с тем, чтобы студенты могли самостоятельно анализировать и решать теоретические и практические задачи, связанные с разработкой и реализацией языков программирования. Курс основывается на системах программирования ЛИСП , DELPHI и Visual FoxPro.

## 2. Место дисциплины в структуре основной образовательной программы высшего профессионального образования

Данная учебная дисциплина включена в раздел " Б3.В.8 Профессиональный" основной образовательной программы 010400.62 Прикладная математика и информатика и относится к вариативной части. Осваивается на 3 курсе, 6 семестр.

Данная дисциплина относится к профессиональным дисциплинам.

Читается на 3 курсе 6 семестр для студентов, обучающихся по направлению "Прикладная математика и информатика".

## 3. Компетенции обучающегося, формируемые в результате освоения дисциплины /модуля

В результате освоения дисциплины формируются следующие компетенции:

| Шифр компетенции | Расшифровка приобретаемой компетенции |
|---|---|
| ОК-12 (общекультурные компетенции) | способность работать с информацией в глобальных компьютерных сетях |
| ПК-1 (профессиональные компетенции) | способность демонстрации общенаучных базовых знаний естественных наук, математики и информатики, понимание основных фактов, концепций, принципов теорий, связанных с прикладной математикой и информатикой; |
| ПК-12 (профессиональные компетенции) | способность составлять и контролировать план выполняемой работы, планировать необходимые для выполнения работы ресурсы, оценивать результаты собственной работы; |
| ПК-2 (профессиональные компетенции) | способность приобретать новые научные и профессиональные знания, используя современные образовательные и информационные технологии, |
| ПК-7 (профессиональные компетенции) | способность применять на практике международные и профессиональные стандарты информационных технологий, современные парадигмы и методологии, инструментальные и вычислительные средства (в соответствии с профилем подготовки) |

В результате освоения дисциплины студент:

1. должен знать:

основные методы компиляции языков программирования.

2. должен уметь:

Анализировать и решать теоретические и практические задачи, связанные с разработкой и реализацией языков программирования.

3. должен владеть:

теоретическими знаниями об основных методах разработки и реализации языков программирования,

4. должен демонстрировать способность и готовность:

Применять:

основные методы трансляции языков программирования.

Проявить:

теоретические знания об основных методах разработки и реализации языков программирования.

## 4. Структура и содержание дисциплины/ модуля

Общая трудоемкость дисциплины составляет 2 зачетных(ые) единиц(ы) 72 часа(ов).

Форма промежуточного контроля дисциплины зачет в 6 семестре.

Суммарно по дисциплине можно получить 100 баллов, из них текущая работа оценивается в 50 баллов, итоговая форма контроля - в 50 баллов. Минимальное количество для допуска к зачету 28 баллов.

86 баллов и более - "отлично" (отл.);

71-85 баллов - "хорошо" (хор.);

55-70 баллов - "удовлетворительно" (удов.);

54 балла и менее - "неудовлетворительно" (неуд.).

### 4.1 Структура и содержание аудиторной работы по дисциплине/ модулю

**Тематический план дисциплины/модуля**

| N | Раздел Дисциплины/ Модуля | Семестр | Неделя семестра | Виды и часы аудиторной работы, их трудоемкость (в часах) | | | Текущие формы контроля |
|---|---|---|---|---|---|---|---|
| | | | | Лекции | Практические занятия | Лабораторные работы | |
| 1. | Тема 1. Системы программирования. Основные определения и классификация. | 6 | | 0 | 3 | 0 | дискуссия |
| 2. | Тема 2. Функциональные языки программ-мирования. Язык программирования ЛИСП. Рекурсивное определение функций, обработка линейных и древовидных списков. | 6 | | 0 | 3 | 0 | тестирование |

| N | Раздел Дисциплины/ Модуля | Семестр | Неделя семестра | Виды и часы аудиторной работы, их трудоемкость (в часах) | | | Текущие формы контроля |
|---|---|---|---|---|---|---|---|
| | | | | Лекции | Практические занятия | Лабораторные работы | |
| 3. | Тема 3. Спецификация, верификация и синтез программ. На уровне определений и основных понятий. | 6 | | 0 | 3 | 0 | тестирование |
| 4. | Тема 4. Системы параллельного программирования. Теория взаимодействующих процессов и ее использование для спецификации и анализа параллельных процессов. | 6 | | 0 | 3 | 0 | тестирование |
| 5. | Тема 5. Объектно-ориентированное программирование. Основные определения и принципы. | 6 | | 0 | 3 | 0 | тестирование |
| 6. | Тема 6. Процесс компиляции. Основные понятия об этапах компиляции. | 6 | | 0 | 3 | 0 | тестирование |
| 7. | Тема 7. Определение языка. Грамматики, классификация языков по Хомскому, способы описания синтаксиса и семантики языков программирования. | 6 | | 0 | 3 | 0 | тестирование |
| 8. | Тема 8. Лексический анализ. Основные понятия, распознавание символов, лексические затруднения. | 6 | | 0 | 3 | 0 | тестирование |
| 9. | Тема 9. Синтаксический анализ. LL[k] ? грамматики. Нисходящий синтаксический анализ. Восходящий синтаксический анализ. | 6 | | 0 | 3 | 0 | тестирование |

| N | Раздел Дисциплины/ Модуля | Семестр | Неделя семестра | Виды и часы аудиторной работы, их трудоемкость (в часах) | | | Текущие формы контроля |
|---|---|---|---|---|---|---|---|
| | | | | Лекции | Практические занятия | Лабораторные работы | |
| 10. | Тема 10. . Семантический анализ, таблицы компиляторов ( таблицы символов, таблицы типов, другие таблицы ) | 6 | | 0 | 3 | 0 | тестирование |
| 11. | Тема 11. Распределение памяти. Статическая и динамическая память. | 6 | | 0 | 3 | 0 | тестирование |
| 12. | Тема 12. Генерация кода (создание промежуточного кода, создание машинного кода , оптимизация кода). | 6 | | 0 | 3 | 0 | тестирование |
| . | Тема . Итоговая форма контроля | 6 | | 0 | 0 | 0 | зачет |
| | Итого | | | 0 | 36 | 0 | |

**4.2 Содержание дисциплины**

**Тема 1. Системы программирования. Основные определения и классификация.**

***практическое занятие (3 часа(ов)):***

Изучение типовых конструкций и специфики различных систем программирования.

**Тема 2. Функциональные языки программ- мирования. Язык программирования ЛИСП. Рекурсивное определение функций, обработка линейных и древовидных списков.**

***практическое занятие (3 часа(ов)):***

Изучение методов программирования на языке ЛИСП

**Тема 3. Спецификация, верификация и синтез программ. На уровне определений и основных понятий.**

***практическое занятие (3 часа(ов)):***

Изучение методов и средств спецификации, верификации и генерации программ.

**Тема 4. Системы параллельного программирования. Теория взаимодействующих процессов и ее использование для спецификации и анализа параллельных процессов.**

***практическое занятие (3 часа(ов)):***

Изучение методов и средств спецификации и анализа параллельных процессов.

**Тема 5. Объектно-ориентированное программирование. Основные определения и принципы.**

***практическое занятие (3 часа(ов)):***

Изучение принципов объектно-ориентированного программирования.

**Тема 6. Процесс компиляции. Основные понятия об этапах компиляции.**

***практическое занятие (3 часа(ов)):***

Изучение основных этапов компиляции (этап предварительной обработки, лексический анализ, синтаксический анализ, генерация и оптимизация кода, распределение памяти).

**Тема 7. Определение языка. Грамматики, классификация языков по Хомскому, способы описания синтаксиса и семантики языков программирования.**

***практическое занятие (3 часа(ов)):***

Изучение грамматик и способов описания синтаксиса и семантики языков программирования.

**Тема 8. Лексический анализ. Основные понятия, распознавание символов, лексические затруднения.**

***практическое занятие (3 часа(ов)):***

Изучение методов лексического анализа.

**Тема 9. Синтаксический анализ. LL[k] ? грамматики. Нисходящий синтаксический анализ. Восходящий синтаксический анализ.**

***практическое занятие (3 часа(ов)):***

Изучение методов синтаксического анализа.

**Тема 10. . Семантический анализ, таблицы компиляторов ( таблицы символов, таблицы типов, другие таблицы )**

***практическое занятие (3 часа(ов)):***

Изучение методов семантического анализа.

**Тема 11. Распределение памяти. Статическая и динамическая память.**

***практическое занятие (3 часа(ов)):***

Изучение способов распределения памяти и организации статической и динамической памяти.

**Тема 12. Генерация кода (создание промежуточного кода, создание машинного кода , оптимизация кода).**

***практическое занятие (3 часа(ов)):***

Изучение методов генерации и оптимизации объектного кода

### 4.3 Структура и содержание самостоятельной работы дисциплины (модуля)

| N | Раздел Дисциплины | Семестр | Неделя семестра | Виды самостоятельной работы студентов | Трудоемкость (в часах) | Формы контроля самостоятельной работы |
|---|---|---|---|---|---|---|
| 1. | Тема 1. Системы программирования. Основные определения и классификация. | 6 | | подготовка к тестированию | 3 | тестирование |
| 2. | Тема 2. Функциональные языки программ-мирования. Язык программирования ЛИСП. Рекурсивное определение функций, обработка линейных и древовидных списков. | 6 | | подготовка к тестированию | 3 | тестирование |
| 3. | Тема 3. Спецификация, верификация и синтез программ. На уровне определений и основных понятий. | 6 | | подготовка к тестированию | 3 | тестирование |

| N | Раздел Дисциплины | Семестр | Неделя семестра | Виды самостоятельной работы студентов | Трудоемкость (в часах) | Формы контроля самостоятельной работы |
|---|---|---|---|---|---|---|
| 4. | Тема 4. Системы параллельного программирования. Теория взаимодействующих процессов и ее использование для спецификации и анализа параллельных процессов. | 6 | | подготовка к тестированию | 3 | тестирование |
| 5. | Тема 5. Объектно-ориентированное программирование. Основные определения и принципы. | 6 | | подготовка к тестированию | 3 | тестирование |
| 6. | Тема 6. Процесс компиляции. Основные понятия об этапах компиляции. | 6 | | подготовка к тестированию | 3 | тестирование |
| 7. | Тема 7. Определение языка. Грамматики, классификация языков по Хомскому, способы описания синтаксиса и семантики языков программирования. | 6 | | подготовка к тестированию | 3 | тестирование |
| 8. | Тема 8. Лексический анализ. Основные понятия, распознавание символов, лексические затруднения. | 6 | | подготовка к тестированию | 3 | тестирование |
| 9. | Тема 9. Синтаксический анализ. LL[k] ? грамматики. Нисходящий синтаксический анализ. Восходящий синтаксический анализ. | 6 | | подготовка к тестированию | 3 | тестирование |
| 10. | Тема 10. . Семантический анализ, таблицы компиляторов ( таблицы символов, таблицы типов, другие таблицы ) | 6 | | подготовка к тестированию | 3 | тестирование |

| N | Раздел Дисциплины | Семестр | Неделя семестра | Виды самостоятельной работы студентов | Трудоемкость (в часах) | Формы контроля самостоятельной работы |
|---|---|---|---|---|---|---|
| 11. | Тема 11. Распределение памяти. Статическая и динамическая память. | 6 | | подготовка к тестированию | 3 | тестирование |
| 12. | Тема 12. Генерация кода (создание промежуточного кода, создание машинного кода , оптимизация кода). | 6 | | подготовка к тестированию | 3 | тестирование |
| | Итого | | | | 36 | |

**5. Образовательные технологии, включая интерактивные формы обучения**

происходит в форме практических и самостоятельных работ

**6. Оценочные средства для текущего контроля успеваемости, промежуточной аттестации по итогам освоения дисциплины и учебно-методическое обеспечение самостоятельной работы студентов**

**Тема 1. Системы программирования. Основные определения и классификация.**

тестирование , примерные вопросы:

Углубленное изучение литературы по теме. Решение задач. Выполнение лабораторных работ.

**Тема 2. Функциональные языки программ- мирования. Язык программирования ЛИСП. Рекурсивное определение функций, обработка линейных и древовидных списков.**

тестирование , примерные вопросы:

Углубленное изучение литературы по теме. Решение задач. Выполнение лабораторных работ.

**Тема 3. Спецификация, верификация и синтез программ. На уровне определений и основных понятий.**

тестирование , примерные вопросы:

Углубленное изучение литературы по теме. Решение задач. Выполнение лабораторных работ.

**Тема 4. Системы параллельного программирования. Теория взаимодействующих процессов и ее использование для спецификации и анализа параллельных процессов.**

тестирование , примерные вопросы:

Углубленное изучение литературы по теме. Решение задач. Выполнение лабораторных работ.

**Тема 5. Объектно-ориентированное программирование. Основные определения и принципы.**

тестирование , примерные вопросы:

Углубленное изучение литературы по теме. Решение задач. Выполнение лабораторных работ.

**Тема 6. Процесс компиляции. Основные понятия об этапах компиляции.**

тестирование , примерные вопросы:

Процесс компиляции. Основные понятия об этапах компиляции.

**Тема 7. Определение языка. Грамматики, классификация языков по Хомскому, способы описания синтаксиса и семантики языков программирования.**

тестирование , примерные вопросы:

Углубленное изучение литературы по теме. Решение задач. Выполнение лабораторных работ.

**Тема 8. Лексический анализ. Основные понятия, распознавание символов, лексические затруднения.**

тестирование , примерные вопросы:

Углубленное изучение литературы по теме. Решение задач. Выполнение лабораторных работ.

**Тема 9. Синтаксический анализ. LL[k] ? грамматики. Нисходящий синтаксический анализ. Восходящий синтаксический анализ.**

тестирование , примерные вопросы:

Углубленное изучение литературы по теме. Решение задач. Выполнение лабораторных работ.

**Тема 10. . Семантический анализ, таблицы компиляторов ( таблицы символов, таблицы типов, другие таблицы )**

тестирование , примерные вопросы:

Углубленное изучение литературы по теме. Решение задач. Выполнение лабораторных работ.

**Тема 11. Распределение памяти. Статическая и динамическая память.**

тестирование , примерные вопросы:

Углубленное изучение литературы по теме. Решение задач. Выполнение лабораторных работ.

**Тема 12. Генерация кода (создание промежуточного кода, создание машинного кода , оптимизация кода).**

тестирование, примерные вопросы:

Углубленное изучение литературы по теме. Решение задач. Выполнение лабораторных работ.

**Тема . Итоговая форма контроля**

Примерные вопросы к зачету:

Примерные тесты

1. С помощью каких форм нельзя организовать бесконечный цикл?
a) LOOP
b) DO
* c) DOTIMES
* d) LET
* e) PROGN
2. Когда завершится выполнение цикла (DOTIMES (I 10) (PRINT '*))
a) через 10 миллисекунд после начала
b) после вывода девяти звездочек
* c) после вывода десяти звездочек
d) после вывода одиннадцати звездочек
e) никогда, так как переменная i не изменяется в теле цикла
f) никогда, так как указано только начальное значение переменной цикла
g) цикл не будет выполнен, так как не указан шаг переменной цикла
3. Допустим ли следующий фрагмент Пролог-программы:
a(X, Y) :- b (X), b (Y).
b (X) :- c (X,_).
a(X, Y) :- c (X), d (Y).
c(z).
c(w).
c(v).
a) нет, как не определен предикат d
* b) нет, так как описание процедуры a "разорвано" описанием предиката b
c) нет, так в описании процедуры c отсутствуют переменные
d) нет, так в описании процедуры c предусмотрен только один аргумент
4. Как Пролог-переменные получают конкретное значение

a) в результате выполнения конструкции вида "X=5"

b) в результате вызова специального предиката, например, let (X, 5)

c) в результате сопоставления термов друг с другом их аргументы приобретают новые значения

* d) в результате сопоставления термов друг с другом их свободные аргументы приобретают конкретное значение

5. Допустим ли следующий фрагмент Пролог-программы?

a(X, Y) :- b(X), b(Y).

a(X, Y) :- c(X, Y).

b(X) :- d(X).

c(X, Y) :- d(Y), e(X).

a(X) :- not (d(Y)).

a) да, допустим

b) нет, так как описание процедуры a "разорвано" описанием предикатов b и c

c) нет, так как процедура a имеет переменное количество аргументов

d) нет, так как отсутствуют предикаты d и y

* e) да, если формат всех используемых процедур описан в секции PREDICATES.

6. В какой секции Пролог-программы объявляются списки?

* a) domains

b) facts

c) predicates

d) clauses

e) goal

f) constants

g) globals

h) lists

i) datas

7. Какие элементы могут составлять Пролог-списки

* a) любые термы

b) числа

c) символы

d) строки

e) символы и строки

8. Укажите, какой из элементов списка может быть пустым:

a) голова

* b) хвост

c) и голова, и хвост

d) ни один из указанных

9. Сколько элементов может содержать список, соответствующий шаблону [X, Y | Z]?

a) любое количество элементов

b) не менее одного

* c) не менее двух

d) два элемента

e) не менее трех

f) три элемента

10. Верно ли утверждение, что not (not (X)) эквивалентно X?

* a) да

b) нет

c) может выполняться или нет в зависимости от содержания базы данных программы

11. Верно ли утверждение, not (not (a(X)))эквивалентно a(X)?

a) да

b) нет

* c) может выполняться или нет в зависимости от формы предиката a и содержания базы данных программы

12. Какие ответы могут быть получены в ответ на запрос

likes (x, dogs)?

* a) Yes (да)

* b) No (нет)

c) No solutions (нет решений)

d) 1 solution (1 решение)

13. Какие ответы могут быть получены в ответ на запрос

likes (_, dogs)?

a) Yes (да)

b) No (нет)

* c) No solutions (нет решений)

* d) 1 solution (1 решение)

14. Какие ответы могут быть получены в ответ на запрос

likes (_, _)?

a) неверный запрос

b) нет

* c) нет решений

15. Укажите предикат, выделяющий хвост списка

a) f ([_, X], X)

b) f ([_, X | [ ]], X)

* c) f ([_ | X], X)

d) f ([ [ ] | X], X)

e) f ([| X], X)

f) f ([X | 2], X)

16. Что такое CDR-рекурсия?

a) второе название хвостовой рекурсии

b) рекурсивный вызов в теле процедуры с помощью вспомогательного предиката CDR

* c) рекурсивный вызов, в качестве параметра которого передается хвост "исходного" аргумента-списка

d) рекурсивный вызов, в качестве параметра которому передается "исходный" аргумент-список, из которого удалён хвост

e) рекурсивная реализация процедуры CDR

17. Чем отличается лексический анализ от синтаксического

a) ничем, это одно и тоже

b) на уровне лексического анализа проверяется правильность составления лексем, при синтаксическом анализе проверяется контекст соединения различных элементов языка.

Семестровое задание №1.

Цель: Отработка техники построения программ лексического и синтаксического анализа на основе языка программирования ПАСКАЛЬ(DELPHI). Изложение

дальнейшего материала дается в терминах ПАСКАЛЯ, однако задания можно выполнять также на C++.

Предусматривается программирование отдельных алгоритмов синтаксического анализа и компиляции на основе рекурсивных определений, являющихся наиболее адекватными для упомянутого класса задач. В качестве исходного текста , являющегося объектом обработки, используются язык арифметических выражений и язык функционального программирования на основе ЛИСПа.

Срок выполнения: 2-месяца.

Пояснения к заданию:

1. Язык арифметических выражений.

Выражения строятся из простых переменных , констант (типа integer или real) ,

знаков операций (+ , - , * , / , DIV , MOD) и функций.

Например : (a1+a2)*(b1-b2 DIV b3)-15+F(x,y,z).

2. Язык фукционального программирования.

Программы строятся в виде суперпозиции функций. Например :

F1(G1(x,y),G2(a,b,c)). Каждая функция представляется в виде

полноскобочного представления вида (F , p1, p2, ? , pn) , где

F-наименование функции , p1, p2, ? , pn - аргументы (параметры) ,

каждый из которых является простой переменной , константой

(типа integer , real , boolean) или в свою очередь полноскобочным выражением.

Например (F1,(G1,x,y),(G2,a,b,c)) .

3. Язык префиксного промежуточного представления.

Промежуточное представление программы получается в результате полного синтаксического разложения программы и затем используется для

дальнейшей генерации кода или интерпретации. Промежуточное представление является структурным списком , который описывается

динамическим типом языка ПАСКАЛЬ.

```
KOD
NAME LL RL

type list=^element;
element=record
KOD:0..4;
NAME:string;
LL:list;
RL:list
end;
```

{ 1- если поле NAME определяет имя функции (или знак операции)

KOD= { 2- если поле NAME определяет имя переменной

{ 3- если поле NAME определяет константу

{ 4- поле NAME - пустая строка , поле LL ссылается на другой

{ список (нижний уровень иерархии)

LL-ссылка на нижний уровень иерархии , RL ссылка на следующий элемент данного уровня иерархии.

LL=> NIL , если KOD= 1 | 2 | 3

Пример:

( F , x ,(G,z,15))

NIL

NIL NIL

NIL

NIL NIL NIL

4. Язык постфиксного промежуточного представления.

В постфиксном представлении знак операции (функции) указывается после аргументов.
Вид представления:

В дальнейшем префиксное промежуточное представлении будем называть просто промежуточным представлением, специально выделяя постфиксное представление.

Примерные задания.

1. Преобразовать арифметическое выражение, не содержащее скобок (и соответственно функциональных символов) в промежуточное представление.
Например , A1+B1-X/15.5 .

2. Получить из промежуточного представления арифметическое выражение.

3. Преобразовать полноскобочное представление функциональной программы в промежуточное представление.

5. Получить из промежуточного представления функциональной программы полноскобочное представление.

6.Запрограммировать процесс интерпретации промежуточного представления арифметического выражения, ограничившись только знаками операций + , - ,* , / .
В качестве исходных данных кроме промежуточного представления использовать таблицу значений переменных, организованную в виде массива.

7. Преобразовать арифметическое выражение, не содержащее скобок ( и соответственно функциональных символов) в полноскобочное представление.
Например , A1+B1-X/15.5 => ( - , (+ , A1, B1) , ( / , X , 15.5 ))

8. Преобразовать полноскобочное представление арифметического выражения в обычное арифметическое выражение.
Например , ( - , (+ , A1, B1) , ( / , X , 15.5 )) => A1+B1-X/15.5

9. Выполнить проверку парности открывающих и закрывающих скобок в обычном арифметическом выражении.

10. Выполнить проверку парности открывающих и закрывающих скобок в полноскобочном представлении.

11. Выполнить проверку соответствия типов в операциях (функциях) на основе промежуточного представления. Тип переменной определяется первой буквой ее наименования (I - Integer , R-real , B-Boolean). Если на каком этапе обнаруживается ошибка несоответствия типов , процесс проверки завершается с выдачей сообщения об ошибке (далее не проверять). В качестве исходных данных кроме промежуточного представления использовать таблицу схем операций (функций). Например , (+,Integer ,integer) => integer .

12. На основе промежуточного представления создать линейный список, содержащий список наименований используемых операций (функций).

13. Преобразовать префиксное промежуточное представление в постфиксное.

14. Преобразовать постфиксное промежуточное представление в префиксное.

15.Запрограммировать процесс интерпретации постфиксного промежуточного представления арифметического выражения, ограничившись только знаками операций + , - ,* , / .

В качестве исходных данных кроме промежуточного представления использовать

таблицу значений переменных, организованную в виде массива.

16. Выполнить лексический анализ полноскобочного представления , ограничившись только знаками операций + , - ,* , / (не используя функциональные символы). Суть лексического анализа сводится к проверке правильности написания переменных , констант и знаков операций . Например :

(+,(-,a1,b1),(*,c1,15)) - выражение корректное ; (+,(-,a,b),(*,c1,1d1)) - в выражении допущена ошибка (1d1- не является идентификатором) .

17.Для заданного промежуточного представления получить списки наименований операций , функций , переменных и констант. Например , если промежуточное представление получено из выражения F(X1,255)+G(X1*Y1/16,Y2) , то в результате мы должны получить (+,*) -список операций , (F,G) -список функций , (X1,Y1,Y2) - список переменных , (255,16) - список констант.

18.Для заданного полноскобочного представления получить списки наименований операций , функций , переменных и констант. Например , если промежуточное представление получено из выражения F(X1,255)+G(X1*Y1/16,Y2) , то в результате мы должны получить (+,*) -список операций , (F,G) -список функций , (X1,Y1,Y2) - список переменных , (255,16) - список констант.

Семестровое задание № 2.

Цель: Отработка техники построения программ в объектно-ориентированной

среде.

Каждая из следующих ниже задач предусматривает построение соответствующей экранной формы.

Срок выполнения: 1 месяц.

Примерные задания.

1.Формирование иерархического справочника. Экранная форма предусматривает

вывод одной таблицы из 2-х полей KOD(C,21) и NAME(C,50) , где KOD -код элемента справочника , NAME -наименование . Например :

001 КГУ

001001 АДМИНИСТРАЦИЯ

001002 ФАКУЛЬТЕТЫ

001002001 ВМК

001002002 МЕХМАТ

002 КГТУ

и т.п. Предусмотреть средства корректировки , добавления и удаления записей.

2.Формульный интерпретатор. Предусмотрено использование 2-х таблиц - таблицы формул TABF и основной таблицы TABO, над которой проводятся расчеты по формулам из таблицы TABF .

Структура TABF :

COND(C,20) -поле условия (например P1>0.AND.P2<=0 или .T. -true)

OBF(C,10)- поле для наименования объекта присваивания (например P1)

FORM(C,50) - поле для записи формулы (например P2+P3)

COMM(C,80) - поле комментария

Структура TABO :

P1(N,10,2) , P2(N,10,2) , ? , P10(N,10,2)

Предусмотреть средства корректировки , добавления и удаления записей , а также командной кнопки для запуска расчета.

3.Телефонный справочник. Информационная таблица TABINF имеет следующую структуру : FIO (C,50) -фамилия и.о. , ADDRESS(C70) - адрес , NTEL(C,10) -

-• телефона.Предусмотреть средства корректировки , добавления и удаления записей а также командной кнопки для запроса.

4.Справочник движения поездов. Информационная таблица TABINF имеет следующую структуру :CITY(C,50)-город , DAY(C,12) -дни недели (например '1,2,3') , DEPT (C,5)--время отправления (например '12.30') , ARRT(C,5) - время прибытия , WAGT(C,10)-тип поезда (например 'скорый') .Предусмотреть средства корректировки , добавления и удаления записей а также командной кнопки для запроса.

5.Энциклопедия. Информационная таблица TABINF имеет следующую структуру : WORD(C,50) - термин (например Казань) , CONT (Memo) -поясняющий текст (например 'Столица Татарстана'). Предусмотреть средства корректировки , добавления и удаления записей а также командной кнопки для запроса.

6.Поиск по ключевым словам. Информационная таблица TABINF имеет следующую структуру :CONT(Memo)-текст. По заданному ключевому слову осуществляется выборка записей с вхождением данного ключевого слова. Предусмотреть средства корректировки , добавления и удаления записей а также командной кнопки для запроса.

7.Задача отдела кадров. Информационная таблица TABINF имеет следующую структуру : FIO(C,50) -фамилия и.о. , ADDRESS(C70) - адрес, NTEL(C,10) -

-• телефона, SEX (C,1) -пол , DATRO (Date) -дата рождения , MESTORO(C,50) -место рождения .Предусмотреть средства корректировки , добавления и удаления записей а также командной кнопки для запроса.

8.Задача анализа заработной платы. Информационная таблица TABINF имеет следующую структуру : FIO(C,50) -фамилия и.о. , SUM(N,10,2)-начисленная зарплата.

Задача заключается в выборке первых n-высокооплачиваемых и первых m-низкооплачиваемых сотрудников и нахождении средней зарплаты по каждой из упомянутых групп (параметры m и n определяются запросом). Предусмотреть средства корректировки , добавления и удаления записей а также командной кнопки для запроса.

9.Расчет заработной платы.

Информационная таблица TABINF имеет следующую структуру : FIO(C,50) -фамилия и.о. , TABN(C,5)- табельный номер сотрудника,SUM(N,10,2)-начисленная зарплата,

NAL(N,10,2) - подоходный налог, VID(N,10,2) - сумма к выдаче. Расчет подоходного

налога и суммы к выдаче осуществляется по формулам: NAL=SUM*0.13,

VID=SUM-NAL. Предусмотреть средства корректировки , добавления и удаления записей а также командной кнопки для расчета. Кроме этого необходимо предусмотреть выборку по табельному номеру и фамилии и.о.

10.Генератор таблиц и экранных форм.

На основе специальной таблицы описания автоматически создавать новую таблицу и экранную форму для отображения построенной таблицы. При изменении таблицы описания необходимо автоматически менять структуру генерируемой таблицы и экранной формы. Таблица описания имеет следующую структуру:

FIELD_NAME (C,10) - для имен полей генерируемой таблицы (например FIO),

FIELD_CAP (C,50) - для русифицированных имен (например ФИО),

FIELD_TYPE (C,1) - для типов полей (C,N,D,?), FIELD_LEN (N,3) - длина поля,

FIELD_DEC (N,3) - количество цифр после запятой для типа N.

11.Конвертация в EXCEL - формат.

Построить программные средства для автоматической конвертации таблиц СУБД

в EXCEL - формат. Исходную информацию (какие поля таблицы и в каком порядке) для конвертации вводить в соответствующую экранную форму

12.Русско-английский словарь.

Организуется в виде таблицы DICTION со следующей структурой:

W_RUS(C,20)- для хранения русского слова, W_ENGL(C,128)- для хранения соответствующих слов на английском языке. Предусмотреть сервис в виде экранной формы, позволяющей корректировку , добавление и удаление записей,

а также запрос на перевод слова.

Замечание: каждое задание может быть запрограммировано на любой

доступной системе программирования (VISUAL FOXPRO ,DELPHI, MS SQL,

C++), причем одно и то же задание можно предложить разным студентам, если оно выполняется в разных системах программирования .

Семестровое задание №3.

Цель: Отработка техники построения программ для реализации методов синтаксического анализа на основе языка программирования ПАСКАЛЬ(DELPHI),

C++ , а также любого другого доступного языка программирования .

В качестве исходного текста, являющегося объектом обработки, используются язык арифметических выражений, язык логических выражений, и язык функционального программирования на основе ЛИСПа. В отличие от предыдущих заданий студенты

занимаются не просто программированием готовых алгоритмов, а также берут на себя

часть постановки задачи на программирование ( в частности предусматривается определение системы продукций для грамматики языка и следовательно формальное описание синтаксиса языка). Ввиду достаточной сложности, каждое задание рекомендуется распределить на несколько студентов.

Срок выполнения: 20.05.2009 г.

Пояснения к заданию:

Суть задания заключается в распознавании принадлежности заданной синтаксической конструкции соответствующему языку. Если заданная конструкция содержит синтаксическую ошибку, то процесс анализа завешается

выдачей сообщения об ошибке. Выполнение заданий предполагает определение системы продукций для грамматики языка (правил вывода), разработку детального алгоритма реализации синтаксического анализа. Ниже представлено три языка, для которых следует создать соответствующие синтаксические анализаторы.

1. Язык арифметических выражений.

Выражения строятся из простых переменных , констант (типа integer или real) ,

знаков операций (+ , - , * , / , DIV , MOD) и функций.

Например : (a1+a2)*(b1-b2 DIV b3)-15+F(x,y,z).

2. Язык логических выражений.

Выражения строятся из простых переменных , констант (типа true или false) ,

знаков операций (AND ,OR , NOT ), отношений (=, >,>=,<=). Отношения определены

только для простых переменных и констант(типа integer или real). Нет других выражений типа, например a+c=d-1.

Пример : ((a1=a2) AND (c1>c2)) OR b.

3. Язык фукционального программирования.

Программы строятся в виде суперпозиции функций. Например :

F1(G1(x,y),G2(a,b,c)). Каждая функция представляется в виде

полноскобочного представления вида (F , p1, p2, ? , pn) , где

F-наименование функции , p1, p2, ? , pn - аргументы (параметры) ,

каждый из которых является простой переменной , константой

(типа integer , real , boolean) или в свою очередь полноскобочным выражением.

Например (F1,(G1,x,y),(G2,a,b,c)) .

4. Методы синтаксического анализа.

Пояснения к обозначениям: S - аксиома грамматики (начальный символ), большие буквы обозначают нетерминальные символы, а малые - терминальные.

Задача нисходящего анализа, как правило , сводится к нахождению левого порождения . Процесс нисходящего анализа начинается с начального символа

(аксиомы грамматики) с последующей генерацией предложения языка .

Рассмотрим в качестве примера язык вида { x m y n |m, n > 0 } , определяемый следующими продукциями :

S -> XY

X -> xX

X -> x

Y - >yY

Y -> y

Предложение xxxyy можно сгенерировать с помощью следующего левого порождения :

S => XY => xXY => xxXY => xxxY => xxxyY => xxxyy .

Алгоритм нахождения левого порождения можно наглядно

проиллюстрировать с помощью приводимой ниже таблицы .

| Входная строка | Продукция | Сентенциальная форма |
|---|---|---|
| xxxyy | S => XY | XY |
| xxxyy | X => xX | xXY |
| xxxyy | X => xX | xxXY |
| xxxyy | X => x | xxxY |
| xxxyy | X => x | xxxY |
| xxxyy | Y => yY | xxxyY |
| xxxyy | Y => yY | xxxyY |

Таблица 3.4.2 -1 . Алгоритм нахождения левого порождения .

Задача восходящего анализа , как правило , сводится к нахождению правого порождения . Рассмотрим в качестве примера рассмотренный

в предыдущих разделах язык { x m y n |m, n > 0 } , определяемый следующими продукциями :

S -> XY

X -> xX

X -> x

Y - >yY

Y -> y

Предложение xxxyy можно сгенерировать с помощью следующего правого порождения :

S => XY => XyY => Xyy => xXyy => xxX yy => xxxyy .

В отличие от нисходящего , при восходящем синтаксическом анализе

этапы порождения определяются в противоположном порядке , то есть :

xxxyy=> xxX yy=> xXyy=> Xyy=> XyY=> XY=> S .

Применение продукции грамматики на каждом этапе предусматривает

замену правой части продукции ее левой частью, состоящей из одного символа.

При восходящем синтаксическом анализе правые части продукции

не распознаются до тех пор, пока не будут полностью считаны . В связи с этим требуется хранение частично распознанных правых частей продукций до замены их соответствующими левыми частями. Для этой цели используется стек. Таким образом основные этапы процесса восходящего синтаксического анализа выражаются с помощью двух типов действий .

1) Занесение последнего считанного символа в стек - действие

переноса (SA-shift action) .

2) Замена строки наверху стека посредством применения продукции грамматики - действие свертки (RA - reduce action) .

Алгоритм восходящего синтаксического анализа иллюстрируется на приводимой ниже таблице .

| Входная строка | Стек | Продукция | Сентенциальная форма | SA \| RA |
|---|---|---|---|---|
| xxxyy | | | | |
| xxxyy | x | | | SA |
| xxxyy | xx | | | SA |
| xxxyy | xxx | | | SA |
| xxxyy | xxX | X -> x | xxXyy | RA |
| xxxyy | xX | X -> xX | xXyy | RA |
| xxxyy | X | X -> xX | Xyy | RA |
| xxxyy | Xy | | Xyy | SA |
| xxxyy | Xyy | | Xyy | SA |
| xxxyy | XyY | Y -> y | XyY | RA |
| xxxyy | XY | Y -> yY | XY | RA |
| xxxyy | S | S -> XY | S | RA |

Таблица 3.4.3 -1 . Алгоритм восходящего синтаксического анализа .

Примерные задания.

1.Создать синтаксический анализатор для языка арифметических выражений

на основе нисходящего анализа.

2.Создать синтаксический анализатор для языка арифметических выражений

на основе восходящего анализа.

3.Создать синтаксический анализатор для языка логических выражений

на основе нисходящего анализа.

4.Создать синтаксический анализатор для языка логических выражений

на основе восходящего анализа.

5.Создать синтаксический анализатор для языка фукционального программирования на основе нисходящего анализа.

6.Создать синтаксический анализатор для языка фукционального программирования на основе восходящего анализа.

### 7.1. Основная литература:

1. Информатика: Курс лекций. Учебное пособие / Е.Л. Федотова, А.А. Федотов. - М.: ИД ФОРУМ: ИНФРА-М, 2011. - 480 с.

http://znanium.com/catalog.php?bookinfo=204273

2. Информатика. Базовый курс: Учебное пособие для студентов высших технических учебных заведений / под ред. С. В. Симоновича. 2 - е изд.. - СПб [и др.]: Питер, 2008. - 639 с.

http://z3950.ksu.ru/bcover/0000758670_con.pdf

3. Андрианова А.А., Мухтарова Т.М. Практикум по курсу "Алгоритмизация и программирование" - часть 1. - Казанский государственный университет, 2008.

http://libweb.ksu.ru/ebooks/09_63.pdf

### 7.2. Дополнительная литература:

1.Могилев, А. В. Методы программирования. Компьютерные вычисления / А. В. Могилев, Л. В. Листрова. ? СПб.: БХВ-Петербург, 2008. ? 320 с. URL: http://znanium.com/bookread.php?book=350418

2.Сырецкий, Г. А. Информатика. Фундаментальный курс. Том II. Информационные технологии и системы /Г. А. Сырецкий. ? СПб.: БХВ-Петербург, 2007. ? 846 с. URL: http://znanium.com/bookread.php?book=350042

### 7.3. Интернет-ресурсы:

Википедия - http://ru.wikipedia.org

Интернет-журнал по ИТ - http://www.rsdn.ru

Интернет-портал образовательных ресурсов по ИТ - http://algolist.manual.ru/

Интернет-портал образовательных ресурсов по ИТ - http://www.intuit.ru

Портал математических интернет-ресурсов - http://www.math.ru/

### 8. Материально-техническое обеспечение дисциплины(модуля)

Освоение дисциплины "Языки программирования и методы трансляции" предполагает использование следующего материально-технического обеспечения:

Мультимедийная аудитория, вместимостью более 60 человек. Мультимедийная аудитория состоит из интегрированных инженерных систем с единой системой управления, оснащенная современными средствами воспроизведения и визуализации любой видео и аудио информации, получения и передачи электронных документов. Типовая комплектация мультимедийной аудитории состоит из: мультимедийного проектора, автоматизированного проекционного экрана, акустической системы, а также интерактивной трибуны преподавателя, включающей тач-скрин монитор с диагональю не менее 22 дюймов, персональный компьютер (с техническими характеристиками не ниже Intel Core i3-2100, DDR3 4096Mb, 500Gb), конференц-микрофон, беспроводной микрофон, блок управления оборудованием, интерфейсы подключения: USB,audio, HDMI. Интерактивная трибуна преподавателя является ключевым элементом управления, объединяющим все устройства в единую систему, и служит полноценным рабочим местом преподавателя. Преподаватель имеет возможность легко управлять всей системой, не отходя от трибуны, что позволяет проводить лекции, практические занятия, презентации, вебинары, конференции и другие виды аудиторной нагрузки обучающихся в удобной и доступной для них форме с применением современных интерактивных средств обучения, в том числе с использованием в процессе обучения всех корпоративных ресурсов. Мультимедийная аудитория также оснащена широкополосным доступом в сеть интернет. Компьютерное оборудованием имеет соответствующее лицензионное программное обеспечение.

Компьютерный класс, представляющий собой рабочее место преподавателя и не менее 15 рабочих мест студентов, включающих компьютерный стол, стул, персональный компьютер, лицензионное программное обеспечение. Каждый компьютер имеет широкополосный доступ в сеть Интернет. Все компьютеры подключены к корпоративной компьютерной сети КФУ и находятся в едином домене.

занятия по дисциплине проводятся в аудитории, оснащенной доской и мелом(маркером), а так же в специализированных компьютерных кабинетах.

Программа составлена в соответствии с требованиями ФГОС ВПО и учебным планом по направлению 010400.62 "Прикладная математика и информатика" и профилю подготовки Математическое и программное обеспечение вычислительных машин и сетей .

Автор(ы):
Еникеев А.И. _____
"__" _____ 201 __ г.

Рецензент(ы):
Бухараев Н.Р. _____
"__" _____ 201 __ г.