Государственное образовательное учреждение
"Приднестровский государственный университет им. Т.Г. Шевченко"

Инженерно-технический институт

Кафедра программного обеспечения вычислительной техники
и автоматизированных систем

УТВЕРЖДАЮ

Заведующий кафедрой ПОВТ и АС


С.Г. Федорченко

«28» августа 2020 г.

ФОНД ОЦЕНОЧНЫХ СРЕДСТВ

по дисциплине

КОНСТРУИРОВАНИЕ КОМПИЛЯТОРОВ

Направление подготовки
2.09.04.04 Программная инженерия

Профиль подготовки
Разработка программно-информационных систем

Квалификация (степень)
выпускника: **магистр**

Форма обучения: **очная, заочная**

Год набора: **2020 г.**

Разработал:
к.п.н., доцент кафедры ПОВТ и АС,


_/С.В. Помян

«28» августа 2020 г.

Тирасполь, 2020

Паспорт фонда оценочных средств по учебной дисциплине

1. В результате изучения дисциплины «Конструирование компиляторов» у обучающегося должны быть сформированы следующие компетенции:

Категория (группа) компетенций	Код и наименование	Код и наименование индикатора достижения компетенции
Общепрофессиональные компетенции и индикаторы их достижения		
-	ОПК-2. Способен разрабатывать оригинальные алгоритмы и программные средства, в том числе с использованием современных интеллектуальных технологий, для решения профессиональных задач	ИД-1$_{ОПК-2}$ Знать современные интеллектуальные технологии для решения профессиональных задач ИД-2$_{ОПК-2}$ Уметь обосновывать выбор современных интеллектуальных технологий и программной среды при разработке оригинальных программных средств для решения профессиональных задач ИД-3$_{ОПК-2}$ Иметь навыки разработки оригинальных программных средств, в том числе с использованием современных интеллектуальных технологий, для решения профессиональных задач
Профессиональные компетенции и индикаторы их достижения		
-	ПК-7. Способен проектировать трансляторы и интерпретаторы языков программирования.	ИД-1$_{ПК-7}$ Знает методы проектирования трансляторов и интерпретаторов языков программирования ИД-2$_{ПК-7}$ Умеет использовать методы проектирования трансляторов и интерпретаторов языков программирования

2. Программа оценивания контролируемой компетенции:

Текущая аттестация	Контролируемые модули, разделы (темы) дисциплины их название	Код контролируемой компетенции (или ее части)	Наименование оценочного средства
РУБЕЖНЫЙ КОНТРОЛЬ	Раздел 1 Место компилятора в системном программном обеспечении Раздел 2 Распознаватели и преобразователи	ОПК-2, ПК-7	Презентация №1 Кейс-задача №1
РУБЕЖНАЯ АТТЕСТАЦИЯ	Раздел 3 Распределение памяти Раздел 4 Генерация промежуточного кода		Кейс-задача №2 Итоговый тест
Промежуточная аттестация		Код контролируемой компетенции (или ее части)	Наименование оценочного средства
№1		ОПК-2, ПК-7	Экзамен

3. Показатели и критерии оценивания компетенции по этапам формирования, описание шкал оценивания

Этапы оценивания компетенции	Показатели достижения заданного уровня освоения компетенции	Критерии оценивания результатов обучения			
		2	**3**	**4**	**5**
Первый этап	ИД-1$_{ОПК-2}$ **Знать** современные интеллектуальные технологии для решения профессиональных задач	Не знает	Знает основные понятия, но не знает особенности применения современные интеллектуальные технологии для решения профессиональных задач	Знает современные интеллектуальные технологии для решения профессиональных задач, но допускает незначительные ошибки	Знает современные интеллектуальные технологии для решения профессиональных задач
Второй этап	ИД-2$_{ОПК-2}$ **Уметь** обосновывать выбор современных интеллектуальных технологий и программной среды при разработке оригинальных программных средств для решения профессиональных задач	Не умеет	Правильно обосновывает выбор современных интеллектуальных технологий и программной среды, но при разработке использует типовые профессиональные решения	Умеет обосновывать выбор современных интеллектуальных технологий и программной среды при разработке оригинальных программных средств для решения профессиональных задач, но допускает незначительные ошибки	Умеет обосновывать выбор современных интеллектуальных технологий и программной среды при разработке оригинальных программных средств для решения профессиональных задач
Третий этап	ИД-3$_{ОПК-2}$ **Иметь навыки** разработки оригинальных программных средств, в том числе с использованием современных интеллектуальных технологий, для решения профессиональных задач	Не владеет	Владеет навыками типовой разработки программных средств.	Владеет навыками разработки оригинальных программных средств для решения профессиональных задач.	Владеет и навыками разработки оригинальных программных средств, в том числе с использованием современных интеллектуальных технологий, для решения профессиональных задач.
Первый этап	ИД-1$_{ПК-7}$ **Знает** методы проектирования трансляторов и интерпретаторов языков программирования	Не знает	Знает основные теоретические понятия теории трансляторов и интерпретаторов языков программирования, но не знает способы проектирования	Знает методы проектирования трансляторов и интерпретаторов языков программирования, но допускает незначи-	Знает методы проектирования трансляторов и интерпретаторов языков программирования

Этапы оценивания компетенции	Показатели достижения заданного уровня освоения компетенции	Критерии оценивания результатов обучения			
		2	**3**	**4**	**5**
				тельные ошибки	
Второй этап	ИД-2$_{ПК-7}$ **Умеет** использовать методы проектирования трансляторов и интерпретаторов языков программирования	Не умеет	Умеет оценить задачу по проектированию трансляторов и интерпретаторов языков программирования, но самостоятельно не может реализовать	Умеет использовать методы проектирования трансляторов и интерпретаторов языков программирования, но допускает незначительные ошибки	Умеет использовать методы проектирования трансляторов и интерпретаторов языков программирования

4. Шкала оценивания

Согласно Положению «О порядке организации аттестации в ИТИ ПГУ им. Т.Г. Шевченко, итоговая оценка представляет собой сумму баллов, полученных студентом по итогу освоения дисциплины (модуля):

Оценка в традиционной шкале	Оценка в 100-балльной шкале	Буквенные эквиваленты оценок в шкале ЗЕ (% успешно аттестованных)
5 (отлично)	88–100	A (отлично) – 88-100 баллов
4 (хорошо)	70–87	B (очень хорошо) – 80-87баллов
		C (хорошо) – 70-79 баллов
3 (удовлетворительно)	50–69	D (удовлетворительно) – 60-69 баллов
		E (посредственно) – 50-59 баллов
2 (неудовлетворительно)	0–49	Fx – неудовлетворительно, с возможной пересдачей – 21-49 баллов
		F – неудовлетворительно, с повторным изучением дисциплины – 0-20 баллов

Расшифровка уровня знаний, соответствующего полученным баллам, дается в таблице, указанной ниже

A	“Отлично” - теоретическое содержание курса освоено полностью, без пробелов, необходимые практические навыки работы с освоенным материалом сформированы, все предусмотренные программой обучения учебные задания выполнены, качество их выполнения оценено числом баллов, близким к максимальному.
B	“Очень хорошо” - теоретическое содержание курса освоено полностью, без пробелов, необходимые практические навыки работы с освоенным материалом в основном сформированы, все предусмотренные программой обучения учебные задания выполнены, качество выполнения большинства из них оценено числом баллов, близким к максимальному.
C	“Хорошо” - теоретическое содержание курса освоено полностью, без пробелов, некоторые практические навыки работы с освоенным материалом сформированы недостаточно, все предусмотренные программой обучения учебные задания выполнены, качество выполнения ни одного из них не оценено минимальным числом баллов, некоторые виды заданий выполнены с ошибками.

D	“Удовлетворительно” - теоретическое содержание курса освоено частично, но пробелы не носят существенного характера, необходимые практические навыки работы с освоенным материалом в основном сформированы, большинство предусмотренных программой обучения учебных заданий выполнено, некоторые из выполненных заданий, возможно, содержат ошибки.
E	“Посредственно” - теоретическое содержание курса освоено частично, некоторые практические навыки работы не сформированы, многие предусмотренные программой обучения учебные задания не выполнены, либо качество выполнения некоторых из них оценено числом баллов, близким к минимальному.
FX	“Условно неудовлетворительно” - теоретическое содержание курса освоено частично, необходимые практические навыки работы не сформированы, большинство предусмотренных программой обучения учебных заданий не выполнено, либо качество их выполнения оценено числом баллов, близким к минимальному; при дополнительной самостоятельной работе над материалом курса возможно повышение качества выполнения учебных заданий.
F	“Безусловно неудовлетворительно” - теоретическое содержание курса не освоено, необходимые практические навыки работы не сформированы, все выполненные учебные задания содержат грубые ошибки, дополнительная самостоятельная работа над материалом курса не приведет к какому-либо значимому повышению качества выполнения учебных заданий.

5. Типовые контрольные задания или иные материалы, необходимые для оценки знаний, умений, навыков и (или) опыта деятельности, характеризующих этапы формирования компетенций при изучении учебной дисциплины в процессе освоения образовательной программы

5.1 Примерный перечень тематик презентации №1

1) LEX – генератор лексических анализаторов
2) Системы автоматизации построения трансляторов. Система СУПЕР.
3) Системы автоматизации построения трансляторов. Система *YACC*.
4) Динамическая компиляция
5) Обзор компилируемых языков программирования
6) Обзор интерпретируемых языков программирования
7) Кросс-трансляторы
8) Виртуальная машина
9) Сборщики мусора в трансляторах, интерпретаторах
10) Основные алгоритмы построения распознавателей с возвратами
11) Основные алгоритмы построения распознавателей без возвратов
12) Табличные распознаватели

5.2. Кейс-задача №1 КЗ1. Примерный перечень тематик и методика выставления баллов, типовой вариант

Тема: Проектирование и реализация распознавателя с возвратами заданного формального языка

Цель кейс-задачи: рассмотреть алгоритмы построения распознавателей с возвратом, закрепить навыки проектирования распознавателя с возвратами, реализовать нисходящий или восходящий распознаватель с возвратами заданного формального языка

Типовой вариант кейс-задачи

Даны:

1) формальный язык: язык содержит арифметические выражения, разделенные символом **;**(точка с запятой). Арифметические выражения состоят из идентификаторов, десятичных чисел с плавающей точкой (в обычной и логарифмической форме), знака присваивания (**:=**), знаков операций **+**, **-**, *****, **/** и круглых скобок.
2) алгоритм распознавателя: алгоритм с подбором альтернативы.
3) грамматика КС-языка:
S → **a :=** F**;**
F → F+T | T
T → T*****E | T/E | E
E → **(**F**)** | **-(**F**)** | **a**

Задание:
1) для заданного формального языка и заданной грамматики реализовать на любом выбранном самостоятельно языке программирования распознаватель с возвратами для отдельной строки по заданному алгоритму;
2) представить дерево вывода и вывод цепочек, используя правила приведенной грамматики, цепочки, принятые распознавателем и отвергнутые;
3) привести классификацию ошибок;
4) реализовать обработку ошибок с указанием типа и локализации ошибки.

Варианты формального языка:
1. Входной язык содержит арифметические выражения, разделенные символом **;**(точка с запятой). Арифметические выражения состоят из идентификаторов, десятичных чисел с плавающей точкой (в обычной и логарифмической форме), знака присваивания (**:=**), знаков операций **+**, **-**, *****, **/** и круглых скобок.
2. Входной язык содержит логические выражения, разделенные символом **;**(точка с запятой). Логические выражения состоят из идентификаторов, констант **true** и **false**, знака присваивания (**:=**), знаков операций **or**, **xor**, **and**, **not** и круглых скобок.
3. Входной язык содержит операторы условия типа **if** … **then** … **else** и **if** … **then**, разделенные символом **;**(точка с запятой). Операторы условия содержат идентификаторы, знаки сравнения <, >, =, десятичные числа с плавающей точкой (в обычной и логарифмической форме), знак присваивания (**:=**).
4. Входной язык содержит операторы цикла типа **for (**…**;** …**;** …**) do**, разделенные символом **;**(точка с запятой). Операторы цикла содержат идентификаторы, знаки сравнения <, >, =, десятичные числа с плавающей точкой (в обычной и логарифмической форме), знак присваивания (**:=**).
5. Входной язык содержит арифметические выражения, разделенные символом **;**(точка с запятой). Арифметические выражения состоят из идентификаторов, римских чисел, знака присваивания (**:=**), знаков операций **+**, **-**, *****, **/** и круглых скобок.
6. Входной язык содержит логические выражения, разделенные символом **;**(точка с запятой). Логические выражения состоят из идентификаторов, констант **0** и **1**, знака присваивания (**:=**), знаков операций **or**, **xor**, **and**, **not** и круглых скобок.
7. Входной язык содержит операторы условия типа **if** … **then** … **else** и **if** … **then**, разделенные символом **;**(точка с запятой). Операторы условия содержат идентификаторы, знаки сравнения <, >, =, римские числа, знак присваивания (**:=**).

8. Входной язык содержит операторы цикла типа **for (**…**;** …**;** …**) do**, разделенные символом **;**(точка с запятой). Операторы цикла содержат идентификаторы, знаки сравнения <, >, =, римские числа, знак присваивания (**:=**).
9. Входной язык содержит арифметические выражения, разделенные символом **;**(точка с запятой). Арифметические выражения состоят из идентификаторов, шестнадцатеричных чисел, знака присваивания (**:=**), знаков операций **+**, **-**, *****, **/** и круглых скобок.
10. Входной язык содержит логические выражения, разделенные символом **;**(точка с запятой). Логические выражения состоят из идентификаторов, шестнадцатеричных чисел, знака присваивания (**:=**), знаков операций **or**, **xor**, **and**, **not** и круглых скобок.
11. Входной язык содержит операторы условия типа **if** … **then** … **else** и **if** … **then**, разделенные символом **;**(точка с запятой). Операторы условия содержат идентификаторы, знаки сравнения <, >, =, шестнадцатеричные числа, знак присваивания (**:=**).
12. Входной язык содержит операторы цикла типа **for (**…**;** …**;** …**) do**, разделенные символом **;**(точка с запятой). Операторы цикла содержат идентификаторы, знаки сравнения <, >, =, шестнадцатеричные числа, знак присваивания (**:=**).
13. Входной язык содержит арифметические выражения, разделенные символом **;**(точка с запятой). Арифметические выражения состоят из идентификаторов, символьных констант (один символ в одинарных кавычках), знака присваивания (**:=**), знаков операций **+**, **-**, *****, **/** и круглых скобок.
14. Входной язык содержит логические выражения, разделенные символом **;**(точка с запятой). Логические выражения состоят из идентификаторов, символьных констант ‘T’ и ‘F’, знака присваивания (**:=**), знаков операций **or**, **xor**, **and**, **not** и круглых скобок.
15. Входной язык содержит операторы условия типа **if** … **then** … **else** и **if** … **then**, разделенные символом **;**(точка с запятой). Операторы условия содержат идентификаторы, знаки сравнения <, >, =, строковые константы (последовательность символов в двойных кавычках), знак присваивания (**:=**).
16. Входной язык содержит операторы цикла типа **for (**…**;** …**;** …**) do**, разделенные символом **;**(точка с запятой). Операторы цикла содержат идентификаторы, знаки сравнения <, >, =, строковые константы (последовательность символов в двойных кавычках), знак присваивания (**:=**).

Примечание:

– римскими числами считать последовательности больших латинских букв **X**, **V** и **I**;

– шестнадцатеричными числами считать последовательность цифр и символов ‘a’, ‘b’, ‘c’,’d’, ’e’ и ‘f’, начинающуюся с цифры (например: 89, 45ac9, 0abc4);

Алгоритмы работы распознавателя:

1) алгоритм с подбором альтернативы;
2) алгоритм сдвиг-свертка.

Варианты грамматики:

Во всех вариантах символ S является начальным символом грамматики; S, F, T и E обозначают нетерминальные символы. Терминальные символы выделены жирным шрифтом. Вместо символа **a** должны подставляться лексемы.

1. S → **a :=** F**;** ***2.*** S → **a :=** F**;**

F → F+T | T
T → T*****E | T/E | E
E → **(**F**)** | **-(**F**)** | **a**

F → F **or** T | F **xor** T | T
T → T **and** E | E
E → **(**F**)** | **not (**F**)** | **a**

3. S → F**;**
F → **if** E **then** T **else** F | **if** E **then** F | **a := a**
T → **if** E **then** T **else** T | **a := a**
E → **a<a** | **a>a** | **a=a**

4. S → F**;**
F → **for** T **do** F | **a := a**
T → **(**F**;** E**;** F**)** | **(;** E**;** F**)** | **(**F**;** E**;)** | **(;** E**;)**
E → **a<a** | **a>a** | **a=a**

Соответствие варианта грамматики варианту формального языка приведены в следующей таблице

№ варианта	**№ варианта грамматики**	**Допустимые лексемы входного языка**
1	1	Идентификаторы, десятичные числа с плавающей точкой
2	2	Идентификаторы, константы true и false
3	3	Идентификаторы, десятичные числа с плавающей точкой
4	4	Идентификаторы, десятичные числа с плавающей точкой
5	1	Идентификаторы, римские числа
6	2	Идентификаторы, константы 0 и 1
7	3	Идентификаторы, римские числа
8	4	Идентификаторы, римские числа
9	1	Идентификаторы, шестнадцатеричные числа
10	2	Идентификаторы, шестнадцатеричные числа
11	3	Идентификаторы, шестнадцатеричные числа
12	4	Идентификаторы, шестнадцатеричные числа
13	1	Идентификаторы, символьные константы (в одинарных кавычках)
14	2	Идентификаторы, символьные константы ‘T’ и ‘F’
15	3	Идентификаторы, строковые константы (в двойных кавычках)
16	4	Идентификаторы, строковые константы (в двойных кавычках)

5.3 Кейс-задача №2 КЗ2. Примерный перечень тематик и методика выставления баллов, типовой вариант

Тема: Проектирование и реализация синтаксического анализатора на основе распознавателя без возвратами заданного формального языка
Цель кейс-задачи: изучение основных понятий теории грамматик простого и операторного предшествования, ознакомление с алгоритмами синтаксического анализа (разбора) для некоторых классов КС-грамматик, получение практических навыков создания простейшего синтаксического анализатора для заданной грамматики операторного предшествования.

Типовой вариант кейс-задачи
Даны:
1) формальный язык: язык содержит арифметические выражения, разделенные символом **;**(точка с запятой). Арифметические выражения состоят из идентификаторов, десятичных чисел с плавающей точкой (в обычной и логарифмической форме), знака присваивания (**:=**), знаков операций **+**, **-**, *****, **/** и круглых скобок.
2) грамматика КС-языка:
S → **a :=** F**;**
F → F+T | T

T → T*****E | T/E | E

E → **(**F**)** | **-(**F**)** | **a**

3) алгоритм распознавателя: выбрать любой из предложенных по своему усмотрению.

*Задание***:** построить и реализовать на любом языке программирования синтаксический анализатор, для чего:

1. Выполнить простейшие преобразования над заданной КС-грамматикой.

2. Проверить принадлежность КС-грамматики, получившейся в результате преобразований, к одному из известных классов КС-грамматик, для которых существуют линейные распознаватели.

3. Если соответствующий класс найден, взять за основу для построения распознавателя алгоритм разбора входных цепочек, известный для этого класса, если найдено несколько классов линейных распознавателей – выбрать из них один по своему усмотрению.

4. Иначе, если соответствующий класс по п. 2 не был найден или же найденный класс КС-грамматик не устраивает разработчиков компилятора – попытаться выполнить над грамматикой неформальные преобразования с целью подвести ее под интересующий класс КС-грамматик для линейных распознавателей и вернуться к п. 2.

5. Если же ни в п. 3, ни в п. 4 соответствующий распознаватель найти не удалось (что для современных языков программирования практически невозможно), необходимо использовать один из универсальных распознавателей.

6. Определить, в какой форме синтаксический распознаватель будет передавать результаты своей работы другим фазам компилятора (*внутренним представлением программы* в компиляторе).

Реализовать выбранный в п. 3 или 5 алгоритм с учетом структур данных, соответствующих п. 6.

В данной работе в заданиях предлагаются грамматики, не требующие дополнительных преобразований. Кроме того, гарантировано, что все они относятся к классу КС-грамматик операторного предшествования, для которых существует известный алгоритм линейного распознавателя. Поэтому создание синтаксического распознавателя для выполнения задания существенно упрощается.

Для грамматик, предложенных в заданиях, известно, что они относятся также к классам КС-грамматик *LR(1)* и *LALR(1),* для которых также существует известный алгоритм линейного распознавателя.

После несложных преобразований эти же грамматики могут быть приведены к виду, удовлетворяющему требованиям алгоритма рекурсивного спуска (или алгоритма анализа для *LL(1)*-грамматик). Этот алгоритм тривиально прост, для его реализации надо выполнить достаточно несложные неформальные преобразования над заданными грамматиками.

При выполнении задания можно пойти любым из рекомендованных путей или построить иной синтаксический анализатор по своему усмотрению.

В качестве основного пути выполнения задания предлагается распознаватель на основе грамматик операторного предшествования.

Варианты формального языка:

1. Входной язык содержит арифметические выражения, разделенные символом **;**(точка с запятой). Арифметические выражения состоят из идентификаторов, десятичных чисел с плавающей точкой (в обычной и логарифмической форме), знака присваивания (**:=**), знаков операций **+**, **-**, *****, **/** и круглых скобок.

2. Входной язык содержит логические выражения, разделенные символом **;**(точка с запятой). Логические выражения состоят из идентификаторов, констант **true** и **false**, знака присваивания (**:=**), знаков операций **or**, **xor**, **and**, **not** и круглых скобок.
3. Входной язык содержит операторы условия типа **if** … **then** … **else** и **if** … **then**, разделенные символом **;**(точка с запятой). Операторы условия содержат идентификаторы, знаки сравнения <, >, =, десятичные числа с плавающей точкой (в обычной и логарифмической форме), знак присваивания (**:=**).
4. Входной язык содержит операторы цикла типа **for (**…**;** …**;** …**) do**, разделенные символом **;**(точка с запятой). Операторы цикла содержат идентификаторы, знаки сравнения <, >, =, десятичные числа с плавающей точкой (в обычной и логарифмической форме), знак присваивания (**:=**).
5. Входной язык содержит арифметические выражения, разделенные символом **;**(точка с запятой). Арифметические выражения состоят из идентификаторов, римских чисел, знака присваивания (**:=**), знаков операций **+**, **-**, *****, **/** и круглых скобок.
6. Входной язык содержит логические выражения, разделенные символом **;**(точка с запятой). Логические выражения состоят из идентификаторов, констант **0** и **1**, знака присваивания (**:=**), знаков операций **or**, **xor**, **and**, **not** и круглых скобок.
7. Входной язык содержит операторы условия типа **if** … **then** … **else** и **if** … **then**, разделенные символом **;**(точка с запятой). Операторы условия содержат идентификаторы, знаки сравнения <, >, =, римские числа, знак присваивания (**:=**).
8. Входной язык содержит операторы цикла типа **for (**…**;** …**;** …**) do**, разделенные символом **;**(точка с запятой). Операторы цикла содержат идентификаторы, знаки сравнения <, >, =, римские числа, знак присваивания (**:=**).
9. Входной язык содержит арифметические выражения, разделенные символом **;**(точка с запятой). Арифметические выражения состоят из идентификаторов, шестнадцатеричных чисел, знака присваивания (**:=**), знаков операций **+**, **-**, *****, **/** и круглых скобок.
10. Входной язык содержит логические выражения, разделенные символом **;**(точка с запятой). Логические выражения состоят из идентификаторов, шестнадцатеричных чисел, знака присваивания (**:=**), знаков операций **or**, **xor**, **and**, **not** и круглых скобок.
11. Входной язык содержит операторы условия типа **if** … **then** … **else** и **if** … **then**, разделенные символом **;**(точка с запятой). Операторы условия содержат идентификаторы, знаки сравнения <, >, =, шестнадцатеричные числа, знак присваивания (**:=**).
12. Входной язык содержит операторы цикла типа **for (**…**;** …**;** …**) do**, разделенные символом **;**(точка с запятой). Операторы цикла содержат идентификаторы, знаки сравнения <, >, =, шестнадцатеричные числа, знак присваивания (**:=**).
13. Входной язык содержит арифметические выражения, разделенные символом **;**(точка с запятой). Арифметические выражения состоят из идентификаторов, символьных констант (один символ в одинарных кавычках), знака присваивания (**:=**), знаков операций **+**, **-**, *****, **/** и круглых скобок.
14. Входной язык содержит логические выражения, разделенные символом **;**(точка с запятой). Логические выражения состоят из идентификаторов, символьных констант ‘T’ и ‘F’, знака присваивания (**:=**), знаков операций **or**, **xor**, **and**, **not** и круглых скобок.
15. Входной язык содержит операторы условия типа **if** … **then** … **else** и **if** … **then**, разделенные символом **;**(точка с запятой). Операторы условия содержат идентификаторы, знаки сравнения <, >, =, строковые константы (последовательность символов в двойных кавычках), знак присваивания (**:=**).

16. Входной язык содержит операторы цикла типа **for (**…**;** …**;** …**) do**, разделенные символом **;**(точка с запятой). Операторы цикла содержат идентификаторы, знаки сравнения <, >, =, строковые константы (последовательность символов в двойных кавычках), знак присваивания (**:=**).

Примечание:

- римскими числами считать последовательности больших латинских букв **X**, **V** и **I**;
- шестнадцатеричными числами считать последовательность цифр и символов ‘a’, ‘b’, ‘c’,’d’, ’e’ и ‘f’, начинающуюся с цифры (например: 89, 45ac9, 0abc4);

*Варианты грамматики***:**

Во всех вариантах символ S является начальным символом грамматики; S, F, T и E обозначают нетерминальные символы. Терминальные символы выделены жирным шрифтом. Вместо символа **a** должны подставляться лексемы.

1. S → **a :=** F**;**
F → F**+**T | T
T → T*****E | T/E | E
E → **(**F**)** | **-(**F**)** | **a**

2. S → **a :=** F**;**
F → F **or** T | F **xor** T | T
T → T **and** E | E
E → **(**F**)** | **not (**F**)** | **a**

3. S → F**;**
F → **if** E **then** T **else** F | **if** E **then** F | **a := a**
T → **if** E **then** T **else** T | **a := a**
E → **a<a** | **a>a** | **a=a**

4. S → F**;**
F → **for** T **do** F | **a := a**
T → **(**F**;** E**;** F**)** | **(;** E**;** F**)** | **(**F**;** E**;)** | **(;** E**;)**
E → **a<a** | **a>a** | **a=a**

Соответствие варианта грамматики варианту формального языка приведены в следующей таблице

№ варианта	**№ варианта грамматики**	**Допустимые лексемы входного языка**
1	1	Идентификаторы, десятичные числа с плавающей точкой
2	2	Идентификаторы, константы true и false
3	3	Идентификаторы, десятичные числа с плавающей точкой
4	4	Идентификаторы, десятичные числа с плавающей точкой
5	1	Идентификаторы, римские числа
6	2	Идентификаторы, константы 0 и 1
7	3	Идентификаторы, римские числа
8	4	Идентификаторы, римские числа
9	1	Идентификаторы, шестнадцатеричные числа
10	2	Идентификаторы, шестнадцатеричные числа
11	3	Идентификаторы, шестнадцатеричные числа
12	4	Идентификаторы, шестнадцатеричные числа
13	1	Идентификаторы, символьные константы (в одинарных кавычках)
14	2	Идентификаторы, символьные константы ‘T’ и ‘F’
15	3	Идентификаторы, строковые константы (в двойных кавычках)
16	4	Идентификаторы, строковые константы (в двойных кавычках)

*Алгоритмы работы распознавателя***:**

1) распознаватели па основе рекурсивного спуска (модификация алгоритма с подбором альтернатив);

2) распознаватели на основе *LL(1)*- и *LL(k)*- грамматик (модификация алгоритма с подбором альтернатив);

3) распознаватели на основе *LR(0)*- и *LR(1)*-грамматик (модификация алгоритма «сдвиг-свертка»);

4) распознаватели на основе *SLR(1)*- и *LALR(1)*- грамматик (модификация алгоритма «сдвиг-свертка»);

5) распознаватели на основе грамматик предшествования (модификация алгоритма «сдвиг-свертка»).

5.4 Типовой Тест промежуточной аттестации

1 Строки, образованные из последовательности 0 и последующей последовательности 1, включая пустую строку, начинающиеся с 0, описываются регулярным выражением:

а) $0+(0|1)^*$;
б) $0(0|1)^*$;
в) $0(01)^*$;
г) $0(0^*1^*)$;
д) $0^*(0|1)$.

2 Правила вида $A \to B$, где $A, B \in V_N$ называются:

а) леворекурсивными;
б) цепными;
в) факторизованными;
г) бесполезными;
д) праворекурсивными.

3 Процедура *procedure B*; *begin if CH= 'b' then begin gc*; *B end else Err end*; реализует рекурсивный спуск для:

а) правила $B \to bB \mid b$;
б) правила $B \to bB$;
в) правила $B \to b$;
г) правила $B \to Bb$;
д) правила $B \to Bb \mid b$.

4 Достаточные условия применимости метода рекурсивного спуска выполняются в правиле:

а) $S \to aS \mid aB$;
б) $S \to Sa \mid aB$;
в) $S \to Sa \mid a$;
г) $S \to aS \mid bB$;
д) $S \to aS \mid a$.

5 В грамматике с правилами $P = \{S \to S{+}T \mid T;\ T \to a \mid S[S]\}$ справедливо:

а) $FIRST(T) = \{a\}$;
б) $FIRST(T) = \{S, T\}$;
в) $FIRST(T) = \{a, +\}$;
г) $FIRST(T) = \{S, T, a\}$;
д) $FIRST(T) = \{a, [,], +\}$.

6 Распознаватели *LR*(*k*)-грамматик основаны на построении:

а) дерева разбора снизу вверх и левостороннем выводе цепочек;

б) дерева разбора сверху вниз и левостороннем выводе цепочек;

в) дерева разбора снизу вверх и правостороннем выводе цепочек;

г) дерева разбора сверху вниз и правостороннем выводе цепочек;

д) дерева разбора сверху вниз и комбинированном выводе цепочек.

7 Если символ в верхушке стека имеет меньшее или равное простое предшествование, чем очередной входной символ, то:

а) МП-автомат выполняет свертку;

б) МП-автомат выполняет выброс;

в) МП-автомат выполняет сдвиг;

г) МП-автомат выдает ошибку;

д) МП-автомат успешно завершает работу.

8 На этапе синтеза компилятор выполняет:

а) группировку символов в лексемы;

б) синтаксический разбор программы;

в) проверку семантической корректности программы;

г) заполнение таблиц идентификаторов;

д) генерацию объектной программы.

9 Многоадресный код с явно именуемым результатом называется:

а) тетрадами;

б) триадами;

в) ПОЛИЗом;

г) синтаксическим деревом;

д) машинной командой.

10 Этап компиляции, на котором символы исходной программы, группируются в отдельные минимальные единицы текста, несущие смысловую нагрузку:

а) лексический анализ;

б) синтаксический анализ;

в) семантический анализ;

г) генерация кода;

д) оптимизация кода.

11. Цепочкой или словом в данном алфавите называется

а) любая бесконечная последовательность символов алфавита

b) любая конечная последовательность символов алфавита

c) любая конечная последовательность слов в алфавите

12. Язык L в алфавите V определяется как

а) L=V

b) L V

c) L=V*

d) L V*

13. Алфавит – это

а) конечное множество цепочек

b) бесконечное множество символов

c) конечное множество символов

14. Грамматики G1 и G2 называются эквивалентными, если

а) $L(G1) \subseteq L(G2)$

b) G1=G2

c) L(G1)=L(G2)

15. Язык, порождаемый грамматикой G – это

a) множество сентенциальных форм

b) множество нетерминальных сентенциальных форм

c) множество терминальных сентенциальных форм

16. Языком, порождаемым грамматикой G=< VT,VN ,P,S> , называется множество

a) L(G)={ α|α VN*, S }

b) L(G)={α|α VT*, S }

c) L(G)={α|α (VT VN)*, S }

17. Если язык L принимается линейно ограниченным автоматом, то L —

a) контекстно-зависимый язык

b) контекстно-свободный язык

c) регулярный язык

18. Если M — недетерминированный магазинный автомат, и L = N(M), то L —

a) контекстно-свободный язык

b) контекстно-зависимый язык

c) регулярный язык

19. Распознают входные слова и отвечают на вопрос, принадлежит ли поданное слово данному языку

a) автоматные сети

b) автоматы-распознаватели

c) автоматы преобразователи

20. Распознавание принадлежности произвольной строки $\boldsymbol{\alpha} \in \mathbf{V_T^*}$языку L(G) называется

a) пустотой языка

b) разрешимостью языка

c) бесконечностью языка

21. На какой фазе работы компилятора получается объектный файл

a) лексический анализатор

b) синтаксический анализ

c) семантический анализ

d) подготовка к генерации кода

e) генерация кода

22. Исполнимый файл является результатом работы

a) компилятора

b) интерпретатора

c) компоновщика (линковщика)

d) загрузчика

23. Проходом работы компилятора называется

a) процесс последовательного чтения компилятором данных из внутренней памяти, их обработки и помещения результата работы во внешнюю память

b) процесс последовательного чтения компилятором данных из внешней памяти, их обработки и помещения результата работы во внешнюю память

c) процесс последовательного чтения компилятором данных из внешней памяти, их обработки и помещения результата работы во внутреннюю память

24. Результатом работы компилятора является

a) выполненная программа
b) объектный файл
c) исполнимый файл

25. Результатом работы интерпретатора
a) выполненная программа
b) объектный файл
c) исполнимый файл

26. Выберите наиболее распространенные способы организации таблиц идентификаторов
a) размещение элементов в порядке поступления
b) используя списки
c) по методу бинарного дерева
d) на основе хэш-функций
e) двумерный неупорядоченный массив

27. Обратная польская запись для выражения A:=B*10+C*K*T
a) A:=B10*CKT**
b) AB10*CK*T*+:=
c) A:=B10CKT*+**

28. Составные компоненты распознавателя
a) лента
b) устройство управления
c) внешняя память
d) внутренняя память
e) язык программирования

29. Распознаватель – это
a) вычислительная машина
b) специальный алгоритм, позволяющий определить принадлежность цепочки языку
c) грамматика, позволяющий определить принадлежность цепочки языку

30. Для каких типов языков может быть решена задача разбора
a) регулярных
b) контекстно-свободных
c) контекстно-зависимых
d) с фразовой структурой

5.5 Вопросы к экзамену по дисциплине «Конструирование компиляторов»

1. Определения компилятора и интерпретатора.
2. Входной язык, целевой язык, язык реализации.
3. Прямой компилятор. Кросс-трансляторы.
4. Виртуальные машины.
5. Компиляция "на лету".
6. Различные способы задания языков в компиляции: грамматики, конечные и магазинные автоматы.
7. Соотношения между различными способами задания языков.
8. Основные задачи лексического анализа. Регулярные выражения.
9. Использование конечных автоматов для лексического анализа.

10. Синтаксический анализ. Контекстно-свободные грамматики.
11. Нисходящие анализаторы. Метод рекурсивного спуска.
12. Восходящие анализаторы.
13. *LR (k)*-анализаторы. Построение *LR (0)*-анализатора. *LR (1)*-анализатор.
14. *LALR*-анализаторы.
15. Неоднозначные грамматики.
16. Различные типы конфликтов. Разрешение конфликтов.
17. Идентификация. Работа с таблицами. Работа с типами.
18. Причины использования промежуточных языков в компиляторах.
19. Атрибутные деревья разбора. Прямая и обратная польские записи.
20. Триады и тетрады. Управление памятью с точки зрения разработчика компилятора. Проблемы управления памятью.
21. Статическое и динамическое размещение памяти.
22. Стековый механизм управления памятью. Управление кучей. Подсчет ссылок и разметка памяти.
23. Задачи оптимизации кода программы. Виды оптимизирующих преобразований. Представления программы, используемые в оптимизирующих преобразованиях. Примеры оптимизирующих преобразований.
24. Задачи анализа потока управления. Граф потока управления. Доминирование.